MODELO DE SISTEMA PARA GERENCIAMENTO DE CONHECIMENTOS EXPLÍCITOS EM ABORDAGENS DE DFA (*DESIGN FOR ASSEMBLY)*

Antonio Francisco Savi

Exemplar de defesa apresentado à Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para a obtenção do título de Doutor em Engenharia Mecânica

ORIENTADOR: Prof. Tit. Eduardo Vila Gonçalves Filho

São Carlos
Abr / 2009

Dedico:

A Deus e a toda minha família

AGRADECIMENTOS

A Deus, que me deu a vida, a inspiração e a luz para seguir o meu caminho.

Ao Prof. Eduardo Vila Gonçalves Filho, pela orientação, motivação, amizade e pela excepcional contribuição à minha formação como pesquisador.

À minha mulher Erika Monteiro de Souza e Savi, por todo amor, compreensão e carinho no dia a dia.

Aos meus pais Gerson Savi e Yolanda Buran Savi, por todo carinho e motivação de seguir em frente.

Às minhas irmãs Silvana, Elisabete, Lourdes e ao meu cunhado Eduardo, por todo carinho e motivação.

As pessoas do Grupo de Engenharia Integrada (GrupoEI) do Núcleo de Manufatura Avançada (NUMA), em especial ao Prof. Henrique, Prof. Daniel, Cristiane, Fernando e Francis, por toda ajuda e carinho.

Aos funcionários e docentes do Departamento de Engenharia Mecânica da EESC, pelo carinho, atenção e tranqüilidade que me proporcionaram para a conclusão desta pesquisa.

À Escola de Engenharia de São Carlos - Universidade de São Paulo, pela oportunidade de desenvolver este trabalho.

A CNPq pelo apoio financeiro.

A todos que direta ou indiretamente contribuíram para a realização desta pesquisa.

RESUMO

SAVI, A. F. **Modelo de sistema para gerenciamento de conhecimentos explícitos em abordagens de DFA (*Design For Assembly*)**. Tese (Doutorado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2009.

Uma importante fonte de vantagem competitiva para muitas empresas é a capacidade de criar projetos de produtos compostos por um número pequeno de partes e de fácil montagem sem deixar de atender às expectativas do consumidor, denominada abordagem DFA - *Design For Assembly* (Projeto para Montagem). Para o reprojeto ou para reduzir o custo do projeto de novos produtos com esse foco, é necessário obter informações que podem estar muitas vezes armazenadas em locais de difícil acesso e nas mais variadas formas de repositórios do conhecimento. Uma maneira de obter essas informações é criar uma classe sistema, chamada *peer-to-peer*, que permite a sincronização e compartilhamento desses documentos entre cada local espalhado numa rede. Esse tipo de sistema busca a descentralização das informações, ou seja, estas ficam espalhadas pela rede (interna ou externa) com a vantagem de que cada organização poderá manter sob sua "guarda" as informações sem nunca dispô-las em servidores de terceiros e estas poderão chegar automaticamente até os usuários por meio de transações XML. Sendo assim, o objetivo deste trabalho é desenvolver uma ferramenta de auxílio à gestão do conhecimento que atue na geração, codificação e transferência do conhecimento sobre técnicas DFA. A avaliação de soluções existentes foi utilizada como metodologia para propor um modelo teórico que especifica o desenvolvimento do sistema. Pode-se concluir que esse tipo de sistema fornece aos participantes meios de coleta de informações mais eficazes, já que informações sobre DFA podem ser consultadas.

Palavras-chave: Gestão do Conhecimento, DFA, XML

ABSTRACT

SAVI, A. F. **Management Model for Explicit Knowledge in DFA (*Design For Assembly)* Approaches.** PhD Thesis - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2009.

An important source of competitive advantages for many organizations worldwide is the capacity to create projects for products consisting of a small number of parts of easy assemblage, nonetheless attending to the consumers' expectations, named DFA Approach - Design for Assembly. In order to re-project or decrease the expenses caused by designing new products aimed at such, it is necessary to retrieve information data which might be stored in inaccessible places and in a large variety of knowledge repositories. A way of retrieving this data is the creation of a system class, called peer-to-peer, which allows these documents to be synchronized and shared among terminals connected to a network. This system aims at decentralizing data, that is, they are spread throughout the network (internal or external) with the advantage that each company may protect the information by not making it available in computers belonging to third parties. Moreover, this information can automatically reach users by means of XML transactions. Therefore, the aim of this study is to develop a tool to help knowledge management work in generating, codifying and transferring knowledge concerning DFA techniques. The assessment of available solutions was used as a method to propose a theoretical model which pinpoints the system development. It can be concluded that this type of system provides the users with means to collect data more efficiently, since DFA data may be accessed.

Keywords: Knowledge Management; DFA (*Design for Assembly*); XML

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
CIM	*Computer integrated manufacturing*
CSCW	*Computer Supported Cooperative Work*
DOM	*Document Object Model*
DTD	*Document Type Definition*
EIP	*Enterprise Information Portal*
GC	Gestão do Conhecimento
GED	Gerenciamento Eletrônico de Documentos
RMI	*Remote Method Invocation*
RPC	*Remote Procedure Call*
SAX	*Simple API for XML*
TI	Tecnologia da Informação
UML	*Unified Modeling Language*
WEB	Abreviatura de WWW
WWW	*World Wide Web*
XML	*Extensible Markup Language*
XSL	*Extendible Style sheet Language*
RSS	*Rich Site Summary*

SUMÁRIO

1 INTRODUÇÃO

Um estudo do *McKinsey Global Institute* (2008) cita que uma importante fonte de vantagem competitiva para muitas organizações é a capacidade de criar produtos compostos por um número pequeno de partes e de fácil fabricação e montagem sem deixar de atender às expectativas do consumidor. Essa preocupação com o projeto do produto justifica-se à medida que se percebe que projetos bem elaborados em relação à facilidade de montagem podem levar ao aumento da lucratividade da organização.

Existe uma abordagem para facilitar o projeto do produto chamado DFX (do inglês: *Design For eXcellence*) definido como uma abordagem baseada no conhecimento que visa desenvolver projetos de produtos que maximizem diversas características.

Como algumas das variáveis incluídas na excelência do produto é sua facilidade de montagem, surge o DFA (*Design For Assembly*) como parte da abordagem DFX. O objetivo principal do DFA é auxiliar o desenvolvimento de produtos que sejam fáceis de montar e fabricar, com o objetivo de reduzir custos. Estima-se que 50% do custo de manufatura está relacionado ao processo de montagem (HOESKSTRA, 1992). Grandes investimentos são necessários para automatizar a montagem de produtos complexos, quando seria muitas vezes mais econômico reprojetar o produto para simplificar a montagem e o processo de manufatura.

Para o reprojeto do produto ou para reduzir o custo do projeto de novos produtos utilizando esse foco, pode ser necessário ter acesso a informações que podem estar muitas vezes armazenadas em locais de difícil acesso e nas mais variadas formas de repositórios do conhecimento.

Para o compartilhamento dessas informações citadas anteriormente pode ser utilizados portais corporativos da organização, pois os usuários internos já têm acesso e usuários externos recebem as informações disponibilizadas por transações XML. Pensando nesse cenário este trabalho apresenta uma proposta para compartilhamento de conhecimentos explícitos referentes a técnicas de DFA interna e externamente à organização. Podemos chamar essa forma de representação do conhecimento referente à DFA de "Caso de DFA" segundo a teoria de Raciocínio Baseado em Casos (RBC) que faz parte da revisão bibliográfica deste trabalho. Essa forma de compartilhamento se daria à medida que a organização que "criou" o caso DFA necessitasse compartilhá-lo com seus colaboradores internos ou com outras organizações. Este compartilhamento pode ser feito através de transações XML (do inglês: e*Xtensible Markup Language*) que permite transmitir o conteúdo de um banco de dados para ser acessado através de *softwares* específicos ou *Browsers* por outros usuários. Com esse conceito pode-se criar um tipo de rede não muito utilizado, a rede *peer-to-peer* (ponto-a-ponto). Esse tipo de rede busca a descentralização das informações, ou seja, estas ficam espalhadas pela rede (interna ou externa) com a vantagem de que cada organização poderá manter sob sua "guarda" as informações e não irá dispô-las em servidores de terceiros. Outro ponto desse cenário é que o usuário não precisa navegar até *site* de cada empresa para buscar um novo Caso de DFA cadastrado, ou seja, essa informação virá até ele automaticamente por meio das transações XML.

OBJETIVO

O objetivo deste trabalho é desenvolver uma ferramenta de auxílio à gestão do conhecimento que atue na geração, codificação e transferência do conhecimento sobre técnicas DFA.

Como o escopo do objetivo principal é amplo foi divido em sub-objetivos para melhorar a compreensão, são eles:

1.1 Propor um modelo que oriente o desenvolvimento de soluções de gerenciamento do conhecimento explícito sobre o DFA;

1.2 Criar uma solução para gerenciamento de conhecimentos explícitos sobre o DFA baseado no modelo;

1.3 Aplicar a solução em um caso exemplo;

No capítulo 3 em que é apresentada a metodologia será revisto com maiores detalhes o objetivo e sub-objetivos descritos acima.

ESCOPO

O trabalho aborda as soluções para gerenciamento de conhecimentos explícitos, dentro de uma área específica, que é o DFA (*Design for Assembly*). Foram empregados conceitos das áreas de estudo sobre a gestão do conhecimento, processo de desenvolvimento de *software,* Raciocínio Baseado em Casos e XML (*Extensible Markup Language*), com o intuito de avaliar as soluções existentes e propor um modelo teórico que especifica como desenvolver sistemas para gestão do conhecimento para abordagens de DFA, denominado Modelo de Sistema para Gerenciamento de Conhecimentos Explícitos em abordagens de DFA. Espera-se que a partir deste modelo diferentes profissionais possam compartilhar conhecimentos referentes a abordagens de DFA, já que tal ação é tão singular nesse meio.

Neste trabalho propõe-se o Modelo e, para verificar sua viabilidade, será desenvolvida uma solução baseada nas suas especificações, formada a partir do desenvolvimento de uma *intranet*. Por fim, pode-se conduzir uma aplicação desta solução em um caso exemplo.

METODOLOGIA E CARACTERIZAÇÃO DA PESQUISA

Para atingir o objetivo proposto e responder às perguntas de pesquisa estabelecidas, que são:

- Como gerar, a partir de tecnologias e ferramentas de mercado, um sistema para gerenciamento de conhecimentos explícitos de maneira integrada a casos de DFA?

Assim, é empregada uma metodologia de pesquisa que pode ser classificada, como pesquisa de campo do tipo participante-observador. Pesquisa de campo é um rótulo que pode ser atribuído a uma coleção de métodos de pesquisa que envolve a observação direta de ocorrências de eventos naturais. A pesquisa de campo, do tipo

participante-observador, tem como característica básica o fato de que todos os participantes sabem que há a presença de um pesquisador e que o pesquisador influencia e participa diretamente nas ações do fenômeno. Existem autores que chamam esse método de pesquisa-ação. Essa pesquisa se caracteriza pelo envolvimento dos pesquisadores e dos pesquisados no processo diferindo-se da pesquisa científica acadêmica. Dentro desse método, o pesquisador do presente trabalho participa do desenvolvimento da especificação do sistema proposto.

Em relação à abordagem de levantamento e manipulação dos dados diz-se que uma pesquisa-ação, geralmente é de natureza qualitativa, por se tratar de uma amostra intencional, em que indivíduos são selecionados a partir de certas características tidas como relevantes pelos pesquisadores e participantes.

ARTIGOS REFERENTES AO ASSUNTO

Existem na literatura artigos referentes ao assunto proposto neste trabalho, mas não houve àquele que se equiparasse totalmente. A seguir são comentadas algumas dessas referências juntamente com suas descrições e análises críticas.

O artigo "Uma experiência baseada em plano de seqüência de montagem para montagens mecânicas" de SWAMINATHAN & BARBER (1995) apresenta o problema de planejamento de seqüência de montagem baseado na filosofia de "plano de reuso". Pesquisas com planejamento de montagem no passado mostraram por completo cada problema. Estas pesquisas mostram casos armazenados de configurações de montagens básicas que podem ser aplicadas em casos novos.

O artigo a resolução de novos problemas de montagem com o relato de problemas anteriores. Este é o RBC (Raciocínio Baseado em Casos) descrito na teoria da atual literatura. Mas o artigo retrata casos em formato de árvore de montagem e não apresenta nenhum *software* para auxiliar a "procura" de casos.

O artigo "Projeto colaborativo para montagem via internet" de YOU & TSOU & YEH (2006) propõe uma arquitetura para o projeto colaborativo de montagem para o ciclo de vida do produto no estágio de desenvolvimento. A arquitetura adota o protocolo STEP baseado em dados representativos e

características CAD extraídas da implementação do desenho via internet. O projeto colaborativo pode garantir o ciclo de vida do produto e influenciar na produtividade da empresa.

Este artigo trata de comunicação entre ferramentas CAD por meio do protocolo STEP. Para que isso aconteça é necessário que a empresa possua um *software* CAD que "converse" com outra através desde protocolo. O protocolo de comunicação STEP , assim como o CORBA foi bastante utilizado no passado, atualmente usa-se o protocolo padrão XML que pode ser "entendido" por qualquer *software* CAD ou até mesmo por outros *softwares* como sistemas Web. Esse tipo de arquitetura pode ser usado por empresas que obtém, em seus arquivos, todas as peças dos seus produtos desenhadas em *softwares* CAD e configuradas para transmissão via protocolo STEP. Seria improvável o uso desse tipo de arquitetura em empresas pequenas devido ao alto custo de manutenção. Outro inconveniente é que a empresa expõe o desenho de seus produtos e não estaria à disposição de qualquer empresa, só entre as empresas configuradas a usar o protocolo. No ponto de vista de comunidade é totalmente nula essa ação, pois a comunicação se dá por tempo determinado e somente entre duas empresas.

No artigo "RBC DFMA: Um sistema baseado em casos usado para o projeto de montagem de partes no estágio de projeto" de CHANG & SU & PRIEST (2006), existem muitos conflitos entre o projeto do produto e o departamento de manufatura no estágio de projeto, projetistas freqüentemente não tem nenhuma experiência para o sucesso do resultado. Estes resultados no projeto do produto têm baixo nível de "montabilidade" e manufaturabilidade. Este artigo propõe um sistema web baseado em decisões inteligentes, RBC-DFMA, conectando com uma base de casos. Resultados de experimentos recentes indicam que um potencial problema de projeto pode ser detectado, experiência de projeto pode ser efetivamente disseminada e treinamentos eficazes podem ser oferecidos para o projetista empregando este sistema.

Este artigo apresenta uma idéia demonstrada por este modelo de sistema, mas com algumas vantagens e desvantagens. Como vantagem, o sistema do artigo oferece treinamento sobre técnicas de montagem para o projetista de produto. Como

desvantagem o sistema não oferece sua base de casos para uma comunidade de empresas, o que dificulta a resolução de novos casos. O sistema *web* oferecido pelo artigo não irá compartilhar casos com outras empresas e como conseqüência tende a diminuir o cadastramento de casos com o passar do tempo.

O artigo "Um ambiente virtual colaborativo para montagem" de CHEN & XU & LI (2005) apresenta um ambiente Virtual Colaborativo para Montagem (AVCM) que foi desenvolvido para construir um típico ambiente virtual colaborativo para permitir engenheiros localizados geograficamente dispersos, mas com as mesmas tarefas. Este artigo apresenta as partes chaves do AVCM, que são: sua arquitetura, alto nível de comunicação e colaboração, orientação de movimento baseado em descoberta de colisão e reconhecimento de constantes de montagem. Este artigo utiliza CAD para o ambiente virtual.

Este artigo apresenta uma colaboração entre engenheiros. O artigo não descreve como é feita a colaboração só diz como é feita a rede virtual que é através do CAD. Para a rede virtual se consolidar entre somente dois participantes é necessário que possuam softwares CAD para isto também é necessário que configurem protocolos de comunicação, como CORBA e STEP, muito usado para este fim. A rede virtual realmente é muito útil para engenheiros tratando do mesmo projeto em locais geográficos diferentes, mas têm limitações de quantidade de pessoas e de software, isto é, a colaboração é só entre duas pessoas e com softwares CAD. Diferente do que acontece com o software do trabalho proposto que é para várias pessoas ao mesmo tempo e com um browser.

Concluindo, existem vários artigos que tratam do DFA com ambiente WEB relacionando-os com *softwares* CAD, ou seja, a colaboração entre empresas é dada pelo *software* CAD com componentes de comunicação como CORBA e STEP. Pode ser visto em Huang & Mak (1999), Ulieru et al (2000), Driskill & Cohen (1995), Xu et al (2006), Stone & Mcadams & Kayyalethekkel (2004), Jayaram et al (2006), Jayaram & Connacher & Lyons (1996), Shyamsundar & Gadh (2000), Roy & Kodkani (1998), Liang & Wei (2006). Ambientes como esses são de difícil uso e de custo elevado para empresas de pequeno porte, que é o foco deste trabalho, sendo que, as empresas necessitam obter o software CAD para realizar a troca de

experiência. Outro fato negativo desta forma de colaboração é o fato de que a empresa compartilha seus desenhos originais, revelando seus “segredos” industriais.

ESTRUTURA DO TRABALHO

A seguir, o capítulo 2 apresenta a fundamentação teórica obtida através da revisão bibliográfica. O item 2.1 discute os conceitos relacionados com gestão do conhecimento, formas de representação de conhecimentos explícitos e redes de informação e comunicação. O item 2.2 descreve conceitos de RBC (Raciocínio baseado em Casos) um tipo de representação de conhecimento explícito. O item 2.3 descreve o conceito de abordagens de DFA, seu papel na integração de organizações, seus objetivos e conceitos. O item 2.4 apresenta conceitos básicos de modelagem de *software* descrevendo a *Unified Modeling Language* (UML). O item 2.5 apresenta conceitos de XML (*Extensible Markup Language*).

O capítulo 3 apresenta a metodologia utilizada para desenvolvimento do projeto de pesquisa.

O capítulo 4 trata do modelo de sistema para gerenciamento do conhecimento explícito em abordagens de DFA e descreve o desenvolvimento do sistema protótipo. O capítulo 5 apresenta as conclusões e trabalhos futuros. Por fim, são apresentadas as referências bibliográficas (capítulo 6) e os anexos (capítulo 7). Os anexos trazem alguns itens que não são relevantes na leitura dos capítulos.

2 REVISÃO TEÓRICA - INTRODUÇÃO AOS TEMAS PRINCIPAIS DO TRABALHO

Este capítulo apresenta os conceitos fundamentais de 5 áreas relacionadas com este trabalho: gestão do conhecimento, RBC (Raciocínio Baseado em Casos), DFA (*Design for Assembly*), modelagem de *software* e XML (*eXtensible Markup Language*).

2.1 GESTÃO DO CONHECIMENTO

Este item apresenta conceitos como dado, informação e conhecimento para chegar justamente a outro conceito: gestão do conhecimento. Mostra formas de representação de conhecimentos explícitos. Para atingir as metas traçadas no processo de gestão de conhecimento de maneira eficiente usa-se o artifício da tecnologia de informação como apoio a esse processo. Apresenta também conceitos de ferramentas utilizadas no apoio à gestão do conhecimento. E finalmente apresenta conceitos sobre interfaces de sistemas *web* para a Gestão do Conhecimento.

2.1.1 Conceitos básicos de gestão do conhecimento

Gestão do conhecimento (*Knowledge Management*, em inglês) é uma disciplina emergente que tem como principais objetivos identificar, criar, registrar e compartilhar o capital intelectual das organizações. O conceito de conhecimento aplicado ao trabalho nas organizações não é novo. Frases contendo a palavra conhecimento, tal como bases de conhecimento e engenharia do conhecimento, existiam antes da gestão do conhecimento se tornar popular. Muitas empresas de *softwares* se empenharam em desenvolver sistemas relacionando atividades de ensino, captura e reuso de experiências adquiridas, mas nunca haviam usado a

expressão “gestão do conhecimento”. A comunidade de inteligência artificial, por exemplo, tem uma longa tradição na tentativa de representar o conhecimento, armazenamento e aplicação (RODRIGUES, 2001).

O foco da gestão do conhecimento é sobre o indivíduo como um especialista e como portador do importante conhecimento que pode ser sistematicamente compartilhado com a organização (SOFFNER, 2006).

Segundo Rus & Lindvall (2002) o conceito de gestão do conhecimento apareceu em meados de 1980 da necessidade de se diferenciar conhecimento de informação e era principalmente usado como um termo de “*business world*”, isto é, o termo estava sendo usado no mundo dos negócios e não mais só entre intelectuais. Os autores dizem ainda que a implementação e uso de gestão do conhecimento têm crescido rapidamente desde a década de 90: 80% das grandes corporações globais possuíam projetos relacionados com gestão do conhecimento e 40% das organizações que apareceram na revista *Fortune* 1000 possuíam profissionais especializados em conhecimento. Os CKO’s (*Chief Knowledge Officer*, em inglês) são executivos seniores que trabalham a infra-estrutura e a cultura da empresa para o compartilhamento do conhecimento.

O conceito de conhecimento não é consensual. A história da filosofia, desde o período clássico grego, está associada a uma busca sem fim para o significado do conceito de “conhecimento”. Neste trabalho não será seguida a epistemologia para adotar uma definição de conhecimento, um ramo da Filosofia que trata especificamente do estudo da ciência. A definição que descreve o intuito desse projeto será apresentada nos próximos parágrafos.

Segundo Soffner (2006), conhecimento é o recurso chave das tomadas de decisão inteligentes, previsões, projetos, planejamentos, diagnósticos, análises, avaliações e julgamentos intuitivos. É criado e compartilhado entre mentes individuais e coletivas. Não surge de bancos de dados, mas aparecem com a experiência, sucessos, falhas e aprendizagem. Desta forma, o conhecimento está nas pessoas e não nos sistemas computacionais.

Alguns autores entendem conhecimento como “o conjunto de crenças mantidas por um indivíduo acerca de relações causais entre fenômenos, entendendo relações

causais como sendo relações de causa e efeito entre eventos ou ações imagináveis e prováveis conseqüências para aqueles eventos ou ações" (SANCHES, HEENE e THOMAS, 1996). Para Nonaka (1994) conhecimento é definido como algo que não pode ser totalmente estruturado, impossível de ser totalmente capturado e ter sua lógica dissecada e só se manifesta quando é utilizado.

A definição de conhecimento, segundo Grant (1996), é discutida desde os primórdios dos tempos. Existem muitas definições de vários autores cada qual focando a definição de conhecimento em sua área. É preciso, entre essas várias definições, encontrar uma que se adapte ao foco do trabalho em questão. Seguiremos a definição de Davenport & Prusak (1998) que é a que mais descreve conhecimento dentro de um sistema revelando *framework* e também casos, freqüentemente usados para desenvolvimento de sistemas. Assim, conhecimento é uma mistura fluida de experiências, valores, informação contextual e intuição, formando um *framework* (um "cenário") na mente de uma pessoa que a habilita a interpretar, avaliar e tomar decisões, acerca de casos, experiências e/ou informações.

Nonaka & Takeuchi (1997) realizaram uma importante distinção entre dois tipos de conhecimento, também chamados de Categorias de Conhecimento. São eles:

- **Tácito**: são os conhecimentos inerentes às pessoas, isto é, as habilidades que esta possui. Trata-se, portanto, da parcela não estruturada do conhecimento, a qual não pode ser registrada e/ou facilmente transmitida à outra pessoa;
- **Explícito**: são os conhecimentos estruturados e capazes de serem verbalizados. Portanto, é a parte estruturada e objetiva do conhecimento. Aquela que pode ser armazenada, transportada e compartilhada por meio de documentos e sistemas computacionais. Faz parte do conhecimento explícito: normas, registros bibliográficos, livros, procedimentos de trabalho e outros.

Conhecimento tácito dificilmente pode ser totalmente externalizado, ou seja, passado para explícito (NONAKA & TAKEUCHI, 1997).

Existem ainda mais duas grandes classes de elementos relacionados com conhecimento dentro das organizações (DAVENPORT & PRUSAK, 1998). São eles:

- **Dado:** é um conjunto discreto e objetivo de fatos sobre um determinado evento. É, portanto, a parcela quantificável e objetiva do estoque de informação e conhecimento de uma empresa, e está armazenado em bancos de dados ou documentos da empresa;
- **Informação:** é uma mensagem contendo um emissor e um receptor e cujo significado envolve uma nova interpretação de algo, baseado em um conjunto de dados. Como, por exemplo: conforme um valor de temperatura e pressão atmosféricas pode-se inferir que deverá chover amanhã, portanto, tem-se uma informação a respeito do clima. Dentro de qualquer empresa há um complexo e contínuo fluxo de informações seja por meios tecnológicos como sistemas computacionais, ou simplesmente através da interação entre as pessoas.

E qual é a relação entre conhecimento, informação e dados? Todas as características do conhecimento podem ser suportadas pela disponibilidade da informação e dos dados. É aí que entra a tecnologia da informação, os bancos de dados, livros, manuais, documentos e apresentações. Apenas uma ínfima parte do conhecimento pode estar armazenada nestes meios, a maior parte está na cabeça das pessoas (SOFFNER, 2006).

Segundo Terra (2002), também não há uma definição padrão sobre gestão do conhecimento, nem um esquema universal dentro do qual se possam alinhar diferentes profissionais.

Assim adotaremos a definição de Loughbridge (1996) por obter o aspecto sistêmico de que o trabalho necessita, diz que "a gestão do conhecimento pode ser definida como a aquisição, troca e uso do conhecimento dentro das organizações, incluindo os processos de aprendizado e os sistemas de informação, requerendo a transformação do conhecimento pessoal em conhecimento corporativo de forma a ser compartilhado e apropriadamente aplicado, sendo sua sistematização vital às organizações".

Gestão do Conhecimento é uma área essencialmente multidisciplinar que envolve conhecimentos dos campos da teoria das organizações, filosofia, psicologia cognitiva, ciência da informação, entre outros (BARROSO & GOMES, 1999).

Segundo Amaral (2002) existe outras distinções de pesquisas realizadas em gestão do conhecimento. Haveria abordagens centradas no armazenamento ou codificação do conhecimento e outras centradas nas pessoas. Exemplos de trabalhos que fazem esta distinção são Blodgood & Salisbury (2001) e Carayannis (1998). Rubenstein-Montano et al (2001) argumenta que o grande desafio é gerar uma abordagem para a gestão do conhecimento que considere ambos os aspectos e dimensões do conhecimento. Para auxiliar essa dinâmica, a tecnologia de informação (TI) oferece inúmeros recursos tecnológicos e ferramentas, e alguns são descritos no item 2.1.3. Rus & Lindvall (2002) afirmam que desenvolvimentos em TI definitivamente têm habilitado compartilhamento de informações.

No entanto, como afirma Carvalho (2000), é importante ressaltar que a TI desempenha um papel de infra-estrutura, pois a gestão do conhecimento envolve também aspectos humanos e gerenciais. Davenport & Prusak (1998) afirmam que a gestão do conhecimento é muito mais do que tecnologia, mas certamente a tecnologia é uma parte importante da gestão do conhecimento.

Este item mostrou conceitos básicos descrevendo classes e tipos de conhecimento e a importância da sua gestão para as pessoas. Uma maneira importante de disponibilizar o conhecimento entre as pessoas é através de formatos acessíveis. Já que o modelo de sistema deste trabalho representa uma forma de organizar documentos (conhecimento explícito) acerca de técnicas de DFA então, formas de conhecimento explícito serão descritas no item 2.1.2.

2.1.2 Formas de representação de conhecimento explícito

De acordo com Davenport & Prusak (1998), para disponibilizar o conhecimento em um formato que seja acessível ao usuário é necessário definir uma codificação. Um aspecto fundamental para atingir eficientemente esses resultados é a forma como esses conhecimentos são representados.

Amaral (2002), baseado em diversos autores, apresenta várias formas de representação de conhecimento explícito, são elas:

- **Não padronizadas**: são antigas bibliotecas, centros de documentação, manuais de qualidade e outras fontes que não possuem uma estrutura única para sua organização;
- **Mapas do conhecimento**: são os descritores (conhecimento explícito) que apontam onde está o conhecimento, isto é, pessoas, documentos e bases de dados que contêm os diferentes conhecimentos explícitos;
- **Narrativas**: são documentos em forma de histórias, descrevendo acontecimentos em um determinado projeto ou atividade registrados pelos membros da organização;
- **Linguagens Estruturadas**: o conhecimento é expresso numa linguagem formada a partir do idioma corrente, adicionando restrições e padrões que tornam o texto menos ambíguo. Um exemplo é o *English-Like* utilizado na área de análise de sistemas;
- **Regras**: o conhecimento é registrado como um conjunto de regras inter-relacionadas que, por serem estruturadas, tornam-se menos ambíguas e de mais fácil localização;
- **Mapas Cognitivos**: basicamente eles são formados por dois construtos: sentenças e flechas (*Constructs* ,em inglês, é um elemento primitivo de uma linguagem de modelagem, possuidor de uma semântica bem definida). As sentenças são frases próximas às regras, as quais são ligadas por flechas que podem indicar conseqüência, antecedência, deduções ou induções; ou mais de uma delas conforme a abordagem. Veja Ramesh & Tirwana (1999), para um exemplo de ferramenta baseada neste tipo de representação;
- **Ontologias**: são especificações explícitas de conceitos relativos a um determinado domínio de conhecimento;
- **Modelos de processo de negócio**: é uma abstração da realidade expressa em termos de algum formalismo (ou linguagem) definido por um método de modelagem em função do objetivo do usuário.

Este item apresentou vários formatos para registrar o conhecimento. Para auxiliar estas representações existem ferramentas (item 2.1.3) que utilizam essas formas.

2.1.3 Ferramentas que apóiam gestão do conhecimento

A definição de ferramenta de gestão do conhecimento, segundo Carvalho (2000), é um tipo específico de *software* que oferece apoio à pelo menos uma das atividades de geração, codificação ou transferência de conhecimento, definidas por Davenport & Prusak (1998) como sendo os processos principais da gestão do conhecimento.

A forma de estruturação dessas ferramentas de apoio à gestão do conhecimento e seus objetivos básicos tornam-se significativamente diferentes. Devem ser mais do que um depósito de informações estáticas e necessárias para a tomada de decisões dentro do processo atual (AMARAL, 2002). Precisam servir de ferramentas de comunicação e troca de experiência entre as pessoas presentes na organização, auxiliando na tomada de decisões e documentando-as. Em decorrência desta nova forma de inserção dentro das organizações, os sistemas de gestão do conhecimento podem conter além do tradicional gerenciamento de informações e documentos (como relatórios e formulários), todo um conjunto de ferramentas que suportem o trabalho em grupo, como por exemplo, sistemas CSCW (*Computer Supported Cooperative Work*), descrito no item 2.1.3.1.

A funcionalidade essencial de uma ferramenta, isto é, o objetivo principal para o qual o *software* foi construído, constitui um parâmetro adicional na classificação das ferramentas. Amaral (2002) e Carvalho (2000) descrevem um conjunto de ferramentas que são comercializadas sob o rótulo de suportar a gestão do conhecimento. Uma síntese destas descrições é mostrada a seguir:

- **Módulos de recursos humanos de sistemas integrados:** por enquanto, as funcionalidades destes sistemas se limitam principalmente ainda aos dados sobre as pessoas, sua formação, carreira, experiências e conhecimentos dominados. Grandes sistemas corporativos, os chamados ERP (*Enterprise Resource*

Planning) como SAP e ORACLE, estão incluindo em seus módulos de recursos humanos funcionalidades para registro e armazenamento de conhecimentos.

- **Ferramentas de modelagem:** estas ferramentas têm sido utilizadas para padronização, registros de procedimentos e estrutura organizacional. Atualmente os fabricantes estão incorporando funcionalidades que permitem o registro de conhecimentos explícitos. É o caso da ferramenta ARIS da IDS-SCHEER que está comercializando uma configuração específica de sua *suite* de ferramentas de modelagem especialmente para projetos de gerenciamento do conhecimento (SCHEER, 1998);
- **Sistemas de gerenciamento de documentos:** esta classe de sistemas suporta a organização e controle dos documentos provendo: armazenamento de documentos permitindo sua classificação por atributos customizados e poderosas ferramentas de busca, controle de fluxo do processo de trabalho (*workflow*), funcionalidades de segurança incluindo *check in* e *check out* de documentos, visualização de documentos, entre outras que permitem o armazenamento seguro de documentos eletrônicos. Alguns exemplos são: *Electronic Data Management* (EDM), *Product Data Management* (PDM) (GUERRERO, 2001);
- **Sistemas *Peer-to-Peer*:** a classe de sistemas *peer-to-peer* é bastante nova, e se baseia em um novo conceito de organização dos dados. Enquanto as soluções anteriores caminham para o paradigma *web* com servidores centrais de dados, os sistemas *peer-to-peer* buscam a descentralização. O princípio fundamental, é o de criar um sistema que permita a sincronização e compartilhamento dos documentos entre cada computador pessoal espalhado na rede, eliminando a necessidade de um servidor de dados centralizado. O *software* Groove, desenvolvido por Groove Networks (GROOVE, 2003) é um exemplo desse tipo de ferramenta. Esse tipo de ferramenta é um dos referenciais teóricos do presente trabalho;
- **Ferramentas para desenvolvimento de sistemas especialistas, engenharia baseada no conhecimento e ontologias:** grupo de ferramentas para gestão do conhecimento que tem origem na área de sistemas especialistas. É formado por soluções comerciais que permitem o armazenamento de regras, ou criação de

ontologias, formando uma base de conhecimento. Sob esta base, é possível construir sistemas de apoio à tomada de decisões relativas ao domínio do conhecimento armazenado. Um exemplo é o ExpertRules (ATLAR, 2002).

- **Business intelligence:** Segundo Jamil (1999), por *Business Intelligence* compreende-se técnicas, métodos e ferramentas que possibilitam ao usuário analisar dados e com base nestas análises emitir respostas que possam subsidiar objetiva e confiavelmente os processos de decisão numa empresa. Entre estas tecnologias podemos citar o uso de *data warehouses* e *data marts* (armazéns de dados), sistemas de suporte à decisão (*DSS-Decision Support System*), sistemas de informações executivas (*EIS-Executive Information System*) e ferramentas de "mineração" (do inglês *mining*) de dados (CARVALHO, 2000)**;**

- **Sistemas de gerenciamento de projeto:** a classe de sistemas para gerenciamento de projeto é mais tradicional. Foi iniciada há muitos anos com soluções que auxiliavam o controle do cronograma e com otimizações do tipo do cálculo de caminho-crítico. Com o passar dos anos e o desenvolvimento da tecnologia de informática eles têm também seguido o conceito de colaboração. Atualmente, estes sistemas têm caminhado no sentido de criação de portais (*workspaces* ou *intranets*) colaborativos para os times de projeto;

- **Ferramentas de *workgroup computing* (CSCW):** trata-se de uma classe de soluções de suporte ao trabalho colaborativo. Estas soluções compreendem funcionalidades integradas de videoconferência, agenda compartilhada, *e-mail*, editores de texto, editores de figura, visualizadores, diários, entre outras. Exemplos desse tipo de ferramenta são o Microsoft Office e o Lotus Notes. Esse tipo de ferramenta é um dos referenciais teóricos do presente trabalho. O item 2.1.3.1 tem por finalidade descrever esse tipo de ferramenta com mais detalhes;

- **Portais de gestão do conhecimento:** reúnem produtos que permitem a criação de portais corporativos, *intranets* e/ou *extranets*. Estes sites não só oferecem a funcionalidade de busca e uma biblioteca de conteúdo classificado, mas também agregam características como comunidades de interesse, grupos de discussão em tempo real, personalização do conteúdo de acordo com a especificação do

usuário (ex: www.intranets.com). Essas ferramentas trabalham com conhecimentos explícitos, principalmente nos formatos não-estruturado e mapas.

- **Sistemas de ensino à distância:** A evolução da telemática e, especialmente da Internet, veio alterar alguns conceitos de difusão e de gestão de informação e também muitos dos conceitos clássicos tradicionais (baseados na interação professor/aluno). Atualmente, assiste-se à entrada na era das Comunidades Virtuais, com a proliferação de escolas virtuais, universidades virtuais, institutos virtuais, turmas virtuais, com cursos e conteúdos acessíveis via *World Wide Web* (WWW), com possibilidade de aulas colaborativas e interações síncronas ou assíncronas, utilizando vários tipos de metodologia e de tecnologia que promovem e permitem o ensino e a aprendizagem através da utilização da *Internet* como dispositivo de mediação entre os vários intervenientes (*e-learning*).

O item atual descreveu alguns tipos de ferramentas de apoio à gestão do conhecimento. O atual trabalho utiliza alguns tipos de ferramentas como veículo para demonstrar o conceito proposto, assim é detalhado em 2.1.3.1 os sistemas CSCW que contribuem com os conceitos de interatividade entre pessoas com suas ferramentas, por exemplo, o Fórum, e em 2.1.3.2 as *intranets* e portais corporativos que são casos de uso de compartilhamento de informação.

2.1.3.1 Sistemas CSCW (Computer Supported Cooperative Work)

Muitas atividades humanas requerem a atuação de grupos para sua execução. Cooperação é um fenômeno que envolve vários processos: comunicação, negociação, coordenação, co-realização e compartilhamento (BARROS, 1994). Portanto, para que as pessoas trabalhem cooperativamente, em um mesmo local ou geograficamente distribuídas, é necessário que exista entre elas um ambiente de apoio à comunicação.

O ambiente computacional que implementa os processos de apoio à cooperação, possibilitando também o trabalho conjunto, bem como a troca de informações, denomina-se sistema de trabalho cooperativo por computador (CSCW-*Computer Supported Cooperative Work*) (ELLIS, GIBBS & REIN, 1991; BORGES, CAVALCANTI & CAMPOS,1995).

Macedo (1999) diz que o principal interesse de pesquisadores da área de CSCW está em facilitar a cooperação entre grupos de pessoas, através da tecnologia computacional, sejam estes grupos numerosos ou não. Para aumentar a eficiência dos sistemas CSCW, estudiosos de outras áreas como psicologia, sociologia, antropologia, ciências cognitivas e administração vêm contribuindo com diferentes perspectivas e metodologias na aquisição de conhecimento sobre grupos e sugerindo como o trabalho em grupo pode ser mais bem apoiado.

Grupos de trabalhos cooperativos nem sempre estão no mesmo local para discutirem e trabalharem pessoalmente. Eles podem estar geograficamente distribuídos e, neste caso, a comunicação via sistemas CSCW torna-se essencial. Os membros podem optar por se conectarem ao sistema ao mesmo tempo (trabalho síncrono) ou em tempos diferentes (trabalho assíncrono) (MACAULY, 1995). Outras formas de classificação podem ser encontradas na literatura (ELLIS, GIBBS & REIN, 1991; NUNAMAKER, 1991; BORGES, CAVALCANTI & CAMPOS,1995).

Existem várias tentativas de classificação de sistemas CSCW. Macedo (1999), baseado em vários autores, apresenta algumas características funcionais principais desses sistemas, que são:

- **Comunicação entre os membros do grupo**: a comunicação é de extrema importância em situações de trabalho em grupo, seja ela pela transmissão de informações ou para tomada de decisões. A maioria das comunicações via computador ainda é restrita a canais textuais podendo ser síncrona ou assíncrona;
- **Compartilhamento de informações**: esta característica é essencial para grupos devido à necessidade de prevenir esforços repetitivos e assegurar que todos os membros do grupo estejam utilizando a mesma informação, de forma que não haja inconsistências;
- **Coordenação e controle de objetos**: em muitas situações, certos objetos devem ser compartilhados entre o grupo por alguma razão com o devido controle de acesso para garantir a integridade de cada versão manipulada;

- **Compartilhamento de espaços de trabalho**: a tela do computador é considerada um espaço de trabalho que pode ser compartilhado, como por exemplo, para apresentar resultados de um *brainstorming* feito por um grupo;
- **Organização e entendimento do processo de trabalho**: as pessoas que trabalham em grupo devem definir a tarefa a ser realizada, entrarem em acordo sobre o conjunto de atividades a ser executado para a realização da tarefa e finalmente informar as decisões a todos.

Este item mostrou um tipo de ferramenta de apoio à gestão do conhecimento. Esse e outros tipos de ferramentas descritos anteriormente podem ser disponibilizados em uma rede de informação chamada *internet*. Esse tipo de disponibilização é descrito a seguir (item 2.1.3.2).

2.1.3.2 Internet e portais corporativos

Com o advento da tecnologia *world wide web* (www ou *web*), a *internet*, rede mundial de computadores, se tornou a sensação e o grande acontecimento na área de tecnologia de informação e de comunicação (SILVA, AMARAL & ROZENFELD, 2000).

Pode-se definir *internet*, *intranet* e *extranet*, de acordo com Laudon & Laudon (1999), como:

- ***Internet***: é uma rede de redes em escala mundial de milhões de computadores. A maioria dos computadores já não opera mais isoladamente, mas como parte de redes de comunicações. Essas redes são interligadas conectando organizações empresariais, governamentais, científicas, assim como indivíduos em todo o mundo;
- ***Intranet***: é uma rede de computadores privativa que utiliza as mesmas tecnologias que são utilizadas na Internet. A tecnologia da internet pode ser utilizada para distribuir e compartilhar informações dentro da organização, bem como interligá-la com o ambiente externo. É uma rede organizacional interna modelada sobre a *web*. Ela utiliza a infra-estrutura de rede já existente da empresa, os padrões de comunicação da *internet* e o *software* desenvolvido para

a *world wide web.* Com a aplicação da tecnologia da *internet* às suas próprias aplicações comerciais, as empresas podem se comunicar e divulgar informações através de sua organização, mantendo afastados os usuários não-autorizados;

- ***Extranet***: pode ser definida com um conjunto de duas ou mais intranets ligadas em rede. Algumas empresas permitem acesso externo limitado a suas *intranets*. As *extranets* são especialmente úteis para interligar as organizações com clientes e parceiros comerciais, com muitas sendo usadas para fornecer dados sobre disponibilidade de produtos, preço, etc.

As *Intranets*: serviços e tecnologias

Há vários anos as empresas vêm apostando em *intranets*. Hoje estão na terceira geração, como mostra a

Figura 2-1. A primeira geração de *intranets* contava apenas com páginas *HTML* estática que precisavam ser mudadas constantemente pelos programadores. A segunda geração já contava com sistemas acessando banco de dados, surgiram os *sites* de compra. A terceira geração se destaca pela integração a outros sistemas, ERPs por exemplo. Em 2001, 70% das grandes empresas brasileiras possuíam uma *intranet* (Revista Forbes, 2001) *apud* (CONECTT, 2003).

A *intranet* foi inicialmente desenvolvida para suprir a demanda pela propagação de informação na organização. Através de *links*, sistemas e várias páginas, a solução baseada nesta tecnologia está sendo utilizada para otimizar o fluxo de informação em diversas empresas.

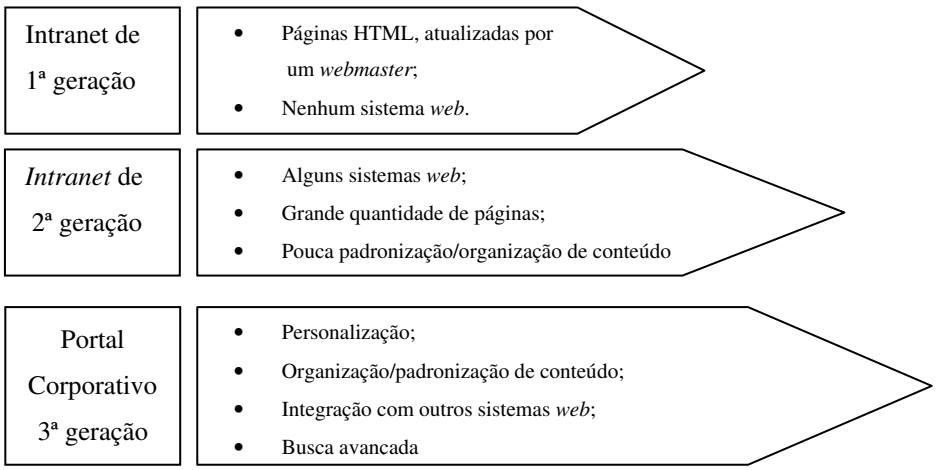


Figura 2-1- Evolução das intranets (CONECTT, 2003)

Mas existem problemas com a forma de utilização que têm sido dadas as *intranets* na maioria das empresas: um "quadro de avisos" eletrônico, utilizado apenas para a divulgação de normas, notícias e outras informações (SOFFNER, 2006).

À medida que as empresas perceberam que apenas essa forma de utilização não era mais suficiente para atender todas as necessidades, surgiu o espaço para um novo tipo de solução: o portal corporativo (3ª geração).

Portais corporativos podem ser definidos como espaços virtuais de trabalho nos quais se encontram as ferramentas de gestão de conhecimento que apóiam a transformação de conhecimento em inteligência de negócios (REYNOLDS & KOULOPOULOS, 1999).

Segundo o Delphi Group (1999), os portais não só oferecem a funcionalidade de busca e uma biblioteca de conteúdo classificado, mas também agregam características como comunidades de interesse, grupos de discussão em tempo real, personalização do conteúdo de acordo com a especificação do usuário e acesso direto a funções especializadas além de permitir o registro do conhecimento explícito pelo usuário.

Os portais corporativos mais complexos, segundo Terra (2002), não oferecem apenas acesso à informação estruturada e não-estruturada. Eles personalizam o acesso à informação, automatizam e aperfeiçoam ciclos complexos de decisões de trabalhadores de conhecimento e podem criar níveis mais profundos de colaboração entre os funcionários. Conseqüentemente, os portais corporativos estão se tornando essenciais para ambientes que exigem uma inovação contínua e de ritmo acelerado.

Batanov & Eloranta (2003) demonstram que os veículos habilitadores da integração empresarial são a tecnologia de informação, comunicação e aplicações, incluindo portais corporativos, baseada em tecnologias *web*.

Cada empresa, de acordo com seu tipo de negócio desenvolve um portal específico. Cavalcanti (2002), descrevendo alguns tipos de ferramentas comuns no mercado em relação a *softwares* de auxílio à gestão do conhecimento observou que os portais corporativos podem ser divididos em cinco aplicações para diferentes áreas do negócio, descritas a seguir:

- **Portal para suporte a vendas**: boa parte das empresas está desenvolvendo portais para dar suporte ao crescente número de representantes de vendas ou parceiros de negócios. Este tipo de portal ganha importância nas empresas de alta tecnologia e outros ramos dinâmicos da indústria. Nestes casos, os profissionais distribuídos pelos mais remotos locais precisam tomar conhecimento dos novos produtos e serviços sendo desenvolvidos para se manterem competitivos. Estes portais em geral oferecem acesso integrado a informações sobre produtos, mercados e clientes. São normalmente desenvolvidos como *extranets* para suportar revendedores, distribuidores ou até mesmo a própria força de venda da empresa;
- **Portal para propagação de melhores práticas**: principalmente atribuído ao ramo de serviços, este tipo de portal busca capturar as melhores práticas desenvolvidas por profissionais altamente especializados. Estes portais servem como meio de treinamento para os funcionários espalhados por diversos lugares, e que se não tivessem o portal estariam criando metodologias, produtos e serviços de suas próprias cabeças, em esforços repetitivos;
- **Portal para inteligência competitiva**: este tipo de portal tem como objetivo unir as diferentes fontes de informação disponíveis como documentos internos, relatórios de "*market-share*" e outras informações externas disponíveis na *internet*, para criar uma fonte de conhecimento que ajude na tomada de decisão;
- **Portal para pesquisa e desenvolvimento**: muitas indústrias e centros de pesquisas desenvolvem pesquisas de maneira cooperativa, distribuindo o trabalho entre diferentes equipes. Nestes casos, os portais corporativos servem para integrar estes esforços, facilitando a comunicação entre as instituições, que em geral possuem grandes bases de conhecimento compartilhadas que aceleram o processo de desenvolvimento de novos produtos;
- **Gerência do conhecimento**: os outros portais definidos não apresentam aplicações específicas e são utilizados para gerenciar o conhecimento da empresa e tornando-a menos dependente de seus empregados.

Segundo Dias (2001) as funções mais importantes de um portal corporativo são suporte à decisão e processamento cooperativo. Alguns autores e fornecedores dão

mais ênfase a uma ou outra função e costumam denominar os portais de acordo com sua principal utilização na empresa. O uso de certas nomenclaturas por parte dos desenvolvedores, entretanto, pode levar a interpretações equivocadas sobre as reais características de seus produtos.

Segundo Balance (2003), existe uma grande diferença entre os níveis de sofisticação encontrados em simples *sites* ou *intranets* e portais corporativos avançados. Os primeiros são instrumentos para departamentos específicos publicarem suas informações ou disponibilizarem seus serviços isolados, já os portais corporativos fornecem a infra-estrutura necessária para desenhar sistemas de informação totalmente customizados e personalizados para cada usuário, esteja este dentro ou fora da empresa.

Não existe uma definição padronizada, sobre quais serviços e funcionalidades deveriam ser incluídos em uma plataforma de portal corporativo, por isso, quando é definido um portal corporativo, faz-se, normalmente, referências a "serviços" porque é isso que importa para os usuários finais e administradores do portal (BALANCE, 2003).

Terra (2002) diz que para considerar um sistema de portal como uma ferramenta de gestão do conhecimento é necessário que ele apresente as seguintes características:

- O conhecimento deve estar disponível democraticamente, ou seja, todos podem acessá-lo, sempre que alguém precisar dele;
- Todos devem ter permissão de colaborar para o repositório de conhecimento;
- Os meios de captura de conhecimento devem estar claros e sistematizados;
- O sistema deve, ao capturar o conhecimento, organizá-lo em uma estrutura simples e de fácil acesso de busca.

Este item mostra a importância da internet para a disponibilização do conhecimento através dos chamados portais corporativos. Mostra ainda alguns tipos de portais existentes no mercado sob o rótulo de ferramentas de auxílio à gestão do conhecimento. O grande desafio nessas ferramentas é a recuperação da informação, que é exatamente a questão de pesquisa do presente trabalho. O item a seguir (2.1.4)

descreve um dos assuntos relacionados com o problema da recuperação da informação, a interface em sistemas *web*.

2.1.4 Interface de sistemas *web*

O termo interface é aplicado normalmente àquilo que interliga dois sistemas. Segundo Moran (1981), a interface de usuário deve ser entendida como sendo a parte de um sistema computacional com a qual uma pessoa entra em contato físico, perceptivo e conceitual.

No processo de interação usuário-sistema a interface é o combinado de *software* e *hardware* necessário para viabilizar e facilitar os processos de comunicação entre o usuário e a aplicação. Os componentes de *hardware* compreendem os dispositivos com os quais os usuários realizam as atividades motoras e perceptivas. Entre eles estão à tela, o teclado, o *mouse* e vários outros. O *software* da interface é a parte do sistema que implementam os processos computacionais necessários para controle dos dispositivos de *hardware*, para a construção dos dispositivos virtuais (chamado *widgets*) com os quais o usuário também pode interagir, para a geração dos diversos símbolos e mensagens que representam as informações do sistema e para a interpretação dos comandos dos usuários (LEITE, 1998).

A interface entre usuários e sistemas computacionais diferencia-se das interfaces de máquinas convencionais por exigir dos usuários um maior esforço cognitivo em atividades de interpretação e expressão das informações que o sistema processa (NORMAN, 1986). Segundo Selker (1996), essa integração entre usuário e computador tem que parecer invisível para o usuário. A interface social, definida assim pelo autor, vem sendo desenvolvida para interação entre pessoas e computadores usando comportamentos humanos familiares. Ele acredita que a interface pode ser mais intuitiva utilizando-se de vários meios como escrita, fala, movimentos dos olhos ou outros gestos.

A introdução da *web* causou uma revolução em projetos de interfaces para usuários. Na área de gestão do conhecimento, Terra (2002), diz que a simplicidade no ambiente de portais corporativos é fundamental para o sucesso de Gestão do

Conhecimento. O projeto gráfico e o *layout* devem ser bastante legíveis (*clean*) e construídos com a participação do usuário. Cada página *web* é um projeto de interface para o usuário e, considerando que no mundo são projetadas milhões de páginas é crucial adotar métodos para sua construção (NIELSEN, 1997).

Profissionais que trabalham com interface devem estudar, segundo Pressman (1987), tópicos relacionados com percepção visual, psicologia cognitiva de leitura, memória humana e raciocínio dedutivo e indutivo. Um exemplo de um tópico relacionado com a percepção visual é descrito por Nielsen (2002), a maioria dos usuários não lêem palavra por palavra em páginas de *internet*, eles fazem o chamado *scan,* ou seja, uma rápida visualização sobre o conteúdo da página. Assim, essas páginas devem conter palavras chaves em destaques, por exemplo, em cores diferentes, subtítulos e listas de conteúdo.

Para adequar-se a esses tipos de ponto de vista, o desenvolvimento de sistemas computacionais interativos não pode ser considerado apenas dentro do escopo restrito da ciência da computação, da engenharia de sistemas, de *software* e de fatores humanos (ergonomia). As habilidades dos usuários, a situação de uso e o contexto onde eles estão são fundamentais para a usabilidade e devem ser considerados no desenvolvimento de sistemas interativos (ADLER & WINOGRAD, 1992). A usabilidade é um conceito que se refere à qualidade da interação de sistemas com os usuários e depende de vários aspectos (LEITE, 1998).

Dentre os aspectos mencionados, por Leite (1998), que são fundamentais para a usabilidade, o aspecto que realmente interessa ao presente trabalho é o contexto. Contexto ou informação de contexto é um termo usado em computação ubíqua. Na visão de Weiser (1991), a tecnologia deve estar integrada ao ambiente e auxiliar os usuários em suas atividades cotidianas.

O item 2.2, descreve conceitos relacionados com o segundo dos cinco conceitos necessários para demonstração da proposta desse trabalho: RBC (Raciocínio baseado em Casos).

2.2 RBC (RACIOCÍNIO BASEADO EM CASOS)

O RBC - Raciocínio Baseado em Casos (do inglês, *Case-Based Reasoning* (CBR)) estabeleceu-se nos últimos anos como uma das tecnologias mais populares e disseminadas para o desenvolvimento de Sistema Baseados em Conhecimento.

RBC é uma abordagem para a solução de problemas e para o aprendizado com base em experiência passada. De uma forma simplificada, podemos entender RBC como a solução de novos problemas por meio da utilização de casos anteriores já conhecidos. Um novo problema que nos é apresentado é resolvido com a reutilização da solução de um problema anterior parecido com o atual. Esta solução pode ser aplicada em sua totalidade ou apenas parcialmente no novo problema, podendo ainda ser modificada de acordo com os requisitos da nova situação.

Assim pode-se definir RBC como "um enfoque para a solução de problemas e para o aprendizado baseado em experiência passada. RBC resolve problemas ao recuperar e adaptar experiências passadas - chamadas casos - armazenadas em uma base de casos. Um novo problema é resolvido com base na adaptação de soluções de problemas similares já conhecidas" (WANGENHEIM & WANGENHEIM, 2003).

Segundo Watson & Marir (1994), o estudo de raciocínio baseado em casos tem seu marco inicial no trabalho de Schank & Abelson (1977) onde foi proposto que nosso conhecimento geral sobre as situações é registrado na forma de *scripts.* Os *scripts* descrevem seqüências de passos ou etapas que nos permitem antecipar como os acontecimentos devem se suceder e realizar inferências a partir dessa expectativa. Os *scripts* são propostos como uma estrutura para a memória conceitual descrever informação sobre eventos estereotipados, como ir a um restaurante ou ir ao médico. Os experimentos mostraram, no entanto, que *scripts* não podiam ser considerados um modelo completo de representação da memória, uma vez que uma pessoa pode "lembrar-se" de *scripts* que são compostos por pedaços de diferentes seqüências de eventos, ou seja, lembrar-se de algo que na verdade foi criação de sua mente. Outros mecanismos atuam na formação da memória e são explicadas por teorias da Filosofia e Psicologia de solução de problemas, formação de conceitos e aprendizado experimental. A teoria da memória dinâmica de Schank foi uma das importantes contribuições para o desenvolvimento da pesquisa na área. A teoria baseou-se na idéia de que não é possível separar experiência, compreensão, memória e

aprendizado e propôs o conceito de pacotes de organização de memória ou MOPs (*memory organization packets*), que utilizam a lembrança de experiências passadas associadas a estereótipos de situações para a solução de problemas e aprendizado. Paralelamente ao trabalho de Schank, Gentner (1983) desenvolveu uma estrutura teórica para o raciocínio por analogia que foi importante para o desenvolvimento de RBC.

Embora raízes filosóficas da teoria de RBC possam se espalhar por trabalhos de diversos pesquisadores no campo da Psicologia ou Ciência da Computação, foram sem dúvida os trabalhos do grupo de Schank, na Universidade de Yale, no início dos anos 80, que produziram o modelo cognitivo de RBC e as primeiras aplicações baseadas nesse modelo.

O RBC também é usado extensivamente para raciocínio de senso comum no dia-a-dia. Quando pedimos uma refeição em um restaurante nos baseamos em outras experiências que tivemos para tomar decisões sobre o que deve ser bom. Quando planejamos nossas atividades domésticas, nos lembramos o que funcionou e o que falhou, e usamos isso para criar nossos planos.

Em 1986, um trabalho alternativo foi desenvolvido na Universidade do Texas, utilizando recursos de classificação heurística e aprendizado de máquina para unificar em um modelo o conhecimento genérico do domínio e o conhecimento específico de casos. O resultado foi o sistema PROTOS com seu modelo de memória de casos (PORTER & BAREISS, 1986). Modelos como esse, que unificam conhecimento genérico e episódico, têm influenciado grandemente os sistemas de RBC, uma vez que o conhecimento do domínio pode melhorar a qualidade do raciocínio, encurtar o caminho de busca da solução ou mesmo preencher lacunas do espaço do problema que os casos naturalmente não cobririam.

Para continuar demonstrando como o RBC pode auxiliar o trabalho com abordagens DFA faz-se necessário revelar os elementos básicos de um sistema desse tipo, o que é feito no próximo item (2.2.1).

2.2.1 Os elementos básicos de um sistema de RBC

A forma principal de representação de conhecimento em um sistema RBC é o caso. Um caso é uma peça de conhecimento contextualizado que registra um episódio em que um problema ou situação problemática foi total ou parcialmente resolvido.

Os elementos básicos de um sistema de raciocínio baseado em casos são:

- **Representação do conhecimento**: em um sistema de RBC, o conhecimento é representado principalmente em forma de casos que descrevem experiências concretas. No entanto, se for necessário, também outros tipos de conhecimento sobre o domínio de aplicação podem ser armazenados em um sistema de RBC, por exemplo, casos abstratos e generalizados, tipos de dados, modelo de objetos usados como informação.
- **Medida de similaridade:** temos de ser capazes de encontrar um caso relevante para o problema atual na base de casos e responder à pergunta quando um caso relembrado for similar a um novo problema.
- **Adaptação:** situações passadas representadas como casos dificilmente serão idênticas às do problema atual. Sistemas de RBC avançados têm mecanismos e conhecimento para adaptar os casos recuperados completamente, para verificar se satisfazem às características da situação presente.
- **Aprendizado:** para que um sistema se mantenha atualizado e evolua continuamente, sempre que ele resolver um problema com sucesso deverá ser capaz de lembrar-se dessa situação no futuro como mais um caso.

Um caso representa tipicamente a descrição de uma situação (problema) conjuntamente com experiências adquiridas (solução) durante sua resolução (veja Figura 2-2).

A essa associação dos dois conjuntos de informações, demonstra-se: a descrição do problema e a respectiva solução.

Casos podem, por exemplo, representar:

- O conjunto de sintomas de um paciente e os passos de um tratamento médico;

- A descrição dos sintomas dos defeitos técnicos apresentados por um equipamento e da estratégia de conserto aplicada;
- Os objetivos de um processo legal e a respectiva jurisprudência;
- A descrição de um pacote de viagem;
- A descrição de abordagens de montagem e suas respectivas técnicas demonstradas passo-a-passo.

O caso pode também conter outros itens, como efeito de aplicação da solução ou a justificativa para aquela solução e sua respectiva explicação (KOLODNER, 1993). Pode ainda ser enriquecido por informações relevantes, como tratados em gestão do conhecimento, ou seja, dados administrativos, melhores práticas ou fóruns realizados para resolver determinado problema.

Caso 1

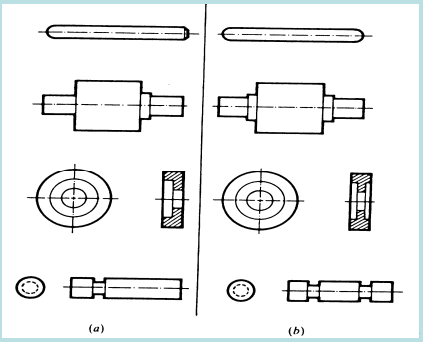
Problemas / Sintomas	
Problema : desenho de um eixo Modelo : EDR 100/05 Material: A1020	Dimensões Raio maior : 60 mm Comprimento: 100 mm
Solução Diagnóstico: figura Obs: não usar esta solução para eixos de comprimento superior a 200 mm	

Figura 2-2 - Exemplo simplificado de um caso

Casos contêm experiências concretas, vividas em uma situação específica. No entanto, podemos também criar casos abstratos, que realizam a junção de experiências adquiridas em um conjunto de situações (BERGMANN, 98).

No caso exposto pela Figura 2-2, são descritos o problema e a solução. A figura no canto inferior direito descreve o antes (lado esquerdo) e o depois (lado

direito) do eixo em questão. Nota-se que o eixo deve conter os dois lados simétricos, sendo assim, diminuiu o tempo de montagem, pois o operador não escolhe o lado para montá-lo.

A recuperação de casos similares é de muita ajuda para a resolução de casos novos. O item a seguir (2.2.2) descreve o conceito de similaridade.

2.2.2 O conceito de similaridade

O objetivo do RBC é a reutilização de soluções conhecidas no contexto de um problema novo, de solução ainda desconhecida. Em função disso, a determinação de exemplos de casos adequados, que não precisam necessariamente ser idênticos à situação atual, é um dos problemas centrais desta técnica.

O conceito de similaridade de casos deve ser útil, de um ponto de vista abstrato. Durante a recuperação de casos procura-se por um exemplo na base de casos, que, no contexto da descrição do problema atual, é útil para determinar a sua solução (BENEDETTI & ABEL, 1999).

É difícil definir exatamente o que significa um caso ser útil para solucionar um determinado problema, como a utilidade deveria ser medida ou o que torna um determinado caso mais apropriado para solucionar um problema do que outro. Em especial, a adequação de um caso para solucionar determinado problema pode ser determinada com certeza somente depois que se tentou elaborar uma solução para este problema com base no caso escolhido. Esta forma de determinação de utilidade, porém, em que um sistema constrói soluções hipotéticas a partir de todos os casos armazenados na base e depois determina qual a melhor para aplicá-la, é impraticável. Portanto, necessita-se de um método heurístico para analisar um caso e julgar sua utilidade sem realizar todo o trabalho de tentar construir uma solução a partir dele e tentar aplicar essa solução, até porque em muitos casos, como em diagnósticos, isto não é possível de ser realizado (BENEDETTI & ABEL, 1999).

Uma das hipóteses básicas de sistemas de RBC é que problemas similares possuem soluções similares. Com base nesta hipótese, o critério *a posteriori* da utilidade de soluções passa a ser reduzido ao critério *a priori* de similaridade de descrições de problema. Esta forma de se proceder é justificada pela premissa de que,

em muitas aplicações, a similaridade de definições de problemas implica na aplicabilidade da solução de um sobre o outro. Portanto a solução descrita em um exemplo de caso escolhido pode ser útil para um novo problema, caso ela (BENEDETTI & ABEL, 1999):

- Permita a solução do problema atual de alguma forma;
- Evite a repetição de um erro anterior;
- Permita uma solução eficiente do problema, que seja mais rápido do que, por exemplo, utilizar uma heurística passo a passo para calcular uma solução;
- Ofereça a melhor solução para o problema de acordo com um critério qualquer;
- Ofereça ao usuário uma solução cuja lógica possa ser compreendida por ele.

Um caso é tão mais útil para a solução de um problema, quanto menor for a necessidade de modificá-lo para adaptá-lo ao problema atual.

O objetivo da determinação da utilidade de casos para um problema atual é o de se determinar uma relação de preferência baseada no critério de similaridade. Está relação de preferência serve para determinar uma ordem parcial sobre os casos da base de casos em relação à utilidade destes casos para o problema particular que o sistema de RBC está tentando resolver.

Resumindo, podemos afirmar que, para a determinação da similaridade em um sistema de RBC, as seguintes premissas devem ser satisfeitas (BURKHARD, 1998):

- A similaridade entre a questão atual e o caso implica utilidade;
- A similaridade é baseada em fatos *a priori*;
- Como casos podem ser mais ou menos úteis em relação a uma questão, a similaridade precisa prover uma medida.

De acordo com o último item do parágrafo anterior a similaridade precisa prover uma medida, e é isso que descreve o item a seguir (2.2.3).

2.2.3 Recuperação de casos

O objetivo da recuperação de casos é encontrar um caso ou pequeno conjunto de casos na base de casos que contenha uma solução útil para o problema ou situação atual. Por exemplo, dada a descrição de um problema de montagem de um eixo em

um produto, um sistema de RBC deveria ser capaz de recuperar um caso descrevendo uma solução apropriada ao problema (por exemplo, mostrar o desenho de um eixo com os dois lados iguais).

Para realizar essa recuperação, é necessário casar a descrição do problema atual como os problemas armazenados na base de casos, aplicando uma medida de similaridade (WANGENHEIM & WANGENHEIM, 2003).

Vários enfoques são aplicados para a realização do processo de correspondência consulta-casos na base para recuperar um conjunto de casos úteis. É necessário discutir primeiramente os princípios gerais da recuperação de casos.

O processo de recuperação de casos pode ser formalmente descrito por meio de um conjunto de subtarefas que devem ser realizadas pelo sistema RBC (WANGENHEIM & WANGENHEIM, 2003):

- **Assessoramento da situação** objetivando a formulação de uma consulta representada por um conjunto de descritores relevantes da situação ou problema atuais;
- **Casamento** objetivando a identificação de um conjunto de casos suficientemente similares à consulta;
- **Seleção** que escolhe o melhor casamento ou casamentos como base no conjunto de casos selecionado.

O assessoramento da situação é a mais complexa das três subtarefas da recuperação, pois utiliza conhecimento e muitas vezes exigem uma interação pró-ativa com o usuário. Para encontrar uma solução adequada ao problema presente, temos de casar nosso problema com os problemas armazenados na base de casos. Problemas do mundo real, no entanto, são muitas vezes mais complexos, apresentando muitas características essenciais. O problema atual pode não se apresentar exatamente da mesma forma que um problema com semântica equivalente armazenado na base de dados. Por isso, durante o processo de recuperação de casos, a meta é recuperar casos potencialmente úteis para a situação presente. Com isso podem-se encontrar casos que são os mais similares à nova situação ao longo de dimensões consideradas importantes no contexto da aplicação. A situação ou problema atual deve ser descrito como entrada para o processo de recuperação, especificando-se o que se está procurando. Isto é realizado por meio de uma consulta

relacionando descritores de entrada – índices que são relevantes para encontrar-se a solução adequada, como sintomas ou características do produto a ser consertado, etc. Esses descritores devem ser elaborados como a entrada do processo de recuperação. Isto é realizado durante o assessoramento da situação (WANGENHEIM & WANGENHEIM, 2003).

Quais descritores de entrada serão importantes? Depende do domínio de aplicação específico do sistema de RBC, por exemplo:

- Quando um médico está buscando pelo diagnóstico de uma doença, ele caracterizará o paciente, por exemplo, idade: 35 anos, sexo: masculino, doenças prévias: malária e descreverá seus sintomas atuais, por exemplo, dor no peito, febre;
- Quando um técnico de serviço está consertando uma impressora e está procurando a causa do problema presente, ele caracterizará a impressora, por exemplo, marca: HP, modelo: 555 e descreverá o sintoma atual, por exemplo, a impressora não funciona;
- Quando procuramos por uma receita de como preparar uma refeição, podemos tentar encontrá-la com base no nome do prato, por exemplo: lasanha de carne.

A identificação de quais descritores é relevante para a solução do problema atual é um dos maiores fatores de complexidade de um sistema de RBC. Quanto menos fatores necessitam ser considerados, tanto mais eficiente será o sistema.

A seção seguinte relata a técnica utilizada para a partir de um conjunto de casos fazer a melhor escolha.

Um grande número de exemplos da vida diária pode ser utilizado para demonstrar como seres humanos utilizam casos conhecidos como forma de resolução de problemas de um modo extremamente natural. Veja alguns exemplos (WANGENHEIM & WANGENHEIM, 2003):

- Ao atender um novo paciente e escutar seus problemas, o médico lembra-se do histórico da doença de outro paciente devido ao conjunto similar de sintomas, e aplica-lhe um tratamento semelhante ao que administrou ao paciente que apresentou aqueles sintomas similares: “Os problemas apresentados nos ouvidos do paciente são parecidos como um caso típico de otite média. Assim vou administrar-lhe um tratamento para otite média”.

- Um técnico de serviço de um determinado tipo de aparelho lembra-se de um defeito similar no tipo de máquina que está tentando consertar: “Essa TV tem os mesmos problemas de uma que eu consertei na semana passada, então vou trocar as placas de saída de áudio”.
- Um profissional jurídico reforça seus argumentos com jurisprudências semelhantes: “Esse caso deve ser decidido como no caso Santos versus de Silva”.
- Um arquiteto estuda as plantas de um prédio já existente ao planejar uma construção similar: “No passado fiz uma casa de praia com três quartos, na encosta de um morro, vou usar o plano daquele caso como uma base”.
- Um vendedor relata para um cliente com necessidades e características semelhantes às de um cliente anterior fatos sobre a venda com sucesso de um determinado produto: “Muitos estudantes ficam neste hotel em Porto de Galinhas”.
- Um projetista de máquinas se depara com a dificuldade de desenhar certo eixo para uma máquina específica: “Este eixo é semelhante ao eixo da máquina X que vi nos arquivos do sistema na semana passada”.

Todas essas situações têm em comum o fato de que uma solução para um problema obtida no passado foi reutilizada para guiar a solução do problema da situação presente.

2.2.4 Técnicas de escolha

A partir do conjunto de casos similares, uma escolha considerada a melhor é realizada. Isto pode ter sido realizado durante o processo de casamento (descrito anteriormente), mas geralmente um conjunto de casos é retornado por essa tarefa. O caso que melhor satisfaz a consulta ou que é mais útil para a solução, é geralmente escolhido por meio da avaliação do grau de casamento ou similaridade de forma mais detalhada. Isto pode ser realizado com uma tentativa de se gerar explicações para justificar atributos não idênticos, como base em conhecimento do domínio de aplicação.

Se um casamento demonstra ser forte o suficiente, uma tentativa de se encontrar um casamento melhor pode ainda ser realizada, por exemplo, com a exploração das

diferenças com os outros casos do conjunto retornado. Se esta tentativa falha, tem-se uma evidência adicional da adequação do caso escolhido.

O processo de seleção gera, tipicamente, conseqüências e expectativas para cada caso recuperado e tenta avaliar conseqüências e justificar expectativas. Isto pode ser realizado com o uso do conhecimento do modelo de conhecimento do domínio do próprio sistema ou com a solicitação ao usuário para que confirme escolhas ou forneça informação adicional. Os casos são eventualmente classificados de acordo com alguma métrica ou critério de ordenação.

A técnica mais simples é a recuperação seqüencial, na qual a medida de similaridade é calculada seqüencialmente para todos os casos na base de dados.

O processo de recuperação seqüencial permite a determinação dos *m* casos mais similares. Para a determinação de uma relação de preferência especifica para a situação aplica-se o conceito de similaridade do sistema sucessivamente a cada um dos casos da base de dados. Todos os casos pertencentes à base são então ordenados de acordo com sua similaridade com a consulta. Como resultado retorna-se então os *m* casos mais similares. O método de recuperação seqüencial é um método de aplicação universal e extremamente fácil de ser implementado para a determinação de casos similares. A principal desvantagem desse método é em relação a bases de dados grandes (WANGENHEIM & WANGENHEIM, 2003).

O item 2.3, descreve conceitos e ferramentas relacionadas com o terceiro dos cinco conceitos necessários para a demonstração da proposta desse trabalho: trata-se do DFA (*Design for Assembly*).

2.3 DFA (*DESIGN FOR ASSEMBLY*)

2.3.1 Contexto

Os conceitos da metodologia de *Design for Manufacturing and Assembly* são conhecidos e aplicados a mais de 180 anos, quando nesta época, Eli Whitney, que trabalhava para o governo norte-americano, recebeu a incumbência de desenvolver um sistema de manufatura de armas. Ele criou uma produção com peças intercambiáveis que substituiu a fabricação manual, onde nenhuma arma era igual a

outra e as peças não podiam ser trocadas. Whitney reprojetou cada peça com uma tolerância limitada e dimensões específicas e padronizadas (RODRIGUES, 2002).

Décadas mais tarde, Henry Ford, com seu automóvel "Model T", desenvolveu um revolucionário sistema de montagem, onde os veículos eram montados manualmente em uma linha de produção, com peças padronizadas e projetadas para atender este processo. Deste modo, Ford atingiu um excelente resultado reduzindo custos de fabricação com alta confiabilidade, qualidade e simplicidade, tornando-se, então, um dos mais famosos engenheiros do setor automotivo.

Roger W. Boltz foi a primeira pessoa e empreendedor a formular a metodologia DFM, apesar de ainda não utilizar esta nomenclatura. Boltz publicou diversos artigos sobre vários processos de manufatura do ponto de vista da engenharia de produtos. Em 1947, todo esse material foi compilado em um único livro chamado "*Production Process – the Productibility Handbook*". Este livro foi editado diversas vezes até 1981(BOOTHROYD, 1992).

Em 1960, a *General Electric* lançou o livro "*Manufacturing Producibility Handbook*" e começou a utilizar o termo "*Producibility*" para identificar essa técnica; anos mais tarde, esse termo começou a ser substituído por *Design for Manufacturability* (BOOTHROYD, 1992).

O *Design for Assembly* surgiu, tempos depois, com Geofrey Boothroyd que ampliou a metodologia do *Design for Manufacturing* para montagem automática, projetando o produto de forma que pudesse atender a essa nova condição. O DFA, além de atender o processo automático, passou também a facilitar a montagem manual dos componentes (BOOTHROYD, 1992).

Em 1968, Boothroyd junto com A. H. Redford publicaram o livro "*Mechanized Assembly*" em que escreveram um guia para engenheiros e projetistas com as técnicas para o desenvolvimento de um produto, atendendo as condições necessárias para a montagem automática e manual (BOOTHROYD, 1992).

Boothroyd ficou bastante conhecido ao lado de outro colega, Peter Dewhurst, pela enorme contribuição que esta técnica trouxe para o desenvolvimento de produtos. A aplicação da metodologia DFA pode trazer uma significante redução de custos e simplificação de produtos.

Atualmente, o conceito do *Design for Manufacturing* foi expandido para outras áreas, ficando genericamente conhecido com *Design for Excellence* (DFX), que engloba o *Design for Manufacturing, Design for Assembly, Design for Environment* entre outros.

2.3.2 Conceitos de DFA (*Design for Assembly*)

Design for Manufacturing é desenvolver um produto que atenda todos os requisitos funcionais, tenha baixo custo de produção e que também seja de fácil manufatura. Essa técnica é composta por vários princípios, conceitos, regras e recomendações que guiam o projetista para o desenvolvimento do melhor produto do ponto de vista da sua fabricação.

Design for Assembly tem por objetivo racionalizar a etapa de montagem por meio da redução do número de peças, analisando separadamente a função, a forma, material e montagem de cada uma delas, desenvolvendo um produto funcional, simples e com baixo custo de produção (RODRIGUES, 2002). Existe entre as duas técnicas pontos em comum como o cálculo do custo, por exemplo (Figura 2-3).

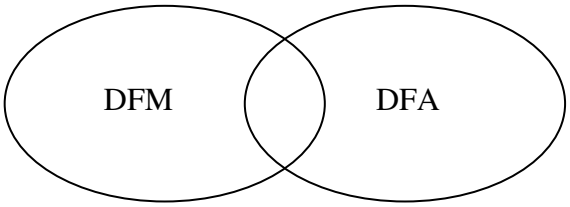


Figura 2-3 - DFM/DFA (BOOTHROYD, 1988)

Boothroyd, Dewhurst e Knigth (1994) dizem que "*Manufacturability*" deve ser entendida como operação de fabricar um componente individual, enquanto "*Assembly*" é a montagem de mais de um componente para formar o produto final; portanto, DFM e DFA devem ser entendidos e aplicados de maneira distinta. Porém, sempre que possível DFM e DFA devem ser aplicados simultaneamente, visando reduzir a complexibilidade tanto da fabricação como da montagem do produto final, evitando, desta forma, que DFM dificulte o processo de montagem por meio das modificações sugeridas, impedindo a plena aplicação do DFA.

Assim um efetivo DFA causa maior eficiência da montagem dos diversos componentes, reduzindo a quantidade de peças e aumentando a qualidade do produto final.

Outras conseqüências são:

- Simplificação dos processos de montagem;
- Redução das operações de manipulação;
- Possibilidade de maior padronização e modularização dos produtos;
- Menor número de passos e ajustes de processamento;
- Menor quantidade de pontos / superfícies de encaixe;
- Redução de problemas de tolerância.

Vê-se, portanto, que a importância de se eliminar componentes não essenciais, principalmente os fixadores separados (parafusos, rebites, etc.), leva o time de projeto a desafiar os limites impostos pelos materiais e processos disponíveis (tecnologia) e procurando inovar soluções mais adequadas.

2.3.3 Causas da não implementação do DFA

Segundo Boothroyd, Dewhurst e Knigth (1994), existem diversas razões para a empresa não implementar o conceito de DFA, como o das resistências humanas ao se encontrar com mudanças, e recusando o grande potencial de melhoria que o DFA pode gerar.

A seguir apresentam-se algumas razões e respostas utilizadas:

- Falta de tempo: é normal que os projetistas reclamem da falta de tempo na execução do produto e que precisam minimizar o tempo de projeto. A verdade é que a implantação do DFA no projeto realmente aumenta o tempo nas fases iniciais do projeto, porém traz benefícios nas fases posteriores, reduzindo-se o tempo total do processo de projeto;
- Baixo volume: quando se tem baixo volume de produção o DFA também deve ser aplicado, pois para os produtos de montagem mais complexa, mesmo de baixo volume, pode reduzir custos de montagem e economizar tempo;

- Baixo custo de montagem: ouve-se também que o custo de montagem já é baixo na produção. No entanto, o DFA não estuda apenas a montagem, mas também o produto, no qual pode ser modificado com o estudo de DFM, reduzindo custos do projeto;
- Resistência humana: cuidado quando se propõem conceitos novos para os engenheiros, projetistas e outros. Eles podem achar que os trabalhos deles estão ruins, por isso a empresa quer a implementação do DFA. Portanto, a idéia da implantação do DFA deve vir deles;
- Prefiro regras de *design*: as regras de design podem guiar projetistas para direções erradas. Geralmente, as regras forçam a criar produtos com geometria mais simples, mais fáceis de serem fabricados. Porém, esse pensamento pode levar no final a maiores custos pela complicação da estrutura do produto e pela má projeção de montagem das peças. Então o DFA une os conceitos guiando o projetista a produtos mais simples e efetivos para processos de montagem resultando na redução de custos.

É importante ressaltar essas causas da não implementação do DFA e também localizar onde as técnicas de DFA estão no processo de desenvolvimento de produtos. O próximo item (2.3.4) descreve essa localização.

2.3.4 Conceitos de desenvolvimento de produtos

É possível encontrar discussões acerca do processo de desenvolvimento de produtos em literaturas relacionadas às disciplinas da engenharia, da qualidade, da administração geral, administração da produção e administração de marketing, assim como da engenharia de produção.

Clark & Fujimoto (1991) sugerem a seguinte definição para o processo de desenvolvimento de produtos: "… é o processo pelo qual uma organização transforma dados sobre oportunidades de mercado e possibilidades técnicas em informações de valor para a produção comercial." Esta é uma definição clássica e que continua válida, bastando que fiquem claros os contornos, início e fim, deste processo; ou melhor, o que se entende por cada uma destas informações: oportunidades de mercado, possibilidades técnicas e informações para a produção.

Conforme apontado por Rozenfeld et al (2000), cada vez mais tem sido ampliado o escopo do processo de desenvolvimento de produto, ou seja, cada vez mais surgem adendos a definição criando-se assim novos definições sobre o tema.

A disciplina de engenharia que comporta maior sistematização no processo de desenvolvimento é a engenharia mecânica. No trabalho de PAHL & BEITZ (1995) é possível encontrar uma detalhada descrição de princípios de projeto mecânico, os quais deveriam ser utilizados na etapa de projeto preliminar (*embodiment design*). Burato (2003) apresenta uma série de métodos de projeto adequados ao desenvolvimento de máquinas e dispositivos de mecânica de precisão. Boothroyd (1988) desenvolveu todo um método de adequação do projeto à estrutura de fabricação de uma empresa mapeando os principais processos de fabricação em diferentes tipos de indústrias.

Em sua obra, Kaminski (2000) explica que o desenvolvimento de produto deve considerar todo o ciclo de produção e consumo; deste modo, propõe a divisão do processo de desenvolvimento do produto em sete etapas, que vão desde a busca das necessidades de mercado, até o descarte final do produto, conforme mostrado a seguir:

1. Estudo de viabilidade;
2. Projeto básico;
3. Projeto executivo;
4. Planejamento da produção;
5. Planejamento da disponibilização do cliente;
6. Planejamento do consumo ou utilização do produto;
7. Planejamento do abandono do produto.

Sousa (1998) explica as definições das etapas do processo do projeto sob o ponto de vista de Ullman apud Sousa (1998), Pahl & Beitz apud Sousa (1998) e Ertas & Jones (1996). Ullman apud Sousa (1998) afirma que o processo deve ser baseado no conceito do ciclo de vida do produto e constituído de seis fases:

1. Desenvolvimento / planejamento de especificações;
2. Projeto conceitual;
3. Projeto do produto;
4. Produção;

5. Uso ou serviço;
6. Descarte ou renovação do produto.

Pode-se notar que, apesar dos autores citados utilizarem nomenclaturas e divisões diferentes, o conteúdo das etapas do desenvolvimento é similar.

Para Boothroyd, Dewhurst e Knigth (1994) o conceito de DFA deve ser aplicado com maior ênfase no inicio da fase conceitual do projeto, pois assim os custos de mudanças são mais baixos e o tempo de duração do projeto pode ser reduzido, disponibilizando o produto mais rápido para o mercado e com preço mais baixo. Nessa fase do projeto conceitual existem apenas conceitos do produto, por exemplo, os desenhos podem ser mudados, pois ainda não há produção, isso acarreta um custo baixo de mudança. Ao contrário de quando o produto já está desenvolvido, os custos para modificações são mais altos e mais difíceis para eventuais mudanças.

A Figura 2-4 a seguir mostra a aplicação do DFA no processo do projeto.

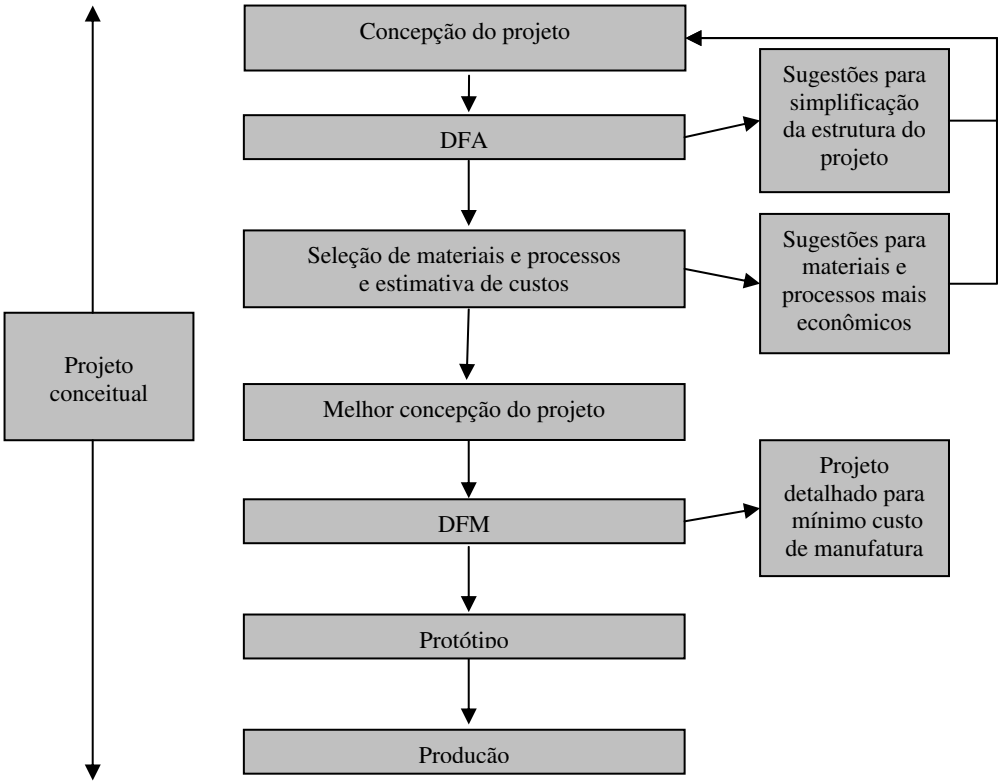


Figura 2-4 - Estrutura da aplicação no processo do projeto (AMARAL, 2002)

Durante o projeto conceitual, as técnicas de DFA são utilizadas para avaliar quais das concepções alternativas são viáveis para serem detalhadas no projeto

preliminar. Deve-se dar atenção a informações sobre manufatura, material, produto e montagem (RODRIGUES, 2002).

Deve-se analisar, ainda nesta fase de projeto, se as alternativas apresentadas até esse ponto atendem aos requisitos de funcionalidade do produto o e aos requisitos de manufatura e montagem.

A implementação do DFA é mais adequada quando a empresa trabalha com a engenharia simultânea, conceito demonstrado no item a seguir (2.3.5).

2.3.5 Engenharia Simultânea

Segundo Chiosoli & Toledo (2000), existem várias definições de engenharia simultânea. De um modo geral, a proposta principal da engenharia simultânea é o desenvolvimento do produto no menor prazo possível através da execução temporal de diversas fases das atividades de engenharia em paralelo, atendendo todos os requisitos por todos os elementos do ciclo de vida de um produto.

Pesquisadores de origem alemã, inglesa e americana discutiam a adoção de metodologias de projeto que permitissem maior produtividade das equipes de engenharia. Esse trabalho culminou nos estudos que defendiam a engenharia simultânea como prática de projeto com poder de gerar produtos a custo reduzido, em menor tempo e com maior qualidade. Existem muitos autores que justificam esse trabalho, são eles: Prasad (1996), Prasad (1997), Clausing (1994) e Ribbens (2000) e ainda Pahl & Beitz (1995) e Pugh (1990), pioneiros nessa abordagem. A metodologia DFA é mais facilmente alcançada quando se adota a metodologia de Engenharia Simultânea para o desenvolvimento de produtos, ou seja, por exemplo, a redução do prazo de desenvolvimento e planejamento da produção, uma vez que várias etapas acontecem simultaneamente.

Segundo Boothroyd, Dewhurst e Knight (1994), usando-se a contabilidade tradicional de custo, o projeto representa, aproximadamente, 5% do custo total de um produto, sendo que cerca de 70% do custo do produto é influenciado durante esta fase do projeto, enquanto que o material, custos diretos e indiretos podem contribuir com 30% restante. Então a partir da perspectiva da manufatura, menos do que 30% do custo do produto pode ser afetado por iniciativas de melhorias, partindo do ponto

que o produto já está definido. Já Rozenfeld & Vega (1995) e Omokawa (1999) afirmam que a influência do projeto no custo do produto chega a 85%, ou seja, acima dos 70% previstos por Boothroyd, Dewhurst e Knight (1994). Portanto, baseando-se nas duas fontes citadas anteriormente, conclui-se que o projeto do produto exerce uma enorme influência na competitividade da empresa, principalmente em custo e o uso do DFA conjuntamente com engenharia simultânea pode potencializar essa economia.

Para Rodrigues (2002), o desenvolvimento de um novo produto ou a mudança de um já existente, não são tarefas fáceis para projetistas. Assim, os engenheiros devem analisar vários pontos importantes, não apenas do produto, mas também da manufatura, vendas, serviços de pós-vendas, qualidade, etc. Porém, ninguém possui conhecimentos suficientes para atender todos os assuntos das áreas envolvidas.

Nesse caso, a técnica de Engenharia Simultânea tem o objetivo de formar um time de projeto com pessoas de diversos departamentos, com especialidades diferentes. Podem-se agrupar vários tipos de profissionais, como: engenheiro de desenvolvimento de produto, engenheiro de manufatura, de planejamento, de processos, pessoal de marketing, vendas, entre outros.

Com pessoas oriundas de diversos departamentos e de diferentes experiências, pode-se resultar em otimização do processo do desenvolvimento do produto que atenda requisitos funcionais, técnicos, de manufatura, etc.

As vantagens de implantar a metodologia de Engenharia Simultânea são (CLAUSING, 1994):

- Redução de custos, pois os problemas são encontrados com mais facilidade e em menor tempo durante o desenvolvimento do projeto onde os custos de modificações ainda não são elevados;
- Redução do prazo de desenvolvimento e planejamento da produção, uma vez que várias etapas acontecem simultaneamente;
- O produto final é considerado de melhor qualidade e desempenho devido à miscigenação de diferentes pontos de vista;

- Maior compatibilidade entre a engenharia de produto e a engenharia de manufatura, já que o produto deve ser compatível com a linha de produção existente na empresa.

Algumas desvantagens para implantar a metodologia são:

- Dificuldade de gerenciar times de trabalho em relação a projetistas individuais;
- Dificuldade de algumas pessoas da engenharia trabalhar em grupos compostas por especialista de outras áreas;
- Custo para reunir grupo de especialistas no desenvolvimento do projeto.

Para aplicar DFA, a técnica de Engenharia Simultânea é a mais indicada para otimizar o desenvolvimento do produto, melhorando os resultados e o desempenho do mesmo.

O time de projeto formado dentro do conceito de Engenharia Simultânea deve seguir os devidos procedimentos do gerente para ter um bom andamento do projeto. Cada representante tem um único objetivo que é desenvolver o melhor produto e de melhor qualidade. Portanto, por exemplo, o representante de *marketing* deve informar ao grupo o que vender e para quem vender, enquanto que o engenheiro do produto vai estudar como será a melhor forma de satisfazer a necessidade do cliente. Já o engenheiro de manufatura indica se é possível fabricar o produto almejado pelo pessoal de produto, assim por diante. Todas as etapas são feitas simultaneamente, o que reduz o tempo de se locomover para reuniões, já que a Engenharia Simultânea tem representante de cada setor.

As vantagens de usar Engenharia Simultânea na aplicação de DFA são:

- A determinação da seqüência de montagem pelo time durante o desenvolvimento do produto ajuda a reduzir os custos de produção e tempo do projeto, pois nesta fase do projeto ainda é possível corrigir falhas de projeto que custariam muito para a manufatura, posteriormente, em termos de compra de equipamentos mais sofisticados e facilidade de montagem;
- Antecipação do planejamento da padronização de operações. Isso pode influenciar na qualidade do produto final. Quanto mais padronizadas forem as operações, menores serão os erros cometidos pelos operadores e maior será a eficiência da operação.

2.3.6 DFX (Design for Excellence)

Atualmente, o mercado consumidor exige muito mais do produto do que apenas o preço mais atrativo trazido pela aplicação do DFA. Em função da grande concorrência e para atrair a atenção dos clientes, o produto deve possuir também outros atributos (KUO & HUANG & ZHANG, 2001).

Neste contexto, surgiu o *Design for Excellence* (DFX), que é definido por Bralla (1996) como sendo "uma base para aprimorar ao máximo o projeto do produto nas suas características desejáveis, como alta qualidade, confiabilidade, facilidade de manutenção, segurança, facilidade de uso, preocupação com o meio ambiente, redução do prazo de disponibilização para vendas e ao mesmo tempo reduzindo os custos de manufatura e manutenção do produto".

O *Design for Excellence* é uma metodologia desenvolvida para estabelecer regras, procedimentos e métodos, de modo a guiar os projetistas para um produto que atenda todos os requisitos esperados pelo mercado consumidor (SCHMIDT, 1998).

A Figura 2-5 mostra a abrangência da técnica DFX.

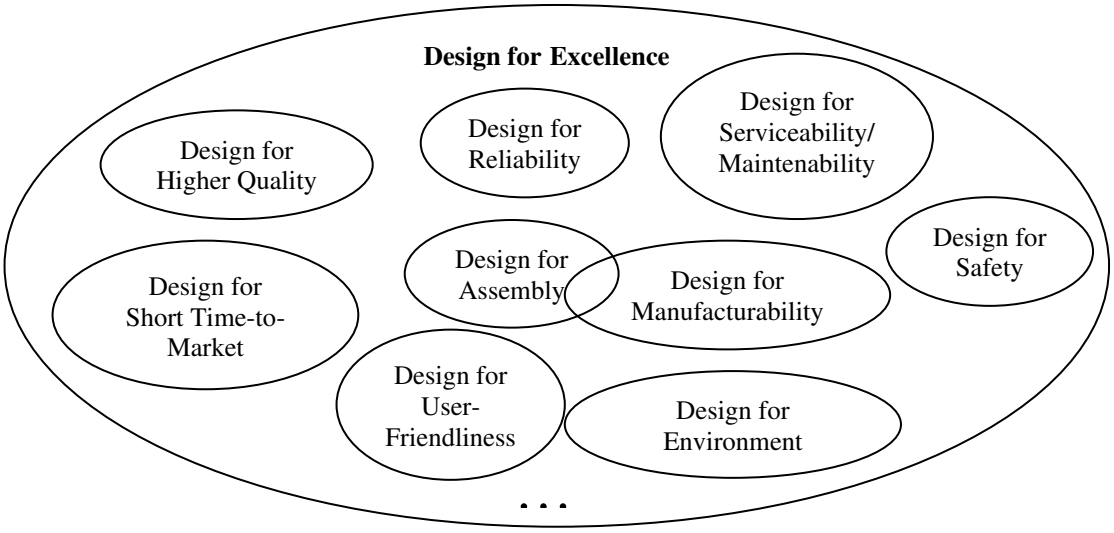


Figura 2-5 - *Design for Excellence* (BOOTHROYD, 1988)

Pode-se notar que apesar do DFX ter sido desenvolvido por meio do sucesso dos conceitos da metodologia DFMA, o seu escopo passou a englobar também essa técnica. Verifica-se também que o DFM e o DFA se complementam, formando o

DFMA. Outra observação é que não existem apenas essas abordagens e intersecções descritas na Figura 2-5, mas muitas outras.

A sigla DFX possui vários significados. Segundo Bralla (1996), DFX é o projeto para todos os fatores que o produto deve ter e para Meerkamm (1994) significa projeto para propriedades.

A seguir serão detalhadas algumas abordagens que fazem parte do DFX:

Design for Higher Quality

Corbett et al. (1991) definem qualidade como satisfazer ou exceder as expectativas do cliente. A qualidade do produto existe quando esse é capaz desempenhar as funções determinadas em projeto nas condições especificadas, atingindo a completa necessidade do mercado consumidor em relação àquele produto. Bralla (1986) afirma ainda que a qualidade do produto pode ser medida de acordo com a satisfação do cliente. Portanto para que o produto possa ser produzido conforme o projeto, este deve atender requisitos de manufatura; deste modo torna-se necessária a aplicação do DFMA e do *Design for Higher Quality*, durante o desenvolvimento do produto, para alcançar a qualidade desejada pelo mercado consumidor.

Design for Reliability

A confiabilidade do produto pode ser definida como: " A medida da habilidade de um produto operar com sucesso, quando solicitado, por um período de tempo pré-determinado, e sob condições ambientais específicas. É medida como uma probabilidade" (*European Organization for Quality Control*, 1965). Deve-se considerar, durante o projeto, que a confiabilidade está diretamente relacionada como o número de componentes existentes no produto; quanto maior, menor será a confiabilidade, pois a possibilidade de ocorrência de uma falha aumenta proporcionalmente, como observado em Moss (1996), Moubray (1997), Lewis (1987), Lafraia (2001), Souza (2003), Doyle (1996), Dias (2003), Dhillon (1983), Kowalski (2000). A curva da banheira, como é conhecida, representa o comportamento da taxa de falhas de um determinado componente ou sistema em função do tempo (Figura 2-6).

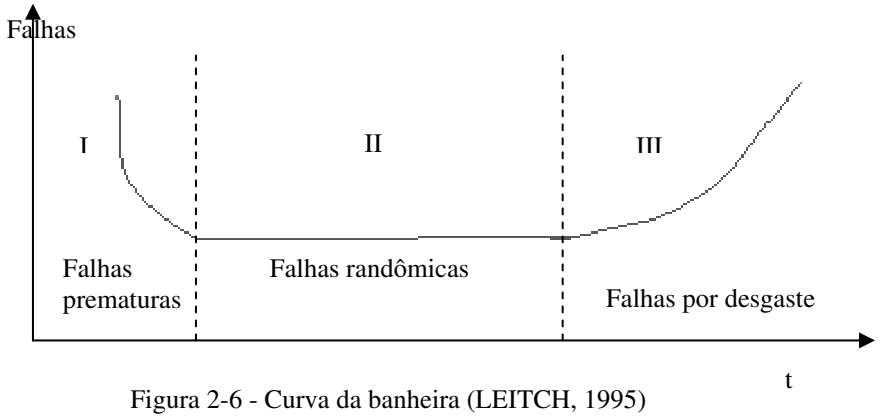


Figura 2-6 - Curva da banheira (LEITCH, 1995)

Pode-se notar que no início a taxa de falhas é elevada, mas decresce com o tempo. A segunda região possui taxas de falhas constante e a chance de falhar passa ser aleatória, não mostrando indícios de que irá ocorrer. A terceira região representa o fim da vida operacional, a taxa de falhas cresce devido ao envelhecimento funcional dos componentes, fadiga e corrosão. O projeto do produto deve ser eficiente a ponto de prevenir que ocorram falhas prematuras, causadas por montagem ou componentes defeituosos (área I da curva). Neste contexto, o DFMA desponta, ao lado de *Design for Reliability*, como uma ferramenta para simplificar o projeto do produto, facilitando sua montagem, evitando que erros causem a falha prematura dos componentes do produto, aumentando assim a confiabilidade.

Design for serviceability / Maintenability

Visa desenvolver o produto de forma que apresente facilidade de manutenção e serviços durante todo a sua vida. Os custos de manutenção de um produto podem causar insatisfação do cliente, e é por esta razão que este aspecto deve ser amplamente abordado durante o desenvolvimento do projeto do produto (CARDOSO, 2000). Na maioria dos casos, a aplicação do DFMA e a conseqüente simplificação do produto fazem com que automaticamente a manutenção deste seja beneficiada. Porém, apenas o DFMA não é o suficiente para resolver problemas de manutenção do produto, o que torna necessário a aplicação de princípios específicos para esta finalidade (DEPARTAMENT OF DEFENSE, 1997).

Bralla (1996) cita algumas regras que devem ser aplicadas quando se objetiva melhorar e reduzir custos de manutenção e serviços, como por exemplo: aumento da confiabilidade, garantir que componentes estejam visíveis, projetar módulos que

possam ser trocados, utilizar peças padronizadas, projetar sistemas auto-diagnosticáveis, incorporar avisos que indiquem a necessidade de manutenção, manuais que indiquem e orientem a manutenção.

Design for Safety

Tem como objetivo garantir a segurança do produto, do ponto de vista da manufatura, do uso e do descarte.

Assim como a confiabilidade e a qualidade, a segurança é um atributo que deve nascer com o produto, pois implementá-la depois é muito complicado e, algumas vezes, impossível.

Segundo Bralla (1996), não existe produto algum que possa ser considerado totalmente seguro. Variações nas condições de uso, falta de entendimento do produto pelo cliente e outros fatores fazem com que exista um risco de segurança. O projetista deve, porém, encontrar um equilíbrio entre os custos de manufatura, outros atributos desejados e o custo da segurança do produto.

Sempre que for identificado algum ponto de possível risco de acidente durante o mau uso, o projetista deve informar o usuário através de avisos no produto e no manual; porém, estas falhas devem ser evitadas ao máximo com a aplicação de sistemas de segurança adicionais.

As empresas devem estar cientes de que não é apenas o projeto que torna o produto mais ou menos seguro; a manufatura pode causar falhas imperceptíveis que ocasionam falta de segurança para o usuário. Durante sua fabricação, o produto deve ser testado de modo a garantir o seu correto funcionamento.

Bralla (1996) cita algumas regras que devem ser aplicadas quando se objetiva melhorar a segurança de produtos, como por exemplo:

1. Projetar mecanismos e dispositivos que evitem acidentes em caso de falha do produto;
2. Desenvolver sistemas que evitem acidentes em caso de falha humana durante o uso do produto;
3. Eliminar quinas pontiagudas e afiadas que possam causar cortes e machucados nos usuários;

4. Projetar capas protetoras, onde existem movimentos mecânicos, para evitar a entrada de objetos estranhos e mãos, e proteger durante a manutenção do produto;
5. O projetista deve garantir o aterramento de produtos elétricos para evitar choques;
6. O projetista deve utilizar materiais com alta resistência a impacto, principalmente quando se trata de produtos para crianças, afim de evitar que, em uma queda o produto se quebre formando quinas, pontas afiadas, etc. que possam causar ferimentos.

Design for the Environment

Também conhecido como *Green Design*, foi elaborado visando a diminuir ou eliminar os problemas de poluição e degradação do meio ambiente.

Além dos benefícios conhecidos, o advento da tecnologia e urbanização das cidades trouxe também graves problemas para o meio ambiente, que estão sendo notados recentemente. A poluição do ar, água e solo e o desmatamento florestal são as principais preocupações dos cientistas e da população de todo o planeta.

Porém, os projetistas têm trabalhado no desenvolvimento de produtos que agridam menos o meio ambiente. Ainda não se pode falar em eliminação de componentes e materiais responsáveis pela poluição, mas na redução de tudo aquilo que possa gerar alguma conseqüência para a natureza (FRANCISCO JUNIOR, GIANETTI e ALMEIDA, 2003 e ADDOUCHE, 2003).

Segundo Schoech et al. (2000), o DFE integra os aspectos ambientais diretamente no processo interno do projeto do produto.

Bralla (1996) descreve o escopo do *Design for the Environment* como sendo o desenvolvimento de produtos, considerando os seguintes aspectos:

1. Matéria-prima: reduzir a poluição do ar, água e terra causada durante a extração da matéria-prima, que possa de alguma maneira prejudicar os trabalhadores, do local, e as famílias que vivem nas proximidades;
2. Manufatura: reduzir a poluição do ar, água e terra causada por materiais gasosos, líquidos e sólidos provenientes do processo de fabricação do produto. Reduzir a poluição sonora da fábrica;

3. Distribuição e vendas: reduzir a poluição causada durante o transporte e manuseio do produto;
4. Uso: reduzir a poluição do ar, água e terra causada pela emissão de gases durante o uso do produto, pelo vazamento de fluidos e gases de refrigeração, bem como a poluição sonora;
5. Descarte: reduzir ou eliminar a poluição do ar, água e terra causada pelo descarte inapropriado de produtos radioativos, fluidos e gases poluentes.

A obrigatoriedade de atender legislações ambientais, preocupação com a possibilidade de escassez de matéria-prima e manter a boa imagem de empresa ecologicamente correta diante do público consumidor, são fatores que fazem as empresas estudarem, planejarem métodos e criarem ferramentas para que os requerimentos ambientais possam influenciar o projeto do produto e haja viabilidade econômica para sua implantação. Na Europa, por exemplo, a meta para indústria automobilística até 2015 é fabricar automóveis com 95 % de seus componentes recicláveis (HARRISON & BLOUNT, 2000).

Design for User-Friendliness

Pode também ser chamado de *Design for Human Factors* e *Design for Ergonomics*. Auxilia os projetistas no desenvolvimento de produtos que são de fácil entendimento por parte de seus clientes, seguros, ergonômicos, confiáveis e que atinjam a satisfação do cliente.

Bralla (1996) afirma que *User-Friendliness* maximiza a utilidade do produto, melhorando sua eficiência, segurança e conforto; produtos famosos pela sua facilidade de uso geralmente são mais fáceis de comercializar.

Bralla (1996) criou alguns princípios para serem aplicados durante o desenvolvimento do produto, como por exemplo:

1. O produto deve ter funções simplificadas, de modo que necessite de um número menor de atividades e conhecimento do usuário para a utilização;
2. O controle do produto deve parecer óbvio para o usuário, de modo que uma pequena ação dele resulte no resultado desejado. O produto não deve confundir o usuário;

3. O projetista deve prever mecanismos que evitem o uso incorreto do produto;
4. O uso de *displays* é um artifício bastante utilizado e desejado pelo cliente para facilitar o uso do produto, indicando funções e apresentando informações importantes;
5. Uma das maneiras de evitar erros durante a utilização do produto é colocar instruções visíveis para que o usuário possa ter acesso mais rápido e fácil;
6. A utilização de funções já conhecidas pelo usuário é uma das maneiras de fazer com que o produto se torne amigável.

Design for short time-to-market

Pode ser definido como sendo o desenvolvimento de um projeto em que se gasta o menor tempo possível para disponibilizar o produto para o mercado consumidor. Este período de tempo é contado a partir da idéia de modificar ou desenvolver um produto até o momento em que ele é comercializado.

As empresas devem analisar cada caso separadamente, pois algumas vezes é mais vantajoso gastar mais para adiantar o desenvolvimento que economizar. Bralla (1996) cita em seu livro que uma empresa americana realizou um estudo em que mostra que um produto que entra no mercado com seis meses de atraso, mas com gastos dentro do planejado, perde cerca de 33% de lucro em um período de 5 anos. Já um produto que é lançado no tempo determinado, mas com gasto extra de 50%, tem seu lucro reduzido em apenas 4% no mesmo espaço de tempo.

Bralla (1996) cita ainda algumas regras com o objetivo de reduzir o prazo de desenvolvimento e liberar o produto mais rapidamente para o mercado consumidor, como por exemplo:

1. Uso de componentes padronizados reduz o tempo de desenvolvimento, teste, aprovação e de desenvolvimento para a produção de um componente novo;
2. Uso de sistemas, procedimentos e materiais padronizados reduzem o tempo de negociação e testes;
3. Uso de sistemas modulares, especialmente se estes já são utilizados por outros produtos;

4. Fazer certo na primeira vez, ou seja, evitar erros que demandem tempo para serem corrigidos, atrasando o projeto;
5. Projetar o produto que não necessite de ferramentas com alto prazo de desenvolvimento.

Design for Disassembly

Segundo Desai & Mital (2003), no contexto da engenharia, a desmontagem pode ser definida como o processo organizado de desmontar um produto sistematicamente montado (conjunto de componentes). Os produtos podem ser desmontados para permitir a manutenção, facilitar a instalação inicial, o reparo e a modificação em campo ou em uso e promover a valorização dos componentes e materiais no fim da sua vida útil como a atividade de reuso, re-manufatura e reciclagem.

Dessa forma, o DFD é condição necessária para que os produtos possam ser economicamente recicláveis. Conforme Desai & Mital (2003), a desmontagem pode ser classificada em dois tipos: não destrutiva e destrutiva. Essa última significando a demolição descontrolada da estrutura do produto. A não destrutiva pode ser classificada em desmontagem total ou seletiva. A total pode não ser economicamente viável, pois deve-se considerar as imposições de prazo, fatores econômicos e presença de materiais tóxicos e perigosos. A segunda é a mais adequada, pois a desmontagem é planejada para que seja feita em subconjuntos pequenos e/ou peças simplesmente.

Segundo Duarte (1997), os benefícios mais evidentes do DFD são:

- Componentes de maior importância podem ser recuperados;
- Melhor separação dos metais, sem contaminação;
- Partes desmontáveis de não metálicos podem ser reprocessadas.

Segundo Addouche (2003), um sistema de desmontagem é um sistema de produção cujo objetivo é a recuperação total ou parcial dos componentes de um ou mais produtos manufaturados de acordo com seus respectivos destinos no final da vida útil. A desmontagem manual é um método dispendioso por estar relacionado aos custos salariais de mão-de-obra, no entanto é muito mais flexível, ou seja eficaz na obtenção de materiais mais “puros” , de frações mais rentáveis, taxa de deterioração ou destruição baixa e capacidade de trabalhar com sistemas complexos. No caso de

um sistema de processo de desmontagem automatizado, há um ganho notável no prazo operacional.

Design for Recyclability

Mediante parágrafos anteriores, pode-se afirmar que o sucesso do projeto para o *Design for Recyclability* - DFR dependerá muito de um bom resultado do DFD. De uma forma geral, o sucesso do DFR e do DFD estão intimamente relacionados e devem ser aplicados conjuntamente e fazem parte da abordagem do DFE, como por exemplo, os subprodutos da desmontagem de um veículo. Schoech et al. (2001), diz que o DFR deve ser aplicado objetivando a reciclagem da matéria-prima do componente, taxa ótima de desmontagem do produto e tratamento adequado do produto no seu final de vida.

No cenário Europeu, segundo Addouche (2003), a recuperação e reciclagem do aço são atividades que geram rendimentos significativos. Nos países industriais, mais de 70% do aço dos bens de consumo que chegam no final de vida é recuperado totalmente. Somente na França, mais de 18 milhões de toneladas do aço são produzidas por ano. Em 1993, equivalente a 60% da produção relativa ao ano, cerca de 10,3 milhões de toneladas, pôde ser coletado.

A vantagem dos aços quanto à coleta e à triagem é o caráter magnético do material, o que lhe permite ser atraído por imantação. Assim, após compactação e esmagamento, os fragmentos e resíduos obtidos são valorizados em quase que sua totalidade pela siderurgia e fundições.

Um exemplo relaciona o conteúdo em massa dos principais componentes e materiais do automóvel produzido no Brasil. Observa-se que os metais são preponderantes em relação aos outros materiais e, portanto, possuem grande importância no ciclo da reciclagem automotiva (Tabela 2-1)

Tabela 2-1 - Conteúdo relativo da composição em massa de cada componente e material em um automável (UNILIVE, 2005)

Componentes / Materiais	Composição mássica relativa do carro
Metais	82,48%
Pneus	4,25%
Vidros	3,35%
Não-metálicos	2,96%
Plásticos	1,68%

Borracha	1,42%
Bateria	1,29%
Espuma	1,29%
Chicote elétrico	0,52%
Óleos	0,41%
Tecidos	0,35%
Outros	0,52%

2.3.7 Princípios básicos do DFA

O *Design for Assembly* leva em consideração a função, forma, o material e a montagem de cada peça de modo a desenvolver um produto funcional e simples, minimizando e/ou incorporando peças do sistema e reduzindo o custo do produto e da montagem. Neste contexto, foram criados vários princípios que auxiliam o projetista a desenvolver um produto.

Em um conceito bastante lógico, a melhor maneira de reduzir os custos de montagem é reduzindo o número de peças e o custo do material e, garantindo que os componentes remanescentes sejam fáceis de montar e de produzir.

No processo de montagem, as peças agrupam-se para formar o produto final. Este processo pode ser dividido em três funções básicas: manipulação, composição e conferência; além de envolver algumas tarefas especiais como empacotamento, ajustagem, tratamento de superfície, etc. Segundo Andersen, Kahler e Lund (1988) definem-se funções básicas como:

- Manipulação: é a função de posicionar dois ou mais objetos em uma determinada posição relativa; engloba a captação, orientação e transporte dos objetos. Isso significa que é um processo de seleção e preparação dos componentes para a composição ou conferência, e transporte para os sistemas seguintes de produção, montagem ou embalagem;
- Composição: é a função de assegurar esta posição relativa contra efeitos externos. Seu objetivo é criar uma conexão permanente entre os componentes. Este processo pode ser alcançado através da forma, força ou material;
- Conferência: é a função de certificar se as etapas acima estão sendo executadas conforme especificado. Ou seja, são processos pelos quais a presença e posições dos

componentes são conferidas em adição à qualidade do produto acabado. Caso ocorra a necessidade de operações adicionais de manipulação e composição como conseqüência da conferência, pode-se chamá-las de ajuste

Um sistema de equipamentos pode ser integrado para atender algumas funções de montagem. Os diferentes tipos de montagem são:

- Montagem manual: executada por operários que utilizam equipamentos auxiliares simples e/ou passivos, como por exemplo, mesas, fixadores e ferramentas;
- Montagem automática: composta de um sistema de equipamentos que segue um programa lógico pré-definido;
- Montagem mecanizada: realizada por um sistema de montagem híbrido, onde algumas operações são realizadas manualmente. Pode-se falar em sistemas semi-automáticos para o caso de equipamentos programáveis manualmente;
- Montagem flexível: quando o sistema de montagem permite variações de determinadas características do produto através da modificação ou adaptação de algumas operações.

No DFA, outros aspectos também são considerados, tais como: projeto para a flexibilidade, racionalização funcional, processos de alimentação, aperto e inserção e suas relações estruturais. O time de projeto realiza decisões envolvendo: a estrutura do produto, o número de peças, a geometria dos componentes, os métodos de união, as tolerâncias de montagem, composição de superfícies e materiais.

Os princípios de DFA estão relacionados à minimização do custo da montagem dentro das restrições impostas pela necessidade de atender encaixe, forma e função de montagem. Estas restrições podem ser humanas, mecânicas ou referentes à seqüência de montagem e desmontagem. Segundo Boothroyd, Dewhurst e Knight (1994), a melhor maneira de alcançar essa minimização é através dos princípios apresentados nos subitens seguintes.

2.3.7.1 Minimização do número de peças

Este é um dos princípios mais evidente e importante de todos, pois com a redução do número de peças, o custo do produto, da montagem e o tempo de montagem também podem ser reduzidos, simplificando o produto final. Sempre que

possível, as peças devem ser combinadas, formando o menor número possível para a montagem, eliminando operações de montagem e submontagem.

Segundo Boothroyd, Dewhrust e Knight (1994), para verificar se o componente é realmente necessário no produto, as seguintes perguntas devem ser respondidas. A resposta positiva significa que a peça deve continuar separada das demais e que não pode ser eliminada.

As perguntas são:

1. O componente deve possuir movimento relativo ao conjunto?
2. O material do componente deve ser diferente do material do conjunto?
3. O componente deve ser separado para permitir a desmontagem e remontagem do conjunto?
4. Este componente é realmente necessário?
5. Estes componentes podem ser combinados?

Sousa (1998) comenta ainda, que o componente deve permanecer separado dos demais quando existe alguma restrição técnica de manufatura e quando a combinação e eliminação dificultam o acesso a outras montagens.

Bralla (1996) cita alguns aspectos que devem ser analisados para combinar as peças de um produto. São eles:

1. Incorporar dobradiças: as dobradiças podem ser incorporadas em vários materiais plásticos e flexíveis;
2. Incorporar molas: molas podem ser incorporadas em materiais como metais, plásticos e, em alguns casos, fibras;
3. Uso de elementos de encaixe, incorporados na peça, em vez de parafusos;
4. Incorporar elementos como guias e coberturas

Em sua obra Bralla (1996) cita ainda um trecho do texto de Stoll que diz: “Menor número de peças significa menos tudo que é necessário para manufaturar um produto. Isto inclui tempo de engenharia, desenhos e números de peças, controle de produção e inventário; número de pedidos de compra, vendedores, etc.; número de *bins*, *containers*, armazém, *buffers*, etc.; número de equipamentos de manuseio, número de movimentos, etc.”

2.3.7.2 Montagem modular ou com componente-base

A montagem modular ou componente-base é o princípio que usa apenas uma única base sobre a qual todos os outros componentes serão montados. Sem esta base, a montagem pode consistir em trabalho sobre muitas montagens, cada uma com suas necessidades de manipulação e composição e a montagem final requerendo extensivo reposicionamento e rearranjo. Essa técnica permite também a incorporação de componentes de fixação e características de alinhamento, facilitando a montagem e reduzindo o custo do produto.

A montagem modular tem como objetivo diversificar produtos a partir da combinação de módulos intercambiáveis e funcionalmente independentes. Este conceito traz como vantagem uma maior agilidade e flexibilidade ao processo.

2.3.7.3 Padronização de componentes

A padronização de componentes implica na redução da variação de peças em uma linha de montagem, redução de tempo com engenharia no desenvolvimento de novos componentes, redução no manuseio, otimização da montagem, padronização de ferramentas, redução de treinamento de pessoal e aumento da qualidade e confiabilidade do produto, entre outros.

Deste modo, sempre que possível, o conceito de padronização de componentes deve ser utilizado, a fim de facilitar a montagem, diminuir custos com equipamentos, peças estocadas na linha e custo final do produto.

Este conceito deve ser amplamente utilizado quando se trata de componentes de fixação, pois, quanto mais padronizados forem, menor será o número de ferramentas, treinamento e variação na linha de montagem.

Aplicando a padronização dos componentes tem-se uma redução das variações desnecessárias dos produtos. Caso contrário implicaria no aumento do inventário, elevando a necessidade de treinamento da mão-de-obra, aumentando as tarefas de manufatura. Todos esses fatores diminuem a qualidade e confiabilidade do produto deixando-o com o custo mais elevado.

A padronização de produtos em uma mesma linha de montagem ajuda a reduzir o custo do produto final, e deve ser aplicado sempre que possível. Quando o produto

é padronizado, custos com mão de obra, treinamento, ferramentas, etc. podem ser reduzidos. Sousa (1998) diz que, quando a padronização é inevitável, sempre que possível deve-se incorporar características para todas as variantes do produto numa submontagem comum.

2.3.7.4 Projeto de peças com características autofixadoras

Projetar as peças com características autofixadoras no produto é sempre útil, pois reduz números de elementos fixadores como parafusos, porcas, arruelas e rebites. Todos estes elementos não agregam valor para o produto. Porém, não são todos os produtos que podem ter essa característica. Se a fixação necessita de esforços relativamente altos, devem-se usar soldas, rebites, entre outros.

Segundo Sousa (1998), alguns caminhos para se usar a autofixação são:

1. Projetar encaixes em peças plásticas (exemplo: *snaps*);
2. Criar características do tipo *tab-in-slot* em chapas metálicas de modo a utilizar o menor número de fixadores.

É importante também considerar a desmontagem, sendo que encaixes do tipo *snaps* são mais fáceis de desmontar, enquanto que parafuso e porcas são mais difíceis, a adesivos e soldas são dificílimos. A Figura 2-7 mostra um exemplo de peças com elementos autofixadores.

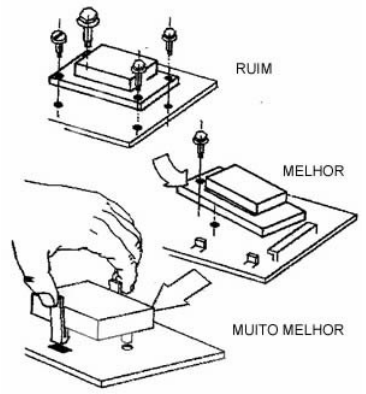


Figura 2-7 - Característica auto-fixadora (BOOTHROYD, 1988)

2.3.7.5 Montagem empilhada ou unidirecional

Durante o projeto de um produto, deve sempre dar preferência para a montagem unidirecional, utilizando sempre a lei da gravidade, ou seja, de cima para baixo (Figura 2-8). A montagem empilhada ou unidirecional reduz o número de reorientações do componente durante sua montagem, facilitando o processo de fabricação.

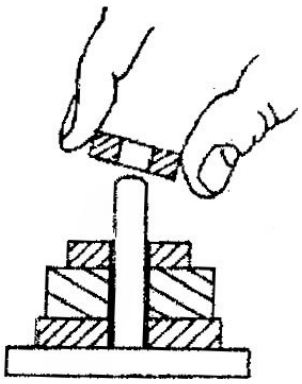


Figura 2-8 - Montagem unidirecional (BOOTHROYD, 1988)

2.3.7.6 Projetar peças com características de autolocalização

Os projetistas devem, sempre que possível, desenvolver componentes com características autolocalizadoras, visando uma montagem precisa, sem ajustes, rápida e fácil (Figura 2-9). Peças com estas características reduzem o tempo de montagem e aumentam a qualidade do produto, além de permitirem que a montagem seja feita automaticamente por um equipamento. Este princípio reduz treinamentos dos operadores, simplifica o processo e assegura que o desempenho do produto não será prejudicado, gerando aumento de qualidade e redução no tempo de montagem.

Características autolocalizadoras podem ser conseguidas no componente por intermédio de chanfros, rebaixos, alargamento de folgas/tolerância, etc.

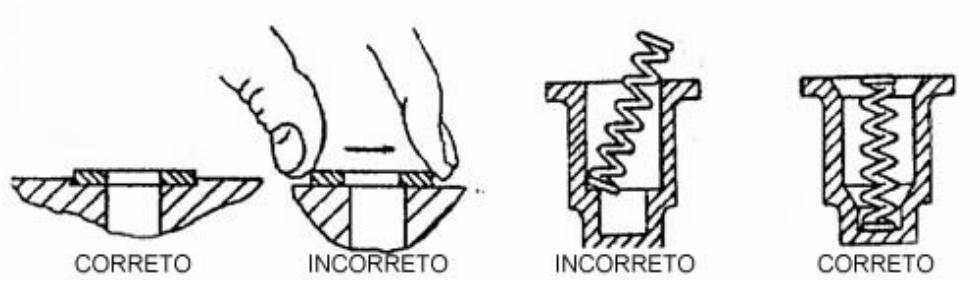


Figura 2-9 – Autolocalização (BOOTHROYD, 1988)

2.3.7.7 Minimização de níveis de montagem

Sempre que possível, os componentes devem ser agrupados em submontagens para melhorar a flexibilidade na programação e no planejamento do processo de montagem. Portanto, a seqüência de montagem pode ser planejada ou eliminada para minimizar a perda de tempo na mudança de ferramental.

Reduzindo o número de montagens ou submontagens em um processo, simplificam-se as especificações, documentação e o layout da fábrica, aumentando o espaço físico para otimizar operações da linha de produção.

A submontagem de componentes, além de facilitar a manufatura, traz também benefícios para o cliente, pois este processo aumenta a qualidade e confiabilidade do produto e facilita e simplifica a manutenção do mesmo.

2.3.7.8 Facilidade de manipulação de peças

Devem-se projetar componentes com peso reduzido e que permitam fácil manipulação, facilitando a montagem e reduzindo o tempo com a operação.

Segundo Sousa (1998), os principais fatores que afetam a manipulação são:

1. Geometria: pode ser simplificada pelo emprego de formas regulares;
2. Rigidez: evitar materiais macios, moles, pontiagudos ou frágeis(Figura 2-10);

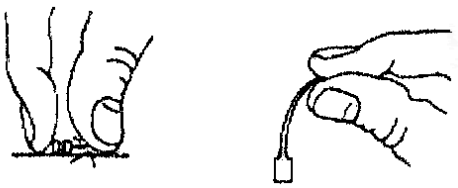


Figura 2-10 - Facilidade de manipulação de peças (BOOTHROYD, 1988)

3. peso: evitar componentes pesados;

4. Tamanho: não utilizar peças muito pequenas, muito grandes ou escorregadias que possam dificultar o manuseio (Figura 2-11);

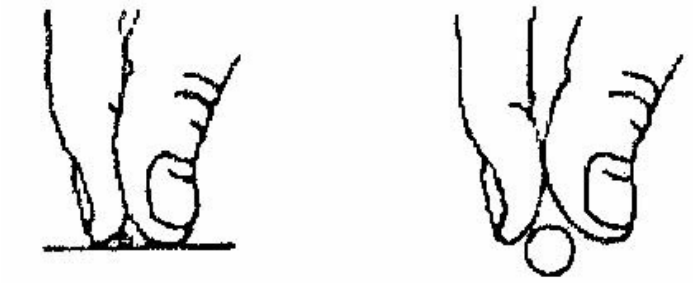


Figura 2-11 - Tamanho de peças (BOOTHROYD, 1988)

5. Simetria: peças simétricas reduzem a orientação e a ocorrência de falhas durante a montagem; se a simetria não for possível, projetar características obviamente assimétricas (Figura 2-12 e Figura 2-13);

Figura 2-12 – Simetria (BOOTHROYD, 1988)

Figura 2-13 – Assimetria (BOOTHROYD, 1988)

6. Não utilizar peças que se aninham ou se emaranham (Figura 2-14);

Figura 2-14 - Peças que não emaranham (BOOTHROYD, 1988)

7. Considerar o empacotamento individual das peças;

8. Usar furos ovais para evitar ajustes;
9. Usar propriedades elásticas dos plásticos como uma vantagem;
10. Facilitar o acesso ao componente, maximizando o espaço disponível;
11. Evitar utilizar peças que necessitem serem manipuladas por duas ou mais mãos; ou dois operadores.

2.3.7.9 Projeto para estabilidade

Desenvolver componentes para estabilidade é projetar peças que não sejam sujeitas a movimentos durante a montagem. As peças devem permanecer estáticas durante a operação. Deste modo, o operador não precisa se preocupar e perder tempo com o processo, pois os componentes envolvidos permanecerão imóveis durante a montagem, não havendo risco de perdê-los.

2.3.7.10 Minimização das necessidades de ajustes

A eliminação ou redução de ajustes durante a montagem ajuda a diminuir o tempo do processo, facilitando a manufatura e aumentando a qualidade e confiabilidade do produto; deste modo, esta técnica deve ser adotada sempre que possível. A

Figura 2-15 mostra que, com o furo oval o operador não necessita de ajuste.

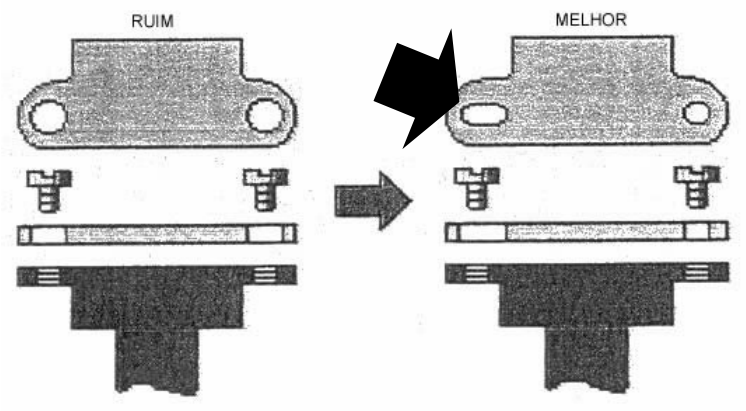


Figura 2-15 - Necessidade de ajustes (BOOTHROYD, 1988)

2.3.7.11 Otimização da seqüência de montagem

A seqüência de montagem deve ser determinada através da coerência na montagem do produto. Quanto mais otimizado for, melhor será a manufatura do produto, reduzindo tempo e facilitando a sua montagem.

Uma das técnicas utilizadas para determinar a seqüência de montagem é determinar a seqüência de desmontagem. Segundo Sousa (1998), duas suposições são feitas:

1. Desmontagem é um processo no qual cada peça pode ser retirada da estrutura, sem prejudicar a estrutura de submontagem;
2. A seqüência de montagem é o inverso da seqüência de desmontagem.

Segundo Sousa (1998) , uma seqüência eficiente é aquela que:

- Possui o menor número de passos;
- Evita o risco de danificar as peças;
- Evita posições instáveis ou inseguras para o produto, operários ou equipamentos durante a montagem.

Quanto mais otimizado for, melhor será a manufatura do produto, facilitando a montagem e redução de tempo.

2.3.8 Conceito de Poka-Yoke

Quando se fala de montagem podemos considerar também defeitos de montagem. Existe um conceito, geralmente utilizado conjuntamente com abordagens de montagem que é o *Poka-Yoke*, como pode ser visto a seguir.

As estratégias adotadas para “zero defeito” segundo Shingo (1986) são:

a. Não:

- Produzir produtos que não sejam necessários;
- Movimentar produtos desnecessariamente;
- Manter estoques desnecessários.

b. Produzir detectando erros ou possíveis anormalidades através da(o):

- Padronização;
- Auto-controle;

- CEP (Controle Estatístico de Processo);
- Fluxo de processo;
- *Poka-Yoke*.

O conceito de *Poka-Yoke* já existe há muitos anos e foi inicialmente desenvolvido pelo Engenheiro Japonês Shigeo Shingo e serviu como uma ferramenta extremamente útil para atingir zero defeito. Em japonês *Yokeru* significa evitar e *Poka* significa erros, ou seja, dispositivos que evitam erros ou mais conhecido como "método à prova de erros".

A frase "à prova de erros" significa impossibilitar que peças, ferramentas, equipamentos e montagem (DFA), possam ser usadas de forma imprópria.

Muitos erros surgem em um ambiente de produção todos os dias, e quando defeitos são produzidos denota-se desperdício. Quando não é descoberta a causa, não é possível atender os requisitos dos clientes mais exigentes, diminuindo a qualidade do produto. Assim a finalidade do *Poka-Yoke* é detectar e prevenir erros no processo, afim de que defeitos possam ser eliminados na fonte, assegurando a qualidade já na estação de trabalho. Logo, detectando e prevenindo erros, conseqüentemente se elimina desperdício, reduzindo por fim, custos de produção.

Antes de mostrar a idéia de conceito de *Poka-Yoke* é necessário entender quais são os erros existentes em uma planta fabril. A maioria dos erros é cometida pelos seres humanos. Existem ao menos dez tipos como apresentado abaixo (SHINGO, 1988):

1. Desatenção do operador no trabalho esquecendo de montar uma peça. Medidas preventivas: concientizar o operador em checar os processos em intervalos pré-determinados;
2. Falta de compreensão do processo. Medidas preventivas: treinamento especializado e atenção no trabalho;
3. Erros de identificação. Medidas preventivas: treinamento e atenção no trabalho
4. Falhas por motivo de trabalho lento e/ou desacelerado. Medidas preventivas: trabalho padronizado e treinamento especializado;
5. Erros por falta de padronização. Medidas preventivas: trabalho padronizado e instruções de trabalho;

6. Falhas surpresas, como falhas por falta de calibração. Medidas preventivas: trabalho padronizado e manutenção de equipamentos;
7. Erros intencionais. Medidas preventivas: educação e disciplina.

As falhas ocorrem por várias razões, mas na sua maioria podem ser evitadas identificando-se a causa do problema e tomando providências para prevení-las usando métodos *Poka-Yoke* e medidas preventivas como mostradas na lista anterior.

Existem oito princípios básicos para implementação de *Poka-Yoke*, segundo Shingo (1988), são eles:

1. Qualidade no processo: faça com que os processos sejam impossíveis de serem montados errados;
2. Erros e defeitos podem ser eliminados: deve-se assumir que erros podem ser evitados. Sempre existem métodos para eliminá-los;
3. Pare de fazer errado e comece a fazer corretamente: elimine todo os "porém", aquela frase que diz: Nós sabemos que não é certo, porém...;
4. Não pense em desculpas, pense em como fazer direito: ao invés de pensar em desculpas, pense em como realizar processos de maneira certa;
5. Você tem 60% de chances de sucesso – implemente sua idéia agora: analise a causa e pense na solução. Se a solução tem 60% de chance de sucesso, implemente. Mudanças e detalhes na solução podem ser trabalhadas depois da implementação da idéia;
6. Erros e falhas podem ser reduzidos a zero quando todos trabalharem para eliminá-los: quando um projeto almeja zero de defeitos e falhas, não pode ser atingido por uma única pessoa. É importante que todos os funcionários da empresa trabalhem juntos para eliminá-los;
7. Dez pessoas pensando é melhor que uma: o *brainstorm* de cada indivíduo é importante, mas a sabedoria e a criatividade que surgem entre mais pessoas são valiosas. Portanto, o trabalho em time é a chave da eficiência das melhores idéias;
8. Procure a causa do problema usando os 5 P's e um C: quando um defeito ocorre, deve-se estudar a raiz da causa perguntando: Porque ocorreu o defeito? e através da resposta, pergunte novamente "Por quê". Pergunte "Por quê" pelo menos cinco vezes para descobrir o motivo do defeito ou falha. Assim, por final, pergunte "como deverá ser consertado?" e implemente a solução em prática.

Existem diversas formas de evitar erros. Podem-se classificar dispositivos a prova de erros em:

- Instrumentos de detecção: são ferramentas (sensores) que detectam alguma anomalia ou falha no processo e pára a linha de montagem;
- Ferramentas restritivas: asseguram a presença e a correta posição das peças para permitir que o processo inicie;
- Dispositivos de sinalização: alertam o operador que uma anormalidade ocorreu, capacitando-o a tomar as medidas corretivas necessárias para restabelecer as condições normais de funcionamento;
- Dispositivos para seleção de modelos: identificam imediatamente o modelo que será processado e ajusta os parâmetros do equipamento que executará a operação;
- Indicadores visuais: podem, através de código de cores, alertar visualmente, otimizando a capacidade do operador de reconhecer ou instalar uma determinada peça ou ferramenta; ou via instruções na tela, geralmente relacionada aos procedimentos de testes, visa assegurar que todos os passos e a correta seqüência do teste serão executadas.

2.3.9 Casos de DFMA

Estes casos foram extraídos do site da empresa do Boothroyd e Dewhurst (BOOTHROYD DEWHURST, 2007) que é uma das maiores referências em DFMA e retrata vários casos de DFA.

Helicóptero Longbow Apache

Um estudo examina a efetividade da metodologia *Design for Manufacturing* e *Assembly* (DFMA) usadas pelo projeto, manufatura, qualidade e engenharias que suportam o desenvolvimento do Helicóptero *Longbow* Apache. Dados são obtidos através de times de Desenvolvimento de Produto Integrado (DPI) por muitas áreas de reprojetos do protótipo do Helicóptero *Longbow*. Resultados desse estudo mostram que através da aplicação do DFMA houve uma significativa redução de custo e peso.

QSTAR espectrômetro

Existiu um caso a poucos anos atrás quando surgiu uma oportunidade para desenvolver um espectrômetro de massa híbrido combinado com tecnologia de *time-of-flight*. O novo projeto alargou a escala de componentes que os laboratórios podem analisar e identificar – de nomear componentes simples pra distinguir diferenças entre moléculas tão extensas e complexas como o DNA.

QSTAR *mass spectrometers* é um instrumento produzido em baixa escala. O time de manufatura reconheceu que a facilidade de montagem influenciaria no custo, na manufatura eficiente e desempenho em campo do produto. Como originalmente construído, o produto era difícil de ser montado e necessitava de dias para testar e procurar pontos para melhorar o desempenho.

Reprojetando o QSTAR obteve-se menos componentes e, planejar componentes que são auto-encaixáveis e auto-localizadores, também foi uma importante prática de simplificação. Os engenheiros da QSTAR usaram o *software* DFA para analisar o projeto existente e para criar inovativos caminhos para consolidar peças e eliminar montagens dificultosas. Depois da aplicação do *software* DFA o time de projeto apresentou melhorias no produto e a versão do QSTAR que foi produzida conseguiu tolerâncias mecânicas sem requerer ajustamentos após a montagem.

Dell Computadores

Por volta de 1998, a Dell associou-se com outra grande empresa e obteve um crescimento industrial de 2,5 vezes.

Customizar a montagem de equipamentos de computadores requer operadores, tempo e espaço. Em fato, o reprojeto de um produto, como o chassi Optiframe® dos computadores pessoais, reduziu um custo estimado em 15 milhões de dólares. A redução de custo em material da integração do chassi e a cadeia de suprimentos relacionada otimizou um programa que era de 11,6 milhões de dólares em 1998 e a previsão de 35 milhões em 1999. De um simples reprojeto, reduziu-se expressivas quantias.

<u>As mudanças no projeto do Optiframe</u>

O principal objetivo do projeto foi o de focar no produto e no processo conseguindo resultados expressivos, como:

- Criação de comunalidade através da linha de produto. O chassi Optiframe, que era usado em todos os sete modelos da plataforma de computadores corporativos, passou também a serem usados, na época, nos modelos Optiplex da linha de computadores pessoais.
- Redução do custo em 17% e montagem mecânica em 25 %. Isso ofereceu a Dell realocar seus recursos em outros produtos e mesmo construir novas facilidades;
- Redução do tipo de parafuso em 67%;
- Fazer o produto ser mais serviço e reduzir o custo de manutenção em 25 %.

Whirlpool Sweden – forno microondas

Whirlpool Sweden localizada em *Norrköping* é o Centro de tecnologia global de forno de microondas da empresa. Sua manufatura crescia um milhão de fornos ao ano, na década de 90. A empresa produz algumas plataformas de fornos, com varias opções e vários modelos. O mercado primário é a Europa embora haja vendas em outros continentes.

A empresa produzia cinco mil fornos por dia. A montagem era exclusivamente manual.

O programa de treinamento em DFA da *Norrköpin*g foi iniciado com dois objetivos. O primeiro foi de ensinar duas equipes de oito pessoas como aplicar o DFA usando o *software* especializado. No time multifuncional incluíam-se engenheiros mecânicos e elétricos, técnicos em microondas, projetistas de sistemas de ventilação e pessoal diretamente envolvido com a produção e montagem.

2.4 MODELAGEM DE *SOFTWARE*

O próximo item (2.4.1) relata a importância de modelar um sistema antes de sua construção ou renovação. Para sua construção é necessária uma ferramenta de modelagem e, o item 2.4.2 descreve uma linguagem de modelagem gráfica para descrever o *software,* a UML (*Unified Modeling Language*). O item 2.4.3 descreve os conceitos dessa linguagem.

2.4.1 Contexto

Desenvolver um modelo para um sistema antes de sua construção ou renovação é essencial. Bons modelos são essenciais para comunicação entre equipes de projeto e para determinar possíveis desvios na sua arquitetura inicial. Como a complexibilidade dos sistemas aumentam, então se torna cada vez mais importante uma boa técnica de modelagem (FURLAN, 1998).

Segundo Fowler (2000) existe muitos fatores adicionais de sucessos de projetos de desenvolvimento de *software*, para isso uma rigorosa padronização de linguagem de modelagem é essencial.

Em se tratando de um modelo, sua construção pressupõe a especificação de uma linguagem de representação com sintaxe e semântica suficientemente ricas para comunicar toda a complexidade requerida pelo idealizador do modelo, e formal o bastante para evitar ambigüidades nesse processo de comunicação.

Como o ciclo de vida evidenciará padrões organizacionais envolvidos no processo de Desenvolvimento de *Software*, seria natural utilizar linguagens de representação nesse processo, como por exemplo, a UML (*Unified Modeling Language*).

2.4.2 Unified Modeling Language (UML)

A Linguagem Unificada de Modelagem é uma linguagem de modelagem gráfica para descrever um *software*. Ela simplifica o complexo processo de análise, projeto e construção de *software* criando visões do sistema que está sendo construído (RATIONAL, 2003).

Um método pressupõe um modelo de linguagem e um processo. O modelo de linguagem é a notação que o método usa para descrever o projeto. Os processos são os passos que devem ser seguidos para se construir o projeto.

A linguagem do modelo é uma parte muito importante do método que corresponde ao ponto principal da comunicação. A UML é uma linguagem padrão para visualizar, especificar, construir e documentar artefatos de um sistema baseado em *software*.

Os autores da UML preocuparam-se com a modelagem de sistemas concorrentes e distribuídos. Por isso os esforços são concentrados em um metamodelo comum, que unifica as semânticas, e em uma notação comum que fornece uma interpretação humana destas semânticas.

A UML reuniu vários recursos existentes em diversos métodos orientados a objetos. Uma síntese desses vários métodos de especificação para criação da UML é mostrado na Figura 2-16.

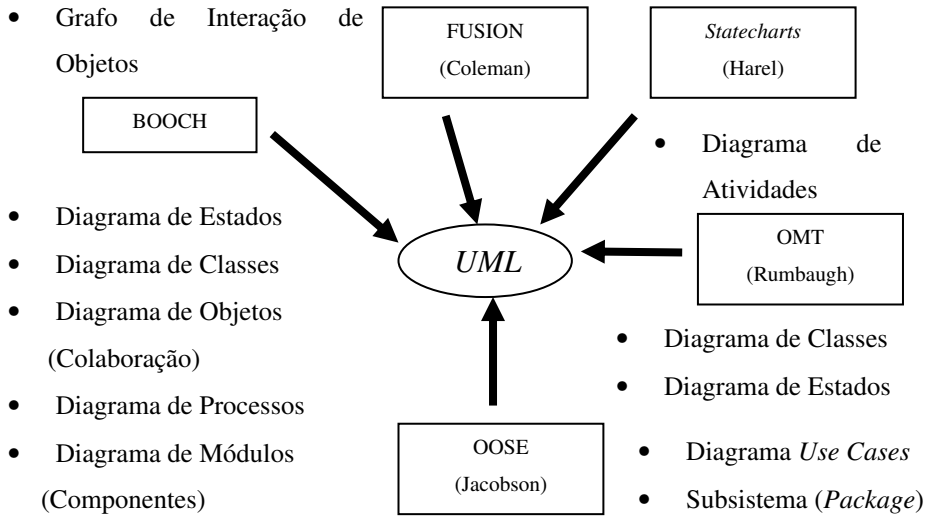


Figura 2-16 - Vários métodos de especificação para criação da UML (BOOCH, 2000)

Esse conceito aglutinou o que havia de melhor em vários métodos existentes, tendo recebido a colaboração dos principais metodologistas da época (FURLAN, 1998).

2.4.3 Conceitos da UML

A *Unified Modeling Language* (UML) é uma linguagem de modelagem não proprietária de terceira geração. A UML não é um método de desenvolvimento, o que significa que ela "não diz" o que fazer primeiro e em seguida ou como projetar seu sistema, mas ela auxilia a visualizar o desenho e a comunicação entre objetos.

Basicamente, a UML permite que desenvolvedores visualizem os produtos de seu trabalho em diagramas padronizados. Junto com uma notação gráfica, a UML também especifica significados, isto é, semântica. É uma notação independente de

processos, embora o RUP (*Rational Unified Process*) tenha sido especificamente desenvolvido utilizando a UML.

É importante distinguir entre um modelo UML e um diagrama (ou conjunto de diagramas) de UML. O último é uma representação gráfica da informação do primeiro, mas o primeiro pode existir independentemente.

A UML define uma notação e um metamodelo. As notações são todos os elementos de representação gráfica vistos no modelo (retângulo, setas, o texto, etc.), ou seja, é a sintaxe do modelo de linguagem. A notação do diagrama de classe, descrito a seguir, define a representação de itens e conceitos tais como: classe, associação e multiplicidade. Um metamodelo é um diagrama de classe que define de maneira mais rigorosa a notação.

Neste trabalho, será apresentada a notação UML através de alguns diagramas usados na especificação.

O Processo de Desenvolvimento de *Software* com UML está estruturado em quatro fases:

- Concepção: quando se especifica a visão do sistema;
- Elaboração: quando se faz o planejamento das atividades necessárias, dos recursos requeridos, a especificação do sistema e *design* da sua arquitetura;
- Construção: desenvolvimento do produto como uma série de interações incrementais;
- Transição: fornecimento do produto para o usuário (fabricação, distribuição e treinamento)

Em cada uma dessas fases, diferentes artefatos são produzidos usando diferentes atividades e técnicas. Compõe-se de 9 diagramas:

- **Diagrama de Classe -** mostra as classes que compõem o sistema e as relações entre elas (por exemplo, a herança). Trata de um aspecto estático e estrutural do sistema;
- **Diagrama de Objeto:** mostra alguns objetos (instâncias das classes) e as relações entre eles (qual objeto cria ou usam quais outros). É muito parecido ao primeiro (objetos são instâncias das classes). As classes definem a estrutura abstrata do sistema, os objetos já supõem que um programa está sendo executado e que instâncias são criadas para enviar e receber mensagens;

- **Diagrama de Caso de uso:** mostra como o sistema vai interagir com os usuários (pessoas ou outros sistemas);
- **Diagrama de Seqüência:** mostra interações entre vários componentes do sistema;
- **Diagrama de Colaboração:** mostra, também, interações entre vários componentes do sistema;
- **Diagrama de Atividade:** mostra como um subsistema ou um objeto realiza uma operação. Também conhecido como diagrama de fluxo de dados;
- **Diagrama de Estados:** (*Statechart diagram*) mostra como um subsistema ou um objeto realiza uma operação. Parecido ao precedente, mas concentra mais sobre eventos e ações que eles produzem;
- **Diagrama de Componentes:** mostra a organização dos componentes. Trata da implementação do sistema;
- **Diagrama de Distribuição:** (*Deployment diagram*) mostra a configuração física (*hardware*) sobre qual o sistema será instalado.

Estruturando ao longo do tempo das atividades dos componentes do processo, têm-se as fases normais do ciclo de vida do *software*, que, em geral, são:

- Análise de requisitos: levantamento de dados (descrição do que o sistema deve fazer);
- *Design*: como o sistema será construído na fase de implementação;
- Implementação: a produção do código que resultará em um sistema executável;
- Teste: a verificação de todo o sistema.

O item 2.5, demonstra conceitos de uma linguagem chamada de marcação que é usada para transferência de banco de dados na web: o XML.

2.5 XML (*EXTENSIBLE MARKUP LANGUAGE*)

Desde meados da última década, temos testemunhado um crescimento espantoso da *Web* e, aliado a ele, observado as limitações da linguagem HTML (*HyperText Markup Language*).

Em resposta à crescente demanda de extensões bem como da necessidade de interoperabilidade, houve um esforço inicial de estender a linguagem HTML

provendo-a com folhas de estilo em cascata, mais comumente conhecido como CSS (*Cascading Style Sheet*). Entretanto, essa solução constituiu-se apenas num paliativo. Como resultado, um esforço coordenado pelo W3C (*WWW Consortium*) foi realizado visando oferecer uma nova linguagem que pudesse satisfazer às necessidades de interoperabilidade, escalabilidade e flexibilidade, permitindo-se fácil extensão.

Surge, então, a linguagem XML (*Extensible Markup Language*) que é caracterizada por prover independência de dados bem como separar conteúdo de apresentação. Um programa em XML compreende a descrição de dados, tornando-se possível seu processamento por uma aplicação. XML tem sido cada vez mais, utilizada por desenvolvedores de aplicações devido ao suporte que ela oferece tanto a interoperabilidade quanto funcionalidade da *Web*. Trata-se de uma linguagem baseada em texto a qual permite qualquer pessoa escrever um código em XML, sendo ele, facilmente, tanto compreensível às pessoas quanto manipulável aos computadores (GOLDFARB & PRESCOD, 1999).

O suporte que a XML oferece a separação entre conteúdo e apresentação, é um aspecto de suma importância na linguagem. Considere, por exemplo, um repositório de dados. XML poderia ser usada para armazenar e exibir informações estáticas desse repositório.

No entanto, adicionalmente, ainda é permitido manipular as informações, ou seja, extrair, filtrar, buscar e ordenar informações. Essas características, juntamente com suporte à interoperabilidade, constituem fatores determinantes na escolha de tecnologia para desenvolver aplicações orientadas para *Web*. Além disso, o 'casamento' da linguagem XML com Java é aspecto chave para o desenvolvimento de aplicações orientadas para *Web*.

Torna-se imperativo ainda acrescentar que XML é uma linguagem simples, possui estrutura de dados rica, permite a troca e exibição de conteúdo de bases de dados e pode ser utilizada como formato para troca de mensagens na comunicação entre aplicações. Dentre essas, a troca de mensagens na comunicação entre aplicações de empresas oferece um meio de comunicação de baixo custo para aplicações B2B (*Business to Business*) e esse é uma das áreas que também pode tirar proveito da tecnologia XML contanto que protocolos seguros (com recursos de criptografia) sejam usados para assegurar as comunicações (GOLDFARB, 1998).

Segundo Marchal (2000), quando se inicia o estudo de XML, nota-se que muitas pessoas começam a aprender a montar um XML "bem-formado" (*well formed*), a aprender a utilizar algumas das APIs existentes para manipulação de XMLs como DOM e SAX. Também começa a aprender que é possível validar o XML com DTDs e/ou *schemas*, a tomar conhecimento do XSL que permite formatar o XML de diferentes formas, mas sempre aparece a clássica pergunta: mas para que exatamente serve o XML? ou onde posso utilizar XML?

Oliveira (2001) diz que uma das utilizações do XML é facilitar a troca de dados entre aplicações. Mas já não existem outras formas de se fazer isto? Já não existem vários formatos de arquivos que se destinam a isto como .ini, .dbf e outros arquivos de dados? E as técnicas para interligar aplicações de forma que elas possam trocar dados como RPC ou RMI? Por que inventar o XML se já existiam diversas formas possíveis para se realizar a comunicação entre sistemas? A resposta está na própria pergunta. Existem muitas formas possíveis. Ou seja, não existe um padrão único que possa ser facilmente utilizado por qualquer sistema em qualquer plataforma ou sistema operacional.

2.5.1 Principais Utilizações

De acordo com Connolly (1999) existem várias utilizações para o XML. Destacam-se algumas delas:

1. **Babel:** quanto maior é uma empresa, maior é a quantidade de dados que ela possui e com a qual tem que lidar. Inevitavelmente os dados de uma empresa são originados de diversas fontes em vários sistemas diferentes. A forma de armazenagem de toda esta informação varia mais ainda, podendo estar em arquivos texto, planilhas eletrônicas, bancos de dados, e-mails, etc. Com a informação espalhada em diversos pontos, invariavelmente surge a necessidade de que dois sistemas diferentes troquem informações entre si para gerar algum benefício maior. E quando os dois sistemas não falam a mesma língua? Estes dois sistemas podem estar em computadores, sistemas operacionais, redes, e em linguagens diferentes. Como então passar o dado que um possui para o outro de forma que o destinatário compreenda o que lhe foi enviado? Atualmente, existem várias formas de se fazer

isto. Poderíamos escrever o dado a ser passado em um banco de dados e a outra aplicação leria diretamente disto. Mas com isto teríamos que liberar acesso ao banco para a outra aplicação (talvez isto não seja conveniente) e também poderíamos ter a situação complicada se a outra aplicação não puder acessar o banco de dados tão facilmente por algum motivo (a linguagem poderia não apresentar *drivers* de comunicação eficientes com o banco). Outra solução seria escrever o dado em um arquivo texto, colocá-lo em algum local da rede que seja acessado por ambas as aplicações e fazer a aplicação de destino tratar o arquivo. Mas neste caso teríamos que escrever um código para analisar o arquivo texto, cuidando de contar os caracteres, identificar final de linha, e todos aqueles códigos que sempre aparecem em um *parser* (é um programa de computador, ou apenas um componente de um programa, que serve para analisar a estrutura gramatical de um *input*, manipulando os *tokens*, que são segmentos de texto ou símbolos que podem ser manipulados). Além disso, com arquivos texto é difícil expor relacionamento entre os dados. Existem outras soluções possíveis, mas o ponto é que sempre temos alguma complicação na implementação delas, quando o que deveria realmente nos preocupar é a implementação da lógica de negócio de aplicação e não questões de conexão, tratamento de arquivos, etc.

2. Múltiplas Visões: tradicionalmente a *Web* é composta por páginas formatadas com HTML e outros arquivos suportados que são transmitidos através do HTTP (*Hypertext Transfer Protocol*). Os navegadores enviam uma requisição para um servidor *Web* e este interpreta a requisição e responde à requisição. Como XML é muito parecido com o HTML ele também pode ser transportado através do HTTP. Há até algum tempo atrás, apenas computadores *desktop* conseguiam acessar a *internet*, mas atualmente um número cada vez maior de dispositivos já consegue fazer o mesmo. Devido a isto, pode-se precisar gerar diversas visões do mesmo dado para cada um dos dispositivos. Pode-se desenvolver sistemas *web* que ao identificar o dispositivo que fez a requisição formate o documento XML de formas diferentes. Uma das ferramentas para isto é o XSL (*eXtensible Stylesheet Language*) (ESPECIFICAÇÃO, 2006).

2.5.2 Estrutura de um XML

Foi comentado sobre o uso do XML, mas nada melhor do que ver com o que ele se parece. Basicamente um documento XML contém *tags* e texto. As *tags* definem elementos de dados e o texto fornece o dado real representado no documento (LIGTH, 1999). A sintaxe básica para um elemento XML é mostrada a seguir:

```
<nome_do_elemento>Texto</nome_do_elemento>
```

Podem-se colocar elementos dentro de outros de forma a representar informação relacionada. O exemplo a seguir mostra como agrupar o nome e endereço para um empregado no elemento.

```
<empregado>
  <nome>Maria Aparecida</nome>
  <endereço>Rua Terra, 13</endereço>
</empregado>
```

O ANEXO A têm maiores detalhes sobre a estrutura do XML.

Documentos XML geralmente consistem de uma série de elementos que se repetem. Podemos pensar em cada elemento como uma linha de uma tabela. Seguindo o exemplo anterior uma empresa que possua vários empregados poder-se-ia armazenar seus dados em um XML.

Elemento é o ponto central de um documento XML, mas eles não são os únicos itens de informação que estão presentes em um documento XML (RAY, 2001).

Os itens que compõem um XML são:

- Declaração XML;
- Instruções de processamento;
- Comentários;
- Elemento raiz;
- Elementos filhos;
- Elementos vazios;
- Atributos.

Instruções de Processamento

Instruções de processamento também podem aparecer em um documento XML. Se presentes, instruções de processamento começam com um sinal de menor e uma interrogação (<?) e terminam com uma interrogação seguida de um sinal de maior (?>), de forma semelhante à declaração XML. Entretanto, a declaração XML não é um exemplo de instrução de processamento.

Instruções de processamento fornecem informações adicionais ao *parser* XML para ajudá-lo a interpretar melhor o documento. Por exemplo, podemos utilizar uma instrução de processamento para aplicar uma folha de estilo que transforme a aparência do XML quando exibido em um navegador *Web* (PIMENTEL, 2001). Como no exemplo abaixo:

```
<?xml-stylesheet type="text/xsl" href="stylesheet.xsl"?>
```

Comentários

Comentários podem ser colocados em qualquer lugar dentro de um documento XML, exceto dentro de uma *tag*. Comentários são cercados por <-- e --> e podem abranger várias linhas. A restrição é que não podemos colocar -- dentro de um comentário. Comentários aninhados também não são permitidos.

Elementos

Como dito anteriormente, elementos são os itens básicos de um XML. São de dois tipos: aqueles que possuem conteúdo e aqueles que são vazios. Quando o elemento possui conteúdo, o texto deve estar entre a *tag* de abertura e *tag* de fechamento, como a seguir:

```
<dado>texto do dado</dado>
```

Elementos vazios são aqueles que não possuem texto e não possuem elementos filhos dentro dele. Se você precisa de um elemento que não precisa de nenhum texto ou filho associado, uma forma de representá-lo é:

```
<elementovazio></elementovazio>
```

Uma abreviação seria utilizar

<elementovazio/>

A barra antes do fechamento da *tag* indica que a *tag* representa ao mesmo tempo o começo e o fim do elemento.

Elementos Filhos

Dados em um documento XML são hierárquicos. Elementos podem conter outros elementos que são chamados de elementos filhos. No exemplo a seguir o elemento <nome> é um elemento filho do elemento <empregado>:

```
<empregado>
  <nome>Maria Aparecida</nome>
</empregado>
```

Atributos

Você pode ainda utilizar atributos para fornecer informações adicionais sobre um elemento. Os valores dos atributos devem ser sempre cercados por aspas (simples ou duplas). Alguns exemplos:

1) A informação do tipo do telefone é armazenada no atributo tipo do elemento telefone:

<telefone tipo="celular">95644545</telefone>

2) Se o valor do atributo contiver aspas duplas utilize aspas simples:

<pessoa nome='Jonas "Cenoura" da Silva'/>

3) Se o valor do atributo contiver aspas simples utilize aspas duplas:

<pessoa nome= “Jonas 'Cenoura' da Silva” />

Vejamos os dois exemplos a seguir:

Primeiro exemplo:

```
<empregado sexo= “masculino” >
  <nome>Fernando</nome>
</empregado>
```

Segundo exemplo:

```
<empregado>
  <sexo>masculino</sexo>
```

```
  <nome>Fernando</nome>
</empregado>
```

No primeiro exemplo sexo é um atributo do elemento empregado e no segundo exemplo é um elemento filho. Não existe uma regra de quando usar atributos e quando usar elementos filhos. Recomenda-se utilizar atributos apenas informações que não são relevantes para os dados.

Alguns problemas aos se utilizar atributos são:

- Atributos não podem conter múltiplos valores (elementos podem);
- Atributos não podem descrever estruturas (elementos podem);
- Atributos são mais difíceis de manipular via código.

2.5.3 XML formado corretamente

A especificação do XML disponibilizada pelo *W3 Consortium* define que a estrutura básica de um XML bem formado consiste de um prólogo e de um elemento raiz (SPENCER, 1999).

O prólogo contém metadados a respeito do resto do documento e é composto da declaração XML, das instruções de processamento e das definições de DTD ou XML *Schemas*.

Os dados de um documento XML consistem de um único elemento chamado elemento raiz. O elemento raiz contém todos os outros elementos e atributos no documento.

Quando criamos um documento XML devemos ter o cuidado de seguir as restrições sintáticas e de conteúdo. Todo documento XML deve ser formado corretamente e válido.

Para ser carregado por um *parser* o documento XML deve ser bem formado. Para isto ele deve seguir as seguintes regras (CARLSON, 2002):

- **Um único elemento raíz:** só pode haver um único elemento raiz no documento XML. Todos os outros elementos devem ser definidos dentro do documento raiz.
- **Abrir e fechar todas as *tags*:** O XML requer que todas as *tags* abertas devem ser fechadas. Se você quiser um elemento vazio pode utilizar a sintaxe <elemento/>.

- **Capitalização consistente:** XML distingue letras maiúsculas de minúsculas. Por exemplo, para uma *tag* aberta com <empregado> não pode ser finalizada com </Empregado>.
- **Elementos aninhados corretamente:** os elementos devem ser cercados completamente por seus elementos pais.
- **Valores de elementos envoltos por aspas:** o valor de um elemento deve estar entre aspas (simples ou duplas).
- **Sem atributos repetidos em um elemento:** se for necessário representar dados repetidos, como múltiplos empregados, deve utilizar elementos e não atributos, porque todos os atributos dentro de um elemento devem ter nomes únicos.

Para ser válido um documento XML é aquele que satisfaz um DTD ou XML *Schema*, quando fornecidos.

3 METODOLOGIA

Neste capítulo é descrito o método empregado. Inicia-se com a exposição do problema de pesquisa, e a discussão sobre sua relevância e justificativa. Na seqüência são apresentados os objetivos a serem atingidos. São apresentados também os aspectos referentes à escolha do método e instrumentos de pesquisa e, apresentam-se as etapas do trabalho.

3.1 JUSTIFICATIVA DO TRABALHO

Este trabalho tem como parcela de sua justificativa o rol de argumentos que será apresentado no item 3.2, no momento que foi exposto o problema de pesquisa. Conforme será apresentado, ele contribui com conhecimentos e um modelo que permitem a construção de soluções para gerenciar os conhecimentos explícitos em abordagens de DFA.

Este trabalho pode contribuir com o desenvolvimento de soluções que venham facilitar o trabalho multidisciplinar e a troca de conhecimentos entre pesquisadores e profissionais de empresas que atuam nesta área.

3.2 EXPOSIÇÃO DO PROBLEMA DE PESQUISA

A relevância do DFA é patente, conforme mencionado no item 2.3. E é na literatura internacional que se destacam algumas publicações, como pode ser visto em Boothroyd (1988), Boothroyd & Alting (1992) e Boothroyd, Dewhurst e Knigth (1994).

Nesse contexto, este trabalho teve como fonte motivadora, o problema prático de pesquisa, identificado durante o estudo de abordagens de DFA, ocasionado na dificuldade de se encontrar uma informação disponibilizada em várias formas de

armazenamento. Deste modo, para apresentar de maneira salutar e precisa sua importância e fundamentação são imprescindíveis a definição do problema e o objetivo que são delineados no contexto metodológico.

Segundo Kerlinger (1980, p. 321) "a pesquisa é dirigida para a solução de problemas práticos e especificada em áreas delineadas e da qual se espera melhoria ou progresso de algum processo ou atividade ou o alcance de metas práticas" Nesse ínterim, o objetivo primeiro do projeto, por ser o mais genérico, é desenvolver um sistema que busca na rede mundial de computadores, a *internet,* abordagens DFA que complementem uma base de dados local.

Com a problemática acima definida, fez-se necessário buscar inferências para a mencionada demanda na literatura acerca da gestão do conhecimento. Em especial, procuraram-se ferramentas e tecnologias de informática que poderiam registrar os conhecimentos explícitos sobre abordagens de DFA.

De acordo com as evidências demonstradas na revisão bibliográfica, foram estabelecidas algumas tecnologias e ferramentas que podem ser empregadas para o incremento de alternativas para a gestão de conhecimentos. Essas tecnologias e ferramentas, desde linguagens de programações básicas até ferramentas sofisticadas que incluem suporte para trabalho em grupo e armazenamento de dados distribuídos, foram utilizadas porque simplesmente podem construir de maneira eficaz o protótipo do modelo proposto. Embora a diversidade tenha sido extensa, foram identificados, por meio de subsídios literários, vários problemas pautados nestas tecnologias.

Cervo e Bervian (1983) destacam a relevância das referências teóricas publicadas em documentos. Com isso, há que se ressaltar a inicial ausência de unicidade entre os conhecimentos explícitos, descritos de variadas formas, atributo relevante no que diz respeito à composição do conhecimento armazenado. Podem-se considerar casos de DFA como um dos formatos de conhecimentos explícitos sobre essa abordagem, em que o diferencial incorre na percepção, conexão e amarração do todo e não apenas em partes desconexas. Ao examinar a abordagem de DFA durante a concretização de um projeto, seria salutar a possibilidade de consulta e, até mesmo de acordo com Oliveira (2000), proceder à investigação, de forma a explorar, diretamente do caso de DFA, utilizando-se de uma diversidade de fontes de

conhecimentos explícitos, tais como livros, artigos científicos, dissertações, teses, testes, formulários, opiniões, casos, mídia, entre outros.

Ressalta-se que as ferramentas observadas por Amaral (2002) e Carvalho (2000) descritas no item 2.1.3, foram identificadas como imprecisas acerca da sua função diante da gestão do conhecimento. Essencialmente, descreviam recursos e funcionalidades sem evidenciar a maneira de utilizá-las, seus potenciais e limitações quando inseridas em um programa de gerenciamento de conhecimentos em uma organização. Assim, ficam evidentes os percalços de se obter uma ferramenta que formalize os conhecimentos explícitos durante a concepção dos casos de DFA, em que a origem de um problema básico se mostrou muito relevante. O que consta é a ausência de conceitos teóricos e discussões que enfocam o assunto capaz de relacionar os recursos disponíveis nestas ferramentas com o efetivo esforço de gestão do conhecimento e aprendizagem organizacional em si; mais precisamente, existia uma lacuna entre pesquisas e teorias que versam acerca de ferramentas computacionais e entre outras que tratam com as ações ligadas à gestão do conhecimento, incorrendo na ausência de conceitos e modelos atrelando estas duas áreas.

Por conseguinte, a ausência de conhecimento em uma forma satisfatória que norteasse os profissionais no emprego e aplicação das tecnologias e funcionalidades das ferramentas comerciais existentes, no fomento de soluções para o registro de conhecimentos explícitos podem ser soluções aptas a proporcionar o avanço da gestão do conhecimento e, mais especificamente, nos casos de DFA. Segundo Booth, Colomb e Willians (1995), este problema poderia ser caracterizado como um problema prático, que poderíamos formular com a seguinte questão: ***Como gerar, a partir de tecnologias e ferramentas de mercado, um sistema para gerenciamento de conhecimentos explícitos de maneira integrada a casos de DFA?***

Nesse sentido, com a definição acima da questão de pesquisa do trabalho, que motivou a proposta de pesquisa, mais especificamente, no *Problema de Pesquisa*: ***Propor um modelo conceitual que auxilie a criação de soluções para o gerenciamento de conhecimentos explícitos sobre DFA***.

O problema de pesquisa é apontado por evidenciar uma lacuna no conhecimento, e em razão disso, incorre em um espaço a ser preenchido na

compreensão de um assunto (BOOTH, COLOMB & WILLIANS, 1995; KÖCHE, 2002 e MÁTTAR NETO, 2002). Tal lacuna é justamente o potencial da aplicação de ferramentas de Gestão do Conhecimento na solução do problema de apresentação do conhecimento explícito sobre abordagens de DFA. A existência desta lacuna impede a construção de forma eficiente e eficaz de ferramentas que auxiliem o registro e análise do conhecimento explícito sobre abordagens de DFA.

3.3 OBJETIVOS DO TRABALHO

Complementando o que foi colocado como objetivo do trabalho de pesquisa, no capítulo 1, pode-se dizer mais especificamente que o objetivo principal deste trabalho é:

1. Propor um modelo teórico que especifica como desenvolver sistemas para gestão do conhecimento explícito para abordagens de DFA.

Este é o objetivo central do trabalho que, uma vez atingido, implicaria na demonstração de um importante caminho a ser seguido nas pesquisas sobre ferramentas para gestão do conhecimento. Mais do que pesquisar novas ferramentas levando em consideração apenas à tecnologia de informação, dever-se-ia compreender melhor quais as funcionalidades existentes, quais são as mais importantes para a gestão dos conhecimentos explícitos e como elas poderiam ser reunidas de forma a auxiliar efetivamente estes gerenciamentos. Responder estas questões é fundamental para os profissionais que enfrentam o problema de avaliar, escolher e implementar sistemas de informação.

Para atingir este objetivo optou-se por realizar uma pesquisa complexa. Adotou-se o caminho de propor um modelo, empregá-lo como base para o desenvolvimento de uma solução específica e, por fim, avaliá-lo a partir da aplicação desta solução.

Desta forma poder-se-ia analisar com maiores detalhes os problemas, vantagens e as limitações de se trabalhar com arquiteturas para o desenvolvimento de sistemas para o gerenciamento de conhecimentos explícitos. Outra vantagem de se optar por este caminho é ter como subproduto um modelo e uma solução específicos que podem ser futuramente aprimorados e utilizados nas avaliações cada vez mais

profundas que venham demonstrar a real potencialidade desta proposta. Neste trabalho espera-se atingir o objetivo mais modesto, mas não menos complicado, de verificar a viabilidade da criação de um modelo deste tipo, dando forma a esta linha de pesquisa.

Uma vez definido este direcionamento, foram desenvolvidos os três objetivos secundários abaixo listados:

1.1 Propor um modelo que oriente o desenvolvimento de soluções de gerenciamento do conhecimento explícito sobre o DFA

1.2 Criar uma solução para gerenciamento de conhecimentos explícitos sobre o DFA baseado no modelo

1.3 Aplicar a solução em um caso exemplo

1.3.1 Escolher e implantar a solução no caso

1.3.2 Analisar os resultados da solução quanto ao gerenciamento de conhecimentos explícitos

1.3.3 Analisar a aplicação da solução quanto ao desenvolvimento de abordagens de DFA

3.4 MÉTODO DO TRABALHO

Para a realização da pesquisa definiu-se quatro grandes etapas, todas se guiando por um mesmo referencial teórico, abordagem de pesquisa e método de pesquisa conforme apresentado na Figura 3-1. Para cada uma destas etapas foram definidos instrumentos a serem aplicados. O conhecimento gerado é, portanto, resultante da soma das análises e informações coletadas em cada uma destas etapas, que se complementam e em determinados pontos se sobrepõem, permitindo uma avaliação mais precisa do objeto de estudo.

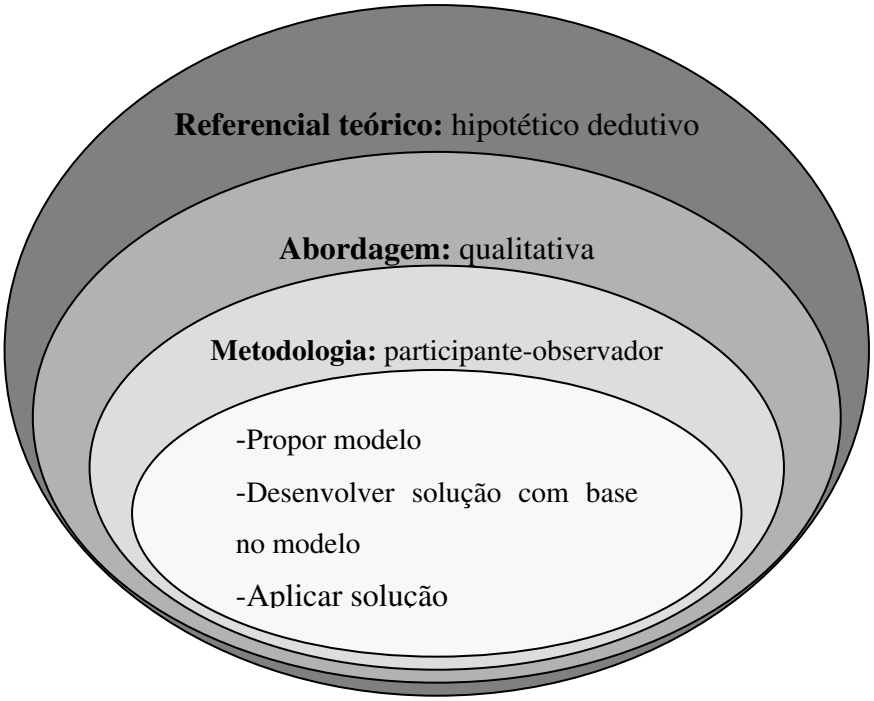


Figura 3-1 - Referencial Teórico e Métodos

Cada um destes elementos será apresentado a seguir junto com a justificativa de suas escolhas, implicações, vantagens e limitações de sua adoção.

3.4.1 Referencial Teórico

Denomina-se aqui como referencial teórico a categoria de métodos de pesquisa mais amplos, ou seja, aqueles diretamente relacionados com as correntes que estudam métodos de pesquisa dentro da área de epistemologia (conforme Pádua, 1996). Nas pesquisas em áreas que abordam problemas sociais e de organizações, como a deste trabalho, é fundamental que se identifique e avalie, dentro do contexto da metodologia, o referencial teórico que é assumido. Conforme o referencial adotado, diferentes concepções da realidade, da personalidade e da interação homem-mundo são assumidas, os quais, por sua vez, terão implicações diretas nas análises e resultados obtidos. Diferentes referenciais limitam ou expandem a compreensão do mundo segundo um conjunto específico de aspectos (PADUA, 1996).

Conforme a visão de mundo positivista deste referencial, a realidade é tida como objetiva e formada por fatos. O papel do pesquisador é identificar estes fatos, montando teorias capazes de controlá-los e predizê-los.

A escolha deste referencial para o trabalho foi motivada por sua prova ser mais robusta que os métodos indutivos e dedutivos (conforme a crítica feita por Karl Popper no início do século XX); e por, além disso, serem as variáveis envolvidas na pesquisa suficientemente concretas e circunscritas a um momento específico para dispensar métodos que considerassem a interpretação do pesquisador e o processo histórico (PÁDUA, 1996 e MARTINS, 1994).

3.4.2Abordagem da Pesquisa

Segundo Cervo & Bervian(2002) e Bryman(1989) pode-se classificar um trabalho científico, segundo a abordagem, definindo-se entre Pesquisa Qualitativa e Pesquisa Quantitativa. Na pesquisa quantitativa o pesquisador tem a oportunidade de definir claramente as variáveis e de medí-las. Já na abordagem qualitativa a ênfase deixa de ser a codificação e medição das variáveis e procura-se captar interpretações, discursos e estruturá-los de forma a construir o conhecimento (BRYMAN, 1989). Neste último tipo, as variáveis não são totalmente quantificáveis. Estas abordagens também não são excludentes podendo coexistir conjuntamente em uma mesma pesquisa.

Como o objeto desta pesquisa é bastante novo e as variáveis não são bem definidas, a abordagem predominante escolhida é a da pesquisa qualitativa. A análise será baseada principalmente nas discussões e análises do processo de definição do modelo conceitual, passando pela sua implantação na forma de uma solução e na avaliação de sua aplicação. É importante manter a predominância desta abordagem, mesmo na avaliação da aplicação (uma etapa que seria teoricamente mais quantificável), porque, dado o grau de inovação e por ser a primeira aplicação do modelo proposto, este método permitiria captar uma riqueza maior de detalhes e possíveis variáveis e problemas não previstos.

3.4.3 Método de Pesquisa

A realização do trabalho foi dividida em quatro etapas que, juntas, envolvem a criação de um modelo, seu uso para a construção de uma ferramenta de gestão de conhecimentos explícitos específica para um determinado caso e a avaliação desta ferramenta quanto sua aplicação neste caso. Esta opção reflete a adoção de um método que poderia, segundo a classificação de Dane (1990), ser identificado como o de pesquisa de campo do tipo participante-observador (*participant-as-observer*).

Segundo esta autora (DANE, 1990), pesquisa de campo é um rótulo que pode ser atribuído aos métodos de pesquisa que envolve a observação direta dos eventos conforme sua ocorrência natural. No tipo de pesquisa participante-observador mencionado, o pesquisador influencia e atua diretamente nas ações do fenômeno e todos os participantes têm conhecimento que se trata de um pesquisador. A este mesmo enfoque há autores que empregariam o rótulo de pesquisa-ação.

Um forte motivo para a escolha deste método que, desde o início mostrou-se mais promissor, foi a não existência de ferramentas ou modelos conceituais anteriores ao proposto em conformidade com as características explicitadas, impedindo, portanto, que o estudo fosse baseado em análises de casos ou outra técnica de observação direta.

Ao mesmo tempo, outros métodos, como optar por uma pesquisa teórica ou *surveys* e casos com análise profunda sobre os temas DFA e Gestão do Conhecimento, não pareciam suficientes para gerar os conhecimentos capazes de suportar a criação desta ferramenta especial de registro da gestão do conhecimento. Isto porque dificilmente seria possível observar, partindo-se destes métodos, com o mesmo nível de detalhes todos os problemas de ordem conceitual e tecnológica envolvidos com a criação de um modelo deste tipo que o enfrentamento do problema pelo próprio pesquisador, conforme o método adotado pode proporcionar.

Porém, não se devem desprezar as limitações do método escolhido. A maior delas é o fato de não permitir que sejam facilmente tecidas relações causais entre variáveis (DANE, 1990). Outra limitação importante é quanto aos resultados gerados que apresentam problemas de validade externa, isto é, os resultados encontrados nestes tipos de pesquisa, na grande maioria das vezes, são válidos somente para o

caso estudado e não há como reproduzir o experimento. As discussões não poderão refutar totalmente os problemas apresentados, pois, estará sendo discutida uma forma particular de modelo e aplicação desenvolvidas por indivíduos (o pesquisador e seu grupo de pesquisa) possuidores de uma bagagem e experiências únicas, não reprodutíveis e que deverão, devido a esta formação básica, enfocar mais ou menos determinados aspectos do problema durante a realização da proposta e sua avaliação.

Estas restrições, se analisadas, estão condizentes com os objetivos propostos no trabalho, o de viabilidade e funcionamento do modelo. Este método é importante também neste momento porque servirá como um passo inicial para que no futuro possa existir uma proposta de modelo e ferramentas a serem exploradas por métodos de pesquisa mais objetivos e generalizáveis.

3.5 ETAPAS DO TRABALHO

Cada uma das etapas para a realização do trabalho é descrita à seguir em maiores detalhes.

3.5.1 Etapa 1: Proposição do Modelo para Gerenciamento de Conhecimentos Explícitos sobre o DFA

Nesta etapa pretende-se responder ao objetivo 1.1, de propor o modelo. Nela o pesquisador desenvolverá o Modelo para Gerenciamento de Conhecimentos Explícitos sobre o DFA e analisará a viabilidade da proposta segundo a experiência obtida durante este trabalho de desenvolvimento.

3.5.2Etapa 2: Desenvolvimento de uma solução para gerenciamento de conhecimentos explícitos no DFA baseado no modelo

Esta etapa visa atingir o objetivo 1.2, criando uma ferramenta baseada no modelo. Para tanto, definir-se-á um grupo como caso. Os instrumentos de pesquisa aplicados são notas de campo e análise de documentos, dado que, as únicas fontes de informação são os pesquisadores envolvidos e os documentos gerados durante o trabalho de desenvolvimento.

O desenvolvimento da solução será realizado em quatro grandes atividades, envolvendo: um desenho geral da solução; a especificação e desenvolvimento de uma *intranet*; a preparação de modelos de documentos e diretórios na rede; e a análise dos resultados.

Enquanto na etapa anterior os resultados esclarecem sobre a possibilidade de obtenção do modelo em si, nesta etapa complementa-se esta demonstração verificando a capacidade de se empregar o modelo na construção de soluções.

4 MODELO DE SISTEMA PARA GERENCIAMENTO DE CONHECIMENTOS EXPLÍCITOS EM ABORDAGENS DE DFA (*DESIGN FOR ASSEMBLY)*

4.1 ESCOPO GERAL DO MODELO

Há atualmente um conjunto grande de informações a respeito de técnicas de DFA distribuídas em várias formas de armazenamento: ferramentas, livros, artigos, pessoas, etc. Sendo assim, cada uma dessas formas de armazenamento possui informações diferentes, pois existem: diferentes autores, boas e más práticas de aplicação, práticas que só se aplicam sob um determinado ambiente, ou seja, as informações estão descentralizadas o que causa dificuldade de acesso. Uma forma de organizar essa descentralização é por meio da realização de um conjunto de atividades dedicadas a garantir e incentivar a criação, registro e compartilhamento do conhecimento, a denominada Gestão do Conhecimento (GC).

A idéia básica de como trabalhar com o conhecimento a respeito de práticas de DFA, armazená-lo e reutilizá-lo é o enfoque deste trabalho. A GC fornece o que é o conhecimento e as formas básicas de armazenamento. O grande problema está na recuperação do conhecimento armazenado. Muitas organizações pouco sabem o conhecimento que tem, não porque não possuem uma política de gestão do conhecimento, mas sim porque não possuem regras para seu armazenamento. Sendo assim fica difícil a recuperação efetiva do conhecimento. Mas qual a forma de armazená-lo? Existe algum tipo de regra com o qual se possa armazenar conhecimento de práticas de DFA?

O Raciocínio Baseado em Casos (RBC) pode fornecer a resposta a essa pergunta. RBC é uma abordagem para a solução de problemas e para o aprendizado com base em experiência passada. De uma forma simplificada, podemos entender

RBC como a solução de novos problemas por meio da utilização de casos anteriores já conhecidos.

Por outro lado, esse projeto visa construir uma ferramenta de auxílio à gestão do conhecimento incentivando uma comunidade de prática (agrupamento de pessoas que compartilham interesses comuns e juntos buscam contribuir para criar novos conhecimentos em uma organização) que atue na geração, codificação e transferência do conhecimento sobre técnicas de DFA.

A forma tradicional de lidar com este problema tem sido o desenvolvimento de um portal corporativo do conhecimento (ferramenta que personaliza o acesso à informação, automatizando e aperfeiçoando ciclos complexos de decisões de trabalhadores de conhecimento e podem criar níveis mais profundos de colaboração entre os funcionários da mesma organização e de organizações diferentes).

Para construir uma ferramenta de compartilhamento de Casos DFA, seria necessário o agrupamento de várias organizações dividindo o custo da construção e manutenção do portal. Sendo que as organizações tendem a "proteger" informações consideradas por elas relevantes para o mercado, deve-se considerar um tipo diferente dos atuais sistemas centralizadores de dados.

Pensando nessas hipóteses, existe um tipo de sistema ainda não muito usado, denominado *peer-to-peer*. Essa classe de sistemas é bastante nova, e se baseia em um novo conceito de organização dos dados e será usada neste trabalho de pesquisa. Enquanto as soluções anteriores caminham para o paradigma *Web* com servidores centrais de dados, os sistemas *peer-to-peer* buscam a descentralização. O princípio fundamental é o de criar um sistema que permita a sincronização e compartilhamento dos documentos entre cada local espalhado na rede, eliminando a necessidade de um servidor de dados centralizado. No caso do projeto proposto, cada organização seria o local determinado, ou seja, cada organização pode hospedar seu portal na Empresa Provedora de *Internet* de sua escolha (ou por ela mesma) onde o gerenciamento das informações seria feito por cada uma dessas organizações, assim como o crescimento de outras funções do portal. A ferramenta proposta se uniria à estrutura *Web* da organização (Portal) sendo que a comunicação entre os portais pode ser feita por requisições XML (do inglês, *Extensible Markup Language*). XML é um padrão para troca de informações entre diversas plataformas, que apenas possibilita a descrição

de dados, em um arquivo de formato texto. Existem várias “linguagens” que fazem uso de aplicações que utilizam os recursos XML, o que torna o XML uma poderosa ferramenta para a publicação de informações.

Essa descentralização das informações oferece uma série de vantagens para cada usuário ou organização. Por exemplo, é uma estrutura de baixo custo e cada uma das organizações pode ou não permitir que outras “enxerguem” informações pontuais, estabelecendo assim uma relação de confiança ou mesmo de comércio de informações.

4.2 DESENVOLVIMENTO DO SISTEMA PROTÓTIPO

Este item mostra os resultados obtidos no desenvolvimento do sistema protótipo empregando o conceito proposto no item 4.1. Deste ponto em diante será demonstrado uma metodologia de desenvolvimento de *software* desenvolvida especificamente para apresentar o sistema em questão. Como exemplo de aplicação da metodologia sugerida será apresentado o próprio sistema proposto neste trabalho.

4.2.1 Metodologia de especificação

O item 4.2.1.1 descreve a origem da metodologia de desenvolvimento de *software* desenvolvida e o item 4.2.1.2 os passos e artefatos gerados. Em especial a estrutura de requisitos é apresentada no item 4.2.1.3.

4.2.1.1 Origens da metodologia de especificação

Para a especificação do sistema procurou-se empregar um processo de desenvolvimento de *software* estruturado, com o objetivo de garantir a qualidade da ferramenta e gerar ao final do trabalho uma documentação rigorosa e clara, permitindo a continuidade da pesquisa por outros pesquisadores.

A metodologia foi desenvolvida através de pesquisa na literatura. Foram estudadas algumas metodologias, mas de complexa e difícil utilização, que são usadas para especificação de sistemas de grande porte e para grandes equipes de desenvolvimento, como por exemplo, CONALLEN (2000).

Baseando-se nestas experiências, no conhecimento sobre programação *web* e na bibliografia sobre engenharia de *software* decidiu-se elaborar uma metodologia simples de desenvolvimento, contendo fases de desenvolvimento e um conjunto de documentos para a especificação. Essa metodologia proposta utiliza alguns artefatos produzidos pela linguagem UML (*Unified Modeling Language*).

Essas fases e a metodologia são descritas no item 4.2.1.2. Para o desenvolvimento da especificação adotou-se duas ferramentas: o RequisitePro e Rational Rose, ambos da empresa Rational, essas ferramentas são descritas também no item 4.2.1.2.

4.2.1.2 Descrição da metodologia de desenvolvimento de software

Existem vários modelos ou paradigmas que definem as fases para o desenvolvimento de *software* (discutidos no item 2.4). Analisando-os foi possível identificar 3 fases genéricas para a divisão do processo de desenvolvimento de *software*, são elas:

a) **Definição de atividades;**

b) **Desenvolvimento do sistema;**

c) **Manutenção do sistema.**

A fase manutenção não é de interesse para este trabalho, uma vez que a especificação foi projetada principalmente para descrever o sistema proposto.

a) **Fase de definição de atividades**

A fase de definição focaliza o "o quê" deve ser feito. Dentro dessa fase as seguintes atividades são desenvolvidas:

1. *Definição de requisitos*: busca-se desenvolver um método para obtenção e armazenamento de requisitos. Quanto à obtenção pode-se realizar pesquisas em ferramentas e em *sites* relacionados com sistemas de informação. Para armazenamento e posterior avaliação e alteração dos requisitos é necessário um método de compilação e avaliação contínua de requisitos “à luz” da análise de sistemas existentes. Portanto, é necessário utilizar um método de registro, análise e comparação de

sistemas baseados em requisitos. No sistema proposto foram utilizados dois métodos para armazenamento de requisitos: empregando o Excel da Microsoft e outro usando o RequisitePro da empresa Rational. Nesta atividade também foram gerados os casos de uso do sistema representados pelos diagramas de casos de uso e também o modelo de perfil do usuário, que demonstra os tipos de usuários existentes no sistema. Em relação aos requisitos, também foram gerados alguns novos com base no conceito proposto.

2. *Definição do escopo*: essa atividade descreve as limitações do sistema quanto a sua construção. Essa delimitação é realizada em aspectos relacionados com a tecnologia empregada (incluindo *software* e *hardware*), as funcionalidades existentes, o programador e o tempo de desenvolvimento. No sistema proposto, a tecnologia empregada foi definida de acordo com o conhecimento do programador.
3. *Definição da estrutura de banco de dados*: essa atividade descreve a estrutura de dados utilizada. No sistema proposto, na confecção desse documento foi utilizado o conceito de MER (Modelo Entidade Relacionamento). A princípio foi necessária a definição de entidades para construção do MER, ou seja, através do levantamento de informações sobre cada área na revisão bibliográfica, incluindo funcionalidades, podem-se dimensionar quais entidades seriam necessárias para a representação do MER do sistema.
4. *Definição da estrutura de telas do sistema*: nessa atividade é confeccionado um desenho da estrutura de telas do sistema, ou seja, um mapa mostrando as páginas *web* e seus relacionamentos. Essa descrição é possível realizar sem a construção de protótipos de telas, pois é considerado apenas o relacionamento entre as páginas. É possível prever a quantidade de páginas que o sistema obterá e quais suas funções. Esse tipo de atividade dá uma visão geral do sistema, pois é possível "enxergar" a interdependência entre as páginas *web* possibilitando ao programador estipular os parâmetros necessários nessa transição. No sistema proposto, a confecção desse desenho foi realizada com a

ferramenta VISIO que, possui ícones para a construção desse tipo de desenho.

b) Fase de desenvolvimento de atividades

A fase de desenvolvimento focaliza o "como" deve ser feito. Essa fase retrata como foi realizada a codificação do sistema. As atividades desenvolvidas foram:

1. *Definição de convenções de programação*: essa atividade descreve convenções, ou seja, regras para a codificação do sistema. Essas regras podem ser descritas com base na literatura que descreve regras para dar nomes a arquivos, nome de tabelas de banco de dados, nome de variáveis, etc. Esse tipo de atividade é imprescindível para o trabalho em equipe. Pode-se evitar uma série de problemas na posterior manutenção do sistema, pois fica fácil localizar arquivos, tabelas e variáveis.
2. *Definição das ferramentas de programação e de banco de dados*: essa atividade mostra quais plataformas e tecnologias de programação podem ser utilizadas na confecção do sistema.
3. *Descrição do sistema*: essa atividade descreve o sistema na forma de protótipo de páginas. São as páginas do protótipo, juntamente com a descrição de cada particularidade. Inicialmente podem ser construídas em PowerPoint, para se ter uma idéia do layout e depois em *HTML* estático para visualizar o trabalho final. *HTML* estático nada mais é do que a construção de páginas de *internet* sem *links* e sem acesso a banco de dados.

O item 4.2.1.3 apresenta, especificamente, os artefatos produzidos para definição de requisitos.

4.2.1.3 Descrição dos artefatos para a definição de requisitos

Artefato é o produto de uma ou mais atividades dentro do contexto do desenvolvimento de um software ou sistema. A gerência de requisitos assemelha-se ao gerenciamento de configuração em uma de suas principais funções: identificação dos itens de configuração (requisitos). Uma importante função adicional é manter a rastreabilidade entre os requisitos. Todo requisito deve ser unicamente identificado.

Além de permitir distinguir cada item, o identificador é o instrumento utilizado para associar o requisito aos artefatos produzidos em etapas posteriores no processo de engenharia de *software*. A utilização de ferramentas para gerenciamento de requisitos permite controlar os custos e utilizar eficientemente a estrutura de dados disponível. Trata-se de um campo fértil de pesquisa em Engenharia de Requisitos (REIFER, 2000) (RAMESH & JARQUE, 2001) (WANG & LAI, 2001) (CLELAND-HUANG & ZEMONT & LUKASIK, 2004) (DAG et al., 2004) (HAYES et al., 2004) (HOFFMAN et al., 2004) (LOCONSOLE, 2004) (DAG et al., 2005) e com diversas soluções para aquisição, como: RequisitePro (IBM RATIONAL, 2007) e DOORS (TELELOGIC, 2007). Infelizmente não se encontram disponíveis ferramentas livres (gratuitas) para gerência de requisitos até o presente momento.

Um processo de engenharia de requisitos deve escolher o conjunto mais apropriado de técnicas, combinando os artefatos gerados de maneira a otimizar o resultado final. A seguir, são listadas as técnicas, encontradas na literatura, que oferecem suporte às respectivas atividades da Engenharia de Requisitos:

- **Elicitação**
    - Facilitadores de comunicação (LEITE & FRANCO, 1993)(DAMIAN et al., 2000);
    - Pontos de vista (SOMMERVILLE & SAWER, 1997);
- **Análise**
    - Metas (MYLOPOULOS & CHUNG & YU, 1999)(LAMSWEERDE, 2001);
    - Casos de uso (JACOBSON et al., 1992);
    - Casos de mau uso (ALEXANDER, 2003);
    - Cenários (RIDAO & DOORN & LEITE, 2000)(BREITMAN & LEITE, 2002);
    - Pontos de vista (LEITE, 1989)(FINKELSTEIN & KRAMER & GOEDICKE, 1990)(FICKAS & LAMSWEERDE & DARDENNE, 1991)(NARAYANASWAMY & GOLDMAN, 1992)(EASTER-BROOK

& NUSEIBEH, 1996)(KOTONYA & SOMMERVILLE, 1995)(SOMMERVILLE, 1996)(SOMMERVILLE & SAWYER, 1997);

- Padrões (RIDAO & DOORN & LEITE, 2000) (TORO et al., 2000)(HAGGE & LAPE, 2005).

- **Especificação**
  - Modelos de referência (GUNTER et al., 2000);
  - Métodos formais (WIERINGA, 1995);
- **Validação**
  - Inspeção (SHULL & RUS & BASILI, 2000);
  - Pontos de vista (LEITE, 1989)(LEITE & FREEMAN, 1991);
  - Revisão;
  - Verificação automática (DURÁN & RUIZ & TORO, 2001);
- **Atividades de apoio**
  - Gerenciamento de Requisitos (RAMESH & JARKE, 2001)(ZYLBERMANN & COHEN & GOLDIN, 2003)(DAG et al., 2004)(HAYES et al., 2004)(HOFFMAN et al., 2004)(CLELAND-HUANG & ZEMONT & LUKASIK, 2004)(DAG et al., 2005)(LOCONSOLE, 2004);
  - Classificação e priorização de requisitos (ZANLORENCI & BURNETT, 2000).

A técnica utilizada por este trabalho foi a de análise por Casos de Uso (*Use Cases*, em inglês) que é um tipo de classificador representando uma unidade funcional coerente provida pelo sistema, subsistema, ou classe manifestada por seqüências de mensagens intercambiáveis entre os sistemas e um ou mais atores. Pode ser representado por uma elipse contendo, internamente, o nome do caso de uso. Um Caso de Uso representa uma unidade discreta da interação entre um usuário (humano ou máquina) e o sistema. Um Caso de Uso é uma unidade de um trabalho significante. Por exemplo: o "*login* para o sistema", "registrar usuário" e "criar ficha

catalográfica" são todos Casos de Uso. Cada Caso de Uso tem uma descrição o qual descreve a funcionalidade que irá ser construída no sistema proposto. Um Caso de Uso pode "incluir" outra funcionalidade de Caso de Uso ou "estender" outro Caso de Uso com seu próprio comportamento. A seguir é descrito cada um dos artefatos:

Diagrama de casos de uso (*use cases*): esse artefato foi produzido através de possíveis situações de uso que possam ocorrer no sistema. Os Casos de Uso foram selecionados de modo que, demonstrasse os requisitos apresentados. Existe um relacionamento entre os Diagramas de Casos de Uso (confeccionados com a ferramenta Rational Rose) e a tabela de requisitos (confeccionados com a ferramenta RequisitePro) onde cada caso de uso do Diagrama de Caso de Uso está relacionado com um conjunto de requisitos.

Lista de requisitos: esse artefato foi gerado através do resultado da pesquisa descrita no item 4.2.1.2 (em definição de requisitos). Para demonstração e controle de alteração dos requisitos foi utilizado a ferramenta RequisitePro.

Glossário: lista de palavras e/ou termos utilizados na descrição dos requisitos. Esse artefato funciona como um dicionário traduzindo palavras e/ou termos utilizados durante a confecção dos artefatos.

Fonte: descreve o local onde foi encontrada uma palavra e/ou termo ou onde foi encontrado o próprio requisito (ex.: ferramenta, site, livro, artigo, etc.).

Relacionamento: faz o relacionamento, ou seja, a ligação entre os outros artefatos. Por exemplo, é possível saber através do relacionamento, que um requisito é normalmente encontrado em um tipo de ferramenta, seu significado, sua fonte, em quais casos de uso aparece. A Figura 4-1 apresenta os artefatos descritos.

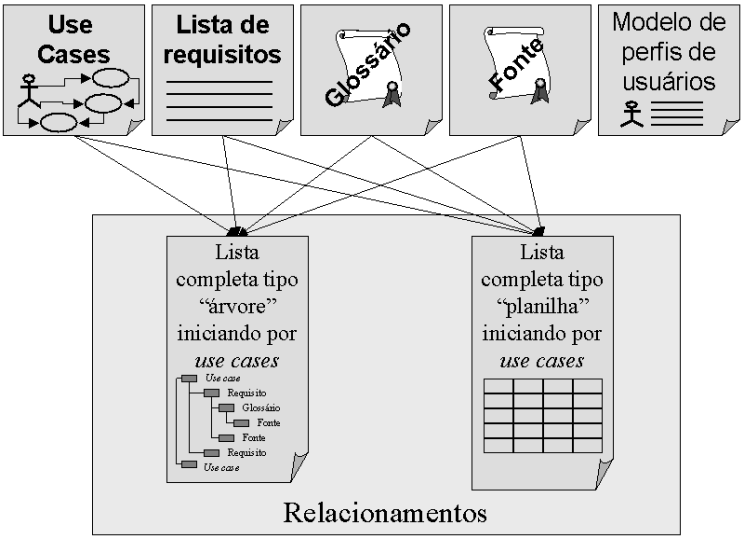


Figura 4-1 Artefatos produzidos na definição de requisitos

O item 4.2.1 descreveu como foram definidos os artefatos da especificação do sistema, o item 4.2.2 apresenta os artefatos produzidos durante a especificação.

4.2.2 Artefatos obtidos no processo de desenvolvimento do sistema

O item 4.2.2.1 mostra o escopo do sistema descrevendo seu objetivo principal. O item 4.2.2.2 mostra como foi feita a definição de requisitos mostrando os artefatos gerados finalizando com o modelo de perfis de usuário. O item 4.2.3 descreve a estrutura de banco de dados descrita pelo modelo entidade-relacionamento) seguido da estrutura de telas (item 4.2.4).

4.2.2.1 Escopo específico do trabalho

O objetivo principal do sistema desenvolvido é permitir a avaliação da viabilidade técnica do conceito e a identificação das restrições e aspectos tecnológicos que devem ser pesquisados para a sua evolução. Isto é, o sistema é um protótipo a ser empregado na avaliação da proposta original.

Considerando este objetivo maior, tem-se como objetivo específico do projeto, o seguinte:

- Ser capaz de apresentar funcionalidades suficientes para demonstrar o sistema proposto no capítulo 1;

Em relação ao desenvolvimento houve uma preocupação em relação ao tempo necessário para o desenvolvimento do protótipo. Havia tempo hábil para a especificação do sistema completo. Assim, decidiu-se especificar e implementar todo o sistema.

Quanto aos recursos de *hardware*, o laboratório de pesquisa NUMA – Núcleo de Manufatura Avançada, localizado na Escola de Engenharia de São Carlos (Campus USP), disponibilizou computadores com os *softwares* necessários para o desenvolvimento e implementação do protótipo. O item 4.2.2.2 apresenta a definição de requisitos descrevendo os artefatos gerados.

4.2.2.2 Definição de requisitos

Os requisitos do sistema foram levantados a partir de pesquisa realizada em sistemas de informação. Como o tema Sistemas de Informação é muito amplo, com base no item 2.1.3 que apresenta ferramentas de apoio à gestão do conhecimento, foi estipulada uma lista de sistemas para a realização da pesquisa de levantamento de requisitos. Os sistemas abordados foram: gestão de qualidade, modelagem de empresa, gestão de comunidades, gestão de melhores práticas, gestão de escopo, gestão de tempo, gestão de riscos, gestão de projetos, gestão de *portfólio*, mecanismos avançados de recuperação, *e-learning*, comunicação, gestão de documentos, gestão de pessoas, gestão de orçamento, gestão de aquisição e gestão de produtividade. Entre os sistemas abordados no levantamento de requisitos existiram aqueles em que não houve nenhum tipo de acesso ao *software*. Assim o levantamento de requisitos se deu através da documentação do sistema (artigos e documentação em *sites*).

Antes de escrever a lista dos requisitos de *software* considerados importantes, foi estabelecido um método e um formato de armazená–los. Inicialmente a opção foi armazenar os dados no formato de planilhas. Existem duas planilhas com os seguintes campos:

- Planilha 1 : sistema: (nome do sistema), tipo de sistema (conforme classificação descrita), fornecedor (empresa que desenvolveu o sistema), descrição (resume a aplicação principal do sistema), *site* (endereço eletrônico do sistema), fonte

(existem sistemas pesquisados através da literatura a respeito do sistema), data (data da pesquisa);

- Planilha 2: Módulo (agrupamento de funcionalidades conforme sua função para desempenhar um papel no sistema), funcionalidades e requisitos.

Neste formato havia facilidade de armazenamento, mas dificuldades de relacionamento e agrupamento de requisitos. Então foi decidido utilizar uma segunda ferramenta para armazenamento de requisitos, o RequisitePro, cujas vantagens são:

- Armazena requisitos em forma hierárquica (árvore de diretórios, conforme Figura 4-2): com isso é possível armazenar requisitos por tipo de sistema;
- Integração com o Microsoft Word: é possível obter relatórios em formato texto (Word da Microsoft);
- Relacionamento: é possível realizar a rastreabilidade de requisitos, ou seja, relacionar as informações nas quais foram baseados;
- Controle de alteração: essa funcionalidade evita que um requisito seja excluído ou alterado sem permissão e controla por quem, quando e porque foi alterado;
- Conexão com diagrama de casos de uso*:* a ferramenta possui conexão com a ferramenta Rational Rose, mais especificamente com diagramas de casos de uso. Com isso é possível relacionar um caso de uso com requisitos.

A Figura 4-2 mostra a estrutura da árvore de requisitos gerada na ferramenta RequisitePro. Os itens da estrutura foram descritos no item 4.2.1.3.

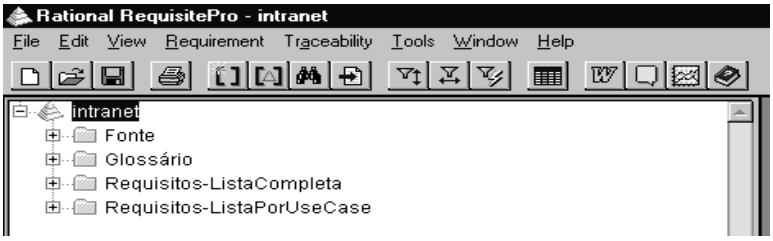


Figura 4-2 - Estrutura de requisitos

O RequisitePro permite a criação de pacotes (pastas em disposição hierárquica). A estrutura criada contém fonte e glossário, lista completa de requisitos e lista de casos de uso.

A lista completa de requisitos mostra todos os requisitos divididos por tipo de sistema. Essa divisão é mostrada na Figura 4-3 com mais detalhes. Por exemplo, todos os requisitos que se encontra em sistemas de comunicação estão nessa pasta. A lista de casos de uso apresenta os diagramas de casos de uso gerados e são ordenados por pacotes. Esses pacotes são intitulados com os nomes dos diagramas de casos de uso gerados na ferramenta Rational Rose (Figura 4-3). Essa ferramenta permite entre outros diagramas, construir os de casos de uso. Existe uma particularidade muito importante nessa ferramenta que é a consistência de dados. É possível fazer um relacionamento com requisitos da ferramenta RequisitePro, já citada anteriormente.

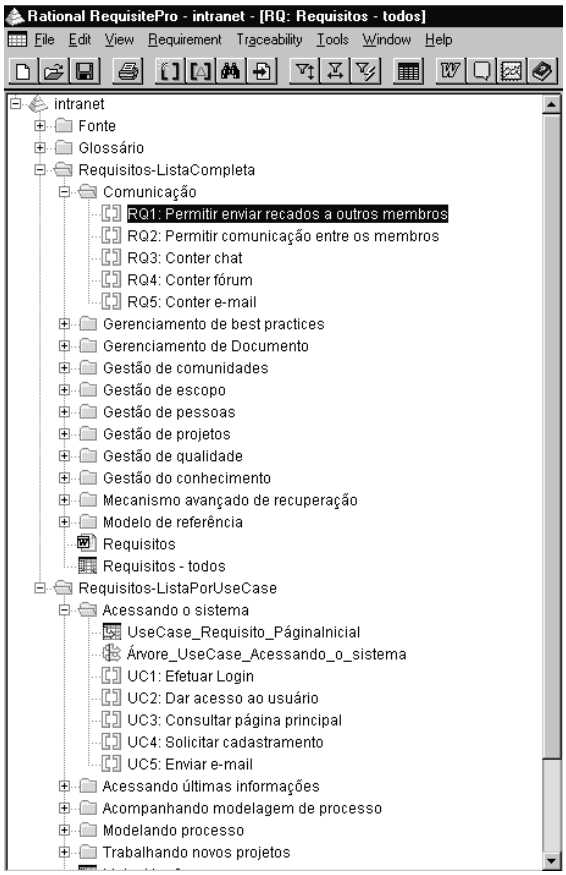


Figura 4-3 - Estrutura de requisitos (árvore aberta)

Outro ponto importante é o reuso de caso de uso, ou seja, é permitido utilizar um caso de uso de um diagrama em outro diagrama. Isso contribui para a consistência dos dados, pois quando o conteúdo de um caso de uso é alterado, todos aqueles instanciados seriam alterados também.

A Figura 4-4 mostra a tela principal da ferramenta, juntamente com a lista de diagramas gerados para a especificação do sistema.

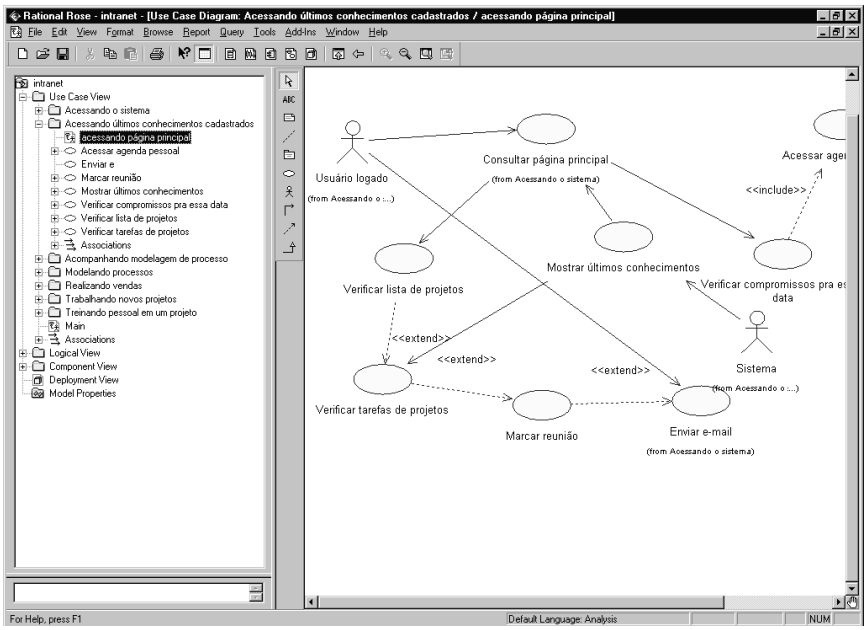


Figura 4-4 - Ferramenta Rational Rose (diagrama de casos de uso)

Os principais requisitos do sistema proposto são mostrados na Tabela 4-1 e na Tabela 4-2:

Requisitos gerais:

Tabela 4-1 - Requisitos gerais

1. Gerenciar fichas catalográficas
2. Publicar página RSS (XML)
3. Visualizar RSS (com *software* de visualização de notícias)
4. Limitar visualização no RSS
5. Buscar elementos* em base de dados
6. Limitar páginas exibidas de acordo com o *login*
7. Limitar operações de acordo com o *login*

Requisitos específicos:

Tabela 4-2 - Requisitos específicos

1.0 Criar
1.1 Ficha catalográfica
1.1.1 Nova figura antes (da aplicação do DFA)
1.1.2 Nova figura depois (da aplicação do DFA)
1.1.3 Preencher ficha catalográfica (nome, tipo, versão, data, responsável, e-mail, link para, visível)
1.1.4 Concluir inclusão
2.0 Alterar
2.1 Ficha catalográfica
2.1.1 Alterar nova figura antes (da aplicação do DFA)
2.1.2 Alterar nova figura depois (da aplicação do DFA)
2.1.3 Alterar ficha catalográfica (nome, tipo, versão, data, responsável, e-mail, link para, visível)
3.0 Excluir
3.1 Ficha catalográfica
3.1.1 Excluir ficha catalográfica
4.0 Visualizar RSS
4.1 Em *software* de visualização de notícias
4.1.1 Cadastrar endereço com extensão XML
4.1.2 Visualizar as páginas permitidas

Em 4.2.2.3 são apresentados quais foram os diagramas de caso de uso gerados na especificação.

4.2.2.3 Casos de uso e requisitos do sistema proposto

Este item mostra os artefatos, casos de uso e requisitos relacionados a casos de DFA.

Os diagramas casos de uso gerados foram:

- Acessar o sistema;

- Buscar caso de DFA;
- Visualizar caso de DFA;
- Inserir caso de DFA;
- Gerenciar usuário;
- Gerenciar palavras chaves.

O *software* RequisitePro permite realizar várias visões para demonstração dos requisitos chamados de relacionamentos (item 4.2.1.3). Para complementar a documentação da especificação em relação a requisitos foi gerada cinco artefatos. Alguns mostram a rastreabilidade dos requisitos desde os casos de uso até a fonte de onde foi baseado o requisito e outros em forma de texto, listam todos os requisitos, glossários e fontes. Esses artefatos são descritos na Tabela 4-3:

Tabela 4-3 - Lista de documentos gerais

Nome documento	**Descrição**
Lista geral "tipo árvore" (casos de uso)	Mostra uma lista "tipo árvore", ou seja, mostra de forma hierárquica todos os requisitos da especificação começando por casos de uso. (mostra requisitos relacionados a casos de uso, glossário e fontes relacionados à requisitos e fontes relacionados à glossário);
Lista "tipo planilha" (requisitos)	Mostra uma lista "tipo planilha" dos requisitos com descrição, quais casos de uso relacionados e qual tipo de ferramenta pertencem;
Documento texto (Requisitos)	Mostra um documento texto (Microsoft Word) de todos os requisitos da especificação;
Documento texto (Glossário)	Mostra um documento texto (Microsoft Word) do glossário da especificação;
Documento texto (Fonte)	Mostra um documento texto (Microsoft Word) das fontes da especificação;

Em 4.2.2.4 é apresentado o artefato modelo de perfis de usuários em forma de tabela. Nela, descrevem-se os tipos de usuários que o sistema poderá conter.

4.2.2.4 Modelo de perfis do usuário

O modelo de perfis do sistema é apresentado na Tabela 4-4. Ele descreve de forma simplificada quais tipos de usuário o sistema suporta e as restrições dos mesmos.

Tabela 4-4 - Perfis de usuários

Perfil	Permissões
Administrador	• Cadastrar: usuário, ficha catalográfica; • Excluir: usuário, ficha catalográfica; • Alterar: usuário, ficha catalográfica; • Publicar página RSS; • Dar permissão: usuário.
Usuário	• Cadastrar: usuário, ficha catalográfica; • Publicar página RSS.

Em 4.2.3 é apresentada a estrutura de banco de dados do sistema. Este artefato não seguiu a estrutura UML e foi elaborado no formato Modelo Entidade Relacionamento (MER).

4.2.3 Estrutura de banco de dados do sistema

O modelo entidade relacionamento (MER) da base de dados do sistema é apresentado na Figura 4-5, destacando-se todas as entidades do sistema.

A entidade usuário representa as pessoas cadastradas no sistema e se relaciona com algumas entidades descritas. Esse relacionamento permite que se registre quem foi o responsável pelo cadastramento. Essa entidade possui um atributo denominado nível, o qual pode assumir um dos valores descritos na tabela de perfis (Tabela 4-4).

A entidade DFA possui um papel fundamental dentro da estrutura. Ela se relaciona com todas as outras entidades, pois é o papel central do *software*. Essa entidade concentra toda "malha" que amarra o foco principal do sistema que é a ficha catalográfica. Mas existem outras entidades importantes nesse processo.

A entidade Pergunta Fórum relaciona-se com a entidade DFA, pois para cada DFA ou ficha catalográfica existe um fórum criado automaticamente. Sendo assim, cada DFA possui várias perguntas do fórum de diferentes autores (relacionamento com a entidade usuário) e relaciona-se com Resposta Fórum, pois para cada pergunta

podem existir várias respostas de diferentes autores (relacionamento com a entidade usuário).

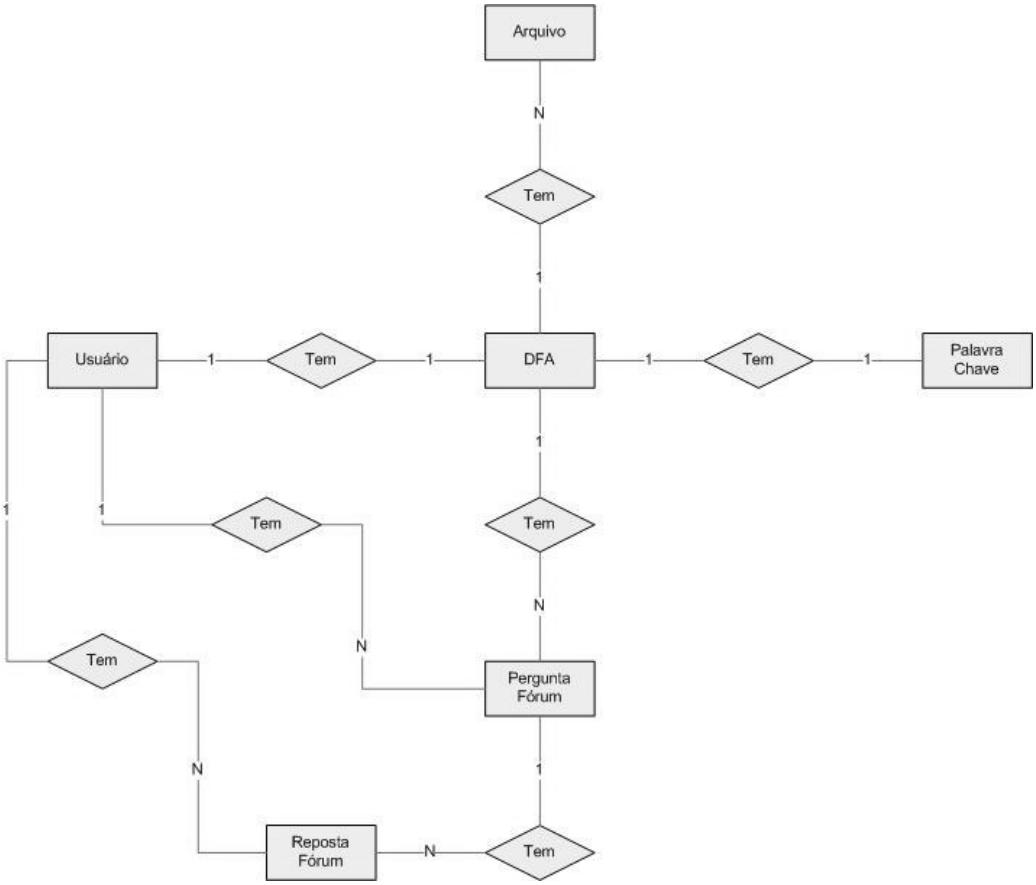


Figura 4-5 - Modelo Entidade Relacionamento

A entidade Arquivo define qual o conjunto de arquivos, ou seja, qualquer tipo de documento texto, planilha, desenho, etc. que possa ser anexado a ficha catalográfica para complementar seu conhecimento. Essa entidade possui relacionamento 1 para N, ou seja, para uma ficha catalográfica podem existir vários arquivos.

Em 4.2.4 é apresentada a estrutura de telas, onde é possível ter uma visão geral do sistema.

4.2.4 Estrutura de telas do sistema

Este é um artefato importante da especificação. Com ele é possível representar esquematicamente as páginas do sistema e seus relacionamentos. Fornece ao programador do sistema uma "visão" geral das páginas, suas funções e relacionamentos com outras. A Figura 4-6 apresenta a estrutura de telas. Páginas de menor importância não foram omitidas no desenho.

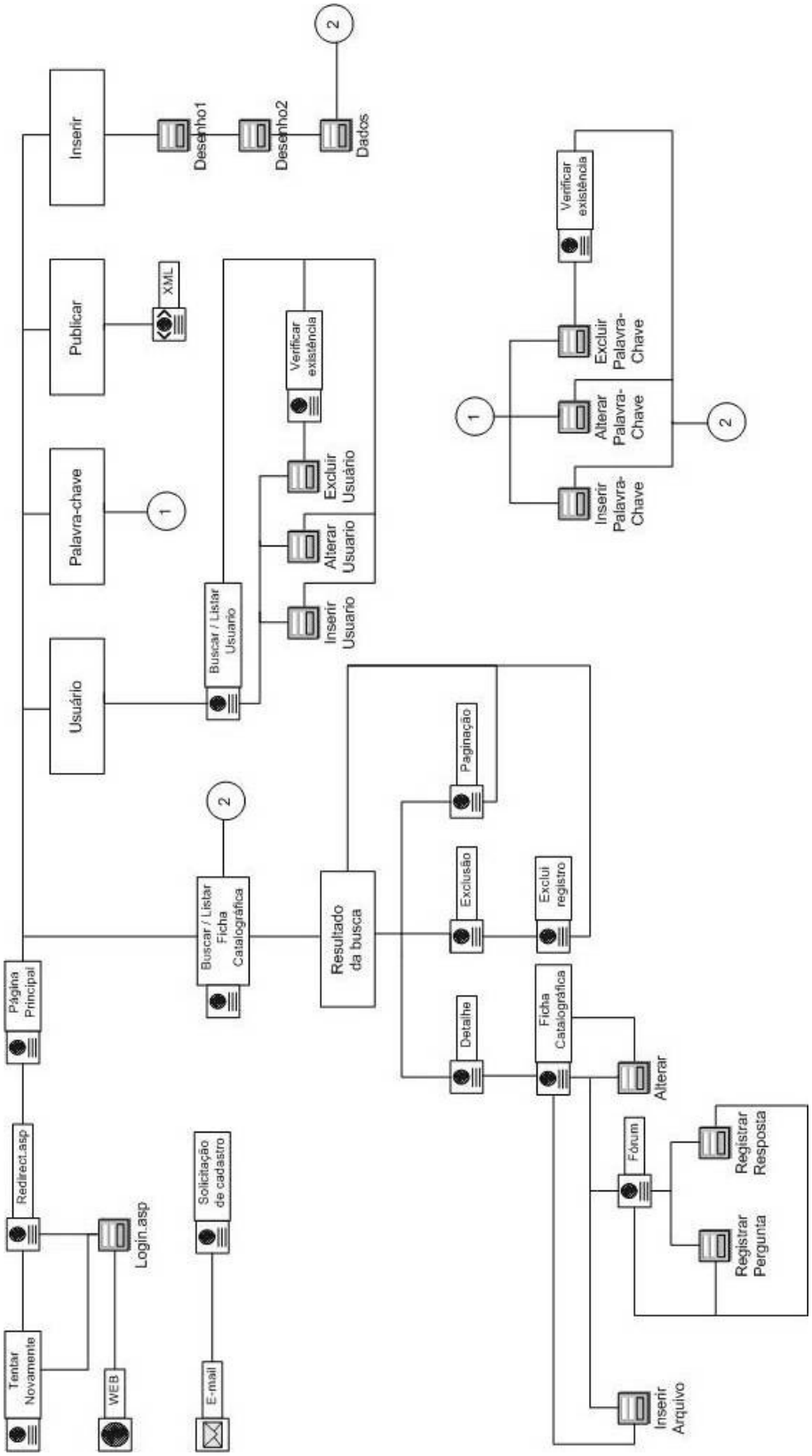


Figura 4-6 - Estrutura de telas

Os itens 4.2.2, 4.2.3 e 4.2.4 apresentaram os artefatos gerados na especificação do sistema. Em 4.2.5 são apresentados artefatos gerados durante a implementação, ou seja, condições a serem satisfeitas para a codificação do sistema.

4.2.5 Implementação do protótipo

A implementação, também chamada de codificação ou desenvolvimento, é uma tradução da especificação para um código (RUMBAUGH, BLAHA & PREMERLANI, 1994). Este item descreve convenções de programação (item 4.2.5.1) onde são escritas as regras para garantir um mínimo de padronização no código. O item 4.2.5.2 mostra as tecnologias usadas para a obtenção do protótipo e o item 4.2.5.3 descreve o sistema mostrando as telas do protótipo.

4.2.5.1 Convenções de programação

Para desenvolvimento do sistema foram empregadas convenções de programação. Essa atitude é importante para a boa confecção e posterior manutenção do sistema (reuso do sistema em outros projetos de pesquisa futuros). É útil em equipes de desenvolvimento de *software*, pois evita erros e economiza tempo na manutenção e compreensão de partes específicas. A seguir, são descritas essas convenções. A organização de diretórios deve ser dividida conforme Tabela 4-5:

Tabela 4-5 - Convenções de programação (organização de diretórios)

Diretório	Conteúdo
Intranet (raiz)	Index.asp, redirect.asp, global.asa e todas as sub-pastas descritas a seguir
Imagem (sub-pasta de intranet)	Todas as imagens utilizadas no sistema
Biblioteca (sub-pasta de intranet)	Todas as funções utilizadas no sistema
css (sub-pasta de intranet)	Todos os css utilizados no sistema
dfa (sub-pasta de intranet)	Todos as páginas relacionadas à modelo
Arquivo (sub-pasta de intranet)	Todos arquivos de uploads
Obs: os nomes das pastas devem ser em letra minúscula e sem acentuação e deve retratar o ambiente do sistema que será codificado	

Nome de tabelas e de campos de banco de dados: os nomes de tabelas devem conter o nome do ambiente em que vai ser usado, por exemplo: dfa. Os nomes das tabelas de relacionamento devem conter os nomes das tabelas relacionadas separadas

por traço baixo, por exemplo: dfa_alteracao. Os campos das tabelas devem conter primeiro o nome da tabela e depois, separado por traço baixo, o nome do campo, por exemplo: dfa_exclusao. Todos os nomes aqui tratados devem ser escritos em minúsculo e sem acentuação.

Nome de páginas *web*: o nome da página *web* deve expressar sua função e escrito em letra minúscula, sem acentuação e separado do traço baixo se houver mais de uma palavra e, ainda não deve conter palavras de ligação, por exemplo: cadastro_usuário.asp

Código: o código deve respeitar a identação dos comandos e funções. Deve possuir comentários sucintos e claros, separando os blocos de codificação. A chamada de funções da biblioteca deve estar no final do código.

Abertura e fechamento da base de dados: o *alias* da abertura da base de dados deve receber o nome abreviado, em minúsculo e sem acento da tabela a ser aberta, por exemplo: dfa (da tabela dfa).

Em 4.2.5.2 são descritas a tecnologia (*hardware* e *software*) utilizada na confecção do sistema.

4.2.5.2 Ferramentas de programação e de banco de dados

A infra-estrutura do sistema foi implementada sobre plataforma Windows com os seguintes servidores:

- Windows 2000 Server com IIS 5.0 (*Internet Information Server*) e interpretador ASP (*Active Server Pages*), como servidor de páginas *web.*
- Windows Access 2003 como banco de dados relacional.

Entre as tecnologias utilizadas na implementação, destacam-se:

- Códigos escritos em VBScript e empregando a tecnologia ASP (*Active Server Pages*): linguagem de criação de scripts embutidas em HTML no servidor;
- Banco de dados Access: sistema para gerenciamento de base de dados;

- Meta-linguagem XML (*Extensible Markup Language*): linguagem de marcação apropriada para a representação de dados, documentos e demais entidades cuja essência fundamenta-se na capacidade de agregar informações;
- DTD (*Document Type Definition*): gramática que rege a estrutura e composição dos documentos XML;
- Linguagem JavaScript;
- Ferramentas de programação: Dreamweaver e Fireworks do pacote MX da empresa Macromedia.

Em 4.2.5.3 descreve-se, no formato de telas, o sistema protótipo proposto.

4.2.5.3 Apresentação do protótipo

Esse item faz a apresentação das principais páginas do sistema. Foram confeccionadas com as ferramentas do pacote MX da Macromedia: *Dreamweaver* e *Fireworks*. A seguir são descritas algumas páginas do sistema suficientes para compreender o funcionamento básico.

Página principal: a Figura 4-7 mostra a página principal do sistema. Esta página mostra onde estão concentradas algumas funcionalidades do sistema. O número 1 mostra a janela onde é feita a busca no sistema. É muito importante procurar pela eficiência da busca por casos, sendo assim foi implementada a busca por palavras chave. Basta segurar o Ctrl pressionado e escolher quais palavras chave participam da busca. A busca pode ser feita simultaneamente por nome ou parte dele e/ou palavra chave.

O número 2 mostra o resultado da busca. Contém o nome do caso de DFA, o usuário que cadastrou o registro, a palavra chave correspondente, visualiza, ou seja, se esse registro é visível no XML ou não, a data do cadastramento, exclusão (número 4), ou seja, se deseja excluir esse registro e detalhe (número 5), que direciona a uma página chamada ficha catalográfica, onde o registro é exposto com maiores detalhes.

O número 3 demonstra a paginação feita nesta página de internet. A paginação nada mais é do que mostrar o resultado da busca em números determinados de

registros, neste caso de 5 em 5. Basta acessar o link (3) para acessar as outras páginas com os demais registros resultantes da busca.

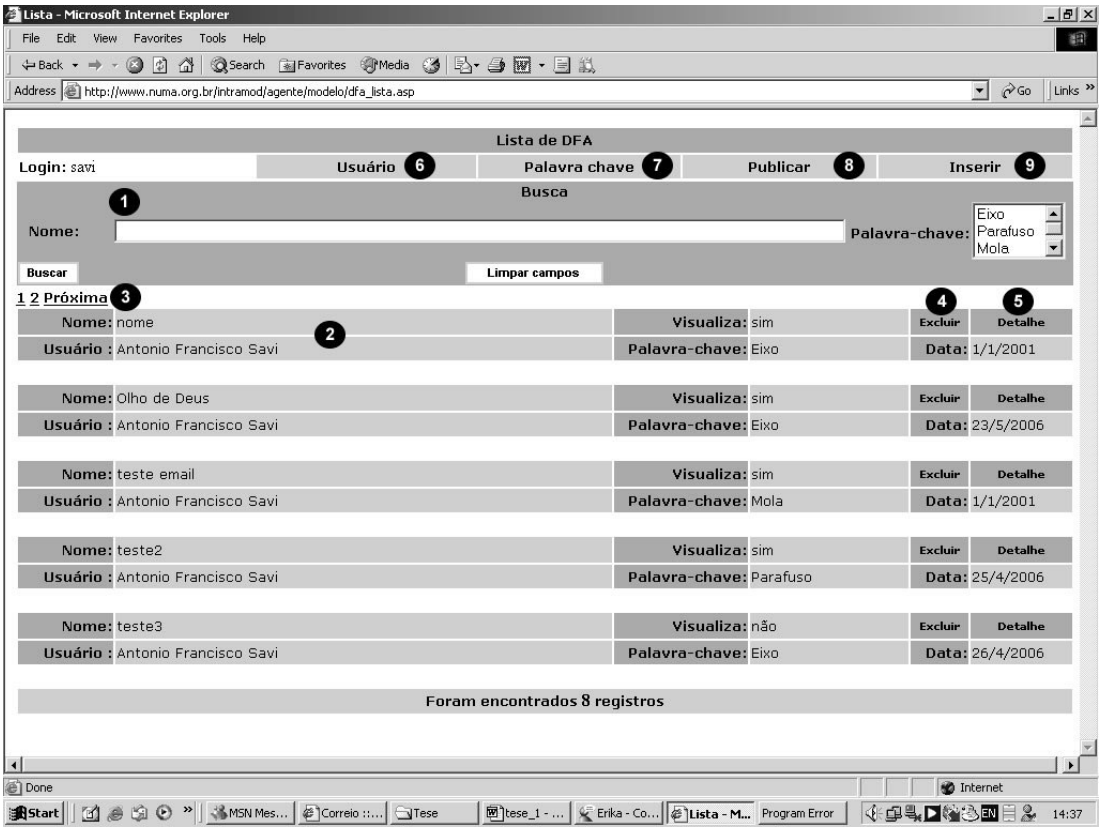


Figura 4-7 - Página principal do sistema

O número 6 arremete o usuário para o gerenciamento de usuários, ou seja, inclusão, alteração e exclusão de usuários.

O número 7 direciona para o gerenciamento de palavras chave, que realiza a inclusão, alteração e exclusão de palavras chave.

O número 8 realiza a "construção" da página XML. Essa página pode ser acessada por *softwares* clientes de leitura de notícias, *softwares* esses gratuitos encontrados na *internet* ou pelo próprio *Browser.* Através do uso desses *softwares* é possível criar a rede *peer-to-peer* (ponto a ponto) e estabelecer conexão com vários servidores de empresas espalhadas geograficamente.

O número 9 direciona para uma página de inserção. Esta página "constrói" a página de ficha catalográfica, contendo todos os dados e figuras.

Inserção da Ficha Catalográfica: ao clicar no link "Inserção" o usuário encontra uma primeira tela que é demonstrada a seguir na Figura 4-8.

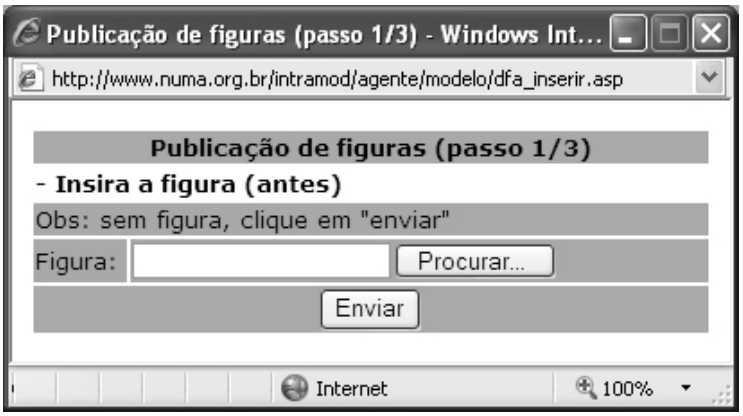


Figura 4-8 - Inserção (Passo 1/3)

Essa tela pede ao usuário a primeira figura que vai fazer parte da Ficha Catalográfica, ou seja, essa seria a figura "antes" da aplicação do caso de DFA. Se o caso não necessitar de figura basta o usuário clicar em enviar e será arremetido para outra tela. Isso acontece também com a tela da Figura 4-9 que representa figura "depois" da aplicação do caso.

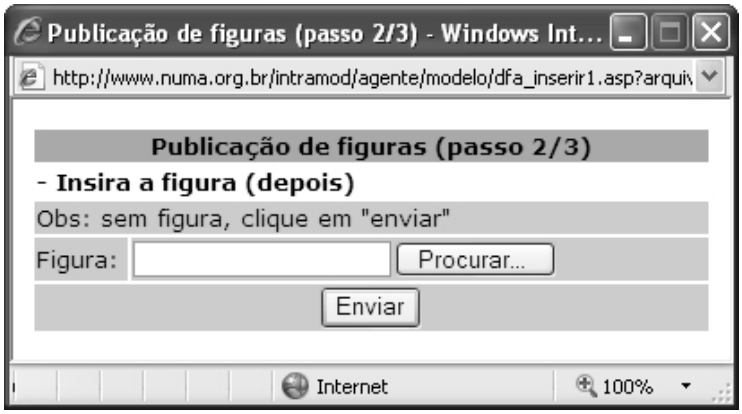


Figura 4-9 - Inserção (Passo 2/3)

O próximo passo pede ao usuário alguns dados mostrados na Figura 4-10. Entre eles alguns são trazidos automaticamente pelo sistema, como o Usuário, E-mail e a Data. Os demais campos são: o nome, é o que identifica o caso juntamente com a palavra chave; a palavra chave é previamente cadastrada em outra página da ferramenta e faz parte da busca da casos da ferramenta; tipo e versão, são apenas campos documentacionais, ou seja, não participam da busca; data, usuário e e-mail são campos já cadastrados na página de gerenciamento de usuários da ferramenta;

texto, descreve o caso em si, ou seja, é nesse campo que o usuário irá explicar o Caso de DFA; link possibilita cadastrar uma página Web interna ou externa que complemente o Caso DFA; visível, possibilita que essa Ficha Catalográfica seja vista externamente por outras empresas.

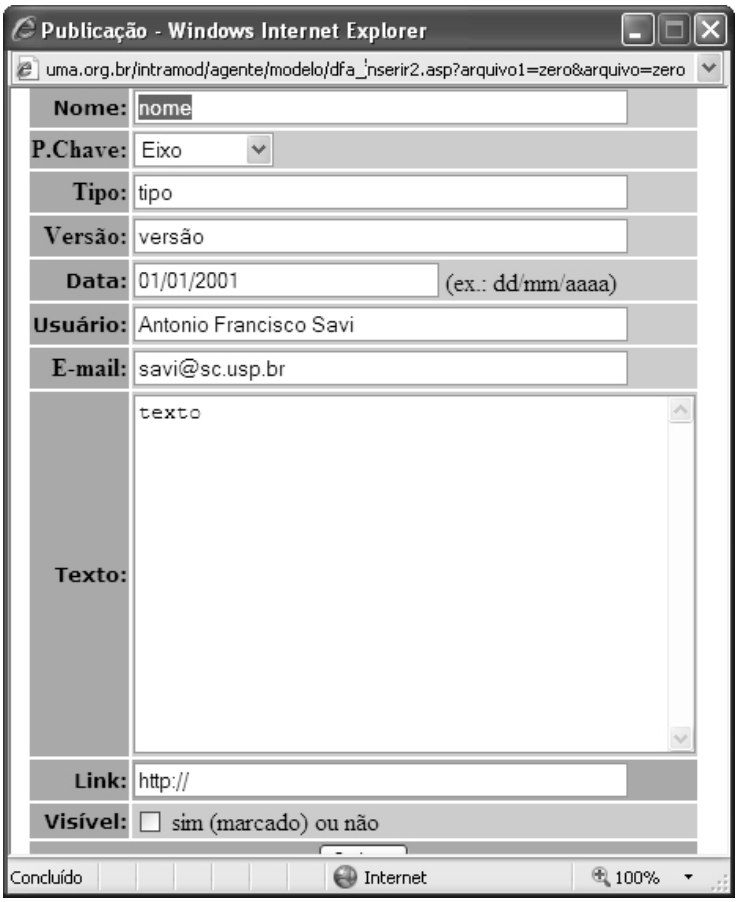


Figura 4-10 - Inserção (Passo 3/3)

Página usuário: esta página cadastra, altera e exclui usuários pertencentes ao sistema (Figura 4-11). O número 1 é a busca que pode ser feita por nome e/ou deferido ou indeferido. A busca por nome pode ser truncada, ou seja, procurar por parte do nome (obs: o nome se refere ao nome completo do usuário, nome e sobrenome). O sistema tem como padrão a busca completa, ou seja, toda vez que esta página for acionada vai ser feita uma busca total como resultado (nome em branco e ambos).

O número 2 é o resultado da busca. Como resultado são obtidos o nome completo do usuário, *login* do sistema, e-mail do usuário, empresa ou instituição

onde trabalha e seu status, que pode ser deferido, se está ativo no sistema ou indeferido, se não esta ativo no sistema.

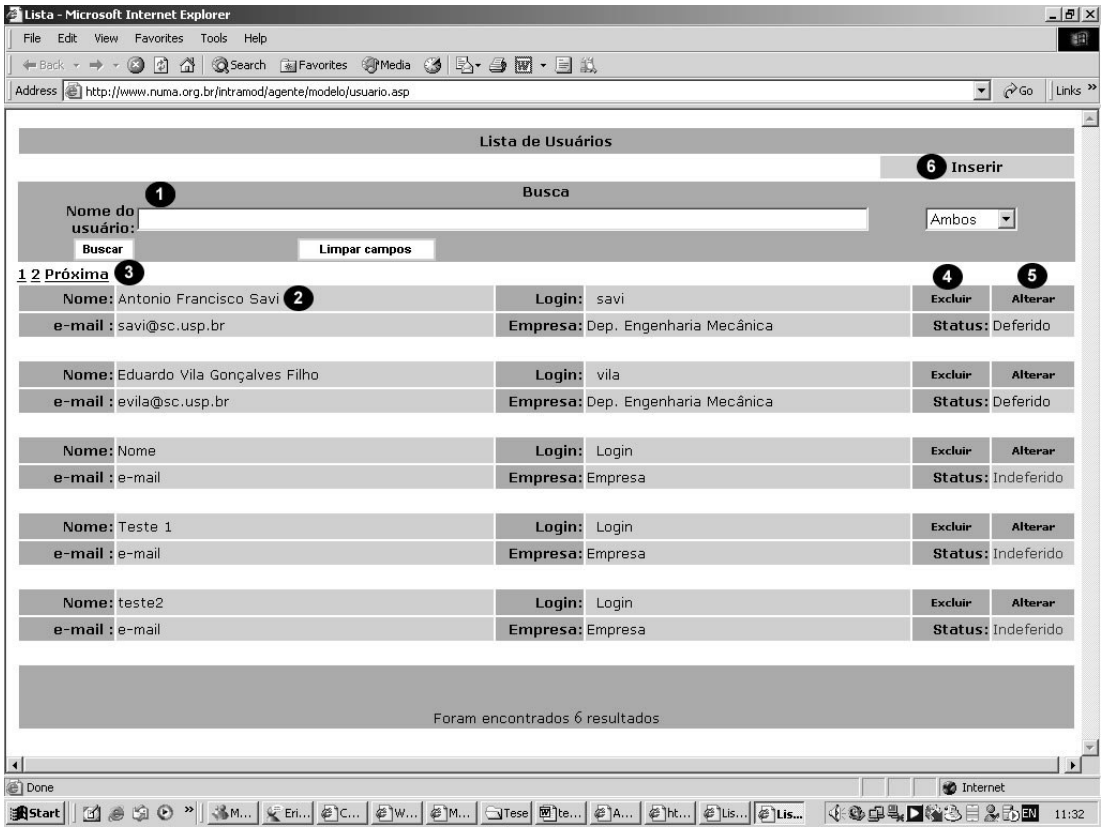


Figura 4-11 - Gerenciamento de usuários

O número 3 informa que esta página possui paginação. Todo o resultado vai ser mostrado em intervalos de 5 registros. Para acessar as outras páginas basta clicar no link indicado (número 3).

O número 4 direciona para uma página de exclusão. Esta página indica se existe algum registro com o nome de tal usuário, ou seja, se existe algum registro cadastrado por este usuário. Se existir, não é possível fazer a exclusão, o sistema impede a exclusão deste usuário. Se não existir , a exclusão é feita normalmente.

O número 5 leva o usuário para uma página de alteração. Nesta página são mostrados todos os dados que podem ser alterados, inclusive o deferimento e indeferimento do usuário.

O número 6 direciona para uma página de inserção de usuário. Lá existe todos os campos a serem preenchidos para a criação do usuário, inclusive a senha para acessar o sistema.

Página de palavras chave: é onde é feito o gerenciamento das palavras chave. O gerenciamento consta da inclusão, alteração e exclusão da palavra chave (Figura 4-12). Porém existe uma particularidade na exclusão. Para se fazer a exclusão é necessário que não exista registro com a palavra chave em questão. Para isso o usuário é direcinado a uma tela onde pode ser feita a alteração de palavra chave do determinado registro onde se encontra a palavra chave a ser excluída. Depois de feita a alteração a exclusão é feita normalmente.

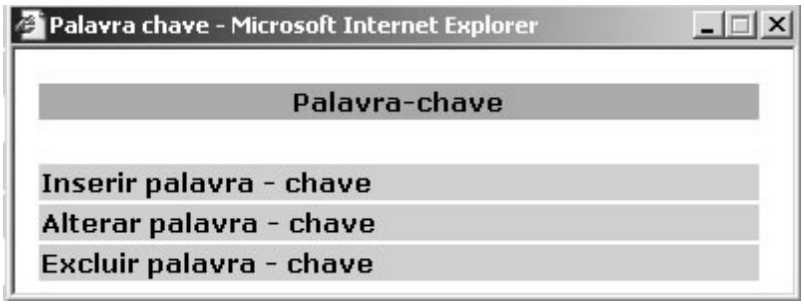


Figura 4-12 - Gerenciamento de palavras chave

Página ficha catalográfica: onde é mostrado tudo o que foi catalogado para o caso de DFA, ou seja, essa página contém os registros como nome, palavra chave, tipo, data ,usuário que cadastrou, etc. Para descrever o caso existe um campo texto, onde o usuário pode explanar sobre o caso referido seguido de duas figuras. Essas figuras servem para demonstrar o "antes" e o "depois" da aplicação do caso. Ainda nessa página existe um campo para a inserção de arquivos que tem uma relação de 1 para N, ou seja, é permitido a inserção de N arquivos para um caso. Para complementar cada caso ou ficha catalográfica, contém um fórum próprio. Esse fórum é criado na relação de 1 para 1, ou seja, para cada ficha catalográfica é criado um fórum específico. Este fórum funciona da seguinte maneira: o usuário cria perguntas que podem ser respondidas por ele ou por outros usuários. Só quem cria a pergunta pode excluí-la. O mesmo acontece com com responde, só quem responde a pergunta pode excluí-la. A cada pergunta ou resposta é registrado o nome do usuário. A seguir a Figura 4-13 demonstra a página descrita anteriormente.

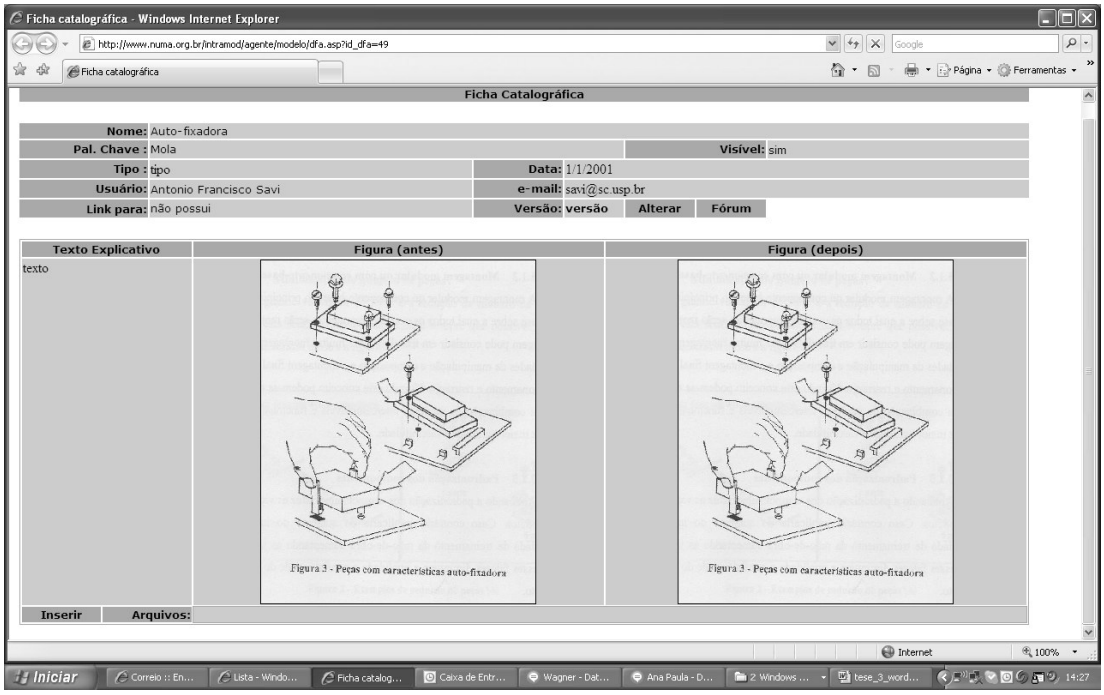


Figura 4-13 - Ficha Catalográfica

A Figura 4-14 demonstra o Fórum criado para cada Ficha Catalográfica.

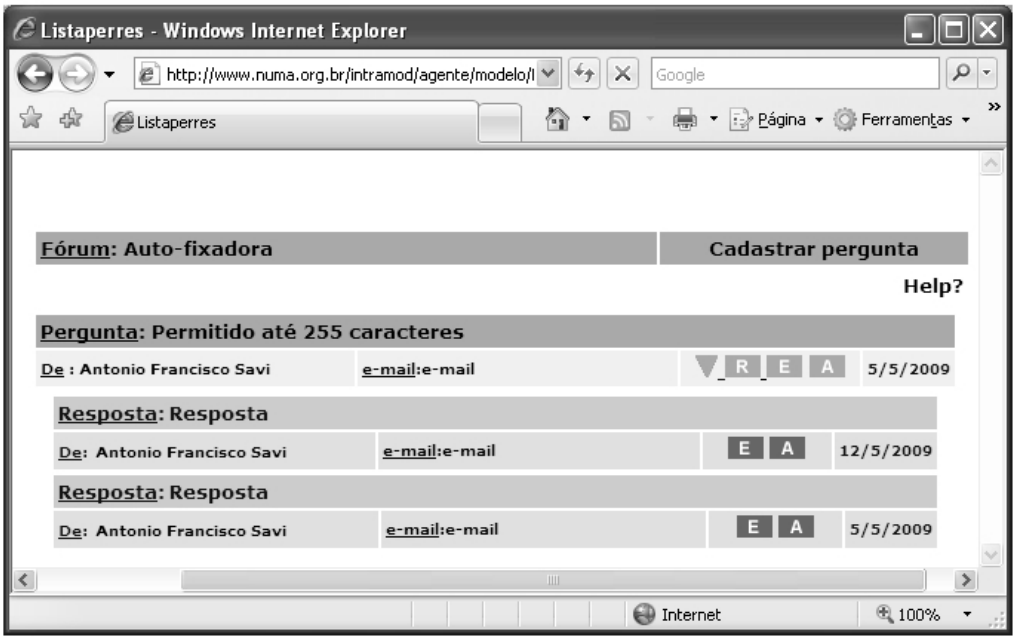


Figura 4-14 – Fórum

Publicação do conteúdo via XML: o link "Publicar" que se encontra na página principal e marcado como número 8 (na demonstração da página principal) é demonstrado abaixo na Figura 4-15.

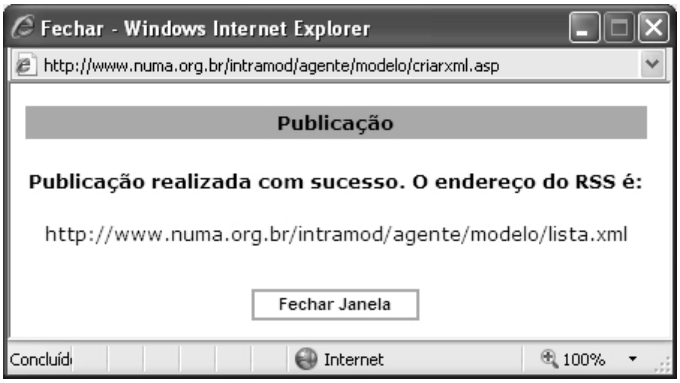


Figura 4-15 – Publicar

Esse link consiste em criar a página XML que será lida pelos *softwares* de leitura de XML ou pelo próprio *Browser*, já que atualmente esses *Browsers* leêm esse tipo de arquivo de uma forma "amigável", ou seja, que o usuário possa entender seu conteúdo. O conteúdo mostrado à partir da criação dessa página XML é àquele que foi marcado com "visível" na criação da Ficha Catalográfica. Sendo assim, o conteúdo marcado como "Não Visível" só poderá ser visualizado pela empresa criadora da Ficha Catalográfica em questão.

Ainda em relação ao protótipo desenvolvido houve uma contribuição de um pesquisador da área de DFA que analisou o protótipo. Esta análise foi realizada subjetivamente, ou seja, as perguntas não são apresentadas com respostas gradativas, sendo assim, a opinião do avaliador se sobressai em cada pergunta realizada. O avaliador é profissional da área de engenharia e especialista em técnicas de DFA. A seguir apresenta-se a questionário com as respectivas respostas.

1. É clara a proposta do sistema?

   Sim, segundo o especialista, a proposta do sistema é clara. Há uma lacuna no que se refere à pesquisa com DFA. Essa lacuna é devido ao conhecimento referente à DFA estar em vários locais, ou seja, em livros, anotações e nas cabeças das pessoas, assim o conhecimento se perde facilmente. Esse último local, na cabeça das pessoas, é o local mais provável de perder-se, pois basta a pessoa sair da organização e todo o conhecimento irá com ela.

2. A interface do sistema é amigável?

Sim, segundo o especialista, não houve dificuldade de criar, ler e excluir informações.

3. A busca de casos de DFA é eficiente?

Sim, a busca do sistema é feita por um ou somando-se dois ou mais argumentos. Pode-se fazer a busca por nome, por palavra chave, por nome e palavra chave e por nome e duas ou mais palavras chaves. Segundo o especialista não houve nenhum problema em encontrar alguma informação cadastrada.

4. O modo de visualização de casos (detalhe) é satisfatório?

Sim, segundo o especialista a visualização do cabeçalho, no detalhe, fornece algumas características do caso cadastrado, como data, usuário e e-mail. O texto explicativo, a figura antes e depois, o fórum e os arquivos anexos são importantes para a descrição do caso.

5. A realização de um fórum para cada caso é importante?

Sim, segundo o especialista, o fórum é um documento onde várias pessoas podem dar a sua contribuição ao caso exposto, ou seja, um caso pode ser resolvido de várias maneiras em um diálogo entre várias pessoas.

6. Como é a navegação pelas perguntas e respostas do fórum?

Segundo o avaliador, a navegação ocorreu sem nenhum problema. As perguntas e respostas estão expostas com cores e identação diferentes. Isso ajuda muito a visualização das perguntas e respostas que são apresentadas por ordem cronológica, ou seja, a resposta ou pergunta mais nova fica em primeiro lugar na lista.

7. O modo de anexar arquivos em casos (ficha catalográfica) é satisfatório?

Houve a necessidade de criar um local para inserção de arquivos na ficha catalográfica. Existe essa necessidade porque há várias formas de armazenamento de conhecimento e, arquivos de vários tipos estão entre essas formas. Segundo o avaliador é importante essa inserção para troca de conhecimento, pois existem arquivos de desenhos que não podem ser visualizados na web, como Autocad e Solid Edge, por exemplo.

8. Na página principal, como é a inserção de casos de DFA?

A inserção de casos de DFA, ou criação da ficha catalográfica é feita em três etapas. Primeiro o sistema pede uma figura que se identifica com “antes”, ou seja, é a figura de uma peça antes da aplicação do DFA, se não houver, simplesmente clique no botão inserir que o sistema remete o usuário para o próximo passo. O segundo passo é a figura “depois” e funciona exatamente como o primeiro, exceto pelo fato de que a figura é a depois da aplicação do DFA. Isso ilustra como o DFA resolveu um problema pontual mostrando como era antes e como foi resolvido. O terceiro passo é um pequeno questionário perguntando nome (que é colocado automaticamente pelo login do sistema), tipo, palavra chave, data (que é colocado automaticamente pelo sistema), e-mail e o texto explicativo. Segundo o avaliador é um espaço importante onde o usuário pode “explicar” o caso de DFA exposto.

9. Os passos de inserção de casos de DFA são claros?

Sim, segundo o avaliador os passos não só de inserção, mas de alteração são bem claros.

10. O módulo de gerenciamento de usuários é satisfatório?

Nesse módulo funciona o gerenciamento de usuários, ou seja, onde é feito o cadastramento para o uso do sistema com o *login* e senha próprios. Com essa ação o usuário pode inserir, alterar e excluir casos, mas deixando sua identificação automaticamente (nome do usuário). Segundo o avaliador seria coerente um usuário não poder excluir nem alterar o caso cadastrado por outro. Não era foco de o sistema ter um complexo controle de usuário, mas a sugestão é pertinente.

11. Ficou clara a utilização da publicação?

A princípio, segundo o avaliador, não ficou clara a utilização da publicação. A publicação é um recurso usado para que o usuário não tenha a necessidade de procurar casos novos pelos sites (esse sistema) de várias empresas. Existe um tipo de *software* que lê notícias em vários sites diferentes, chamado leitor de notícias. Basta o usuário configurar esse software com o endereço mostrado no

link “publicar” e ele avisará o usuário quando alguma empresa cadastrar casos novos. Lembrando que para aparecer no leitor de notícias, o usuário deve marcar o item “visível” no terceiro passo da inserção. Se o item “visível” estiver desmarcado ele só será visto dentro do site da própria empresa. Isso é um recurso para evitar que um “segredo” da empresa seja mostrado para todos.

12. O funcionamento do módulo de palavras chaves é satisfatório?

Sim, segundo o avaliador o módulo é satisfatório, pois existe a inserção, alteração e exclusão. Há uma particularidade na exclusão onde só pode ser feita se não existir nenhuma palavra chave pertencente a um caso. Se existir o sistema avisa quais casos e cria um link para alteração da palavra chave naquele caso. Só após a mudança pode ser feita a exclusão.

Existem outras sugestões dadas pelo avaliador que serão apresentadas a seguir:

- Falta de um *logout* na tela principal, onde o usuário possa sair do sistema;

- Falta uma confirmação quando é feita uma exclusão de caso. Esta sendo feita diretamente, sem perguntar ao usuário se deseja realmente excluir o caso;

- Falta colocar na tela de *login* o nome do sistema, o autor e um pequeno histórico.

5 CONCLUSÕES E TRABALHOS FUTUROS

Existe atualmente um conjunto grande de ferramentas que apóiam a gestão do conhecimento. Uma questão fundamental para a eficiência dessas ferramentas é a quantidade de informações, que tende a crescer exponencialmente. Isto dificulta a recuperação e em casos extremos pode impedi-la. Dentro desse cenário, tem sido dada uma grande atenção para as ferramentas de busca. A principal contribuição desse trabalho consiste em explorar outra estratégia de recuperação, diferente da proposta por essas ferramentas tradicionais.

O principal desafio foi propor um conceito para recuperação de informações, baseado em Raciocínio Baseado em Casos (RBC) e viabilizado pelo emprego das teorias de Gestão de Conhecimento (no caso a Palavra-Chave). Foram realizados estudos de ferramentas de gestão do conhecimento e DFA e uma revisão bibliográfica. No início procurou-se abordar o problema delimitando-o como sendo o da construção de um sistema que simula os sistemas já existentes no mercado.

Compreendia-se sistema como sendo um *software stand alone*, ou seja, sem comunicação com outros *softwares*. Porém, as referências acerca deste tipo de problema na literatura, a qual possui uma ênfase nas regras gerais de construção, isto é, componentes de comunicação (protocolos de comunicação) como CORBA e STEP, gerou um impasse. Desejava-se explorar prioritariamente a arquitetura de informação, isto é, como as informações relacionadas com aprendizagem poderiam ser integradas com informações de DFA. Esse problema estava bastante relacionado com o mecanismo de integração. No entanto, existe na literatura um conceito conhecido como RBC (Raciocínio Baseado em Casos), que vem tratando destes tipos de sistemas e do assunto de validação de informação. Ele forneceu os conceitos chave que levaram à criação do conceito, identificação das funcionalidades e sua aplicação.

A primeira contribuição deste trabalho é, portanto, a formalização mais rigorosa do conceito de validação e recuperação da informação aplicada aos sistemas de apoio à gestão do conhecimento. Nesta formalização inclui-se a descrição de potenciais benefícios do uso do conceito, são eles:

- Poderia haver maior interesse do usuário em relação ao uso do sistema, pois existe realmente uma interação entre o sistema e o usuário, já que o sistema fornece informações sem o usuário necessitar procurá-la;
- Enriquece o DFA ou parte dele com informações de como foram executadas atividades;
- Fornece aos participantes meios de coleta de informações mais eficazes, já que informações sobre DFA podem ser consultadas;
- Fornece à equipe de projeto de produto um ambiente de colaboração de informações, fazendo com que não exista a concentração de informações em pontos isolados;
- Facilita o desenvolvimento de novos casos de DFA, que podem ser construídos a partir da comparação e da análise das imperfeições de casos pré-existentes.

A formalização do conceito permitiu a construção de um protótipo para verificar sua viabilidade e comportamento. O conceito de RBC prevê que existem informações que qualificam a situação do usuário, por exemplo, seu nível de aprendizado. No conceito proposto faltou incluir a figura do mecanismo que compare casos, e defina o comportamento apropriado para o sistema. Logicamente, tal função é implícita no conceito proposto. Porém, a falta de uma ênfase maior, quando no emprego do conceito, gerou a falta de entidade e módulos específicos para sua implementação. Como conseqüência, seu funcionamento depende de funções espalhadas por todo o *software*. A incorporação de novas regras fica então prejudicada, pois depende de alterações em vários dos objetos e mudanças de atributos em várias entidades da base de dados. Deveria, portanto, ser incluído um novo elemento no conceito, uma regra ou outro dispositivo, que seja capaz de armazenar a “inteligência” para o filtro da informação conforme o contexto.

Deve-se notar que este aspecto foi outra contribuição importante desse trabalho. Ele sugere, para trabalhos futuros, a melhoria do conceito com a formalização de um mecanismo de regras, editáveis pelos usuários, que estabeleceriam ações a serem realizadas pelo sistema conforme as informações de contexto.

Com relação à especificação, um primeiro aspecto importante a ser notado é a dificuldade na definição de requisitos dado a inexistência de clientes. Assim, essa atividade foi realizada com base em comparações de outros sistemas. Não foram encontradas metodologias para identificar requisitos a partir de sistemas existentes. A comparação foi feita por meio da experiência própria do pesquisador. Cuidados, como o de identificação das fontes de cada requisito, diminuíram e tornaram mais sistematizada esta atividade, porém, acredita-se que há espaço para desenvolvimento de ferramentas que possam auxiliar comparações para esse fim.

Outro ponto é a desconsideração dos requisitos não-funcionais. A consideração de todos os requisitos, funcionais e não funcionais, demandaria um esforço muito grande, principalmente se considerarmos que, de acordo com o caráter inovador do projeto, existia, no seu início, pouco conhecimento sobre as reais possibilidades desse tipo de conceito. Decidiu-se, como estratégia, concentrar os esforços da pesquisa nos requisitos funcionais e, demonstrada sua viabilidade, proceder a verificação não-funcionais em projetos complementares. Um exemplo de como esse tipo de requisito pode ser importante são os requisitos não-funcionais de desempenho. O protótipo foi analisado quanto à execução ou não da tarefa, mas não se avaliou quão bem ele é capaz de realizá-la. Assim, mesmo que demonstrada a viabilidade em termos de função, o sistema pode ser inviável em situações reais, onde demonstrar um desempenho minimamente suficiente é condição necessária para ser utilizado.

Ainda na especificação, a base de dados ficou complexa e mudou várias vezes no decorrer da codificação trazendo inconsistência e necessidade de retrabalho. Sugere-se, para projetos futuros, um estudo pormenorizado dessas entidades e a inclusão da entidade “regras” no banco de dados.

É possível pensar na necessidade, no caso da construção de um sistema comercial a partir do conceito proposto, de se utilizar tecnologias adicionais tais

como pré-processamento de páginas, a fim de garantir um desempenho minimamente suficiente.

Além dessas considerações, deve ser notado que outra contribuição desse trabalho é o próprio protótipo desenvolvido. Ele poderá ser reutilizado em outros trabalhos que venham implementar novas funcionalidades baseadas no conceito de RBC.

Trabalhos Futuros

Durante o trabalho observou-se algumas lacunas no que se refere à especificação e implementação. São problemas, limitações ou novos caminhos identificados durante sua realização do escopo do trabalho. Recomendam-se os seguintes trabalhos futuros:

- Implementar toda a especificação realizada nesse trabalho com realização de testes com empresas reais;
- Dar seguimento a desenvolvimento do conceito com a introdução de um elemento decisório do sistema que faça a ligação entre validação de informação e DFA;
- Uma terceira linha de pesquisa poderá incluir à metodologia de especificação outros diagramas da UML, como diagramas de estado e diagrama de atividade, por exemplo;
- Outra linha de pesquisa poderá abordar a influência dos requisitos não-funcionais, visto que muitos requisitos não funcionais afetam diretamente o tipo de arquitetura que deverá ser utilizada caso eles venham a ser satisfeitos. Alguns autores destacam os requisitos não-funcionais como sendo de crucial importância para a obtenção de *softwares* que atendam as expectativas dos clientes (CHUNG, 1993; BOEHM & IN, 1996; DARDENNE, LAMSWEERDE & FICKAS, 1993; CYSNEIROS & LEITE, 1999). Uma grande parte dos aspectos de qualidade de um sistema é expressa por requisitos não-funcionais que, muitas vezes, são também denominados “atributos de qualidade de um *software*” (BOWEN, 1985).

6 REFERÊNCIAS BIBLIOGRÁFICAS

ADDOUCHE, S. (2003). **Contribution à une démarche de conception optimisée dês processus de désassemblage.** 203f. These (Grade de docteur de l´universite de Franche-Comte en Automatique et Informatique), Cap. 1, p. 1-23.

ADLER, P.; WINOGRAD T. (1992). **Usability**: turning technologies into tools. Oxford: Oxford University Press.

ALEXANDER, I. (2003). Misuse cases: **Use cases with hostile intent**. IEEE Software, v. 20, n.1, p. 58-66, jan.

AMARAL, D. C. (2002). **Arquitetura para gerenciamento de conhecimentos explícitos sobre processo de desenvolvimento de produto**. Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos (Tese de Doutorado).

ANDREADEN, M. M.; KAHLER, S.; LUND, T. (1998) **Design for Assembly.** 2 ed. United Kingdom: IFS Publications.

ATLAR. **XpertRule knowledge builder**. Disponível em: <http://www.attar.com/pages/info_kb.htm> Acesso em: 24 Jul. 2008.

BALANCE CONSULTING. **Entendendo os diferentes níveis de sofisticação de um Portal Corporativo**. Disponível em: <http://www.balanceconsulting.com.br/> Acesso em: 23 Fev. 2008.

BARROS, L. A. (1994). **Suporte a ambiente distribuídos para aprendizagem cooperativa**. Rio de Janeiro: COPPE, Universidade Federal do Rio de Janeiro (Tese de Doutorado).

BARROSO, A.C.O.; GOMES, E.B.P (1999). **Tentando entender gestão do conhecimento.** Rio de Janeiro: Comissão Nacional de Energia Nuclear.

BATANOV, D., ELORANTA, E. (2003). **Advanced Web technologies for industrial applications**. Computers in Industry, v.50, p.123-125.

BENEDETTI, F.; ABEL, M. (1999). **Modelagem de Textos Jurídicos em XML Usando Raciocínio Baseado em Casos**. Simpósio Internacional de Gestão do Conhecimento/Gestão de Documentos (ISKM/DM). Curitiba, PR: CEFET-PR, outubro.

BLOODGOOD, J.M.; SAILSBURY, W.D. (2001). **Understanding the influence of organizational change strategies on information technology and knowledge management strategies**. Decision support systems.

BOEHM, B.; IN, H. (1996). Identifying quality-requirement conflicts. **IEEE Software**, Los Alamitos, v.13, n.2, p.25-35, Mar.

BOOCH, G., RUMBAUGH, J., JACOBSON, I. (2000). **UML**: guia do usuário. Rio de Janeiro:Campus.

BORGES, M.R.S., CAVALCANTI, M.C.R.,CAMPOS, M.L.M. (1995). **Suporte por computador ao trabalho cooperativo**. XV Congresso da sociedade brasileira de computação.

BOOTH, W. C.; COLOMB, G. G.; WILLIAMS, J. M. (1995) **The Craft of Research**. Chicago: University of Chicago Press.

BOOTHROYD, G. (1988). Make it simple: **design for assembly**. Mechanical Engineering.

BOOTHROYD, G.; ALTING, L. (1992). **Design for assembly and disassembly**. In: Anuals of the CIRP, v.41/2, p.625-636.

BOOTHROYD, G.; DEWHURST, P.; KNIGTH, W. (1994) **Product design for manufacture and assembly.** New York: Marcel Dekker.

BOWEN, T.P. (1985). **Specification of software quality attributes**. New York: Rome Air Development Center, Grifiss Air Force Base.

BRALLA, J. G. (1996). **Design for Excellence.** New York: McGraw-Hill.

BRALLA, J. G. (1986). **Handbook of product design for manufacturing.** New York: McGraw-Hill.

BREITMAN, K.K.; LEITE, J.C.S. do P. (2002). **Managing user stories**. In: EBERLEIN, A.; LEITE, J.C.S. do P. (Ed.). Proceedings of the International Workshop on Time Constrained Requirements Engineering. Essen, Germany: [s.n.].

BRYMAN, A. (1989). **Research Methods and Organization Studies**. London: Unwin Hyman.

BURKHARD, H.D.(1998) Extending Some concepts of CBR – **Foundations of Case Retrieval Nets, in Case Based Reasoning Technology from foundations to applications**. Springer-Verlag , p.17 –50.

CAVALCANTI, M.R. Portais corporativos: **Tópicos Especiais em Banco de dados.** Disponível em: <http://genesis.nce.ufrj.br/dataware/Tebdpos2001_3/Trabalhos/Portais/TRAB-PORTAL-Marcus.doc> Acesso em: 08 Mar. 2008.

CARAYANNIS, E.G. (1998). **The strategic management of technological learning in projetct program management:the role of extranets, intranets and intelligrnt agents in knowledge generation, diffusion, and leveraging**. Technovation.

CARDOSO, I. A. P. (2000). Elaboração de políticas de manutenção: **uma abordagem voltada à análise de confiabilidade**. São Paulo: Escola Politécnica, Universidade de São Paulo, 189 p. (Dissertação de Mestrado).

CARLSON, D. (2002). Modelagem de aplicações XML com UML : **aplicações práticas de e-business**. São Paulo : Pearson Education do Brasil.

CARVALHO, R.B. (2000). **Aplicações de *Softwares* de Gestão do Conhecimento**: Tipologia e Usos. Belo Horizonte: Escola de Ciência da Informação, Universidade Federal de Minas Gerais (Dissertação de Mestrado).

CERVO, A. L. e BERVIAN, P. A. (2002). **Metodologia Científica**. São Paulo: Prentice Hall.

CHANG, G.; SU, C.C.; PRIEST, J.W. (2006). **CBR-DFMA: A case-based system used to assembly part design in the early design stage**. ASME Internacional Mechanical Engineering Congress and Exposition, IMECE 2006, Chicago, IL, United States, nov.

CHEN, X.; XU, N.; LI, Y. (2005). **A virtual environment for collaborative assembly**. Second International Conference on Embedded Software and Systems (ICESS'05), IEEE Computer Society, China.

CHIUSOLI, R. F. Z.; TOLEDO, J. C. (2000). Engenharia simultânea: **estudo de casos na indústria brasileira de autopeças.** In: CONGRESSO BRASILEIRO DE GESTÃO DE DESENVOLVIMENTO DE PRODUTO. São Carlos: Univesidade Federal de São Carlos, p.10-19.

CHUNG, L. (1993). **Representing and using non-functional requirements**: a process oriented approach. Thesis (PhD) - University of Toronto, Toronto, 1993.

CLARK, K. B.; FUJIMOTO, T. (1991) Product development performance: **strategy, organization and management in the world auto industry**. Harvard Business School Press, Boston, Massachussets, United States.

CLAUSING, D. (1994). Total Quality Development: **a step-by-step guide to world-class concurrent engineering**. The American Society of Mechanical Engineers, New York.

CLELAND-HUANG, J.; ZEMONT, G.; LUKASIK, W. (2004). **A heterogeneous solution for improving the return of investment of requirements traceability**. In: IEEE International Requirements Engineering Conference. Kyoto, Japão: [s.n.], p. 230-239.

COHEN, M. (1993). **Introducción a la lógica y al método científico**. Buenos Aires: Amorrortu.

CONECTT MARKETING INTERATIVO S.A. **Introdução a portais corporativos**. Disponível em : <http://i3.conectt.com.br/download/Doc2-IntroducaoaPortaisCorporativos.pdf> Acesso em : 24 Fev. 2008.

CONNOLLY, D. (1997). **XML principles, tools, and techniques**. Sebastopol, CA : O'Reilly & Assoc.

CORBETT, J.; DOONER, M.; MELEKA, J.;PYM, C.(1991). **Design for manufacture – Strategies, Principles and Techniques**. Addison – Wesley Publishing Company.

CYSNEIROS, L.M.; LEITE, J.C.S.P. (1999). Integrating non-functional requirements into data model. In: IEEE INTERNATIONAL SYMPOSIUM ON REQUIREMENTS ENGINEERING, 4., 1999, Ireland. **Proceedings…** New York: IEEE. P.162-171.

DAG, J. N.; et al. (2004). Speeding up requirements management in a product software company: **Linking customers wishes to product requirements through linguistic engineering.** In: International Requirements Engineering Conference. [S.I.: s.n.], p. 1-12.

DAG, J. N.; et al. (2005). **A linguistic-engineering approach to large-scale requirements management.** IEEE Software, v. 22, n. 1, p. 32-39, jan.

DAMIAN, D.E.H.; et al. (2000). **Using different communication media in requirements negotiation.** IEEE Software, v. 17, n. 3, p. 28-36, out.

DANE, F.C. **Research methods** (1990). California: Brooks/Cole Publishing Company.

DARDENNE, A.; LAMSWEERDE, A.; FICKAS, S. (1993). Goal directed requirements acquisition. **Science of Computer Programming**, Amsterdam, v.20, n.1/2, p.3-50, Apr.

DAVENPORT, T. H.; PRUSAK, L. (1998). **Conhecimento empresarial***:* como as organizações gerenciam seu capital intelectual. Rio de Janeiro: Campus.

DAVYDOV, M. M. EIP**: the second wave**. Disponível em : <http://www.intelligententerprise.com/000301/supplychain.shtml>. Acesso em: 30 Mar. 2008.

DEPARTAMENT OF DEFENSE (1997). United Sates: **MIL-HDBK-470A – Designing and developing maintainable products and systems**. Washington D. C. , Dod.

DESAI, A.; MITAL, A. (2003). **Evaluation of disassemblability to enable design for disassembly in mass production.** Internacional Journal of Industrial Ergonomics, Cincinnati, p. 1-17, v. 32, april

DIAS, A. (2003). **Projeto para confiabilidade aplicado ao processo de implantação de uma rede de gás.** Product Management &Development: Revista Brasileira de Gestão de Desenvolviento de Produtos, n. 2.

DIAS, C.A. (2001). Portal Corporativo: **conceitos e características**. Brasília: Universidade de Brasília, (Dissertação de Mestrado).

DHILLON, B.S. (1983). **Reliability engineering in systems design and operation**. New York: Van Nostrand Reinhold, 319 p.

DOYLE, R. L. (1996). Mechanical Reliability. **Design for reliability.** In IRESON, W. G.; COOMS, Jr., C. F. ; MOSRO, R. Y. **Handbook of Reliability Engineering and Management.** New York: MacGraw Hill, p.19.1-19.27.

DRISKILL, E.; COHEN, E. (1995). **Interactive design, analisys and illustration of assemblies**. Department of Computer Science, University of Utah, Salt Lake City, Utah.

DUARTE, M. D. (1997). **Caracterização da rotulagem ambiental de produtos.** Florianópolis: Universidade Federal de Santa Catarina (Dissertação de Mestrado).

DURÁN, A.; RUIZ, A.; TORO, M.(2001). **An automated approach for verification of software requirements.** In: JIRA'2001 Proceedings. Seville:[s.n.].

EASTER-BROOK, S.; NUSEIBEH, B. (1996). **Using viewpoints for inconsistency management engineering.** In: IEEE Software Engineering Journal, v. 11, n. 1, p. 31-43, jan.

ECKERSON, W. Business portals*:* **drivers, definitions, and rules**. <http://www.viador.com/pdfs/SeyboldWhitePaper.pdf>. Acesso em : 15 Abr. 2008.

ELLIS, C., GIBBS, S., REIN, G. (1991). Groupware**: Some Issues and Experiences**. Communications of the ACM.

ERTAS, A.; JONES, J. C. (1996). **The engineering design process.** New York: John Wiley & Sons, 614 p.

ESPECIFICAÇÃO DO XML – Disponível em:< http://www.w3.org/TR/REC-xml>.Acesso em : 29 Mar. 2008.

EUROPEAN ORGANIZATION FOR QUALITY CONTROL (2001). Notas de aula, Prof. Gilberto F. M. Souza, Confiabilidade de produtos e sistemas, curso de mestrado de Engenharia Automotiva, Escola Politécnica da Universidade de São Paulo.

FICKAS, S.; LAMSWEERDE, A. van; DARDENNE, A. (1991). **Goal-directed concept acquisition in requirements elicitation.** In: 6th International Workshop on Software Specification and Design. Como, Italy: [s.n.], p. 14-21.

FINKELSTEIN, A.; KRAMER, J.; GOEDICKE, J.K.(1990). **Viewpoints oriented software specification.** In: IEEE COMPUTER SOCIETY. 3rd International Workshop on Software Engineering and its Applications. Toulouse, France, p. 337-351.

FINOTTI, M. B. ; PERUSSO, A. O. ; PEIXOTO, M. O. C. ; GONÇALVES FILHO, E. V. (1999). DFMA em ação: **a metodologia aplicada no desenvolvimento de uma secadora centrifuga.** In: XV Congresso Brasileiro de Engenharia Mecânica, Águas de Lindóia.

FOWLER, M. (2000) UML essencial: **um breve guia para a linguagem padrão de modelagem de objetos**. Porto Alegre: Bookman.

FRANCISCO, M.; GIANNETI, B. F.; ALMEIDA, C. M. V. B.**Ecologia Industrial: Projeto para o meio ambiente**.Disponível em: < http://www.hottopos.com/regeq12/art5.htm>. Acesso em: 28 ago. 2008.

FURLAN, J. D. (1998) Modelagem de objetos através da UML: **The Unified Modeling Language**. São Paulo: Makron.

GIL, A. C. (1991). **Como elaborar projetos de pesquisa**. São Paulo: Atlas.

GIL, A.C. (1999). **Métodos e técnicas de pesquisa social**. São Paulo: Atlas.

GOLDFARB, C. F. (1998). **The XML handbook** .Upper Saddle River, NJ : Prentice Hall.

GOLDFARB, C.F.; PRESCOD, P. (1999). **The XML handbook.** Prentice Hall PTR Upper Saddle River, NJ, USA.

GRANT, R.M. (1996) **Toward a knowledge-based theory of the firm**. Strategic Management Journal, p. 109-122.

GROOVE. Groove networks. Disponível em: <http://www.groove.net/> Acesso em: 28 jan. 2008.

GUNTER, C. A. et al. (2000). **A reference model for requirements and specifications.** IEEE Software, v. 17, n. 3, p. 37-43, may.

HACKOS, J. T., REDISH, J. C.(1998) **User and task analysis for interface design**. Computer Publishing.

HAGGE, L.; LAPPE, K. (2005). **Sharing requirements engineering experience using patterns.** In: IEEE Software, v. 22, n. 1, p. 24-31, jan.

HARRISON, L. A.; BLOUNT , G. N. (2000). Business Model Approach: **Design "Versus" Economic Considerations of Automative Recycling.** SAE Technical Paper Series, Detroit, p. 1-8, march.

HAYES, J. H. ; et al. (2004). Helping analysts trace requirements: **An objective look**. In: International Requirements Engineering Conference. Kyoto, Japão: IEEE Computer Society, p. 249-259.

HOFFMANN, M. ; et al. (2004). **Requirements for requirements management tools**. In: IEEE International Requirements Engineering Conference. Kyoto, Japão: IEEE, p. 301-308.

HUANG, G.Q.; MAK, K.L. (1999). **Design for manufacture and assembly on the internet**. Elsevier – Computers in industry , v. 38, p. 17-30.

IBM RATIONAL. RequisitePro. Disponível em: http://www-306.ibm.com/software/adwtools/reqpro/ . Acesso em: 20 jul. 2008.

JACOBSON, I. et al.(1992). Object-Oriented Software Engineering: **A use case driven approach**. [S.I.]: Addison-Wesley.

JAYARAM, S. et al (2006). **A virtual assembly design environment.** School of Mechanical and Materials Engineering, Washington State University.

JAYARAM, S.; CONNACHER, H.I.; LYONS, K.W.(1996). **Virtual assembly using virtual reality techniques**. Elsevier - Computer-Aided Design, v. 29, n.8, p. 575-584.

KAMINSKI, P.C. (2000). **Desenvolvendo produtos com planejamento, criatividade e qualidade.** Rio de Janeiro: Livros Técnicos e Científicos Editora S.A.

KERLINGER, F. (1980). **Metodologia da pesquisa em ciências sociais**. São Paulo: EPU.

KÖCHE, J. C. **Fundamentos de Metodologia Científica**. Petrópolis: Vozes, 1997.

KOTONYA, G.; SOMMERVILLE, I. (1995). **Requirements Engineering With Viewpoints.** Reino Unido.

KOWALSKI, R. (2000). Maintainability and Reliability.In IRESON, W. G.; COOMS, Jr., C. F. ; MOSRO, R. Y. **Handbook of Reliability Engineering and Management.** New York: MacGraw Hill, p.22.1-23.1.

KUO, T. C.; HUANG, S. H.;ZHANG, H. C. (2001). Design for manufacture and design for "X": **concepts applications and perspectives.** Computer & Industrial Engineering, p. 1-20, v. 41, may.

LAFRAIA, J. R. B. (2001) **Manual de confiabilidade, mantenabilidade e disponibilidade.** Rio de Janeiro: Qualitymark Petrobrás , 388 p.

LAMSWEERDE, A. (2001). Goal-oriented requirements engineering: **A guided tour**. In: 5th International Symposium on Requirements Engineering. Toronto, EUA: [s.n.], p. 249-263.

LAUDON, K.C., LAUDON, J.P.(1999). **Sistemas de informação**. Rio de Janeiro: LTC.

LEITCH, R. D. (1995). **Reliability analysis for Engineers – An introducion**. New York: Oxford Science Publications.

LEITE, J.C. (1998). **Modelos e formalismos para engenharia semiótica de interfaces de usuários**. Tese (Doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 1998.

LEITE, J. C. do P. (1989). Viewpoint analysis: **A case study**. ACM J. Software Engineering Notes, v.14, n. 3, p. 111-119.

LEITE, J.C.S. do P.; FRANCO, A.P.M. (1993). **A strategy for conceptual model acquisition.** In: Proceedings of IEEE International Symposium on Requirements Engineering. [S.I.: s.n.], p. 243-246.

LEITE, J. C. S. do P. ; FREEMAN , P. A. (1991). **Requirements validation throgh viewpoint resolution.** IEEE Transactions on Software Engineering, v. 17, n. 12, p. 1253-1269, dez.

LEWIS, E.E. (1987). **Introduction to reliability engineering.** New York: John Wiley & Sons, 435 p.

LIANG, J. S.; WEI, P.W. (2006). **Conceptual design system in a Web-based virtual interactive environment for product development. London:** Springer-Verlag, nov.

LIGHT, R. B.(1999) **Iniciando em XML**. São Paulo : Makron Books.

LOCONSOLE, A. (2004). **Empirical studies on requirements management measures**. In: Doctoral Symposium, International Conference on Software Engineering. [S.I.]: IEEE Computer Society, p. 42-44.

LOHN, L.L. (2000) **Designing the instrucnional interface**. Computers in human behavior, p. 161-182.

LOUGHBRIDGE, M.E.D (1996). **Intellectual capital and knowledge management**. IFLA Journal, p. 299-301.

MACAULY, L. (1995). **Human computer interaction for software designers**. Internacional Thomson Computer Press.

MACEDO, A.A. (1999) **Explorando tecnologias hipermídia e de trabalho cooperativo em um ambiente de apoio ao ensino**. São Carlos: Instituto de Ciências Matemáticas e de Computação,Universidade de São Paulo (Dissertação de Mestrado).

MARCHAL, B.(2000). XML **: Conceitos e aplicações**. São Paulo : Berkeley Brasil.

MARCHIONINI, G. (1995). **Information seeking in electronic environments**. Australia: Cambridge.

MARTINS, G. A. (1994). Manual para Elaboração de Monografias 2ª. ed. São Paulo: Ed. Atlas.

MÁTTAR NETO, J. A. (2002). **Metodologia científica na era da informática**. São Paulo: Saraiva.

MCGUIGAN, C. A. (1975). **The add-a-word spelling program** (Working Paper No. 53). Experimental Education Unit, Seattle, WA: University of Washington.

MCKINSEY & COMPANY. Disponível em: http://www.mskinsey.com/mgi/publications/index.asp#brtop. Acesso em: 21 de Abr de 2008.

MEERKAMAMM, H. (1994)**. Design for X – A core area of design methodology.** Journal of Engineering Design, vol. 5, n. 2, p. 145-163.

MORAN, T. (1981). The Command language grammars: a representation for the user interface of interactive computer systems. International Journal of Man-Machine Studies, London, v.15, p.3-50.

MOSS, R. (1996). **Design for reliability.** In IRESON, W. G.; COOMS, Jr., C. F. ; MOSRO, R. Y. **Handbook of Reliability Engineering and Management.** New York: MacGraw Hill, p.5.1-5.16.

MOUBRAY, J. (1997). **Reliability-centered maintenance.** New York: Industrial Press, 440 p.

MURRAY, G. **The portal is the desktop**. Disponível em: http://www.groupcomputing.com/Back_Issues/1999/MayJune1999/mayjune1999.html. Acesso em: 20 Mar. 2008.

MYLOPOULOS, J.; CHUNG, L.; YU, E. (1999). **From object-oriented to goal oriented requirements analysis.** Communications of the ACM, v.42 , n. 1, p. 31-37, jan.

NARAYANASWAMY, K.; GOLMAN, N. (1992). Lazy consistency: **A basis for cooperative software development**. In: International Conference on Computer-Supported Cooperative Work. Toronto, Ontario, Canada: [s.n.], p. 257-264.

NONAKA, I., TAKEUCHI, H. (1997). **Criação de conhecimento na empresa**. Rio de Janeiro: Campus.

NONAKA, I. (1994). **A dynamic theory of organizational knowledge creation**. Organization Science, p. 14-37.

NORMAN, D. (1986). Cognitive engineering. In: NORMAN D.A.; DRAPER, W. (Ed.). **User centered system design**: new perspectives on human-computer interactions. Hillsdale: Lawrence Erlbaum. p.31-61.

NIELSEN, J. (1997). Something is better than nothing. **IEEE Software**, Lo Alamitos, v.14, n.4, p.27-28, July/Aug.

_____. (2002). Affordances and design. Disponível em:< http://www.jnd.org

/dn.mss/affordances-and-design.html>. Acesso em: 20 June 2008.

NUNAMAKER, J.F. (1991). **Eletronic meeting sistems to support group work**. Communications of the ACM.

OLIVEIRA, T. C. (2001). **Using XML and frameworks to develop information systems**. Rio de Janeiro : PUC.

OMOKAWA, R. (1999). Utilização de sistemas PDM em ambientes de engenharia simultânea**: o caso de uma implantação em uma montadora de veículos pesados.** São Carlos: Escola de Engenharia de São Carlos – Universidade de São Paulo (Dissertação de Mestrado).

PADUA, E. M. M. (1996). **Metodologia da pesquisa**: abordagem teórico-prática. Editora Papirus.São Paulo.

PAHL, G. & BEITZ, W. (1996). Engineering design: **a systematic approach** – 2Rev.ed. Springer-Verlag London Limited, London, Great Britain.

PIMENTEL, M. G. C. (2001). **Hiperdocumentos XML na web.** São Carlos : ICMC/USP.

PRASAD, B. (1996) Concurrent engineering fundamentals: **integrated product and process organization**. New Jersey, United States, Prentice Hall.

_____. (1997) Concurrent engineering fundamentals – **volume II: integrated product development**. New Jersey, United States, Prentice Hall.

PRESSMAN, R.S. (1987). **Projeto de interface com o usuário**. London: Elsevier Applied Science. p.601-632.

PUGH, S. (1990). Total design: **integrated methods for successful product engineering**. Addison Wesley, London, United Kingdom.

RAMESH, B.; JARQUE, M. (2001). **Toward reference models for requirements traceability**. IEEE Transactions on Software Engineering, v. 27, n. 1, p. 58-93, jan.

RAY, E. T. (2001). **Aprendendo XML**. Rio de Janeiro : Campus.

RATIONAL SOFTWARE CORPORATION. **UML Resource Center**. Disponível em: <http://www.rational.com>. Acesso em: 22 Jan. 2008.

REIFER, D.J. (2000).Requirementes management: **The search for nirvana**. IEEE Software, v.17, n. 3, p. 45-47, may.

RIBBENS, J. (2000). **Simultaneous Engineering for new product development – Manufacturing Applictions.** John Wiley &Sons. Inc, 332p.

RIDAO, M.; DOORN, J.; LEITE, J.C.S.do P. (2000). **Uso de patrones en la construcción de escenarios**. In: WER. [S.I]: WER, p. 140-157.

RODRIGUES, A.A. (2002). **Aplicação da metodologia design for manufacturability and assembly na indústria automobilística.** São Paulo: Escola Politécnica (Dissertação de Mestrado).

RODRIGUES, M. V. (2001) Gestão do Conhecimento: **reinventando a empresa para uma sociedade baseada em valores intangíveis**. Rio de Janeiro: IBPI Press.

ROY, U.; KODKANI, S.S. (1998). **Collaborative product conceptualization tool using web technology.** Elsevier – Computers in Industry, v. 41, p. 195-209, sep.

ROZENFELD, H.; AMARAL, D.C.; TOLEDO, J.C.; CARVALHO, J. (2000) O processo de desenvolvimento de produtos e processos na fábrica do futuro. In.

ROZENFELD, H.; VEGA, H. A. (1995). **Ambiente distribuído de soluções para suportar a engenharia simultânea.** Máquinas e metais, março, p. 211-223.

RUBENSTEIN-MONTANO, B.; LIEWBOWITZ, J.; BUCHWALTER, J., McCAW; NEWMAN, B.; REBECK, K. (2001). **A systems thinking framework for knowledge management**. Decision support systems.

RUS, I. , LINDVALL, M. (2002). Knowledge Management in Software Engineering. **IEEE Software**, p. 26-27, May-June.

SCHOECH, H. ; FLORIN, H.; KREISSIG, J. ; EYERER, P. (2000). **LCA Based Design for Environment in the Automative Industry.** SAE Technical Paper Series, Detroit, p. 1-7, april.

SANCHES, R., HEENE, A., THOMAS, H. (1996). Towards the theory and pratice of competence-based competition. In: SANCHES, R., HEENE, A., THOMAS, H. (Ed.). **Dinamics of competence-based competition**. Oxford: Elsevier.

SCHEER, A. V. (1998). ARIS: **business process modeling**. Berlin: Springer-Verlag.

SHINGO, S. (1988). **Poka-Yoke – Improving Product Quality by Preventing Defects.** Cambrige: Productivity Press, INC.

SHINGO, S. (1986). Zero Quality Control: **Source Inspection and Poka-Yoke System.** Cambrige: Productivity Press, INC.

SCHMIDT, S. (1998). Preventive Optimisation of Costs and Quality for the Total Life Cycle - **Design for manufacture, assembly, service, environment (DFMA).** SAE Technical Paper Series, Graz, p. 1-9, dec.

SELKER, T.(1996). New paradigms for using computers. **Communications of the ACM**, Volume 39, Issue 8.

SHYAMSUNDAR, N.; GADH, R. (2001). **Internet-based collaborative product design with assembly features and virtual design spaces.** Elsevier – Computer-Aided Design, v. 33, p. 637-651, oct.

SHULL, F,;RUS, I.; BASILI,V.(2000**). How perspective-based reading can improve requirements inspections.** IEEE Software, v. 33, n. 7, p. 73-79, jul.

SILVA, E. L.; MENEZES, E. M. (2000) **Metodologia da pesquisa e elaboração de dissertação**. Florianópolis: Laboratório de Ensino a Distância da UFSC.

SILVA FILHO, A. M. (2004). **Programando com XML** .Rio de Janeiro : Elsevier/Campus.

SOFFNER, R. **Curso sobre Gestão do Conhecimento**. Disponível em : <http://www.soffner.eng.br>. Acesso em: 16 Nov. 2008.

SOMMERVILLE, I.(1996). **Software Engineering**. 5 ed. Massachussets: Addison-Wesley.

SOMMERVILLE, I.;SAWER, P. (1997).Viewpoints: **principles, problems and pratical approach to requirements engineering.** Annals of Software Engineering, v.3, p. 101-130.

SOUZA, G. F. M. (2003). **Análise de confiabilidade aplicado ao projeto de sistemas mecânicos.** São Paulo: Escola Politécnica da Universidade de São Paulo.

SPENCER, P. (1999). XML design and implementation : **programming with XML, ASP, and IE5**. Olton, Birmingham, England : Wrox.

SOUSA, A. G. (1998). **Estudo e Análise dos Métodos de Avaliação da Montabilidade de Produtos Industriais no Processo de Projeto.** Florianópolis: Universidade Federal de Santa Catarina (Dissertação de Mestrado).

STONE, R.B.; MCADAMS, D.A.; KAYYALETHEKKEL, V.J. (2004). **A product architecture-based conceptual DFA technique**. Elsevier – Design Studies, v. 25, p. 301-325.

SWAMINATHAN, A.; BARBER, K.S. (1995). APE : An Experience-based Assembly Sequence Planner for Mechanical Assemblies. In: iEEE International Conference on Robotics and Automation. University of Texas at Austin, Austin, Texas.

TELELOGIC. DOORS. Disponível em:< http://www.telelogic.com/products/doorsers/doors/infex.cfm> Acesso em: 20 jul. 2008.

TERRA, J.C.C. (2002). **Portais corporativos: a revolução na gestão do conhecimento**. São Paulo: Negócio.

THIOLLENT, M. (1998). **Metodologia da pesquisa-ação**. São Paulo: Cortez.

TIWANA, A. (2000). **The knowledge management toolkit**. Upper Saddle River: Prentice-Hall.

TORO, A.D. et al. (2000). **Identificación de patrones de reutilización de requisitos de sistemas de información.** In: WER2000. [S.I.: s.n.].

ULIERU, M. et al (2000). **A multi-resolution collaborative architecture for web-centric global manufacturing.** Elsevier – Information sciences, v. 127, p. 3-21.

VALERI, S. G., ROZENFELD, H.(2000). **Aplicação do Processo de Revisão Gerencial de Fases em uma Indústria Automotiva no Brasil.** São Carlos: Escola de Engenharia de São Carlos, Universidade de São Paulo (Dissertação de Mestrado).

VIEGAS, W. (1999). **Fundamentos de metodologia científica**. Brasília: EdUnB/Paralelo 15.

WANG, Q.; LAI, X. (2001). **Requirements management for the incremental development model.** In: Asia-Pacific Conference on Quality Software. [S.I.]: IEEE, p. 295-301.

WHITE, C. **The enterprise information portal marketplace**. Disponível em :< http:// www.decisionprocessing.com/papers/eip1.doc>. Acesso em: 20 Abr. 2008.

WHITE, C. **Decision Threshold**. Disponível em : < http:// www.intelligententerprise.com/991611/feat1.shtml>. Acesso em: 20 Abr. 2008b.

WIERINGA, R. J. (1995). Requirements Engineering: **Framework for understanding**. [S.I.]: Wiley.

XU, Y. et al (2006). **A collaborative virtual environment for real time assembly design**. Association for Computing Machinery, Hong Kong, p. 14-17, Jun.

YIN, R. K. (1994). **Case study research**: design and methods. USA: Sage Publications.

YOU, C.F.; TSOU, P.J.; YEH, S.C. (2006). **Collaborative design for an assembly via the internet**. Springer-Verlag: London, jan.

ZANLORENCI, E. P. ; BURNETT, R. C. (2000). REQAV: **Modelo para descrição, qualificação, análise e validação de requisitos.** In: IDEAS2000: Jornada Ibero Americana de Ingeneria de Requisitos Y Ambientes de Software. [S.I.:s.n.], p. 61-72.

ZYLBERMANN, D.; COHEN, Y.; GOLDIN, L. (2003). **The road to requirements maturity**. In: IEEE International Conference on Software – Science , Technology & Engineering (SwSTE'03).[S.I.]: IEEE Computer Society, p. 71-78.

7 ANEXOS

7.1 ANEXO A – ESTRUTURA DO XML

```
<empregados>
  <empregado>
    <nome>Maria Aparecida</nome>
    <endereco>Rua Terra, 13<;/endereco>
  </empregado>
  <empregado>
    <nome>Jonas Silva</nome>
    <endereco>Av. Afonso Pena, 1305<;/endereco>
  </empregado>
  <empregado>
    <nome>Carlos Roberto</nome>
    <endereco>Rua Taberna, 200<;/endereco>
  </empregado>
</empregados>
```

Um XML exemplo seria:

```
<?xml version="1.0" encoding="ISO-8859-1">
<?xml-stylesheet type="text/xsl" href="stylesheet.xsl"?>
<!-- Empregados da Empresa -->
<empregados>
  <empregado>
    <nome idade="26">Maria Aparecida</nome>
    <endereco>Rua Terra, 13</endereco>
    <estagiario/>
  </empregado>
```

</empregados>

Declaração XML

De acordo com SILVA FILHO (2004), a declaração XML no começo de um documento XML não é obrigatória, mas é a melhor forma de dizer a um processador que o documento trata-se definitivamente de um documento XML.

A sintaxe da declaração XML é a seguinte:

<?xml version="1.0" standalone="yes/no" encoding=="ISO-8859-1"

- Dos seus três atributos, *version* é o único obrigatório e deve ter o valor "1.0" (a versão 1.1 deverá sair em breve);
- O atributo *standalone* especifica se outros arquivos são necessários para renderizar o documento, tais como Document Type Definition (DTD) ou XML Schemas. Já o atributo encoding especifica o conjunto de caracteres em uso no documento. Alguns dos valores mais comuns para o atributo são:
  - *ISO-8859-1* - Alfabeto latino 1. Usado para a maioria das linguagens do oeste europeu. Similar ao ASCII de 8 bits. Permite o uso de acentuação.
  - *UTF-8* - Unicode de 8 bits.
  - *UTF-16* - Unicode de 16 bits usado para conjuntos de caracteres internacionais.

Outros possíveis valores para o atributo *encoding* são ISO-10646-UCS-2, ISO-10646-UCS-4, ISO-8859-1 to ISO-8859-9, ISO-2022-JP, Shift_JIS, EUC-JP.

7.2 ANEXO B - ARTIGOS PUBLICADOS

SAVI, A.F.; GONÇALVES FILHO, E. V.; SAVI, E. M. S. **Engineering and the sustainable development**. In: Environmental and Health Would Congress 2006, Santos - SP.

SAVI, A. F.; GONÇALVES FILHO, E. V.; SAVI, E. M. S. **Development a support solution knowledge management: an integration of information of DFA (Design for Assembly)**. In: Global Congress on Manufacturing and Management, Santos, 2006.

SAVI, A. F.; GONÇALVES FILHO, E. V.; SAVI, E. M. S. **Rules of storage of information of DFA (Design for Assembly) based on cases (CBR – case-based reasoning).** In: Global Congress on Manufacturing and Management, Santos, 2006.

SAVI, A.F.; GONÇALVES FILHO, E. V.; SAVI, E. M. S. **Desenvolvimento sustentável : a engenharia atendendo a necessidade das atuais gerações sem comprometer a capacidade das futuras em satisfazer as suas necessidades**. In: II Congresso Brasileiro de Sistemas, Ribeirão Preto, 2006.

SAVI, A.F.; GONÇALVES FILHO, E. V.; SAVI, E. M. S. **Sistema de apoio à Gestão do Conhecimento em abordagens de DFA – *Design for Assembly***. In: II Congresso Brasileiro de Sistemas, Ribeirão Preto, 2006.

SAVI, A.F.; GONÇALVES FILHO, E. V.; SAVI, E. M. S. **Gerenciando o conhecimento referente a DFA (*design for assembly*) utilizando regras baseadas em casos (RBC)**. In: II Congresso Brasileiro de Sistemas, Ribeirão Preto, 2006.

SAVI, A.F.; GONÇALVES FILHO, E. V.; SAVI, E. M. S. **The Project for Manufacture and Assembly (DFMA) and the Accounting of Costs as elements key for the improvement of the product**. In: IV Global Conference on Sustainable Product Development and Life Cycle Engineering, São Carlos, SP, 2006.

SAVI, A.F.; GONÇALVES FILHO, E. V.; SAVI, E. M. S. **Engineering and Sustainable Development**. In: IV Global Conference on Sustainable Product Development and Life Cycle Engineering, São Carlos, SP, 2006.

SAVI, A.F.; GONÇALVES FILHO, E. V.; SAVI, E. M. S. **Gerenciamento de conhecimentos explícitos em abordagens de DFA – *Design for Assembly*** In: I WOPEPRO – Workshop de pós-graduação em Engenharia de Produção, São Carlos, SP, 2006.

SAVI, A.F.; GONÇALVES FILHO, E. V.; SAVI, E. M. S. **Engenharia apoiando o desenvolvimento sustentável**. In: XIII SIMPEP - Simpósio de Engenharia de Produção, Bauru, SP, 2006.

7.3 ANEXO C - METODOLOGIA DE DESENVOLVIMENTO DE *SOFTWARE*

O padrão de proposta de desenvolvimento de *software* especifica que itens devem fazer parte de produtos que serão desenvolvidos.

O conteúdo da proposta é mostrado na Figura 7-1, juntamente com o tipo de documento e a ferramenta utilizada para criá-lo. A Figura 7-1 mostra somente os aspectos técnicos da metodologia sendo que, aspectos gerenciais como gestão de projetos e gestão de qualidade serão futuramente abordados.

A seguir serão detalhados os itens existentes na Figura 7-1.

- Especificação
  - **Escopo do sistema**
    - Nome do sistema e dos componentes principais;
    - Descrição do sistema: descreve-se, usando português estruturado, um texto detalhando o sistema;
    - Missão do sistema: descreve os objetivos do produto que deverá ser desenvolvido no projeto;
    - Lista de funções: as necessidades que se pretende atender e os benefícios esperados desdobrados por função;
  - **Definição de requisitos**
    - Requisitos: visa à captura das necessidades dos usuários em relação ao produto;
    - Casos de uso: é uma técnica para capturar informação sobre como um sistema ou negócio trabalha, ou de como se deseja que trabalhe. Não pertence estritamente ao enfoque orientado a objeto, é uma técnica para captura de requisitos;
    - Descrição de telas: é gerado em HTML estático e descrevem as principais telas do sistema. Através dos requisitos e dos casos de uso é possível desenvolver essa descrição de telas que, seria como a primeira versão de um protótipo sem acesso ao banco de dados;

- **Estrutura de banco de dados**
  - Modelo entidade relacionamento: é o modelo mais usado atualmente para projeto conceitual de banco de dados. Tem por base de que o mundo real é formado por um conjunto de objetos chamado entidade e pelo conjunto de relacionamentos entre essas entidades;
- Definição: neste item define-se qual será a linguagem de programação, o banco de dados e algumas convenções de programação que serão utilizadas na fase de implementação. Como convenções de programação são descritos:
  - Diretórios: formato de nome e organização de pastas (ou diretórios) na máquina onde será desenvolvido o sistema;
  - Nomes de páginas e funções: define-se o formato de nomes de páginas *Web* e funções usadas no desenvolvimento do sistema;
  - Siglas: define-se formato de siglas para nomes longos (de páginas, funções, figuras)e para variáveis utilizadas no sistema;
  - Glossário: é construído um glossário de todas as definições citadas acima para melhor entendimento do código gerado. Assim uma pessoa que não participa do desenvolvimento poderá entender o formato de nomes, siglas, etc.
- Implementação: essa fase usa todas as outras fases como base para a implementação do sistema. Aqui é construído o primeiro protótipo e obtém-se a versão final utilizando o clássico modelo de desenvolvimento de *software* em espiral, assim a cada iteração obtém-se uma versão mais completa do sistema;
- Teste: nesta fase são feitos os testes com o produto desde a primeira versão. Todas as fases são aplicadas a cada iteração do modelo em espiral de desenvolvimento de *software*. Desde a especificação até o teste, a cada iteração da espiral o sistema é aprimorado.

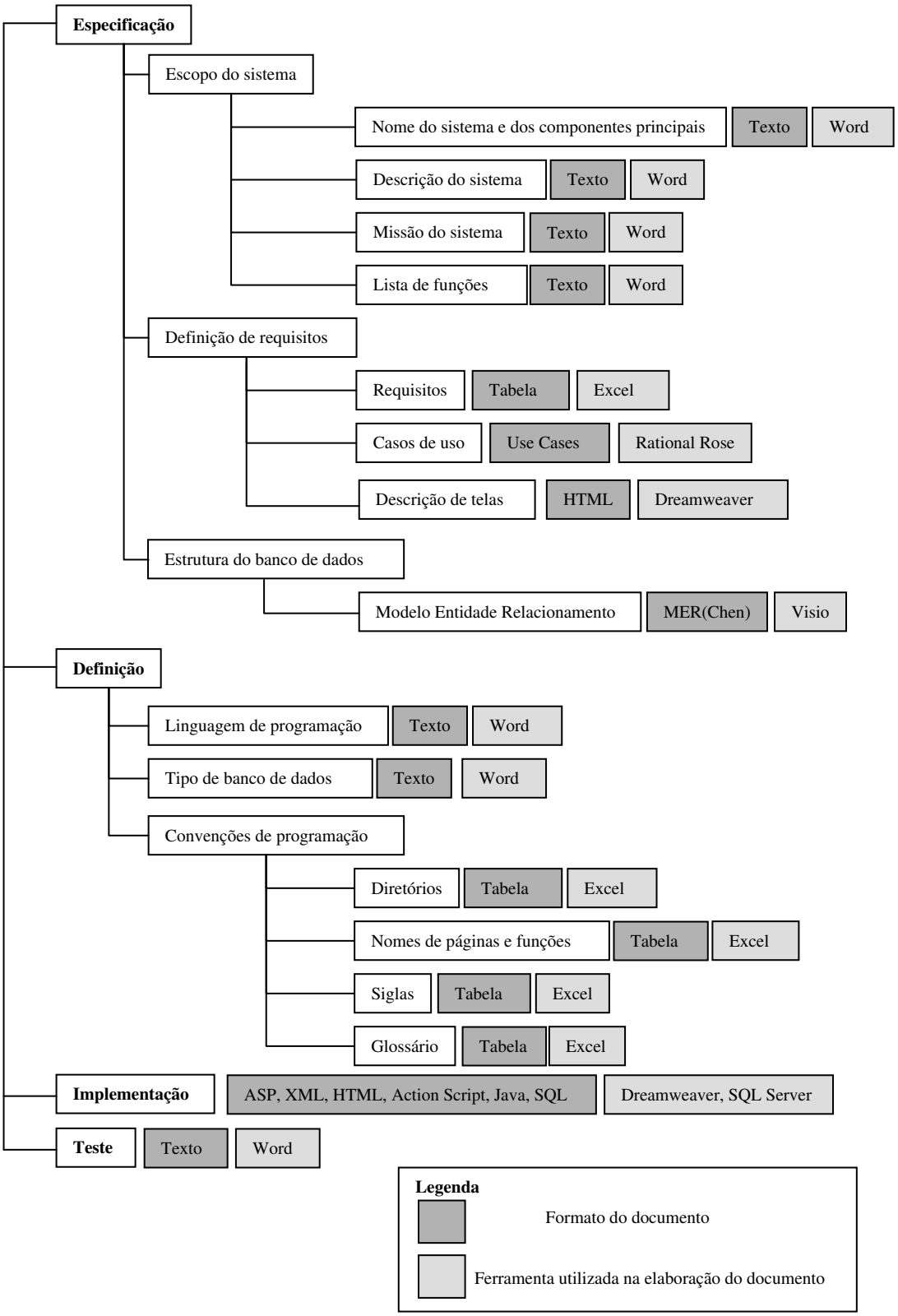


Figura 7-1 - Metodologia de desenvolvimento de software