UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

OTTO ROBERT LESSING

PROPOSTA DE UM MODELO PARA AUXÍLIO NA TOMADA DE DECISÃO EM ENGENHAIRA DE SISTEMAS

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação

LUIZ FERNANDO J. MAIA

Florianópolis, Dezembro de 2001.

PROPOSTA DE UM MODELO PARA AUXÍLIO NA TOMADA DE DECISÃO EM ENGENHARIA DE SISTEMAS

OTTO ROBERT LESSING

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação Área de Concentração Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Prof. Fernando A. O. Gauthier, Dr.
- Coordenador -

Banca Examinadora

Prof. Luiz Fernando J. Maia, Dr.
(orientador)

Prof. João Bosco da Mota Alves, Dr.

Prof. Vitório Bruno Mazzola, Dr.

Prof. Cristiane A. Gresse Wangenheim, Dra.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

UML	-	Linguagem de Modelagem Unificada
RBC	-	Raciocínio Baseado em Casos
SGBD	-	Sistema Gerenciador de Banco de Dados
SAD	-	Sistemas de apoio à decisão

RESUMO

Este trabalho apresenta a proposta de um modelo para auxilio na tomada de decisão em Engenharia de Sistemas, para determinar qual o melhor ou os melhores produtos de software que estão disponíveis no mercado de acôrdo com a necesssidade do usuário.

Como pôde ser comprovado pelo modelo desenvolvido neste trabalho, o usuário mediante uma determinada necessidade pode acessar o modelo e através de uma pesquisa direta poderá ter acesso a uma tecnologia específica fornecendo as informações básicas do produto a ser consultado, bem como, no caso de uma pesquisa indireta, o usuário informará suas necesidades, e em cima deste contexto o modelo retornará qual o tipo de tecnologia ou tecnologias que mais se adaptam à sua realidade no momento da consulta ao software.

O software leva em consideração as características da norma ISO/IEC 9126-1 (NBR 13.596-4) como a funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade, bem como, também, leva em consideração, suas subcaracterísticas procurando identificar e mensurar dentro deste contexto, os atributos específicos a cada subcaracterística.

ABSTRACT

This work presents the proposal of a model for aid in the taking of decision in Engineering of Systems, to determine which the best or the betther software products than they are available in the market in agreement with the user's need.

As it can be checked by the model developed in this work, the user by means of a certain need cannot access the model through a direct research he can have access to a specific technology supplying the basic information of the product to be consulted, as well as, in the case of an indirect research, the user will inform its needs, and on top of this context the model will come back which the technology type that more they adapt to its reality in the moment of the consultation to the software.

The software takes in consideration the characteristics of the norm ISO/IEC 9126 (NBR 13.596) as the functionality, reliability, usufully, efficiency, maintenance and portability, as well as, also, it takes in consideration, its sub-características trying to identify and measure inside of this context, the specific attributes to each sub-características.

1. INTRODUÇÃO

Durante décadas o desenvolvimento de software tem-se concentrado nos aspectos estruturais e de funcionamento do produto. A interface de utilização, ou Interface Homem-Computador, usualmente tem seu projeto significativamente influenciado pelo funcionamento. Com bastante frequência, o projeto da interface de utilização de produtos prioriza a facilidade de implementação ante a facilidade de utilização (KANO, 1984).

O contexto em que softwares estão sendo desenvolvidos está estritamente ligado a quase cinco décadas de evolução dos sistemas computadorizados, onde se tem observado as mais variadas tecnologias sendo investigadas e desenvolvidas para auxiliar usuários na tomada de decisão em Engenharia de Sistemas. Entretanto as técnicas de projeto centrado no usuário são até hoje pouco adotadas pelos produtores de software, especialmente no Brasil.

Desta forma, o resultado imediato desta abordagem é a frustração dos usuários diante da dificuldade de obter no software o apoio necessário para a execução de suas tarefas, o que, em situações extremas, leva ao abandono do produto.

A atenção crescente para estas questões tem levado à discussão de padrões internacionais de ergonomia, incluindo padrões normativos que permitam ser possível, à comunidade de desenvolvedores e usuários, instrumentalizar a avaliação das qualidades operacionais mínimas dos produtos de software. Um destaque importante nesta área é a norma internacional ISO/IEC 9126-1 (NBR 13.596-4) que fornece seis características subdivididas em subcaracterísticas as quais descrevem, com um mínimo de sobreposição, a qualidade de software. Tais caracteristicas são aplicáveis a qualquer tipo de software. A norma supracitada está voltada para as pessoas relacionadas à aquisição, desenvolvimento, uso, avaliação, suporte, manutenção ou auditoria de software.

Um fator que agrava ainda mais esta situação é a tendência de aumento da disponibilização de novas tecnologias, complementando com a existência de uma

grande dificuldade por parte dos estudiosos da área em se reciclar com estas novas tecnologias que estão sendo disponibilizadas, levando-se em consideração os recursos necessários.

Esta discussão pode vir a auxiliar sobremaneira as empresas que desenvolvem software, bem como os usuários que são os principais atingidos na sua tomada de decisão.

Levando em consideração o exposto, ressalta-se o tema central desta dissertação: a proposta de um método de modelagem para auxílio na tomada de decisão em Engenharia de Sistemas.

1.1. JUSTIFICATIVAS

A presente pesquisa fundamenta-se pelos seguintes motivos:

- Modelar ajuda a projetar corretamente o sistema e garantir a qualidade da solução escolhida entre outras possíveis soluções alternativas;

- Permitir a mensuração de softwares nos aspectos quantitativos, qualitativos e seus fatores de influência potencial;

- Buscar soluções que possam auxiliar na solução dos problemas apresentados;

- Clarear a forma como as tecnologias estão dispostas no mercado; possíveis soluções a que se propõem e como interagem.

1.2. OBJETIVOS

O objetivo deste trabalho é propor um modelo para auxílio à tomada de decisões na Engenharia de Sistemas, possibilitando pesquisar diretamente a tecnologia desejada ou através de pesquisa indireta obter a tecnologia que mais se adapta ao seu ambiente de desenvolvimento.

Mais especificamente, pretende-se:

- Aprofundar o conhecimento através de estudo em métodos de modelagem;
- Pesquisar a maneira como as diversas tecnologias de software se relacionam;
- Investigar bancos de dados baseados em conhecimentos;
- Apresentar um modelo para auxílio na tomada de decisão em Engenharia de Sistemas.

1.3 METODOLOGIA DE TRABALHO

O trabalho seguirá as etapas de estudo descritas abaixo:

- Investigar tecnologias na área de software, buscando através de suas características e subcaracterísticas mensurar os aspectos relevantes a cada uma delas;
- Estudar e aprofundar conceitos relativos a Banco de Dados baseados em conhecimentos para desenvolver e implementá-lo em uma segunda etapa na modelagem;

- Pesquisar os métodos existentes de modelagem e propor dois modelos, para auxilio na tomada de decisão em Engenharia de Sistemas, bem como avaliar o comportamento da sua modelagem.

1.4. ESTRUTURA DO TRABALHO

Esta dissertação está assim dividida:

Capitulo 1: Contém a introdução, justificativa, objetivos, metodologia e a estrutura do trabalho.

Capitulo 2: serão apresentadas diversas abordagens que tratam da avaliação de softwares; fundamentos sobre qualidade; qualidade de softwares; normas técnicas ISO de avaliação da qualidade; mensuração de softwares; importância da equipe na modelagem e desenvolvimento de softwares. O Objetivo deste capítulo é evidenciar a importância do desenvolviemtno do software com qualidade;

Capítulo 3: Aborda os diversos Bancos de Dados existentes, procurando dar uma maior enfâse ao Banco de Dados Baseado em Raciocinio, onde, posteriormente, numa segunda etapa, será incluído no modelo para auxílio na tomada de decisão em Engenharia de Sistemas;

Capítulo 4: Proposta de um modelo lógico desenvolvido em Análise Estruturada, descrevendo o Modelo Entidade-Relacionamento, juntamente com suas chaves, atributos e o dicionário de dados. Apresenta também as vantagens e desvantagens do modelo desenvolvido em Análise Estruturada;

Capítulo 5: Propõe o modelo lógico desenvolvido em Análise Orientada a Objeto (UML), relacionando a Descrição do caso de Uso, Diagrama de Caso de Uso (Use-

Case), Diagrama de Sequência, Descrição e funcionalidade das Classes e diagrama de classes. Apresenta as vantagens e desvantagens do modelo desenvolvido em UML , bem como fazendo um comparativo entre as duas tecnologias;

Capítulo 6: Apresenta a interface pertinente aos dois modelos lógicos (Análise Estruturada e Análise Orientada a Objeto (Unified Modeling Language) – UML), onde procura demostrar a funcionalidade do software através de suas classes, relacionamentos, objetos e operações.

2. AVALIAÇÃO DA QUALIDADE DE SOFTWARE

Este capítulo trata do estado da arte sobre fundamentos que envolvem a avaliação de softwares, fundamentos sobre qualidade, qualidade de softwares, normas técnicas ISO de avaliação da qualidade, mensuração de softwares, importância da equipe na modelagem e desenvolvimento de softwares.

Fica clara a importância do aporte de muitas áreas do conhecimento. Dentre elas destaca-se a avaliação de software que possibilita um conhecimento das características e subcaracteristicas dessa área para iniciar o mapeamento ou caracterização da proposta de um método de modelagem para auxilio na tomada de decisão em Engenharia de Sistemas.

O estudo de critérios e princípios de qualidade de software não se restringe às atividades acadêmicas. Fornecedores de software de portes variados procuram se adaptar a esta nova realidade. Uma das normas que mais está se destacando no mercado é a norma ISO/IEC 9126-1, International Standard, Information Tecnology – Software quality characteristics and metrics – Part-1: Quality characteristics and sub-characteristics [ISO9126-1].

Os textos a seguir descrevem, resumidamente, os conceitos de qualidade e normas técnicas existentes que asseguram a qualidade do produto e a dificuldade em mensurar softwares, onde a equipe de desenvolvimento tem papel fundamental no método de modelagem do desenvolvimento de softwares.

2.1. QUALIDADE

Independentemente do compromisso com uma definição de qualidade (ou algo que definitivamente qualidade é), muitos conceitos têm sido utilizados sobre qualidade:

a) para Dunn "é adequação ao uso através da percepção das necessidades dos clientes" (DUN, 1990);

b) para Freedman "é perseguição às necessidades dos clientes e homogeneidade dos resultados do processo" (FREEDMAN, 1990);

c) para Levson "é o conjunto de características incorporadas ao produto através do projeto e manufatura que determinam o grau de satisfação do cliente" (LEVSON, 1991).

e) para Rook "é rápida percepção e satisfação das necessidades do mercado, adequação ao uso dos produtos e homogeneidade dos resultados do processo" (ROOK,1990).

A satisfação das necessidades do cliente parece ser, para Dunn e Rook dependente da capacidade de capturar correta e completamente essa necessidade. O ato da transformação da necessidade em especificação de projeto e processos parece ser importante; já a homogeneidade dos resultados do processo parece ser importante para Dum. Analisando-se os termos desses conceitos, percebe-se duas características interessantes ao contexto do presente trabalho: o enfoque sobre a especificação dos itens acima mencionados e o enfoque sobre a constância de resultados dos processos.

2.1.1. Certificação de Qualidade

Um aspecto interessante da qualidade é que não basta que ela exista, mas que ela seja reconhecida pelo cliente. Por este motivo, é necessário que a mesma tenha algum tipo de certificação oficial, emitida com base em um padrão.

Muitas propagandas de empresas falam de certificação ISO-9000, que nada mais é do que um padrão de qualidade, reconhecido mundialmente, pelo qual empresas são

avaliadas. Para que seja possível realizar uma avaliação e um julgamento, é necessário haver um padrão ou norma. Existem alguns organismos normalizadores reconhecidos mundialmente:

- ISO - Intemational Organization for Standardization;
- IEEE - Instituto de Engenharia Elétrica e Eletrônica;
- ABNT - Associação Brasileira de Normas Técnicas.

A norma ISO-9000, por exemplo, foi criada pela ISO para permitir que todas as empresas do mundo possam avaliar e julgar sua qualidade. Neste sentido, ao existir um padrão único mundial, uma empresa do Brasil pode garantir a qualidade de seu trabalho a uma da Europa, mesmo não tendo contato com esta.

A Certificação em uma norma ou padrão é a emissão de um documento oficial indicando a conformidade com determinada norma ou padrão. É claro que, antes da emissão do certificado, é preciso realizar todo um processo de avaliação e julgamento, de acordo com uma determinada norma. Embora uma empresa possa auto-avaliar-se ou ser avaliada por seus próprios clientes, o termo Certificação costuma ser aplicado apenas quando efetuado por uma empresa independente e idônea, normalmente especializada neste tipo de trabalho. No Brasil,o INMETRO é o órgão do governo responsável pelo credenciamento das instituições que realizam a certificação de sistemas de qualidade [ISO9000].

2.1.2. Fatores de Qualidade de Software

Os fatores que afetam a qualidade de software podem ser categorizados em dois grupos: o que pode ser medido de forma direta (número de erros, linhas de código, etc.) e os que podem ser medidos de forma indireta (usabilidade, manutenibilidade, etc.). Para cada uma destas características deve ocorrer medição.

Os testes e os mecanismos de medição complementam, mas não solucionam todo o processo de construção do software, pois em todas as dimensões existe o fator humano no processo de construção, o qual é sempre esquecido e é fator-chave de sucesso em qualquer software. Para controlar o que não se pode medir e vice-versa, deve-se questionar sempre o caminho escolhido para desenvolver o software, pois a qualidade não deve ser a meta, porém deve ser o caminho para a satisfação de todos.

MCCALL (1977) e seus colegas, propuseram uma categorização útil dos fatores que afetam a qualidade de software. Esses fatores da qualidade de software focalizam três aspectos importantes de um software: suas características, sua manutenibilidade de mudanças e sua adaptabilidade a novos ambientes.

Existem quatro razões para medir produtos, processos e recursos relacionados a software: para compreender, avaliar, estimar e melhorar.

Segundo VALLE (2000), "A primeira das razões pela qual medições são realizadas é para que se possa compreender – ganhar entendimento dos processos, produtos, recursos e ambientes; e para estabelecer base de dados para comparações com verificações futuras".

Com tais perspectivas de compreensão, avalia-se para determinar o status atual com relação ao planejado. Medidas são sensores que permitem a percepção de quando projetos e processos estão "saindo dos trilhos" para que se possa trazê-los novamente sob controle. Também avalia-se para verificar se objetivos de qualidade foram atingidos e para verificar os impactos da tecnologia e processos de melhoria sobre produtos e processos.

Estima-se para que se possa planejar. Medições para estimativa envolvem ganhos de compreensão sobre relacionamentos entre processos e produtos e também construção de modelos destes relacionamentos, onde os valores que são observados a alguns atributos possam ser usados na estimativa de outros. Isto é feito porque faz-se necessário estabelecer metas tangíveis para custo, cronograma e qualidade - para que

recursos apropriados sejam aplicados. Projeções e estimativas baseadas em dados históricos também ajudam a analisar riscos e fazer relações de custo/benefício.

Mede-se para melhorar. Informações quantitativas são obtidas para auxílio na identificação de barreiras, ineficiências, fraquezas e outras oportunidades para melhoria da qualidade do produto e da performance dos processos. Medidas ajudam também a planejar e mapear os esforços de melhoria.

2.1.3. Norma iso/iec 9126-1 (nbr 13.596)

A ISO/IEC 9126-1, International Standard, Information Tecnology – Software quality characteristics and metrics – Part-1: Quality characteristics and sub-characteristics; Jan / 1997 (CD) [ISO9126-1], é uma das mais antigas da área de qualidade de software e já possui sua tradução para o Brasil, publicada em agosto de 1996 como NBR 13596. Estas normas listam o conjunto de características que devem ser verificadas em um software para que ele seja considerado um "software de qualidade". São seis grandes grupos de características, cada um dividido em algumas subcaracterísticas:

a) **funcionalidade:** trata-se das funções que satisfazem as necessidades dos usuários:
   a.a) adequação: é apropriado ao uso, conforme especificado;
   a.b) acurácia: geração de resultados nos níveis conforme acordado;
   a.c) interoperabilidade: capacidade de interação com outros softwares, conforme especificado;
   a.d) conformidade: de acordo com normas e leis em vigor;
   a.e) segurança de acesso: capacidade de evitar o acesso não autorizado;

b) **confiabilidade:** indica se o software mantém um determinado nível de

desempenho, sob condições pré-determinadas:

b.a) maturidade: freqüência de falhas causadas por defeitos no software

b.b) tolerância a falhas: capacidade de manter um determinado nível de desempenho em caso de falhas;

b.c) recuperabilidade: capacidade de restabelecimento aos níveis de desempenho especificados, em caso de falhas;

c) **usabilidade:** atributos que indicam o esforço necessário ao uso do software:

c.a) inteligibilidade: esforço necessário para o usuário compreender o conceito lógico da aplicação;

c.b) apreensibilidade: esforço necessário para o usuário aprender a usar o software

c.c) operacionalidade: esforço necessário para o usuário operar o software

d) **eficiência:** relacionamento entre o nível de desempenho do software e a quantidade de recursos utilizados:

d.a) comportamento em relação ao tempo: refere-se aos acordos sobre tempos de resposta e de processamento do software;

d.b) comportamento em relação aos recursos: refere-se aos acordos sobre a quantidade de recursos usados durante o uso do software;

e) **manutenibilidade:** atributos que evidenciam o esforço necessário à execução de modificações especificadas:

e.a) analisabilidade: identifica o esforço necessário à identificação de problemas;

e.b) modificabilidade: identifica o esforço necessário a remoção de problemas ou adaptação a mudanças;

e.c) estabilidade: evidências sobre os riscos de efeitos inesperados em caso de mudanças;

e.d) testabilidade: identifica o esforço necessário para a execução de testes em caso de modificações;

f) **portabilidade:** capacidade de operar em ambientes operacionais diferentes:

f.a) adaptabilidade: identifica a capacidade de adaptar-se a ambientes diferentes, conforme especificado;

f.b) capacidade para ser instalado: identifica o esforço necessário à instalação do software;

f.c) conformidade (quanto à portabilidade): atributos do software que identificam o nível de padronização no que se refere à portabilidade;

f.d) capacidade para substituir: identifica o esforço necessário para usar o software em substituição a outro já instalado.

Dentro desta norma ISO/IEC 9126-1 foi editada outra norma complementar ISO/IEC 14598-4 devido à necessidade de mais detalhes sobre como avaliar a qualidade de um software. As características e subcaracterísticas da norma ISSO/IEC 9126-1 apenas começaram o trabalho, isto é, faltava definir, em detalhes, como atribuir um conceito para cada item. Afinal, sem uma padronização, que valor teria uma avaliação?

A ISO, consciente deste problema, está finalizando o trabalho em um conjunto de guias para a Avaliação da Qualidade segundo a norma ISSO/IEC 9126-1. Estes guias descrevem, detalhadamente, todos os passos para que se avalie um software. Embora o trabalho nesta norma ainda não esteja totalmente pronto, já existem informações detalhadas sobre o que será esta norma, quando ela for oficialmente publicada.

Esta norma trará muitos recursos interessantes aos avaliadores, já que trata o processo de avaliação em grande detalhe. Ela leva em conta a existência de três grupos interessados em avaliar um software, o que define os três tipos básicos de certificação.

Certificação	Quem realiza	Finalidade
de 1ª . parte	Empresas que desenvolvem software	Melhorar a qualidade de seu próprio produto
de 2ª . parte	Empresas que adquirem software	Determinar a qualidade do produto que irão adquirir
De 3ª. parte	Empresas que fazem certificação	Emitir documento oficial sobre a qualidade de um software

Esta norma se constituirá, na verdade, de seis documentos distintos relacionados entre si. Uma das normas é a 14598-4 (processo para adquirentes).

2.1.3.1 - Norma 14598-4 (processso para adquirintes)

A ISO 14598-4, International Standard Information Tecnology – Software product evaluation – Part 4: Process for acquirers; Sep/1996 (CD) [ISO14598-4] está dividida em dois processos distintos: um para aquisição de produtos **de *software* de prateleira e** outro **para aquisição de** produtos de software sob encomenda ou modificações em produtos de software existentes. O Processo para adquirentes é resultado da combinação do processo genérico de **avaliação e** do processo de **aquisição** definido pela NBR 12207 — Processo do Ciclo de Vida do *Sofware.*

O propósito da avaliação no processo de aquisição pode ser a comparação de produtos alternativos ou garantir que um produto desenvolvido ou modificado sob encomenda atenda os requisitos inicialmente especificados.

Outras normas podem apoiar a avaliação no processo de aquisição como:

12119 - quando deseja-se adquirir pacotes de software

14598-5 - quando forem requeridas avaliações independentes.

12207 - a qual define o processo de aquisição a ser seguido pelo Processo para Adquirentes.

2.1.3.1.1 - Processo de Aquisição

O processo de aquisição definido pela NBR 12207 [ISO 12207-1], é constituído das seguintes atividades:

a) Iniciação:

O adquirente inicia o processo de aquisição a partir de uma necessidade em adquirir, desenvolver ou melhorar um sistema, produto ou serviço de software.

Nesta atividade são identificados e analisados os requisitos de sistema, os quais podem ser: de negócio, organizacionais e de usuário, segurança entre outros.

Ao final desta atividade são consideradas opções de aquisição, incluindo análise de risco, custo e benefício para cada opção, sendo alguns exemplos: comprar um produto de prateleira, desenvolver internamente, contratar o desenvolvimento, melhorar um produto existente ou uma combinação das opções anteriores.

Caso a opção escolhida seja a aquisição de produto de prateleira, o adquirente deve assegurar que os requisitos sejam satisfeitos, que a documentação do produto

esteja disponível, os direitos de propriedade sejam gaantidos e que o suporte futuro esteja planejado.

b) Preparação do pedido de proposta

Nesta atividade é realizada a documentação dos requisitos de aquisição. Além disto são determinados os pontos de controle nos quais o progresso do fornecimento deverá ser revisado e auditado.

c) Preparação e atualização do controle

Nesta atividade realiza-se a seleção de fornecedores, baseando-se na avaliação das propostas dos fornecedores, bem como suas capacidades. Um contrato com o fornecedor deve ser preparado e negociado, incluindo custo e cronograma de execução. Quando o processo está andando, o adquirente deve possuir o controle de mudanças no contrato.

d) Monitoração do fornecedor

A monitoração do fornecedor consiste em atividades de avaliação aplicadas durante a execução do contrato levando à aceitação e entrega do produto de *software.*

e) Aceitação e conclusão

A aceitação e conclusão são atividades executadas durante a aceitação do produto e entrega do produto de *software* final, obedecendo os critenos de aceitação previamente definidos durante a atividade de iniciação.

2.2. MENSURAÇÃO DE SOFTWARE

A mensuração tem sido importante para incentivar as organizações a atingirem níveis superiores de maturidade. Programas de mensuração ajudam as organizações e os tomadores de decisões fornecendo informações significativas com relação à qualidade, adequação e progresso evolutivo de processos, produtos e projetos de softwares.

Segundo VALLE (2000), muitos especialistas vêm, até hoje, discutindo as diferenças entre medidas, métricas e indicadores, justamente os três termos mais freqüentes quando se fala em mensuração. O certo é que não existe um consenso quanto à utilização de cada termo, muitas vezes tais termos são usados indistintamente.

2.2.1 - Mensuração é a chave

Mensuração do software é a tecnologia chave para a aquisição da melhoria da qualidade. Para que se compreenda o que está acontecendo e se construa modelos específicos numa organização, é necessário que primeiro se meça dados quantitativos e qualitativos de qualidade e produtividade de software e seus fatores de influência potenciais (WANGENHEIM, 2000).

Cristiane e Aldo Von Wangenheim (2000) declaram que:

Somente com base em dados específicos coletados no ambiente com o qual vamos trabalhar, seremos capazes de construir modelos de qualidade válidos, capazes de refletir a realidade de uma organização específica e de servirem de base efetiva para decisões gerenciais e uma melhoria real da qualidade. Mensuração de software pode ser usada para analisar várias qualidades ou atributos do processo, produto e recursos de software. Por exemplo, pode-se utilizar medidas para analisar a efetividade de tec-

nologias, produtividade, qualidade de produtos (confiabilidade, usabilidade, manutenibilidade etc).

O objetivo da mensuração é a captura de características de entidades relevantes e seus inter-relacionamentos, bem como a sua manipulação de modo formal. Para isso, modelos de qualidade, medidas e procedimentos de mensuração e seus instrumentos devem ser definidos, bem como seus respectivos critérios de interpretação e procedimentos de análise.

2.3. IMPORTÂNCIA DA EQUIPE PARA DESENVOLVER UM TRABALHO COM QUALIDADE

Com o surgimento e desenvolvimento das tecnologias da informática, quando todos buscam técnicas altamente sofisticadas através da pesquisa científica, numa busca incessante ao que se pensa ser o único meio de sobrevivência digna e humana, torna-se imperioso e necessário um perfeito conhecimento e internalização dos fundamentos da ciência que alicerçou todo este processo tecnológico atual.

De acordo com as colocações de CASTRO (2001), um dos treinamentos mais solicitados é o de trabalho em equipe que visa colocar todos na mesma harmonia e sinergismo, enxergando a organização como um todo e não mais com uma visão fragmentada. O treinamento em trabalho de equipe concentra a atenção dos participantes no objetivo comum da empresa com igual responsabilidade e comprometimento de todos. A meta é tornar o funcionário participativo, com comunicação adequada e eficiente, mostrar sua criatividade e proporcionar a troca de idéias nas equipes sem esquecer o propósito da empresa.

A possibilidade de trabalhar harmoniosamente em equipes de trabalho é um dos pré-requisitos para ser um bom líder nos dias de hoje. Essa característica é im-

prescindível, pois a exigência de qualidade é cada vez maior e a cobrança por resultados é uma necessidade.

Hoje é necessária a consciência de que a equipe é algo mais, que deve ser eficiente e eficaz, apresentando resultados condizentes com as necessidades, não importando o tamanho dos departamentos e das equipes. Se são grandes ou pequenas, devem atuar conjuntamente de forma que todas as áreas estejam integradas e trabalhem para alcançar o mesmo objetivo.

As constantes mudanças verificadas nas tecnologias de desenvolvimento, associadas às também contínuas mudanças do perfil e exigência dos clientes, têm demandado dos desenvolvedores e das equipes uma significativa adaptação dos seus processos, práticos e forma de organização.

Se não há mais tempo para o analista adquirir os conhecimentos necessários, como será feito para garantir a qualidade dos softwares a serem desenvolvidos? Experiências de pesquisadores demonstram que o conhecimento não precisa ser todo adquirido por uma única pessoa, ou seja, podem estar "espalhado" pelos componentes da equipe de desenvolvimento.

Estima-se que 80% dos problemas existentes num projeto de tecnologia da informação são decorrentes do relacionamento entre pessoas e não de questões técnicas. Os principais contratempos são os interesses conflitantes, pessoas inadequadas para a função que exercem ou para a situação proposta, a resistência à mudança e a dificuldade de gestão.

O sucesso dependerá da qualidade do processo de gerenciamento do trabalho do grupo envolvido no projeto. Nesse sentido, a equipe precisa ser organizada para o máximo aproveitamento das capacidades de cada componente, sendo que estas devem ser maximizadas através de um constante processo de reciclagem e auto-estudo.

Dentro da área de informática, os especialistas acreditam que o grau de integração

vem aumentando em todas as funções, inclusive naquelas que tradicionalmente trabalhavam de maneira isolada.

O trabalho realizado em qualquer área deve ser como o de um músico, que está sempre pensando em melhor qualidade, maior precisão, afinação, interpretação e melhores instrumentos. Eles estão sempre em treinamento e aperfeiçoamento.

Um software de qualidade é aquele que atende perfeitamente (projeto "perfeito"), de forma confiável (sem defeitos), de forma acessível, de forma segura, e no tempo certo às necessidades. A qualidade e produtividade de uma organização dependem de três fatores: processos, tecnologia e pessoas.

É um trabalho difícil a implantação desses modelos. Porém, o retorno do investimento vem com a diminuição de tempo gasto nas tarefas, nos custos de manutenção dos produtos, certificação da qualidade do trabalho e da equipe. Isso reflete em desempenho, confiabilidade, entrega e preço.

Talvez a frase que defina o que hoje significa um software de qualidade com trabalho em equipe, seja esta de J. M. Juran: "qualidade não se controla , se faz".

2.4. CONSIDERAÇÕES SOBRE MODELOS

Os modelos são representações ou abstrações de determinados pontos de vista de uma realidade. Os modelos são uma tentativa de imitar os sistemas. Dessa forma, servem ao propósito de materializar uma mensagem e também prestam-se para testar certos princípios físicos da realidade que representam (a um custo e risco menores do que se o teste fosse efetuado sobre o sistema real) buscando a predição de determinados padrões de comportamento, constituindo-se, portanto, em importantes ferramentas para o analista de sistemas.

Entretanto, apesar do desenvolvimento de sistemas ser facilitado com o uso de modelos (como é o caso da UML (Unified Modeling Language)), para FURLAN (1998), "criar modelos é um trabalho altamente criativo e não há uma solução final ou uma resposta correta que possa ser verificada ao final do trabalho".

2.4.1 - Por que Fazer a Modelagem

Para entregar um software que satisfaça ao propósito pretendido, será preciso reunir-se e interagir com os usuários de uma maneira disciplinada com a finalidade de expor os requisitos do sistema. Para desenvolver software de qualidade duradoura, será necessário criar uma arquitetura de fundação sólida que aceite modificações.

Para desenvolver software de forma rápida, eficiente e efetiva, com o mínimo de desperdício e de retrabalho de software, será preciso dispor das pessoas certas, das ferramentas adequadas e do enfoque correto. Para fazer tudo isso de maneira previsível e consistente, com uma avaliação dos custos reais do sistema, você precisará de um processo seguro de desenvolvimento que possa ser adaptado às novas necessidades de seu negócio e de sua tecnologia (BOOCH, 2000).

A modelagem é uma parte central de todas as atividades que levam à implantação de um bom software. Construir modelos para comunicar a estrutura e o comportamento desejados do sistema. Construir modelos para visualizar e controlar a arquitetura do sistema. Construir modelos para compreender melhor o sistema que esta sendo elaborando, muitas vezes expondo oportunidades de simplificação e reaproveitamento. Construir modelos para gerenciar os riscos.

2.5. CONCLUSÕES

Este capítulo apresentou considerações importantes com o objetivo de contribuir com o desenvolvimento do modelo, pois quando se pensa em qualidade de um "produto físico", é fácil imaginar padrões de comparação, provavelmente ligado às dimensões do produto ou alguma outra característica física. Quando se trata de software, como podemos definir exatamente o que é qualidade? Parece difícil!

A ISO (Organização Internacional de Padrões) tem se empenhado o suficiente para publicar uma norma que representa a atual padronização mundial para a qualidade de produtos de software.

Esta visão é importante porque ajuda a definir melhor os benefícios que o modelo pode trazer. Dentre estes benefícios pode-se estabelecer uma linguagem comum, construir uma visão compartilhada, oferecer uma estrutura para se priorizar as ações, prover uma estrutura para se realizar avaliações confiáveis e consistentes.

Três pontos devem ser enfatizados (PRESSMAN, 1995):

1. Os requesitos de software são a base a partir da qual a qualidade é medida. A falta de conformidade aos requisitos significa falta de qualidade.

2. Padrões especificados definem um conjunto de critérios de desenvolvimento que orientam a maneira segundo a qual o software passa pelo trabalho de engenharia. Se os critérios não forem seguidos, o resultado quase que seguramente será a falta de qualidade.

3. Há um conjunto de requisitos implícitos que freqüentemente não são mencionados. Se o software se adequar aos seus requisitos explícitos, mas deixar de cumprir seus requisitos implícitos, a qualidade de software será suspeita.

3. BANCOS DE DADOS

As bases de dados foram criadas com a necessidade de um local para armazenar dados após a exclusão de uma aplicação. Assim a tecnologia de Sistemas Gerenciadores de Banco de dados (SGDB's) vem sedo desenvolvida desde o final dos anos 60 com modelo hierárquico. Como uma forma aplicada, o modelo de rede sucedeu o hierárquico. Por volta dos anos 80, já com outro enfoque, surgiram os SGBD's Relacionais.

A orientação a objeto é a modelagem que mais se aproxima da representação do mundo real. Em comparação aos métodos relacionais, as aplicações mais complexas podem ser representadas através de modelagens de dados orientados a objeto.

Hoje, na maioria dos casos, a disseminação por toda a empresa de produtos e lições aprendidas não é suportada. A reutilização de conhecimento de mensuração é feita ad-hoc, de maneira informal, normalmente limitada às experiências pessoais. Para maximizar a produtividade e os ganhos de qualidade pela reutilização de experiências, atividades de reutilização devem ser sistematizadas pelo conhecimento de uma definição de processo de gerenciamento de conhecimento e suportada por uma infraestrutura técnica por toda a companhia.

Neste contexto, o Raciocínio Baseado em Casos (RBC), de acordo com as conclusões de AAMODT (1994), "tem um papel chave, uma vez que provê um pleno suporte para o desenvolvimento de sistemas baseados em conhecimento. As maiores vantagens do RBC, neste contexto, são a recuperação baseada em similaridade para todos os tipos de artefatos e seu foco primário em conhecimento baseado em experiência. A aprendizagem contínua e incremental ocorre naturalmente através do produto da resolução de problemas e da revisão e captura de experiências cada vez que um novo problema é resolvido".

Este capítulo se propõe a investigar, ou seja, a estudar os diversos tipos de Bancos de Dados existentes, bem como suas diferenças, necessidades, vantagens e desvantagens

de forma a deduzir o melhor banco de dados que se possa incluir no desenvolvimento deste projeto.

3.1. DEFININDO BANCOS DE DADOS

Podemos entender por banco de dados qualquer sistema que reúna e mantenha organizada uma série de informações relacionadas a um determinado assunto, em uma determinada ordem. Sendo assim, o sistema de banco de dados é basicamente um sistema de manutenção de registros por computador.

Os sistemas de banco de dados são projetados para armazenar e administrar grande volume de dados. O gerenciamento de dados engloba definições de estruturas para armazenamento da informação e a elaboração de mecanismos de manipulação destas informações, mantendo também a segurança e integridade das informações armazenadas.

Um dos grandes recursos oferecidos pelo banco de dados é fornecer uma visão abstrata das informações, isto é, o sistema abstrai certos detalhes de como são armazenados e mantidos os dados.

Segundo CERÍOLA (1994), "uma definição de banco de dados bem aceita por vários autores seria: Um banco de dados é uma coleção de dados organizados e integrados, armazenados em forma de tabelas interligadas através de chaves primárias e estrangeiras, que constituem uma representação natural dos dados, sem imposição de restrições ou modificações, de forma a ser adequada a qualquer computador, podendo ser utilizado por todas as aplicações relevantes sem a necessidade de serem definidos em programas, pois utiliza as definições existentes nas bases de dados, através do dicionário de dados ativo e dinâmico".

Com o passar do tempo, a dependência pela informática aumentou muito e consequentemente o número de informações a serem processadas cresceu, surgindo assim a necessidade de aplicações maiores e mais complexas. O Sistema Gerenciador Banco de Dados (SGBD) possibilita o uso de um sistema gerenciador mais poderoso onde se podem operar diversas aplicações sobre uma mesma base de dados, garantindo a consistência, integridade e rapidez no processo das informações.

Assim, cada vez mais as aplicações de Banco de Dados tonam-se extremamente complexas, exigindo uma completa e rigorosa formulação, análise das necessidades de informação antes de iniciar o processo de desenvolvimento do projeto e um constante aperfeiçoamento de aplicativos e ferramentas de desenvolvimento.

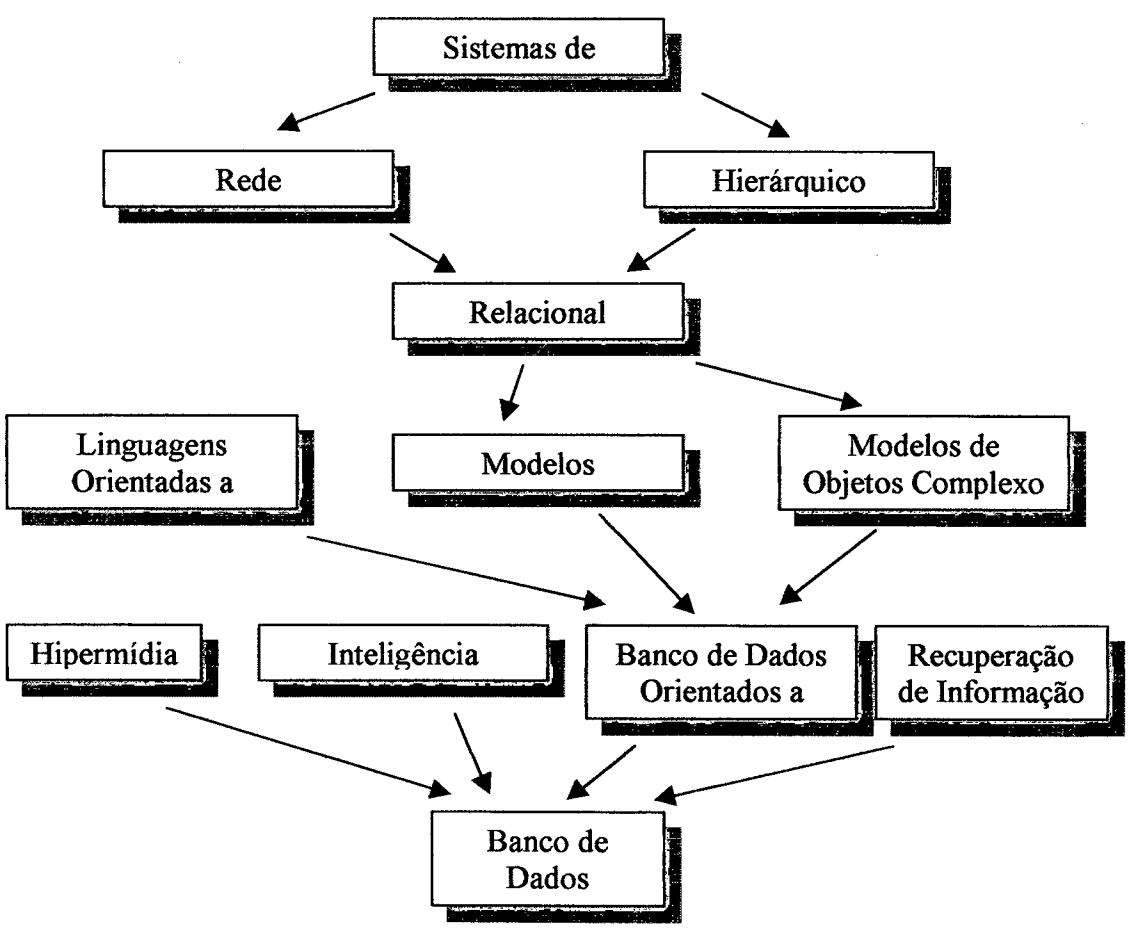

FIGURA 3.1. Evolução do Banco de Dados

3.2. BANCO DE DADOS DISTRIBUÍDOS

Um sistema de Banco de Dados Distribuído consiste em uma coleção de locais conectados através de um sistema de rede de comunicação, no qual cada local é um sistema de banco e dados em seu próprio direito, mas os locais "concordam" em cooperar, de forma que um usuário em qualquer local pode ter acesso a qualquer dado da rede, exatamente como se os dados estivessem armazenados no próprio local do usuário (CERÍOLA 1994).

Os sistemas distribuídos compreendem a atualização das informações cooperativas em aplicações nos computadores micro, mini e grande porte, dentro de um processamento misto. A conexidade entre os computadores é fundamental para implementar o uso de sistemas distribuídos.

Um sistema de Banco de Dados Distribuído é composto de uma rede de bancos de dados locais, armazenado em diversas máquinas, devendo "aparecer" para o usuário com um único banco de dados lógico, instalado em uma só máquina.

3.3 - SISTEMAS GERENCIADORES DE BANCO DE DADOS

Um sistema de gerenciamento de banco de dados (SGDB) consiste em uma coleção de dados inter-relacionados, juntamente com uma coleção de programas que promovem o acesso a esses dados. Para DATE (1997) "O objetivo principal de sistema de banco de dados é promover um ambiente que seja adequado e eficiente para armazenamento e recuperação de informações". Existem muitas vantagens no uso de um SGBD:

- Recuperação de informação;

- Compartilhamento entre usuários;
- Compartilhamento entre aplicações;
- Segurança
- Integridade;
- Extensibilidade;
- Distribuição dos dados.

Havendo a necessidade de compartilhar informações entre usuários com diferentes visões de dados, assim um SGBD de alto desempenho é a base de um ambiente gerenciador de informação e da produtividade que dele deve derivar. É o fator que determina com que rapidez uma organização será capaz de responder a novas oportunidades de negócio, ou será forçada a implementar mudanças sem causar um impacto negativo em sistemas existentes. Oferece a facilidade e a rapidez com que novas aplicações poderão ser desenvolvidas, satisfazendo assim as freqüentes necessidades geradas pelo usuário.

Um sistema gerenciador de banco de dados é utilizado para armazenar dados de uma forma que seja permitido examinar as informações de diversas maneiras.

3.3.1 Sistema Gerenciador de Banco de Dados Relacional

Um banco de dados relacional consiste em uma coleção de tabelas, cada qual designada por um nome único. Uma linha em uma tabela representa um relacionamento entre um conjunto de valores. Uma vez que numa tabela é uma coleção de tais relacionamentos, existe uma correspondência íntima entre o conceito de tabela e o conceito matemático de relação.

O Sistema Gerenciador de Banco de Dados Relacional (SGBDR) é o resultado de pesquisas efetuados por um grupo de pesquisadores da IBM no final da década de 1960, e que só se tornou uma realidade estabelecida no mercado computacional em

princípios da década de 1980. Historicamente, a Oracle Corporation foi a primeira que lançou no mercado um Sistema Gerenciador de Banco de Dados Relacional, instalando em 1979 esse projeto na NASA (*National Aeronautics Space Agency*) nos Estados Unidos (MARTIN, 1995).

Como ferramenta de apoio a linguagem SQL (*Strutured Query Language*), inicialmente desenvolvida como linguagem padrão para pesquisas a banco de dados, hoje não é apenas uma linguagem de consulta, sendo muito mais abrangente por incluir segurança e manipulação de dados.

A evolução do processamento cooperativo e a evolução dos microcomputadores pessoais como culturas separadas deram origem a um lapso tecnológico que os sistemas distribuídos tratam de minimizar e integrar. O papel de cada cultura deve ser levado em consideração no tratamento das informações cooperativas quando surgir a polêmica entre as aplicações centralizadas e as descentralizadas. Deve-se levar em consideração a revolução que o mundo do processamento cooperativo está enfrentando.

3.3.2. Sistema Gerenciador de Banco de Dados Orientado a Objetos

Os bancos de dados orientados a objetos integram a orientação a objetos com aptidões de banco de dados e através destas construções orientadas a objetos, os usuários podem esconder detalhes de implementação de seus módulos, compartilhar a referência a objetos e expandir seus sistemas através de módulos existentes. Por meio dos Bancos de Dados, os usuários podem obter o estado em que os objetos se encontram e estar atualizados entre as várias solicitações de programa, podendo ao simultaneamente compartilhar a mesma informação. Os bancos de dados orientados a objeto combinam os benefícios e conceitos de orientação a objeto com a funcionalidade dos bancos de dados.

Bancos de dados orientados a objeto = orientação a objeto + aptidões de banco de dados.

Os bancos de dados orientados a objetos são o resultado da evolução de várias tecnologias desde a criação dos primeiros sistemas gerenciadores de banco de dados.

3.3.2.1. Modelo de Dados Orientado a Objeto

O modelo orientado a objetos é baseado num conjunto de objetos contendo valores armazenados em variáveis de instância internas. Um objeto possui também métodos que são códigos que manipulam as variáveis contidas nesse objeto. A característica principal das linguagens e ferramentas orientadas a objetos é o conceito de componentes modularizados utilizado para expressar mais naturalmente problemas complexos, tornando assim a programação mais direta e objetiva.

A interface entre um objeto e o resto do sistema é definida como um conjunto de **mensagens**. Uma mensagem pode solicitar ao objeto que realize uma tarefa e retorne um valor, ou que modifique o conteúdo do objeto, mudando seu estado ou valor. Em um ambiente orientado a objetos a interação entre vários objetos é o envio de mensagens uns aos outros.

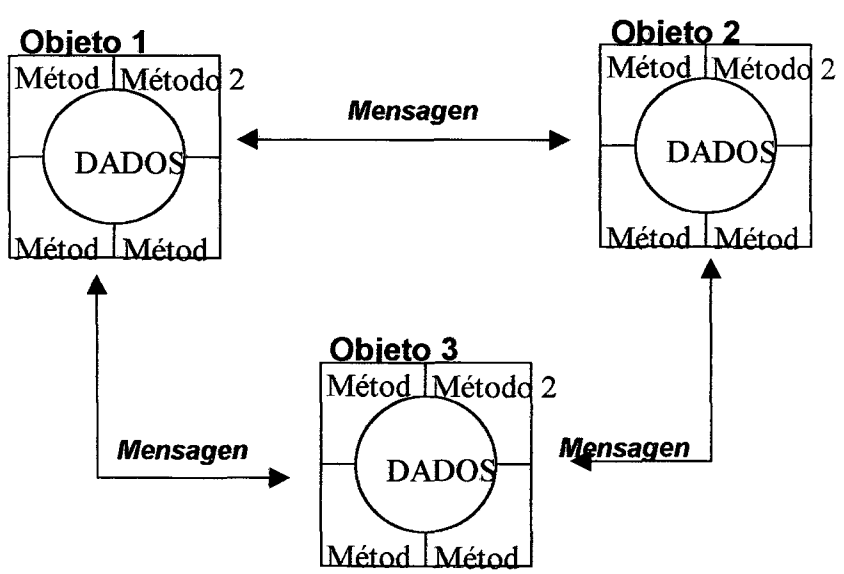

FIGURA 3.3.2.1 - **TROCA DE MENSAGENS ENTRE OS OBJETOS**

Segundo KHOSHAFIAN (1990), em geral um objeto tem associado a ele:

- Um conjunto de variáveis que contém os dados para o objeto. O valor de cada variável é por si próprio um objeto;
- Um conjunto de mensagens às quais um objeto responde;
- Um método que é uma rotina de código para implementar cada mensagem. Um método retorna um valor como resposta a uma mensagem.

Uma vez que a única interface externa apresentada por um objeto é o conjunto de mensagens às quais ele responde, é possível modificar a definição de métodos e variáveis sem interferir em outros objetos. É também possível a substituição de uma variável por um método que calcule um valor. Por exemplo, um objeto pedido pode conter uma variável total que armazene o valor total do pedido ou um método total que calcule o total com base em outras variáveis ou métodos.

3.3.3. Sistema Gerenciador de Banco de Dados em Conhecimento (Data Warehouse)

A promessa atual de futuro, onde o mais importante não será a informação, mas a forma como ela pode ser vista. Na maioria dos casos o enfoque principal dos primeiros bancos de dados era o processamento operacional - geralmente transacional (INMON, 1997).

Nos últimos anos, surgiu um conceito de banco de dados mais sofisticado, (chamado de data Warehouse – grande depósito de dados) – um que atende a necessidades operacionais e outro que atende a necessidades informacionais ou analíticas (INMON, 1997). A divisão em banco de dados operacionais e informacionais ocorre por várias razões:

• Os dados que atendem a necessidades operacionais são fisicamente diferentes dos dados que atendem a necessidades informacionais ou analíticas;

• A tecnologia de suporte ao processamento operacional é fundamentalmente diferente da tecnologia utilizada para prestar suporte a necessidades informacionais ou analíticas;

• A comunidade de usuários dos dados operacionais é diferente da que é atendida pelos dados informacionais ou analíticos;

• As caracteristicas de processamento do ambiente operacional e do ambiente informacional são, fundamentalmente, diferentes.

A maneira moderna de construir sistemas consiste em separar o processamento e os dados operacionais dos informacionais ou analíticos.

O processamento informacional ou analítico atende às necessidades dos gerentes durante o processo de tomada de decisões. Geralmente conhecido como processamento SAD (ou de SADs – sistemas de apoio à decisão). O processamento analítico examina amplos espectros de dados para detectar tendências. Em vez de considerar um ou dois registros de dados (como ocorre no processamento operacional), quando o analista de SAD executa um processamento analítico, muitos registros são acessados (GLASSEY, 1999).

Além disso, o analista de SAD, muito raramente, atualiza dados. Nos sistemas operacionais, os dados estão constantemente sendo atualizados no nível de registro individual. No processamento analítico, os registros estão constantemente sendo acessados e seus conteúdos são agrupados para análise, mas ocorre pouca ou nenhuma alteração dos registros individuais.

No processamento analítico, os requisitos de tempo de resposta são muito atenuados em comparação com os do tradicional processamento operacional. O tempo de resposta analítico alcança de 30 minutos a 24 horas. Para o processamento

operacional, tempos de resposta inseridos nessa escala significariam um absoluto desastre.

A rede que atende à comunidade analítica é muito menor do que a que atende à comunidade operacional. Normalmente, há muito menos usuários da rede analítica do que da rede operacional.

Ao contrário da tecnologia que dá suporte ao ambiente analítico, a tecnologia voltada para o ambiente operacional deve tratar do bloqueio de dados e transações, disputa de dados, deadlock e assim por diante.

3.3.3.1. Mudança de Enfoque

No cerne de um ambiente "projetado" para data warehouse está a percepção de que há fundamentalmente duas espécies de dados: dados primitivos e dados derivados. Dados primitivos são dados detalhados utilizados na condução das operações cotidianas da empresa. Dados derivados são dados resumidos ou calculados de forma a antender às necessidades da gerência da empresa. Dados primitivos podem ser atualizados. Dados derivados não. Dados primitivos consistem basicamente em dados cujos valores são referentes ao presente momento. Dados derivados são, geralmente, dados históricos (GLASSEY, 1999).

3.4. RACIOCÍNIO BASEADO EM CASOS

O *racioclnio baseado em casos* (RBC) (AAMODT et al, 1994, KOLODNER, 1993), é um paradigma para a resolução de problemas que em muitos aspectos é fundamentalmente diferente de outras atividades da inteligência artificial. Ao invés de

criar associações sobre relacionamentos generalizados entre descrições de problemas e conclusões, o RBC tem capacidade para utilizar o conhecimento especifico de experiências antigas e situações de problemas concretos (casos). Um novo problema é resolvido pela busca de um caso similar no passado e sua reutilização no novo problema. Uma segunda diferença importante é que RBC é também uma abordagem para a aprendizagem sustentada e incremental, uma vez que uma nova experiência quando um problema é resolvido, tornando-se imediatamente disponível para problemas futuros (AAMODT, 1994).

Deve-se notar que cada caso usualmente denota uma situação de problema. Uma situação anteriormente experimentada, anteriormente capturada e aprendida de tal forma que possa ser reutilizada na resolução de futuros problemas é então referenciada como um caso passado, caso armazenado ou caso retido. Correspondentemente, um novo caso ou caso não resolvido é a descrição de um novo problema a ser resolvido. Um caso típico é usualmente assumido por conter um certo grau de riqueza de informações contidas em si e uma certa complexidade com respeito a sua organização interna. Possui também outras características como: a possibilidade de ser modificado ou adaptado; e a solução recuperada pode ser aplicada em diferentes contextos de solução de problemas (AAMODT, 1994).

Raciocínio pelo reuso de casos passados é uma forma poderosa, freqüentemente, utilizada por seres humanos para resolver problemas. Pessoas usam casos passados como modelos quando estão aprendendo a resolver problemas, particularmente em aprendizados recentes.

3.4.1. Fundamentos do Raciocínio Baseado em Casos

As tarefas centrais de todos os métodos de raciocínio baseado em casos têm em comum a identificação de uma situação de problema atual, a procura por um caso no passado que seja similar ao novo caso, a utilização deste caso para sugerir uma solução

ao problema corrente, a avaliação da solução proposta e a atualização do sistema pela aprendizagem desta nova experiência (AAMODT, 1994).

3.4.1.1. Representação de Casos

Um caso é uma peça contextualizada do conhecimento representando uma experiência. Ele contém lições passadas, isto é, o conteúdo do caso e o contexto no qual a lição pode ser usada. Um caso pode ser o relato de um evento, uma história, ou algum registro típico.

O problema descreve o estado do mundo quando o caso ocorreu e a solução demonstra a solução derivada para um determinado problema. Em um caso é possível armazenar muitos tipos de dados no qual espera-se que possam ser armazenados em uma base de dados convencional, como nomes, identificação de produtos, valores como temperaturas e notas de texto.

3.4.1.2. O MODELO DE MEMÓRIA DINÂMICA

O primeiro sistema que pode ser referenciado como um sistema de raciocínio baseado em casos foi o CYRUS, baseado no modelo de memória dinâmica de SCHANK (1982). A memória do caso, neste modelo, é uma estrutura hierárquica a qual é chamado pacote de organização de memória episódico (E-MOP's), também referenciado como episódios generalizados (GE). Este modelo desenvolvido da teoria MOP mais geral de Schank. A idéia básica é organizar casos específicos os quais compartilham propriedades similares sob uma estrutura mais geral (um episódio generalizado). Um episódio generalizado contém três diferentes tipos de objetos: normas, casos e índices. Normas são qualidades em comum a todos os casos sob o GE;

índices são qualidades as quais discriminam os casos uns dos outros. Um índice pode apontar para um episódio generalizado mais específico ou diretamente para um caso. Um índice é composto por dois termos: o nome do índice e o valor do índice.

A memória de casos é inteiramente uma rede discriminatória onde um nó é ou um episódio generalizado, o nome de um índice, o valor de um índice ou um caso.

3.4.1.3. O MODELO CATEGORIA & EXEMPLAR

O sistema PROTOS, construído por Ray Bareiss e Bruce Porter, propõe uma forma alternativa para organizar casos em uma memória de casos. Casos são também referenciados como exemplares. A base psicológica e filosófica para este método é a visão de que no mundo real, conceitos naturais podem ser definidos extensionalmente. Além disso, diferentes características são assinaladas com importâncias diferentes na descrição de um caso associado a uma categoria. A memória do caso é embutida em uma estrutura de rede de categorias, relações semânticas, casos e ponteiros de índices. Cada caso associado com uma categoria (PORTER, 1986).

Um índice pode apontar para um caso ou uma categoria. Os índices são de três tipos: ligações para características apontando de descritores do problema (características) para casos ou categorias (lembranças), ligações de casos apontando de categorias para os casos associados (ligação de exemplares) e ligações de diferenças apontando de casos para os casos vizinhos que apenas diferem em um ou em um número pequeno de características. Uma característica é, geralmente, descrita por um nome e um valor.

Entre esta organização de memória, as categorias são interligadas em uma rede semântica, a qual também contém as características e estados intermediários encaminhados por outros termos. Esta rede representa uma formação do conhecimento geral do domínio, o que habilita o suporte explanatório de algumas tarefas de RBC.

3.4.1.4. Indexação

Muitos sistemas de bancos de dados utilizam índices para acelerar a recuperação de dados. Um índice é uma estrutura computacional de dados que pode ser colocada na memória e rapidamente pesquisada. Isto significa que não é necessário percorrer cada registro armazenado no disco, o que pode ser lento. RBC também utiliza índices para acelerar a recuperação.

Informações em um caso podem ser de dois tipos: Informações indexadas que serão utilizadas para recuperação e informações não indexadas que provêm informação contextual de valores para o usuário, mas não são usadas diretamente na recuperação. Índices devem ser preditivos, isto é, endereçar os propósitos para os quais o caso será usado, ser abstrato para permitir ampliação futura da base de casos, e ser concreto para ser reconhecido no futuro (KOLODNER, 1993).

Existem inúmeros métodos de indexação automática, tais como indexação por características e dimensões que tendem a ser preditivas por todo o domínio; Indexação baseada na diferença, a qual seleciona índices que diferenciam um caso de outros; métodos de generalização baseados em similaridade e explanação que produzem um apropriado conjunto de índices para casos abstratos a partir de casos que compartilham um conjunto comum de características ou métodos indutivos de aprendizagem que identificam características preditivas as quais são usadas como índices para aplicações práticas; além disso tudo, índices podem ser escolhidos automaticamente, manualmente, ou pelas duas técnicas (WATSON, 1997).

3.4.1.5. Armazenamento

O armazenamento de casos é um aspecto importante de sistemas RBC, no qual é

refletida a visão conceitual do que é representado no caso e dos índices que caracterizam o caso. Um sistema baseado em casos pode ser organizado em uma estrutura gerenciável que suporta a busca eficiente e métodos de recuperação. Um equilíbrio é encontrado entre os métodos de armazenamento que preservam a riqueza dos casos e seus índices e métodos que simplificam o acesso e a recuperação dos casos relevantes. Estes métodos são usualmente referidos como modelos de memória de casos. Os dois mais influentes modelos de memória de casos acadêmicos são o modelo de Memória Dinâmica de Schank e Koloder, e o modelo de Categoria e Exemplar de Porter e Bareiss. Estas técnicas são ainda largamente utilizadas pela comunidade da ciência cognitiva, porém nenhuma das ferramentas comerciais de RBC disponíveis utiliza estas técnicas.

3.4.1.6. Recuperação

A recuperação dos casos é dependente do método de representação utilizado. Em geral duas técnicas são usadas por aplicações de RBC comerciais:

- *Nearest-neighbor retne vai.*
- *inductive retrie vai.*

3.4.1.6.1. Nearest-neighbor retrieval

Conceitualmente, é uma técnica muito simples. O que é preciso é calcular a distância relativa entre um caso alvo e os outros casos. Aquele que obtiver o menor valor é o caso vizinho mais próximo. Esta solução utiliza-se de alguma medida de similaridade, que pode utilizar-se de pesos diferentes para cada atributo, onde a soma da similaridade de todos os atributos é calculada. Algoritmos similares a estes são usados em muitas ferramentas de RBC para realizar esta recuperação.

3.4.1.6.2. Inductive retrieval

Esta técnica utilizada por muitas ferramentas RBC envolve um processo chamado indução. Indução é uma técnica desenvolvida por pesquisadores de aprendizado de máquina para extrair regras ou construir árvores de decisão a partir de dados passados. Em sistemas RBC, o embasamento em Casos é analisado por um algoritmo de indução para produzir uma árvore de decisão que classifica ou indexa os casos. O algoritmo de indução mais largamente utilizado é chamado de ID3 (QUINLAN, 1986). Este algoritmo constrói uma árvore de decisão a partir de uma base de casos. Utiliza-se de uma heurística chamada de ganho de informação para procurar o atributo mais promissor, a partir do qual a árvore será dividida ao meio.

Estas duas técnicas são amplamente utilizadas em ferramentas de RBC.

*3.4.*1.7. Adaptação

Quando um caso é recuperado, o sistema RBC estará atento para reutilizar a solução sugerida pelo caso recuperado. Em muitas circunstâncias a solução pode ser suficiente. Porém, em outras circunstâncias, a solução pode necessitar de uma adaptação da solução recuperada para as necessidades do caso atual. Adaptações procuram por diferenças proeminentes entre o caso recuperado e o caso corrente e então a aplicação algumas fórmulas e regras que tomam aquelas diferenças no relatório sugerindo a solução final.

Em geral existem dois tipos de adaptação em RBC: Adaptações estruturais que aplicam regras e fórmulas de adaptação diretamente na solução armazenada nos casos, ou adaptações derivativas que reutilizam regras ou fórmulas que geraram a solução original para produzir uma nova solução para o problema corrente. Neste método, a seqüência de planejamento que construiu a solução original deve ser armazenada como

um atributo adicional ao caso. Esta adaptação pode ser apenas utilizada para domínios que estão bem entendidos.

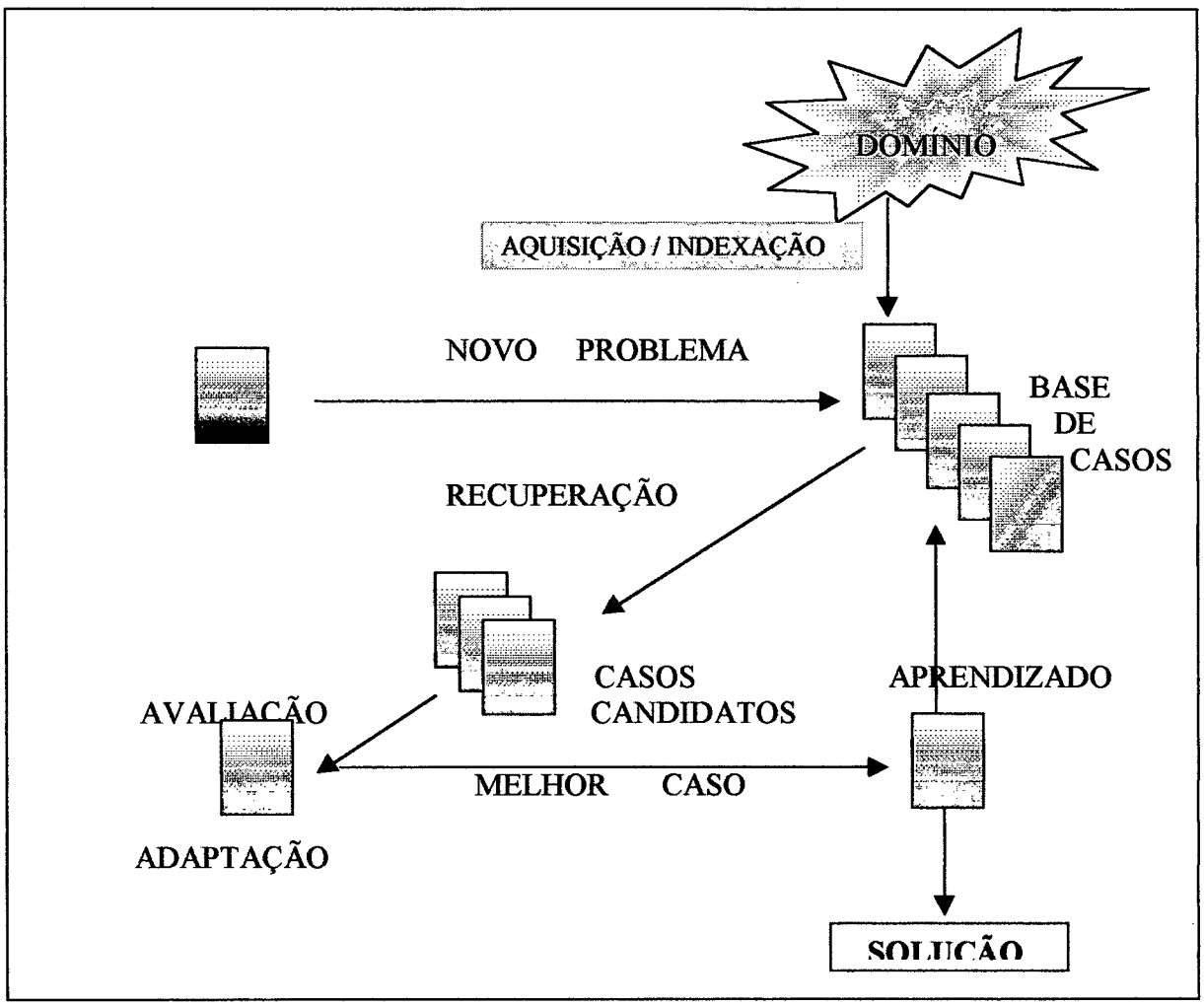

FIGURA 3.4.1.7.1 - Ciclo de solução em um sistema RBC. Fluxo de análise e solução de problemas, mais aquisição de conhecimento em um sistema de RBC.

3.4.1.8. Aprendizado

Este é o processo de incorporação que é útil para reter a partir do novo episódio de um problema resolvido na base de conhecimento existente. A aprendizagem com sucesso ou não da solução proposta é disparada pela saída de uma avaliação e possível reparo. Isto envolve a seleção de qual informação do caso e em que formato é retido,

sucesso ou não da solução proposta é disparada pela saída de uma avaliação e possível reparo. Isto envolve a seleção de qual informação do caso e em que formato é retido, como indexar o caso para posterior recuperaçao para problemas similares e como integrar o novo caso na estrutura de memória.

A etapa de avaliação pode ser programada para a execução automática ou com participação do usuário. A avaliação tem por objetivo avaliar a qualidade da solução adaptada ao problema de entrada para definir se esta tem condições de ser adicionada à memória.

3.4.1.8.1. Extração

Em RBC, a base de casos é atualizada, assim não interessando a forma pela qual o problema foi resolvido. Se ele utiliza um caso prévio, um novo caso pode ser construído ou um caso antigo pode ser generalizado. Se o problema foi resolvido por outros métodos, incluindo o ato de perguntar ao usuário, um caso inteiramente novo será construído. Falhas podem ser extraídas ou retidas como casos de falhas separadas ou como casos completos de problema. Quando uma falha é encontrada, o sistema pode lembrar-se de uma falha similar prévia e utilizar o caso de falha para melhorar o entendimento e corrigir a presente falha.

A inclusão do caso adaptado, reutilizado e avaliado consiste na aprendizagem em um sistema RBC.

3.4.1.8.2. Indexação no aprendizado

O problema da indexação é o problema central e muitas vezes focado problema em RBC. Ele soma as decisões sobre qual tipo de índices a usar para futura recuperação e qual estrutura para procurar espaço para índices. Isto é atualmente o problema da

aquisição de conhecimento e precisa ser analisado à parte da análise e modelagem do domínio do conhecimento.

3.4.1.8.3. Integração

Este é o passo final da atualização da base de conhecimento com um novo caso de conhecimento. Se nenhum caso e índice serão construídos, é o passo principal da retenção.

Pela modificação dos índices dos casos existentes, os Sistemas RBC aprendem a tornar a avaliação mais similar. O ajuste dos índices existentes é uma importante parte da aprendizagem do RBC. Neste caminho, a estrutura de índices tem o papel de ajustar e adaptar a memória de casos para esse uso. O caso recém aprendido pode ser finalmente testado, reentrando o problema inicial e vendo o que o sistema recupera.

3.5. AVALIAÇÃO DE RBC COMO FERRAMENTA DE REPRESENTAÇÃO DE CONHECIMENTO E INFERÊNCIA

Como em outros modelos, neste também há possibilidade de verificar os benefícios e desvantagens que o Raciocínio Baseado em Casos traz consigo KOLODNER (1993), ABEL (1996), WATSON (1994), MARTIN (1995).

Vantagens de RBC:

- Fornecem uma amostragem significativa do tipo de problemas que o sistema deve resolver;

- Facilita a aquisição de conhecimento, especialmente em domínios pouco estruturados e muito complexos, mesmo antes de uma perfeita compreensão do domínio;

- A grande vantagem é que permite a reutilização de conhecimento armazenado em banco de dados e outras fontes;

- Permite o encapsulamento da descrição do conhecimento com a solução aplicada;

- Realizam a manutenção do banco de conhecimento por aprendizagem automática de novos casos com pouca ou nenhuma interferência de um engenheiro de conhecimento.

Desvantagem de RBC:

- Dificilmente os casos estão disponíveis de forma confiável em quantidade suficiente e com boa representatividade sobre o domínio;

- Os algoritmos de recuperação por similaridade não mostram bom desempenho em aplicações reais, onde as comparações feitas por um especialista tendem a ser muito mais sofisticadas e de difícil compreensão;

- Não existem bons algoritmos de adaptação que permitam ao sistema fornecer soluções adequadas ao problema apresentado pelo usuário;

- Sistemas de RBC só obtém bom desempenho quando combinados com o raciocínio baseado em modelos que compensa as falhas na recuperação e adaptação dos casos

4. PROPOSTA DO MODELO LÓGICO EM ANÁLISE ESTRUTURADA

Construir softwares não se restringe a uma questão de escrever uma grande quantidade de software. O segredo estará em criar o código correto e pensar em como será possível elaborar menos software. Isso faz com que o desenvolvimento de software de qualidade se torne uma questão de arquitetura, processo e ferramentas (BOOCH, 2000).

Os modelos fornecem uma cópia do projeto de um sistema. Os modelos poderão abranger planos detalhados, assim como planos mais gerais com uma visão panorâmica do sistema considerado. Um bom modelo inclui aqueles componentes que têm ampla repercussão e omite os componentes menores que não são relevantes em determinado nível de abstração. Todos os sistemas podem ser descritos sob diferentes aspectos, com a utilização de modelos distintos e cada modelo será, portanto, uma abstração semanticamente específica do sistema (BOOCH, 2000).

Este capítulo apresentará o desenvolvimento do modelo proposto , em análise estruturada, sendo apresentado o modelo Entidade-Relacionamento com suas chaves e atributos, bem como, o Dicionário de Dados que constará o nome das tabelas, nome dos campos, tipo dos campos, tamanho dos campos, relacionamentos e tabela de índices.

Ao final do capítulo, será feita a consideração final do modelo desenvolvido em análise estruturada.

4.1. MODELO ENTIDADE-RELACIONAMENTO

Também conhecido como diagrama DER ou E-R, é um modelo em rede que descreve a diagramação dos dados e armazenados de um sistema em alto nível de

abstração. É uma eficaz ferramenta de modelagem para comunicação com o grupo de administração de banco de dados que pode iniciar a verificação de que tipo de chaves, índices ou ponteiros serão necessários para obter acesso eficiente aos registros dos bancos de dados (YOURDON, 1992).

O DER pode ser uma valiosa ferramenta para qualquer sistema com múltiplos depósitos (objetos) e complexos relacionamentos de dados. Ele é inteiramente voltado para os relacionamentos de dados, sem oferecer quaisquer informações sobre as funções que criam ou utilizam os dados.

O modelo Entidade-Relacionamento desenvolvido em análise estruturada encontra-se gravado no CD-ROOM no Access, na pasta "Capítulo4" . O nome do arquivo é "Classes.mdb".

4.2. DICIONÁRIO DE DADOS

A expressão dicionário de dados é quase auto-explicativa. O dicionário de dados é uma listagem organizada de todos os elementos de dados pertinentes ao sistema, com definições precisas e rigorosas para que o usuário e o analista de sistemas possam conhecer todas as entradas, saídas, componentes de depósitos e cálculos intermediários (YOURDON, 1992).

O dicionário de dados define os elementos de dados da seguinte maneira:

- Descreve o significado dos fluxos e depósitos mostrados no DER;
- Descreve a composição de pacotes agregados de dados que se movimentam pelos fluxos, isto é, pacotes complexos que podem ser divididos em itens mais elementares;
- Descreve a composição dos pacotes de dados nos depósitos;
- Especifica os relevantes valores e unidades de partes elementares de informações dos fluxos de dados e depósito de dados;

• Descreve os detalhes dos relacionamentos entre os depósitos realçados em um diagrama E-R.

4.3 TABELAS DO MODELO

O Dicionário de Dados com tabelas do Modelo Entidade-Relacionamento desenvolvido em análise estruturada encontra-se gravado no CD-ROOM no Access, na pasta "Capítulo4" . O nome do arquivo é Dicionário de Dados.

4.4. CONCLUSÕES DO MODELO RELACIONAL

O mundo real é complexo e por mais que a Tecnologia da Informação se esforce em representá-lo por meio de modelos simplistas, o mundo continuará complexo.

Modelos de dados tradicionais, especialmente, os relacionais, são relativamente fáceis de serem implementados. O problema surge quando um desenvolvedor de aplicações tenta forçar o mundo real para dentro de um padrão que essas tecnologias relacionais simples possam entender.

O enfoque tradicional de modelagem baseia-se na compreensão desse sistema como um conjunto de programas que, por sua vez, executam processos sobre dados, dificultando o acompanhamento do usuário no decorrer do desenvolvimento do sistema pela necessidade de aprimoramento técnico.

O resultado é a proliferação da quantidade de tabelas, que interagem de forma complexa e que modelam precariamente os reais relacionamentos de dados. As conexões entre essas tabelas freqüentemente estão escondidas em programas de aplicação, fora da base de dados onde poderiam ser gerenciadas mais efetivamente.

Como o software é aberto e prevê o aumento de novas tabelas e novos recursos, a tendência é que se torne pesado e lento.

O modelo desenvolvido dentro de análise relacional dá uma visão limitada do modelo, sob o ponto de vista do DER e do diagrama de fluxo de dados (DFD) que se limita às funções do sistema, não disponibilizando as informações que a modelagem poderia fornecer para o desenvolvedor e o usuário.

5. MODELO LÓGICO DESENVOLVIDO EM ANÁLISE ORIENTADA A OBJETO - UNIFIED MODELING LANGUAGE - (UML)

A UML é a linguagem padrão para especificar, visualizar, documentar e construir artefatos de um sistema, podendo ser utilizada com todos os processos ao longo do ciclo de desenvolvimento e através de diferentes tecnologias de implementação. Buscou-se unificar as perspectivas entre os diversos tipos de sistemas e fases de desenvolvimento de forma que permitisse levar adiante determinados projetos que antes não eram possíveis pelos métodos já existentes (FURLAN, 1998).

A UML tem o objetivo de:

a) Fornecer aos usuários uma linguagem visual expressiva e pronta para o uso, visando o desenvolvimento de modelos de negócio;

b) Fornecer mecanismos de extensibilidade e de especialização para apoiar conceitos essenciais;

c) Ser independente de linguagens de programação e processos de desenvolvimento;

d) Prover uma base formal para entender a linguagem de modelagem;

e) Encorajar o crescimento no número de ferramentas orientadas a objeto no mercado;

f) Suportar conceitos de desenvolvimento de nível mais elevado, tais como: colaborações, estrutura de trabalho, padrões e componentes;

g) Integrar as melhores práticas.

5.1. DESCRIÇÃO DO CASO DE USO – FORNECEDOR

A descrição do Caso de Uso do Fornecedor, ou seja, como o fornecedor deve proceder para incluir sua tecnologia no modelo está descrito e gravado no CD-ROOM

no "Capítulo5". A descrição do caso está no formato "Word" como "CasousoForn.doc".

5.2. DIAGRAMA DE (USE-CASE) – FORNECEDOR

O Diagrama Use-Case do Fornecedor foi descrito utilizando a ferramenta Rational Rose. Encontra-se gravado no CD-ROOM no "Capítulo5" com o nome de Versão1(Fornecedor),mdl.

5.3. DIAGRAMA DE SEQÜÊNCIA – FORNECEDOR

O Diagrama de Sequência do Fornecedor foi descrito utilizando a ferramenta Rational Rose. Encontra-se gravado no CD-ROOM no "Capítulo5" com o nome de Versão1(Fornecedor),mdl.

5.4. DESCRIÇÃO DE CASO DE USO - CLIENTE

A descrição do Caso de Uso do Cliente, ou seja, como o Cliente deve proceder para consultar sua tecnologia no modelo está descrito e gravado no CD-ROOM no "Capítulo5". A descrição do caso está no formato "Word" como "CasousoCliente.doc".

5.5. DIAGRAMA DE USE-CASE- CLIENTE

O Diagrama Use-Case do Cliente foi descrito utilizando a ferramenta Rational Rose. Encontra-se gravado no CD-ROOM no "Capítulo5" com o nome de Versão1(Usuário),mdl.

O Diagrama de Sequência do Cliente foi descrito utilizando a ferramenta Rational Rose. Encontra-se gravado no CD-ROOM no "Capítulo5" com o nome de Versão1(Usuário),mdl.

5.6. DIAGRAMAS DE CLASSES

O Diagrama de Classes tanto do Fornecedor como do Cliente também foi descrito utilizando a ferramenta Rational Rose. Encontra-se gravado no CD-ROOM no "Capítulo5" com o nome de Versão1(Usuário),mdl e Versão1(fornecedor),mdl.

5.7. PROTÓTIPO DO MODELO PROPOSTO

O protótipo do modelo encontra-se gravado no CD-ROOM, na pasta "Protótipo", onde se localiza o arquivo executável "otto.exe" . O protótipo foi desenvolvido em Clarion. Não há necessidade de instalar o Clarion para que o programa rode na máquina, é só executar o programa.

5.8. CONCLUSÕES DO MODELO UML

Devido às suas características, o modelo multidimensional facilita a modelagem dos dados, pois permite representar estruturas reais complexas sem ignorar aspectos do mundo real e sem ter de forçar esses aspectos a assumirem uma forma que possa ser digerida pela tecnologia. Além disso, há consideráveis vantagens durante a execução de processamentos complexos.

A tecnologia de objetos também usa tipos de dados ricos que refletem

relacionamentos entre dados do mundo real. Com sua modularidade inerente e poderosa interoperabilidade, os objetos conseguem alavancar a produtividade do desenvolvedor.

Esse modelo oferece como principais benefícios:

a) Manter a modelagem do sistema e, em decorrência, sua automação o mais próximo possível de uma visão conceitual do mundo real;

b) Servir de base à decomposição e modelagem do sistema nos dados que é o elemento mais estável de todos aqueles que compõem um sistema de informação;

c) Oferecer maior transparência na passagem da fase de modelagem para a de construção através da introdução de detalhes, não requerendo uma reorganização do modelo;

d) Mostrar as fronteiras do sistema e suas funções principais utilizando atores e casos de uso;

e)Ilustrar a realização de caso de uso com diagrama de seqüência;

f) Representar uma estrutura estática do sistema utilizando diagramas de classe;

Além do que foi mostrado no modelo, há possibilidade de se utilizar outros diagramas, tais como: diagrama de interação, transição de estado, possibilidade de se utilizar estereótipos a partir do momento em que a semântica do modelo se torne mais complexa.

Enfim, a possibilidade de representar o modelo através de seus diagramas, permite ao desenvolvedor do modelo visualizar o software sob vários ângulos, ampliando assim, sua visão global.

5.9 CONCLUSÕES DO MODELO PROPOSTO

O modelo para auxílio na tomada de decisão em Engenharia de Sistemas desenvolvido possibilita ao fornecedor de tecnologia a oportunidade de divulgar e informar sua tecnologia com padrão de qualidade para o usuário.

Permitir ao Usuário a liberdade de investigar as tecnologias com clareza de informações para tomar decisões;

Ganho representativo para Fornecedor e Usuário no tempo e contato de novas tecnologias ou até mesmo, de novas versões do fornecedor disponíveis no mercado.

5.10 TRABALHOS FUTUROS

As perspectivas futuras na área de modelagem é promissora. Com este modelo para auxílio na tomada de decisão em Engenharia de Sistemas há possibilidade de:

a) - Desenvolver no modelo um Banco de Dados Inteligente (Raciocínio Baseado em Casos);

b) - Desenvolver um modelo dessa natureza aplicado a hardware;

c) - Desenvolver um modelo levando em consideração peopleware;

d) - Desenvolver um modelo que integre os três modelos (software, hardware e peopleware), na mesma base de dados utilizando inteligência artificial

e) - Desenvolver um modelo de gerenciamento de informações no escopo do software utilizando inteligência artificial;

f) - Desenvolver um modelo para o processo de normalização de tabelas utilizando inteligência artificial;

g) - Integrar os modelos propostos.

6. INTEFACE DO SISTEMA

Uma interface de utilização bem projetada é construída com base em princípios e processos de desenvolvimento centralizados nos usuários e suas tarefas. No entanto, qualidades mínimas podem ser identificadas e arranjadas de forma a constituir um conjunto coeso que venha a orientar as atividades de projeto e avaliação. O desenvolvimento desse conjunto usualmente é um processo heurístico baseado em experiência, mas que passa por validações e refinamentos até alcançar o nível de detalhamento desejado.

A esse conjunto de qualidades podemos dar o nome de critérios ou princípios. O emprego de princípios e critérios bem definidos é fundamental para o sucesso da concepção da interface [CYB97] porque, da mesma forma que permitem orientar de maneira coerente o projeto, pode, posteriormente, vir a servir como quesitos de avaliação de produtos.

A finalidade deste capítulo é mostrar a interface do sistema desenvolvido em análise relacional, como em análise orientada a objetos – UML. Nas duas modelagens é utilizado a mesma interface.

As telas do protótipo encontram-se gravadas no CD-ROOM na pasta "Capítulo5" com o nome de Telas.doc.

REFERÊNCIAS BIBLIOGRÁFICAS

AAMODT, Agnar, Enric Plasa: **Case Based Reasoning: Fundational Issues, Methodological Variations, and system approaches**, 1994, p. 65.

ABEL, M: **Um estudo sobre Raciocínio Baseado em Casos**. Trabalho Individual. Porto Alegre. CPGCC/UFRGS, 1996, p. 41.

BOOCH, Grady. Rumbaugh James, Jacobson Ival. **UML – Guia do Usuário**. Rio de Janeiro: Campus, 2000, p. 15.

CERÍOLA, Vicent Oswald. **Metodologia para Desenvolvimento de Sistemas em Ambientes de 4ª Geração**. São Paulo: Makron Books, 1994, p.78.

CASTRO, Alexandre Ramires de, Aline Pons Alves, Cassia Antunes & Maria Elizabeth Grass; apud, **Desenvolvendo um trabalho em Equipe com qualidade,** Developers Magazine. Janeiro/2001. p. 34.

DATE, C. J. **Banco de Dados Tópicos Avançados**. Rio de Janeiro: Campus, 1997, p. 56.

DUN, R., Ullmann, **Quality Assurance**, Prentice-Hall, 1990, p. 156.

FEIGENBAUM, V. **Total Quality Control**. McGraw Hill, 1991, p.76.

FREEDMAN, D.P. e G. M. Weiberg, Handbook of Walkthroughs, **Inspections and Tecnical Reviews**, 3a ed. Dorset House, 1990, p. 135.

FURLAN, José Davi. **Modelagem de objetos através da UML**. São Paulo: Makron Books, 1998, p. 45.

GLASSEY, Welch J. D. Inmon W. H. **Gerenciando Data Warehouse**. São Paulo: Makron Books, 1999, p. 167.

INMON, W.H. **Como Construir o Data Warehouse**. Rio de Janeiro: Campus, 1997, p. 16.

ISO/IEC 12207-1, Software life-cycle process; mês / 1994 (DIS).

ISO9000 – Normas de Gestão da Qualidade e Garantia da Qualidade – Diretrizes para Seleção e Uso.

ISO/IEC 9126-1, International Standard, Information Technology – Software quality characteristics and metrics – Part 1: Quality characteristics and sub-characteristics; Jan / 1997.

ISO/IEC 14598-4, International Standard Information Technology – Software product evaluation – Part 4: Process for acquirers; Sep / 1996 (CD).

KANO, N., Seraku, N., Takahashi. F., Tsuji, S. – **Atrractive Quality and Must-Be Quality** (Jan. 1984) – in TQM – Ten Elements for Implementation – Apostila de Curso da Goal-QPC. 1991.

KHOSHAFIAN, Setrag. *Insight Into Object-Oriented Databases*. **Information and Software Technology**. Vol. 32, nº 4. May 1990.

KOLODNER, J.: **Case-Based Reasoning. San Francisco**: Morgan Kaufmann Publishers, 1993, p. 69.

LEVSON N. G., **Software Safety in Embedded Computer** Petri Nets", IEEE Trans. Software Engineering, vol. SE-13, num. 3, 1991.

MARIR, Farhi and WATSON, Ian. Case-based reasoning: a categorizes bibliography. **The Knowledge Engineering Review**, Vol 9:4, 1994. p. 355-381.

MARTIN, James. **ODELL**, J. James. **Análise e Projeto Orientados a Objetos**. São Paulo: Makron Books, 1995, p. 98.

MCCALL, J., P. RICHARDS e G. WALTERS, apud. **Factors in Software Quality**, três volumes, NTIS AD-AO49-014,055, nov/1977.

MICROSOFT 95 Resource Kit. **O consultor profissional do windows** 95, Rio de Janeiro: Campus, 1995, p. 210.

PORTER, B. And Bareiss, R. (1986) PROTOS: **An experiment in Knoledge acquisition for heuristic classification tasks in: Proceedings of the First International Meeting on Advances in Lerning (MAL)**, Les Arcs, France, pp, 159-174.

PRESSMAN. S. Roger. **Engenharia de Software**. São Paulo: Makron Books, 1995. p. 340.

QUINLAN, J.R.: **Induction of Decision Trees**. Machine Learning, 1(1), 81-106, 1986.

ROOK, J., Software Handbook, Elsevier, 1990, p. 250.

SCHANK, R: **Dynamic Memory: A Theory of Learning in Computers and People. Canbridge University Press**, 1982, p. 246.

WATSON, Ian and MARIR, Farhi Case-based reasoning a review. **The Knowledge Engeneering Review**, Vol 9:4, 1994, p. 327-354.

WATSON, Ian: Applyng Case Base Reasoning: Techniques for Enterprise Systems, 1997, p. 120

WANGENHEIM Von Gresse, Cristiane, Wangenheim Von , Aldo. **.Mensuração no melhoramento da qualidade de software**. Developers Magazine dezembro/2000 p. 29.

YOURDON, Edward. **Análise Estruturada Moderna.** São Paulo: Campus, 1992, p. 170.

VALLE, Arthur, Marciniuk Marlon, Melhoretto, Sandro Marcelo & Burnett, Roberto: **Um Roadmap para Métricas de Software: Definições e Histórico**. Developers Magazine, set/2000 p. 30.