Universidade Federal de Santa Catarina - UFSC
Curso de Pós-Graduação em Ciências da Computação

METODOLOGIA PARA ESPECIFICAÇÃO DE SISTEMAS EM AMBIENTE CLIENTE/SERVIDOR ORIENTADA A OBJETOS

por

Gilberto Grandi

Dissertação submetida à Universidade Federal de Santa Catarina para a obtenção do grau Mestre em Ciências da Computação

Profª. Elizabeth S. Specialski, M Sc.
Orientadora

Florianópolis, Março de 1996

METODOLOGIA PARA ESPECIFICAÇÃO DE SISTEMAS EM AMBIENTE CLIENTE/SERVIDOR ORIENTADA A OBJETOS

GILBERTO GRANDI

ESTA DISSERTAÇÃO FOI JULGADA ADEQUADA PARA A OBTENÇÃO DO TÍTULO DE

MESTRE EM CIÊNCIA DA COMPUTAÇÃO

ESPECIALIDADE SISTEMAS DE COMPUTAÇÃO E APROVADA EM SUA FORMA FINAL PELO PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Elizabeth Specialski

Profª. Elizabeth Sueli Specialski, M.Sc.
Orientadora

Prof. Rogério Cid Bastos, Dr.
Coordenador do curso

BANCA EXAMINADORA:

Elizabeth Specialski

Profª. Elizabeth Sueli Specialski, M Sc.

Prof. Fernando Álvaro Ostuni Gauthier, Dr.

Prof. Rogério Cid Bastos, Dr.

A

Bruno, Caroline, Myrnaia e Tatiane

AGRADECIMENTOS

Agradeço a todas as pessoas que contribuíram para o desenvolvimento deste trabalho.

Meu agradecimento especial à professora Elizabeth Specialski, minha orientadora, por ter acreditado e apoiado o presente trabalho. A Otacilio Angelo Bortolon pelo incentivo e apoio.

Agradecimento aos empregados do departamento de informática da ELETROSUL e em especial a Renato T. Scoz, Eduardo M. Wageck, Célio Silvestre, Antonio Manuel Tavares e Emerson R. Baldi.

Agradeço também aos colegas do mestrado, especialmente a Marcos Santos Zarbato, Neilor A. Tonin e Nery Artur Eller.

A ELETROSUL por ter viabilizado este curso.

A UFSC, professores e funcionários, especialmente a nossa secretária Verinha.

AS BASES

... Não existe regra fixa quanto a abordagem. Tudo muda, tudo evolui. A perspectiva histórica apenas proporciona mais autenticidade ao que se faz.

Sigiswald Kuijken (50) violinista belga

SUMÁRIO

Lista de figuras e tabelas ..... **IX**
Lista de abreviaturas ..... **XIII**
Resumo ..... **XIV**
Abstract ..... **XV**

1. Introdução **1**
1.1 Considerações iniciais 1
1.2 Objetivos do Trabalho 2
1.3 Delimitação do problema 2
1.4 Estrutura do trabalho 2

2. Conceitos básicos **3**
2.1 Metodologia 3
2.2 Conceitos de orientação a objetos 5
2.3 Características da orientação a objetos 9
2.4 Conceitos de Cliente/Servidor 10

3. Metodologia **12**
3.1 Análise 12
3.2 Projeto 13
3.3 Notação gráfica 15

4. ANÁLISE **18**
4.1 Definição do problema 19
4.2 Identificar classes e objetos 19
4.2.1 Como encontrar classes e objetos 20
4.2.2 Conservar classes corretas 23
4.2.3 Refinar classes utilizando a herança 24
4.2.4 Registrar as classes 24
4.3 Identificar atributos 25
4.3.1 Como definir os atributos 26
4.3.2 Conservar os atributos corretos 28
4.3.3 Normalizar 29
4.3.4 Registrar os atributos 30
4.4 Projetar o modelo de objetos 30
4.4.1 Ligações e associações 30

4.4.2 Conservar as associações corretas 34
4.4.3 Registrar o modelo 35
4.5 Identificar estruturas 36
4.5.1 Generalização-Especialização 37
4.5.2 Como aplicar a generalização 38
4.5.3 Agregação 40
4.5.4 Generalização versus herança 41
4.5.5 Agregação versus associação 42
4.5.6 Agregação versus generalização 42
4.5.7 Generalização como extensão e restrição 43
4.5.8 Registrar as estruturas 44
4.6 Identificar serviços 44
4.6.1 Identificar serviços 45
4.6.2 Identificar conexões de mensagens 46
4.6.3 Diagrama de fluxo de dados 47
4.6.4 Como construir o DFD 47
4.6.5 Registrar os serviços 52
4.7 A iteração da análise 53
4.8 Documentação da análise 53

5. Projeto 54
5.1 Projeto de objetos 54
5.1.1 Projetar os algoritmos 55
5.1.2 Otimizar o projeto 58
5.1.3 Implementar o controle 60
5.1.4 Ajustar a herança 63
5.1.5 Projetar as associações 67
5.1.6 Empacotamento físico 70
5.2 Projetar a interface humana - GUI 73
5.2.1 A importância de uma interface 73
5.2.2 Análise e projeto da interface 74
5.2.3 Cores e fontes 80
5.2.4 Sistema visual padrão 86
5.2.5 Projeto de ícones 94
5.2.6 Mensagens 97
5.2.7 Multimídia 97

5.3 Distribuição de processos 99
5.3.1 Tipos de processos 99
5.3.2 Equilibrar o processamento 100
5.3.3 Distribuir a carga de processamento 108
5.4. Distribuição de dados 109
5.4.1 Movimentação de dados 110
5.4.2 Formas de distribuição de dados 112
5.4.3 Fatores que influenciam a distribuição 120
5.5 A documentação do projeto 125

6. Conclusão 126

Bibliografia 128

LISTA DE FIGURAS E TABELAS

Figura 2.1 Topologia de uma Rede com Clientes e Servidores, ... 10
Figura 2.2 Interação Usuário e Cliente/Servidor, ... 11

Figura 3.1 Visão geral da análise proposta, ... 12
Figura 3.2 Visão geral do projeto, ... 14
Figura 3.3 Notação de classe de objetos, ... 16
Figura 3.4 Notação para classe, ... 16
Figura 3.5 Notação para cardinalidade, ... 16
Figura 3.6 Notação para Herança, ... 16
Figura 3.7 Notação para Generalização, ... 16
Figura 3.8 Notação para a utilização de Agregação, ... 17
Figura 3.9 exemplo de modelo E-R, ... 17

Figura 4.1 Visão geral do processo de análise [RUM 94], ... 19
Figura 4.2 Estruturas, ... 21
Figura 4.3 Papéis executados, ... 22
Figura 4.4 Agrupar classes específicas, ... 23
Figura 4.5 Exemplo de associações, ... 31
Figura 4.6 Associação ternária, ... 31
Figura 4.7 Atributos de ligação versus atributos de objeto, ... 32
Figura 4.8 Modelagem de uma associação de classe, ... 32
Figura 4.9 Nomes de papéis para uma associação, ... 33
Figura 4.10 Uma associação qualificada, ... 33
Figura 4.11 Relacionamento de controle, ... 33
Figura 4.12 Relacionamento dependente, ... 34
Figura 4.13 Associações não-redundantes, ... 35
Figura 4.14 Exemplo de Estruturas, ... 36
Figura 4.15 Hierarquia com Generalização/Especialização, ... 37
Figura 4.16 Estrutura Gráfica Generalização-Especialização, ... 38
Figura 4.17 Estrutura Gen-Esp de Indivíduos, como hierarquia, ... 39
Figura 4.18 Gen-Esp de Indivíduos, na forma de um entrelaçamento, ... 40
Figura 4.19 Agregação, ... 40
Figura 4.20 Agregação multinivelada, ... 41
Figura 4.21 Herança a partir da generalização, ... 41
Figura 4.22 Agregação e associação, ... 42

Figura 4.23 Agregação e generalização, ... 43
Figura 4.24 Modelo para objetos, ... 44
Figura 4.25 Representação de mensagem, ... 46
Figura 4.26 Diagrama de fluxo de dados para exibição de gráficos, ... 47
Figura 4.27 Valores de Entrada e Saída, ... 48
Figura 4.28 DFD de nível mais elevado para ATM, ... 48
Figura 4.29 DFD do processo executar transação para ATM, ... 49
Figura 4.30 Descrição da função atualizar conta, ... 50
Figura 4.31 Processos, ... 50
Figura 4.32 Fluxo de dados para subdividir um valor, ... 51
Figura 4.33 Depósitos de dados, ... 51
Figura 4.34 Fluxo de controle, ... 52

Figura 5.1 Fragmento de modelo, ... 56
Figura 5.2 Cadeia de associações, ... 58
Figura 5.3 Índice para banco de dados, ... 59
Figura 5.4 A associação como um cache, ... 60
Figura 5.5 Gabarito para componentes de gerenciamento de tarefas, ... 62
Figura 5.6 Acomodar herança única (1), ... 66
Figura 5.7 Acomodar herança única (2), ... 66
Figura 5.8 Herança, ... 67
Figura 5.9 Acomodar herança zero, ... 67
Figura 5.10 Implementação de associação, ... 68
Figura 5.11 Associação de dupla direção, ... 69
Figura 5.12 Esboço de menu inicial, ... 77
Figura 5.13 Agrupar em blocos, ... 78
Figura 5.14 Classes de interação humana, ... 80
Figura 5.15 Botões em 3-D, ... 81
Figura 5.16 Contorno em preto, ... 81
Figura 5.17 Onde as cores são focadas, ... 82
Figura 5.18 Propriedades das cores, ... 82
Figura 5.19 Tipos de cores no Windows, ... 85
Figura 5.20 Tipos e tamanhos de fontes, ... 86
Figura 5.21 Tipos de botões, ... 86
Figura 5.22 Botões gráfico com texto em baixo, ... 88
Figura 5.23 Botões gráficos com texto ao lado, ... 88
Figura 5.24 Botão de deslizar, ... 89
Figura 5.25 Botão de rotação, ... 89

Figura 5.26 Botão de rádio, ... 89
Figura 5.27 Botão de múltipla escolha, ... 89
Figura 5.28 Botão push button, ... 90
Figura 5.29 Menu Pull-down e menu em cascata, ... 90
Figura 5.30 Menu flutuante e menu destacável, ... 91
Figura 5.31 Barra de ferramentas, ... 91
Figura 5.32 Lista drop-down e lista de controle, ... 92
Figura 5.33 Lista drop-down, ... 92
Figura 5.34 Caixa de texto, ... 93
Figura 5.35 Tipos de cursores, ... 93
Figura 5.36 Tipo de ícone para apagar informações, ... 94
Figura 5.37 O desenho de um ícone, iniciando em preto e branco, ... 95
Figura 5.38 Cores em ícones, ... 95
Figura 5.39 Origem da luz, ... 96
Figura 5.40 A utilização do antialias, ... 96
Figura 5.41 Os resultados do antialias, ... 96
Figura 5.42 Exemplo de mensagem, ... 97
Figura 5.43 Formas de distribuição de processos, ... 99
Figura 5.44 Equilíbrio de Processamento [REN 94], ... 101
Figura 5.45 Gerenciamento de Sistemas Ponta-a-Ponta, ... 103
Figura 5.46 Funções de Gerenciamento de Rede, ... 105
Figura 5.47 Perímetros de Segurança, ... 105
Figura 5.48 A Chamada de Procedimento Remoto, ... 106
Figura 5.49 O Ciclo da análise de carga, ... 108
Figura 5.50 Arquivos centralizados, ... 112
Figura 5.51 Tipo de Dados Distribuídos, ... 113
Figura 5.52 Sistemas cliente/servidor de duas camadas, ... 113
Figura 5.53 Arquivos distribuídos em forma de réplica, ... 114
Figura 5.54 Arquivos distribuídos em forma de réplica periódica, ... 115
Figura 5.55 Arquivos distribuídos em forma de réplica contínua, ... 116
Figura 5.56 Arquivos distribuídos em forma de réplicas atualizada, ... 116
Figura 5.57 Arquivos particionados, ... 117
Figura 5.58 Arquivos distribuídos em forma de extrato simples, ... 118
Figura 5.59 Arquivos distribuídos em forma de extrato por tempo, ... 118
Figura 5.60 Arquivos distribuídos em forma de extrato por intervalo, ... 119

Tabela 4.1 Palavras chaves para iniciar o nome de um atributo, ... 29

Tabela 5.1 As cores do VGA, ... 83
Tabela 5.2 A resolução de acordo com o monitor, ... 84
Tabela 5.3 Aparências de botões, ... 87
Tabela 5.4 Seis estados de um botão, ... 87
Tabela 5.5 Freqüência de acesso e fatores de afinidade de uma entidade, ... 111
Tabela 5.6 Vantagens e desvantagens das réplicas, ... 115
Tabela 5.7 Distribuir ou Centralizar, ... 120
Tabela 5.8 Pesos para avaliar a utilização da opção de arquivos, ... 121
Tabela 5.9 Pesos para avaliar a utilização da opção de arquivos, ... 121
Tabela 5.10 Pesos para avaliar a utilização da opção de arquivos, ... 121
Tabela 5.11 Pesos para avaliar a utilização da opção de arquivos, ... 122
Tabela 5.12 Pesos para avaliar a utilização da opção de extrato, ... 122
Tabela 5.13 Pesos para avaliar a utilização da opção de arquivos, ... 122
Tabela 5.14 Pesos para avaliar a utilização da opção de arquivos, ... 123
Tabela 5.15 Pesos para avaliar a utilização da opção de arquivos, ... 123
Tabela 5.16 Resumo das técnicas utilizadas para descentralizar dados, ... 124
Tabela 5.17 Resultado de arquivo para cada local da empresa, ... 124
Tabela 5.18 Freqüência de operações, ... 125

LISTA DE ABREVIATURAS

1FN	Primeira Forma Normal
2FN	Segunda Forma Normal
3FN	Terceira Forma Normal
3-D	Terceira Dimensão
ACID	Atomicidade, Consistência, Isolação e Durabilidade
ACK	Acknowledge character
AOO	Análise Orientada a Objetos
ATM	Automated Teller Machine
CIH	Componente de Interação Humana
C/S	Cliente Servidor
CRUD	Create, Read, Update e Delete
DFD	Diagrama de Fluxo de Dados
DOO	Projeto Orientado a Objetos
EGA	Enhanced Graphic Adapter
E-R	Entidade Relacionamento
OO	Orientação a Objetos
Gen/Esp	Generalização/Especialização
GUI	Graphical User Interface
I/O	Input/Output
POO	Programação Orientada a Objetos
RPC	Report Procedure Call
SGBD	Sistema de Gerenciamento de Banco de Dados
VGA	Vídeo Graphic Adapter

RESUMO

As regras da tecnologia da informação nas empresas, atualmente, não são simplesmente suportar os processos administrativos existentes, mas criar vantagens competitivas através do uso de novas técnicas. Hoje o uso efetivo e eficiente da tecnologia da informação é um fator significativo para o sucesso ou fracasso em qualquer organização.

A tecnologia orientada a objetos é um nova técnica com potencial significativo de benefícios para as empresas juntamente com a utilizações de redes usando a arquitetura cliente/servidor.

Este trabalho refere-se a uma metodologia para a construção de sistemas com os conceitos da orientação a objetos e voltada as característica da arquitetura cliente/servidor.

A metodologia descreve as fases do ciclo de vida de um objeto, a distribuição de dados e de processos na arquitetura cliente/servidor incluindo a utilização de uma interface gráfica com o usuário.

Acredita-se que este trabalho possa ajudar as empresas a construir com sucesso sistemas utilizando estas tecnologias.

ABSTRACT

Information technology rules in companies, presently, are not simply to support the existing administrative processed, but to create competitive advantages through the use of new techniques. Today, the effective and efficient use of information technology is a significant factor for the success or failure in any organization.

The object oriented technology is a new technique with a significant benefit pontencial for companies, along with net utilizations using client/server architecture.

This paper refers to a methodology for system construction with object orientation concepts and turned to client/server architecture.

The methodology describes the fases of an object's life cycle, the data distribution, and processes in the client/server architecture including the use of a Graphical User Interface.

It is believed that this paper can help companies to build systems successfully by using these technologies.

1. INTRODUÇÃO

1.1 CONSIDERAÇÕES INICIAIS

Esta década trouxe um completo remodelamento dos fundamentos dos sistemas de informação nas empresas. Os tradicionais sistemas centralizados estão sendo questionados. A Tecnologia da Informação está caminhando rapidamente para adaptar-se às novas mudanças no mundo dos negócios. A competição global está dirigindo as empresas na busca de uma crescente satisfação de seus clientes e usuários, buscando melhores produtos e serviços orientados aos usuários/consumidores com custos mínimos.

No livro *Visão 2020* [DAV 91] os autores prevêem que os negócios serão todos informatizados, indicando que a informação terá importância cada vez maior. A organização da empresa será mais achatada, eliminado gerências intermediárias. A ênfase estará na utilização de trabalhadores preparados em todos os níveis empresariais, dando-lhes autonomia e ferramentas de produtividade.

O verdadeiro suporte da informática às atividades empresariais são os dados, em primeira instância. O hardware e o software, ou mesmo a arquitetura utilizada, são aspectos secundários, que influenciam indiretamente os resultados do negócio.

Ao se aplicar novas Tecnologias da Informação nas empresas, o foco deverá estar na infra-estrutura necessária, o que não significa apenas adquirir tecnologia, mas envolve mudanças culturais básicas para entender e aceitar as mudanças propostas.

A arquitetura cliente/servidor (C/S) e a Orientação a Objetos (OO) estão sendo apontadas como as tecnologias dos anos 90. Os PCs e as redes locais neles baseadas, amadureceram e se tornaram tecnologias seguras para importantes aplicativos comerciais. A análise de sistemas e o projeto orientados a objetos juntamente com a programação são ferramentas que podem fazer um software mais consistente e produtivo. Sendo assim, a arquitetura C/S é o próximo passo lógico para a utilização do poder crescente dos sistemas em microcomputador e das redes, permitindo a distribuição de dados e de processos. A orientação a objetos é uma técnica para construir sistemas nesta arquitetura, sendo mais produtiva e consistente.

1.2 OBJETIVOS DO TRABALHO

O presente trabalho propõe a especificação de uma metodologia para o desenvolvimento de sistemas baseados no paradigma cliente/servidor e orientados a objetos. A metodologia aborda algumas partes do ciclo de vida de um sistema: a análise orientada a objetos e o projeto orientado a objetos na arquitetura C/S.

1.3 DELIMITAÇÃO DO PROBLEMA

Os propósitos básicos desta metodologia permitem aos analistas utilizar uma ferramenta para a construção de sistemas em qualquer área da empresa (incluindo áreas descentralizadas) com qualidade, produtividade e disponibilidade nas informações.

Algumas características básicas da metodologia proposta são:

- Utilização de tecnologia Orientada a Objetos;
- Sistemas implementados usando o paradigma Cliente/Servidor;
- Especificação de Graphical User Interface (GUI);
- Análise e distribuição de dados;
- Análise e distribuição de processos.

1.4 ESTRUTURA DO TRABALHO

O documento está organizado em 6 capítulos. No capítulo 2 são apresentados conceitos básicos sobre metodologia, análise, projeto, orientação a objetos e arquitetura cliente/servidor.

O capítulo 3 descreve a metodologia e estabelece uma notação gráfica. O capítulo 4 detalha a parte da metodologia proposta, referente aos mecanismos da análise orientada a objetos. O capítulo 5 descreve a parte da metodologia proposta, referente aos mecanismos do projeto baseado em objetos e cliente/servidor. A conclusão do trabalho está no capítulo 6.

2. CONCEITOS BÁSICOS

Neste capítulo são especificados conceitos sobre metodologia, análise, projeto, orientação a objetos e cliente/servidor.

2.1 METODOLOGIA

Uma metodologia para desenvolver sistemas especifica todos os passos que um analista deve dar para construir um sistema. Neste contexto, uma metodologia para desenvolvimento de sistemas refere-se aos componentes iniciais do ciclo de vida do software: análise, projeto e implementação.

A metodologia baseada na Análise Estruturada, proposta por Dikjstra em 1968, é um processo baseado em decomposição funcional onde os sistemas são compostos basicamente de subprogramas. Os procedimentos básicos do sistema são alcançados através do refinamento gradativo dos subprogramas. A desvantagem desta técnica é que ela obriga o programador a fixar sua atenção muito mais nos procedimentos do que nos dados.

Uma metodologia baseada na Orientação a Objetos é um modo de tratar o software com base em abstrações que existem no mundo real. A essência de uma metodologia baseada em objetos deve ser identificar e organizar os conceitos do domínio do problema a ser construído. A vantagem desta técnica é que ela permite um desenvolvimento dos sistemas, independente da linguagem de programação até as etapas finais. Além disso, a Orientação a Objetos pode servir como um meio para a especificação, análise, documentação, interface e para programação.

A definição tradicional das fases de um projeto de desenvolvimento de uma aplicação consiste de:

- definição do projeto e planejamento;
- análise (do negócio e do sistema);
- projeto do sistema (projeto lógico e projeto físico);
- construção do sistema (implementação, teste e documentação);
- testes de aceitação e ajustes;
- operação e manutenção.

A fase de definição do projeto e planejamento consiste na definição do escopo e dos limites do projeto a ser definido. Se a organização não possui um planejamento para a implantação de sistemas, ela deve fazê-lo antes de construir qualquer sistema. A finalidade do planejamento é estabelecer as prioridades de desenvolvimento e manutenção de sistemas em uma seqüência lógica, integrada e incremental.

A análise do negócio deve fornecer um entendimento da natureza dos serviços realizados pela organização e identificar as áreas específicas para as quais presume-se uma necessidade de implantação da tecnologia computacional, a fim de melhorar a forma com que o negócio vem sendo conduzido.

A análise dos requerimentos do sistema deve produzir um entendimento claro das capacidades que devem ser disponibilizadas pelo sistema a ser desenvolvido, baseada na utilização que o sistema terá na organização. Estas capacidades devem ser documentadas de tal forma que os desenvolvedores da aplicação tenham delineados os objetivos que devem ser alcançados e, após desenvolvido, o sistema possa ser validado quanto ao alcance destes objetivos.

O projeto lógico do sistema consiste na identificação dos componentes de software e hardware que satisfazem os requerimentos do sistema. Nesta fase devem ser identificados os relacionamentos entre estes componentes, evitando detalhes técnicos que forcem um mapeamento do projeto para um ambiente de implementação específico.

Na fase do projeto físico são tomadas decisões relativas a arquitetura de hardware específica, pacotes de Sistemas Gerenciadores de Bases de Dados, linguagens de programação e tipos de interfaces. É nesta fase que são tomadas decisões quanto à forma de distribuição dos dados, distribuição de objetos em um ambiente de rede e atribuição de papéis em um ambiente cliente/servidor.

A construção do sistema consiste de três fases: o desenvolvimento propriamente dito, os testes do sistema e a documentação do sistema. A fase de desenvolvimento consiste na utilização de uma linguagem de programação ou de um ambiente de desenvolvimento de aplicações para a implementação do projeto físico do sistema. Os testes são realizados para validar o sistema quanto ao alcance dos objetivos estabelecidos na fase de análise. A documentação do sistema pode ser realizada de diversas formas, seja através da construção de manuais e/ou de documentação técnica do software.

Os testes de aceitação e ajustes devem ser dirigidos tomando por base o modelo definido na fase de análise dos requerimentos do sistema.

A fase de operação e manutenção do sistema é uma atividade que deve ser executada com o controle da versão formal do sistema e com o apoio de uma gerência de configuração.

Este trabalho abordará a etapa de Projeto de Sistemas a que se refere a metodologia tradicional. O projeto de sistemas aqui especificado é dividido em duas etapas: a análise e o projeto de objetos.

2.2 CONCEITOS DE ORIENTAÇÃO A OBJETOS

A utilização de uma metodologia para especificação de sistemas orientada a objetos requer o estabelecimento de conceitos básicos que irão nortear a definição das etapas a serem estabelecidas. A bibliografia consultada [BOO 94], [COD 93], [JAC 92], [MON 94], [RUM 94] e [YOU 95] apresentam várias definições e notações.

Para Yourdon [YOU 95], objeto é uma instância da classe objeto. Um objeto combina a estrutura de dados e seu comportamento. O objeto possui um estado, que são seus dados e é capaz de responder a mensagens, usando as operações definidas para a classe objeto.

Para Booch [BOO 94], um objeto tem um estado, comportamento e identidade; a estrutura e comportamento de objetos similares são definidos em suas classes comuns; os termos instância e objeto são interrelacionados.

Para Jacobson [JAC 92], um objeto é caracterizado por um número de operações e de estados que lembram os efeitos destas operações.

Apesar de que as diversas definições não apresentarem conflitos, optou-se pela adoção das definições estabelecidas por Montenegro [MON 94]. O objetivo é o de garantir uma seqüência lógica de pensamento e consistência na estruturação da metodologia proposta.

Os elementos básicos definidos são: objeto, classe, classe objeto, instância, atributo, serviço, herança, herança múltipla, tipo de acesso, encapsulamento e polimorfismo.

Um objeto é a representação de uma entidade física ou lógica, real ou abstrata. Exemplos de objetos podem ser: um documento, um parágrafo de um documento, uma janela na estação de trabalho, um empregado da empresa, um fornecedor de produtos, um produto, etc.

A um objeto estão associados o seu **estado**, seu **comportamento** e sua **identidade**. O **estado** do objeto é caracterizado pelas propriedades (atributos) que o objeto possui e pelos valores que eles assumem. O **comportamento** define a forma como o objeto reage em conseqüência da mudança de valor em seus atributos e/ou de seu relacionamento com outros objetos. Na definição de comportamento de um objeto, devem ser estabelecidas as ações que podem ser executadas sobre o objeto e os procedimentos associados à estas operações. A **identidade** de um objeto é a propriedade pela qual ele se distingue dos demais. A característica de identidade de um objeto refere-se a uma única ocorrência.

Uma classe representa um conjunto de objetos semelhantes, isto é, objetos com a mesma estrutura de dados (mesmos atributos) e mesmo comportamento. Pode-se, portanto, definir uma classe de objetos "empregado", que represente o conjunto de todos os empregado de uma empresa. A definição de uma classe determinará a estrutura de cada objeto pertencente àquela classe; todos os objetos terão, portanto, os mesmos atributos e o mesmo comportamento. Cada objeto particular consiste de uma "instância" da classe à qual o objeto pertence. Por exemplo, considere a seguinte definição da classe "empregado":

CLASSE Empregado

ATRIBUTOS: Nome
Idade
CPF
Endereço
Cargo
Salário

OPERAÇÕES: Mudar-valor
Ler-valor
Eliminar-registro

Uma instância desta classe pode ser representada pelo objeto:

Nome = João das Quantas
Idade = 27
CPF = 142.135.470-84
Endereço = Rua das Palmeiras, 73, Conjunto A, apto. 333, Sertão Longínquo.
Cargo = Programador
Salário = 850,00

Deste exemplo, pode-se depreender que a diferença entre classe e objeto consiste no fato de que a classe apresenta uma definição estrutural em termos de atributos que os objetos da classe contém, operações que podem ser efetuadas sobre objetos da classe e o comportamento que os objetos da classe apresentam, enquanto que o objeto associa valores aos atributos definidos pela classe e está relacionado com alguma entidade do mundo real. Um objeto é, portanto, uma INSTÂNCIA de sua classe.

A definição das classes pode ser realizada através da utilização do mecanismo de HERANÇA. Este mecanismo permite que se defina uma nova classe reaproveitando definições de uma classe mais genérica, isto é, definindo-se apenas as diferenças resultantes de uma especialização. Como exemplo, considere a classe "empregado", (definida anteriormente) e a classe "cliente", definida a seguir:

CLASSE Cliente
ATRIBUTOS:
- Nome
- CGC/CPF
- Endereço
- Data-última-compra
- Valor-da-compra
- Saldo-devedor
- Vencimento

OPERAÇÕES:
- Mudar-valor
- Ler-valor
- Eliminar-registro

Estas duas classes (empregado e cliente) apresentam diversas características (atributos) e operações comuns. Para efeito de reaproveitamento, pode-se definir uma classe genérica

(por exemplo, "pessoa"), onde são definidas todas as características, comportamentos e operações em comum. Esta classe, assim definida, é denominada "classe superior" ou "classe pai", no contexto da hierarquia de herança. As classes "empregado" e "cliente" são derivadas da classe "pessoa" e denominadas "classe derivada", "classe filha" ou "subclasse". Assim, a definição de uma classe consiste na descrição de apenas aquelas características que não estão incluídas na classe superior da qual ela é derivada. Uma possível definição da classe "pessoa" é dada a seguir:

CLASSE Pessoa

ATRIBUTOS:

Nome
CGC/CPF
Endereço

OPERAÇÕES:

Mudar-valor
Ler-valor
Eliminar-registro

Considerando-se esta definição da classe "pessoa", pode-se obter uma significativa simplificação na definição das classes "empregado" e "cliente". Um modelo contendo estas definições está na figura 3.9.

CLASSE Empregado

DERIVADA DE Pessoa

ATRIBUTOS:

Idade
Cargo
Salário

CLASSE Cliente

DERIVADA DE Pessoa

ATRIBUTOS:

Data-última-compra
Valor-da-compra
Saldo-devedor
Vencimento

No exemplo, as classes empregado e cliente herdam características de uma única classe (pessoa); no entanto, é possível que existam situações onde uma classe herde características de mais de uma classe. Este mecanismo é referenciado como herança múltipla e

pode ser ilustrado com o exemplo de definição de uma classe "empregado-cliente", dada a seguir:

CLASSE empregado-cliente
DERIVADA DE empregado, cliente
ATRIBUTOS:
Desconto

Uma instância desta classe irá conter todos os atributos definidos na classe "empregado" e todos os atributos definidos na classe "cliente" mais o atributo Desconto, definido na própria classe. As operações permitidas para objetos desta classe são as mesmas operações herdadas pelas classes "empregado" e "cliente" da classe "pessoa".

Como pode ser visto neste exemplo, nem todas as classes definidas são instanciáveis, isto é, algumas classes são definidas apenas com o objetivo de agrupar definições comuns a mais de uma classe. Neste trabalho, usa-se o termo classe para identificar classes que não são instanciáveis e o termo classe objeto para identificar aquelas classes que verdadeiramente serão instanciáveis.

O comportamento de um objeto é estabelecido na definição da classe. Ele se refere à ações que devem ser realizadas quando algum evento ocorre. Estas ações podem ser traduzidas por execução de operações ou emissão de notificações e devem refletir um resultado coerente com os objetivos do sistema. Por exemplo, na classe "cliente" um comportamento pode ser a emissão de uma correspondência a um cliente específico, quando a data de vencimento de alguma compra for ultrapassada de 5 dias. Este comportamento pode ser especificado de forma textual ou com o auxílio de alguma ferramenta formal. Cada comportamento especificado para uma classe de objetos é denominado de "método" ou "serviço".

2.3 CARACTERÍSTICAS DA ORIENTAÇÃO A OBJETOS

As principais características apresentadas por uma metodologia de desenvolvimento de sistemas OO são Abstração, Encapsulamento, Herança e Polimorfismo.

A Abstração é o mecanismo que permite a representação de uma realidade complexa em termos de um modelo simplificado, de tal forma que, os detalhes irrelevantes podem ser suprimidos a fim de proporcionar uma compreensão global do sistema.

O Encapsulamento é uma propriedade apresentada por todos os objetos no sentido de que a implementação do objeto não fica visível aos outros componentes do sistema. Com isto, garante-se a integridade do objeto pois apenas podem ser executados procedimentos que foram previamente definidos quando a classe do objeto foi especificada.

A Herança, conforme já citado, é um relacionamento entre classes que permite que uma classe reutilize os atributos e operações definidos em outra classe mais genérica.

O Polimorfismo consiste na capacidade de se usar um mesmo método para diferentes implementações em diferentes níveis de uma hierarquia de classes. Cada classe tem um comportamento específico para o método. Polimorfismo é, portanto, a propriedade de um programa OO de discernir, dentre os métodos homônimos, aquele que deve ser executado. Um exemplo comum é a impressão, onde para cada classe na hierarquia pode haver pequenas mudanças. Ou a operação "mover" que pode atuar de forma diferente nas classes Janela e PeçaXadrez.

2.4 CONCEITOS DE CLIENTE/SERVIDOR

A arquitetura Cliente/Servidor (C/S) é composta por um ou mais servidores e por um ou mais clientes. Na verdade, o conceito C/S representa uma interação entre processos de software executando de forma concorrente, conforme pode ser visto na figura 2.1.

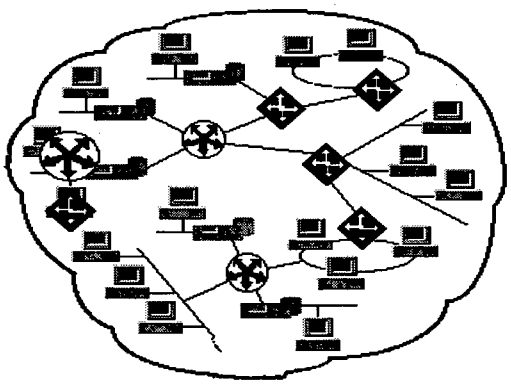


Figura 2.1 Topologia de uma Rede com Clientes e Servidores

O cliente é um processo que roda em uma estação de trabalho e que serve de interface para as aplicações do usuário. O processo cliente emite requisições para um processo servidor e aguarda a resposta da solicitação.

O servidor é um processo residente em um hardware com características específicas para dar suporte aos serviços que ele disponibiliza. Na arquitetura C/S, o servidor não pode iniciar nada, o cliente não pode responder a requisições de outros clientes. Existe uma clara função de diferenciação estabelecida na qual o cliente é o mestre, e o servidor, o escravo.

As interações a partir do cliente para o servidor são pedidos, as interações do servidor para o cliente são respostas, conforme demonstra a figura 2.2.0

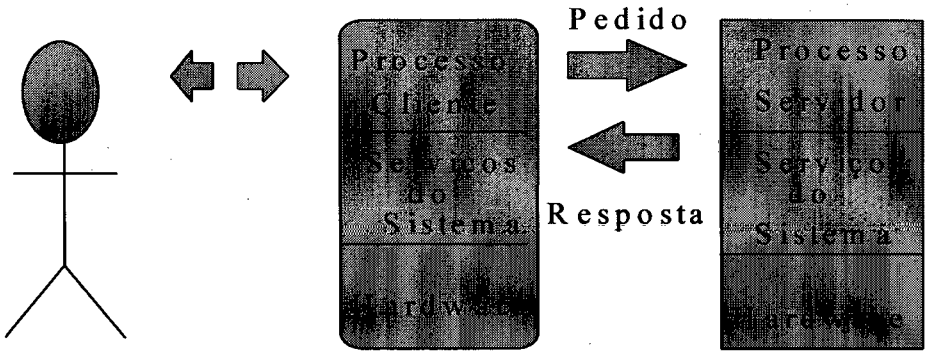


Figura 2.2 Interação Usuário e Cliente/Servidor

3. METODOLOGIA

Este capítulo descreve como será a análise de sistemas, o projeto de sistemas e a notação gráfica utilizada para a metodologia proposta.

3.1 ANÁLISE

Para efetuar a análise o analista estuda o enunciado do problema e constrói um modelo mostrando suas propriedades relevantes. Este modelo é uma abstração precisa e concisa do que o sistema deverá fazer, não como deverá fazê-lo. Os objetos do modelo devem ser conceitos do domínio da aplicação e não conceitos de implementação. A figura 3.1 mostra as fases da análise da metodologia proposta.

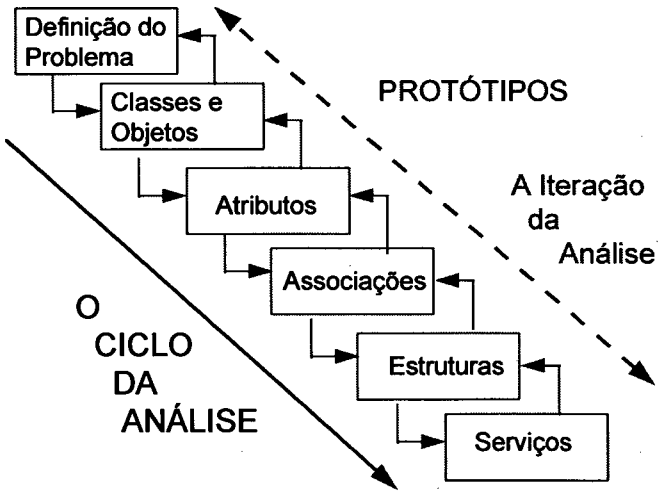


Figura 3.1 Visão geral da análise proposta

O ciclo da análise na metodologia possui as seguintes fases:

- Definição do Problema - descreve o enunciado do problema. A partir dele, o analista entrevista usuários, utiliza seus conhecimentos do domínio do problema e experiências do mundo real para construir o modelo de análise.

- Classe e Objetos - identifica as classes de objetos existentes no domínio do problema.

- Atributos - descreve os dados de cada classe objeto.

- Associações - identificam dependências entre classes. Os atributos que pertencem a uma associação geram uma nova classe.

- Estruturas - é um termo que indica a utilização das estruturas de agregação e generalização.

- Serviços - define os métodos efetuados pelos objetos. Os serviços são especificados através do Diagrama de Fluxo de Dados (DFD).

A meta da análise é especificar integralmente o problema e o domínio da aplicação sem introduzir desvios. Para efetuar a análise de acordo com a metodologia proposta deve-se seguir o ciclo da análise demonstrado na figura 3.1. Como a maioria dos modelos da análise exige mais do que uma etapa para ficar completa, esta metodologia propõe que deve-se atacar a análise iterativamente, preparando uma primeira aproximação para o modelo e, depois, repetir a análise à medida que o seu conhecimento se amplia. Não há necessidade de seguir os itens na ordem especificada. Os protótipos podem ajudar na especificação ou no entendimento do problema e podem auxiliar o analista a dar "feedback" aos seus usuários.

3.2 PROJETO

O projetista constrói um modelo de objetos baseado no modelo de análise, mas contendo os detalhes de implementação. As classes e objetos provenientes da análise são acrescidas das estruturas de dados referentes à linguagem de programação e do ambiente onde será implementada. A fase de análise determina o que a implementação deve fazer, e a fase do projeto de sistema como fazê-lo. O enfoque do projeto de objetos são as estruturas de dados e os algoritmos necessários à implementação de cada classe. Esta fase acrescenta objetos internos de apoio à fase de implementação e otimiza estruturas de

dados e algoritmos. O projeto de objetos é análogo à fase de análise de desenvolvimento de software. A figura 3.2 demonstra o ciclo do projeto na metodologia proposta.

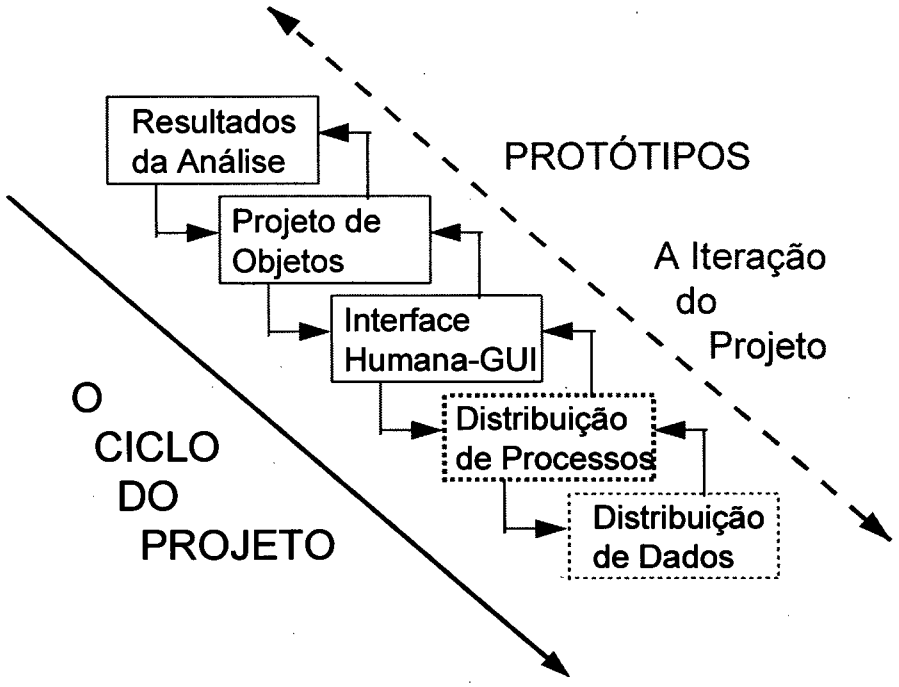


Figura 3.2 Visão geral do projeto

A fase do projeto proposto pela metodologia possui as seguintes etapas:

- Resultados da análise - a partir do resultado obtido na fase da análise, acrescentar detalhes de implementação dos objetos.

- Projeto de Objetos - tomar decisões a respeito da implementação dos objetos.

- Interface Humana GUI - nesta etapa será definida a interface gráfica com o usuário. Ela descreve o ambiente onde atuará o sistema, através da especificação de ícones, botões, listas e menus com os quais o usuário vai interagir.

- Distribuição de Processos - nesta etapa a metodologia prevê a realização de uma avaliação em termos de distribuição dos processos, buscando um equilíbrio de trabalho entre os clientes e os servidores, estabelecendo onde deve ficar o processamento.

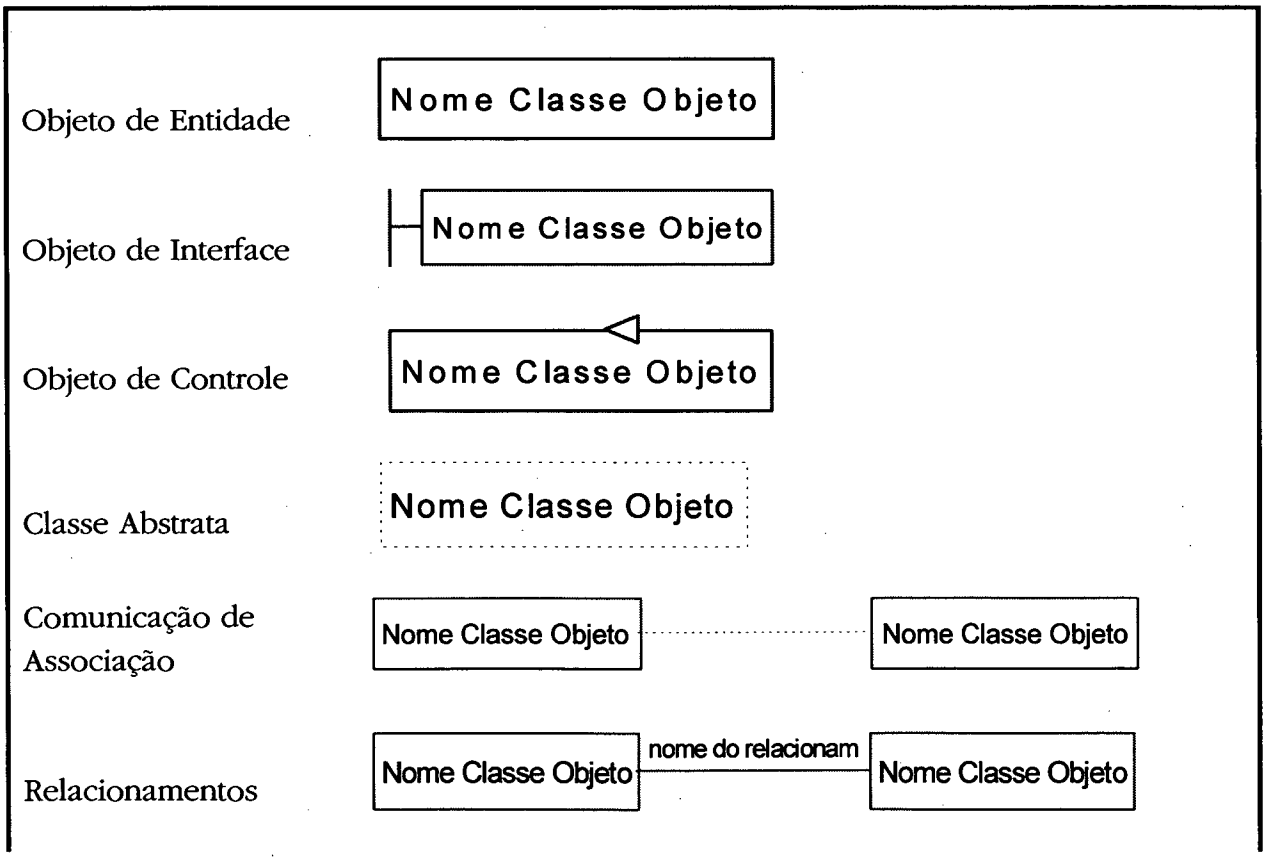
- Distribuição de Dados - nesta etapa a metodologia estabelece formas de distribuição de dados e regras com o objetivo de reduzir o tráfego de dados na rede.

Esta fase da metodologia proporciona uma base para a implementação. As etapas descritas no ciclo do projeto da figura 3.2 não precisam ser executadas em ordem. Assim como na análise, o projeto pode ser revisado e alterado várias vezes, efetuando-se uma iteração entre os seus componentes. Os protótipos podem mostrar retorno mais rápido das atividades junto ao usuário.

3.3 NOTAÇÃO GRÁFICA

A utilização de uma notação gráfica ajuda a visualizar os objetos e seus relacionamentos. A representação gráfica é uma forma amigável de modelar estruturas de objetos tais como: classes, diagramas estruturados de objetos, relacionamentos de herança, agregação, generalização e fluxo de mensagens. A notação desta metodologia foi baseada em Rumbaugh [RUN 94] [COD 92] e Yourdon [YOU 95] com alterações nas notações de herança, agregação e generalização. As alterações introduzidas visam facilitar a compreensão, pois alguns autores usam o mesmo símbolo com significados diferentes.

A figura 3.3 mostra os tipos de classes para especificar um Diagrama Estruturado.



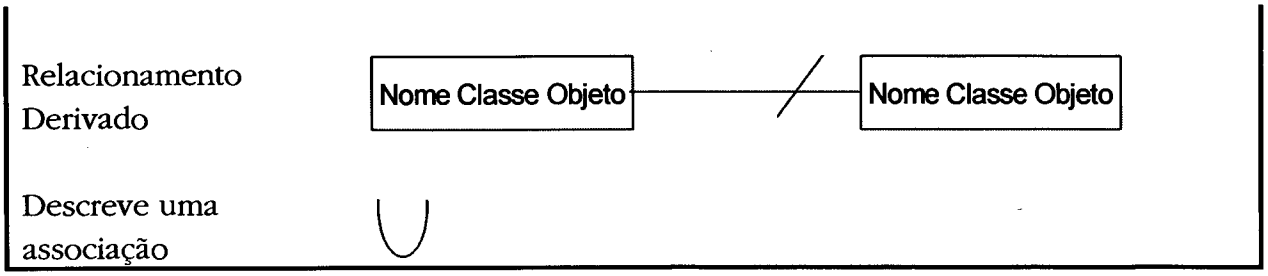


Figura 3.3 Notação de classe de objetos

A figura 3.4 ao lado mostra uma classe, dividida em três partes: Na primeira parte coloca-se o nome da classe. Na segunda parte o nome dos atributos e na terceira o nome das operações.

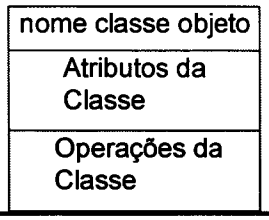


Figura 3.4 Notação para classe

A figura 3.5 especifica a quantidade e a opcionalidade entre relacionamentos de classes.

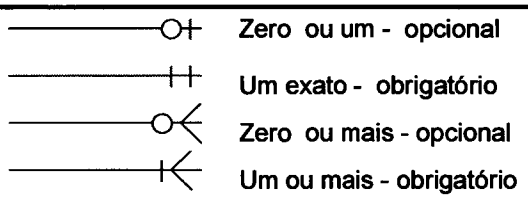


Figura 3.5 Notação para cardinalidade

A figura 3.6 ao lado especifica a notação para indicar a utilização de herança representada por um retângulo contendo um "H".

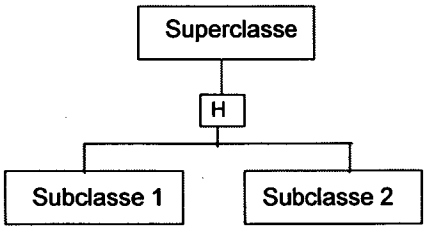


Figura 3.6 Notação para Herança

A figura 3.7 ao lado especifica a notação para indicar a presença de estruturas utilizando a generalização, representada por um retângulo com um "G".

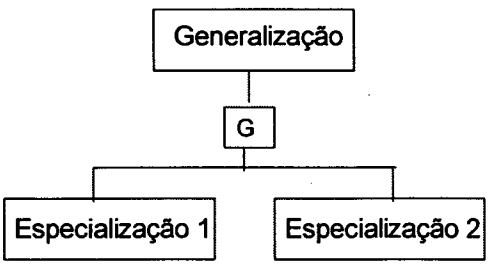


Figura 3.7 Notação para Generalização

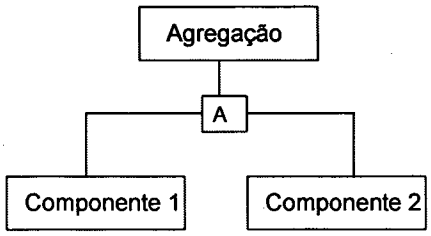
A figura 3.8 ao lado mostra a notação a ser utilizada na modelagem de objetos para indicar a presença de agregação, representada por um retângulo contendo um "A".	

Figura 3.8 Notação para a utilização de Agregação

A figura 3.9 mostra um exemplo de um modelo utilizando a notação gráfica.

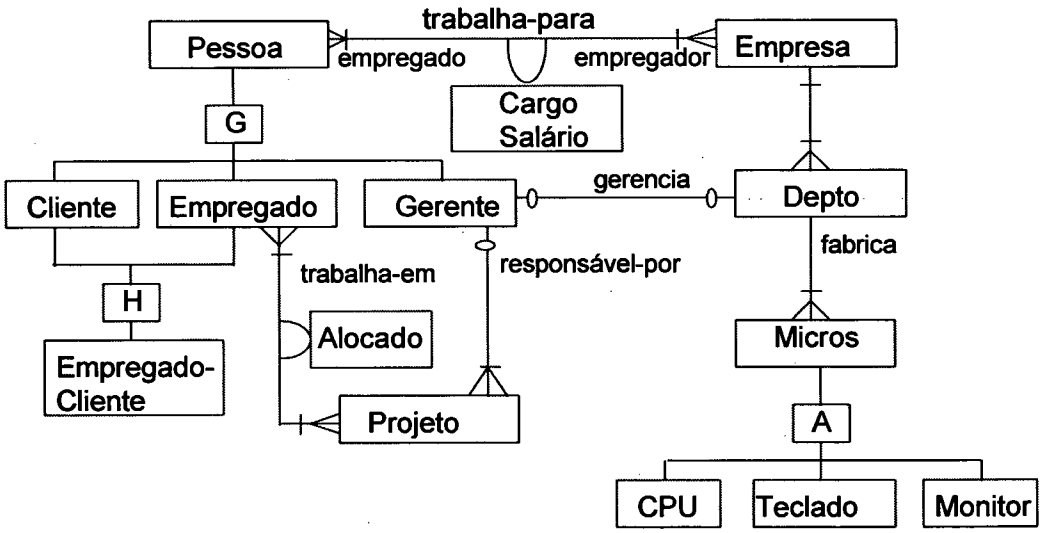


Figura 3.9 exemplo de modelo E-R

A figura 3.9 mostra que Cliente, Empregado e Gerente são generalizações de Pessoa. Empregado-Cliente é uma classe que herda características de Cliente e de Empregado. Muitas Pessoas trabalham em muitas Empresas. A associação entre Pessoa e Empresa se chama trabalha-para. Os atributos desta associação (cargo e salário) formam uma nova classe. A Pessoa exerce a função de empregado e a Empresa a função de empregador. A Empresa tem muitos Departamentos que podem ter um gerente e fabricam CPUs, Teclados e Monitores agregados ao Micros. O gerente pode ser responsável por muitos projetos. A associação entre Projeto e Empregado chama-se Alocado.

4. ANÁLISE

Análise é o estudo do problema antes de qualquer ação. É o estudo do domínio do problema, que leva a uma especificação. Analisar é o processo de extrair as necessidades do sistema.

Este capítulo descreve os passos e ferramentas necessários para efetuar a análise de um sistema orientado a objetos segundo a metodologia proposta. Ele está dividido em oito partes:

- Definição do problema;
- Classes e objetos;
- Atributos;
- Associações;
- Estruturas;
- Serviços;
- A iteração da análise;
- Documentação da análise.

4.1 DEFINIÇÃO DO PROBLEMA

A análise inicia com o enunciado do problema, conforme mostra a figura 4.1 extraída de Rumbaugh [RUM 94]. O enunciado pode ser incompleto e informal: a análise o torna mais preciso e expõe suas inconsistências. O enunciado do problema não deve ser visto como imutável e deve servir como base para o refinamento dos verdadeiros requisitos.

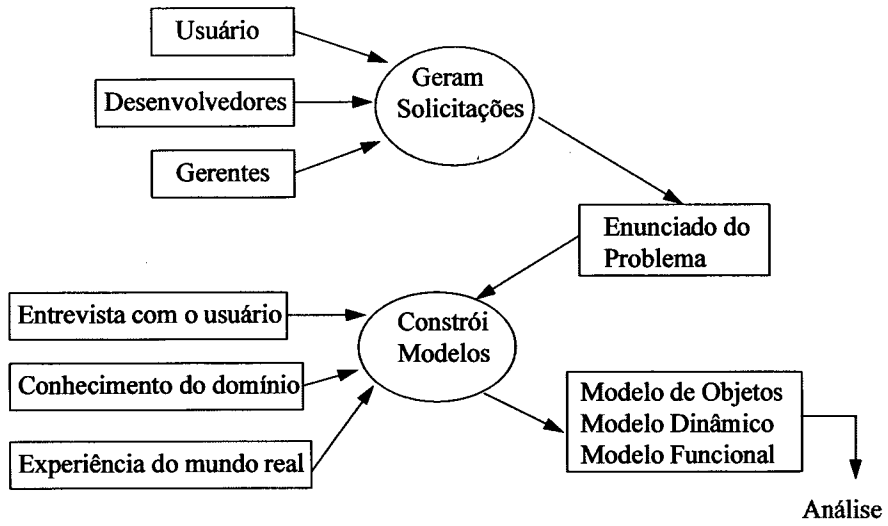


Figura 4.1 Visão geral do processo de análise [RUM 94]

O sistema do mundo real descrito na definição precisa ser compreendido, e suas características essenciais abstraídas em um modelo. Os enunciados em linguagem natural são muitas vezes ambíguos, incompletos e inconsistentes. O modelo de análise é uma representação concisa do problema, permite que se respondam algumas questões e se elabore uma solução.

4.2 IDENTIFICAR CLASSES E OBJETOS

Identificar Classes e Objetos em domínios [COD 92] de problemas auxilia a compreender e a comunicar o que está acontecendo. Os analistas precisam entender primeiro o domínio do problema. Definir Classes e Objetos como uma abstração do mundo real ajuda a entender e a conversar sobre o domínio do problema. As classes e objetos definitivos são encontrados após terminar o modelo de AOO, composto pelos níveis de classe e objeto, atributos, associações, estruturas e serviços.

Cada um destes níveis coloca novos detalhes, até que as classes estejam claramente definidas. As classes e objetos de um sistema não se alteram com o tempo, elas são relativamente estáveis; já os atributos e os serviços para os objetos nesta classe podem mudar radicalmente. O encapsulamento baseado no domínio do problema ajuda a reduzir a volatilidade e o trabalho subseqüente. O encapsulamento de atributos e serviços corresponde a um todo intrínseco. O analista deve enfocar em conjunto o estado (dados) e o comportamento (processos) de um objeto. Processos e dados devem ser considerados em conjunto.

A estratégia desta metodologia para encontrar as classes e objetos consistem em:

- Como encontrar classes e objetos;
- Conservar classes corretas e
- Refinar classes utilizando a herança.

4.2.1 COMO ENCONTRAR CLASSES E OBJETOS

Esta metodologia estabelece um conjunto de itens para determinar se é necessário ou não incluir uma classe objeto em um modelo:

- **O domínio** - estudar o domínio do problema. Entrevistar o usuário para entender sua atividade.

- **Outros sistemas** - verificar resultados de AOO anteriores - domínios iguais ou semelhantes podem ajudar através da reutilização de classes e objetos (herança).

- **Documentos** - ler o documento de solicitação, identificar o propósito (missão) e fatores críticos de sucesso para o sistema em consideração. Pedir ao usuário um resumo conciso sobre o problema. Separar o problema em seções que esclareçam particularmente um assunto, dedicar a estas seções um estudo especial. Consultar livros, enciclopédias e manuais sobre o assunto pode ajudar a entender melhor o problema.

- **Protótipo** - ele é essencial para a análise efetiva. O protótipo mostra como será o sistema e o usuário pode indicar se ele está sendo construído conforme a sua concepção.

- **Estruturas** - para localizar classes e objetos procurar no domínio do problema estruturas de Generalização-Especialização e Agregação que serão apresentadas adiante, são mais produtivas e permitem que o trabalho tenha um alto nível intelectual. A figura 4.2 mostra um exemplo de estrutura, o "G" indica que suas subclasses (veículo passageiros, veículo caminhão, etc) são Generalizações da superclasse veículo. Motor, câmbio e freio são partes do veículo, constituindo uma Agregação , indicada pela letra "A".

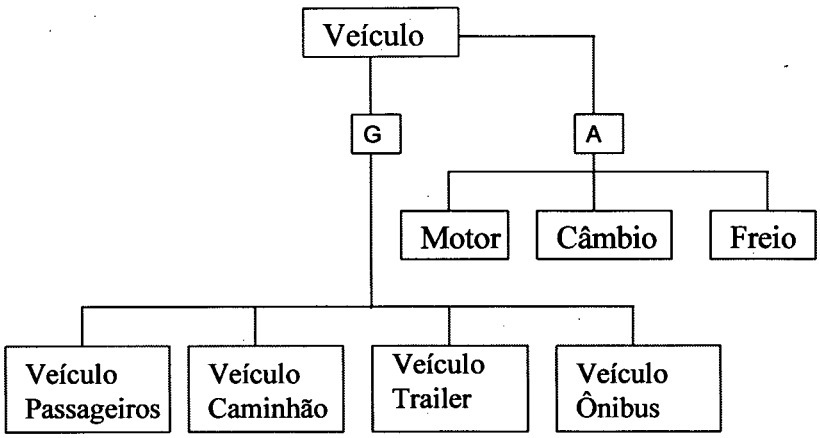


Figura 4.2 Estruturas

- **Lembranças necessárias** - o sistema precisa lembrar-se de alguma informação histórica ou algum evento legal sobre os objetos na classe? Analisar o domínio do problema e depois as responsabilidades do sistema.

- **Processamento necessário** - verificar se um objeto precisa ter algum tipo de comportamento (processamento)? À medida que o comportamento é necessário, então os serviços serão necessários. Para estes serviços, verificar que atributos são necessários e qual é o seu estado.

- **Atributos múltiplos** - este critério ajuda a filtrar classes e objetos potenciais quando um analista chega a um nível muito baixo em sua análise. Se um objeto (por exemplo: Local-Instalado) tem apenas um atributo (por exemplo: Endereço), existem indícios de erros, é provável que seja melhor incluir endereço como um atributo, que pode aparecer em outras classes e objetos, em vez de um objeto individual.

- **Atributos sempre aplicáveis** - identificar um conjunto de atributos que se aplica a cada objeto na classe. Sempre que um sistema souber que existe um objeto, deve haver um valor para cada atributo. Por exemplo, uma classe e objeto Imóveis em um sistema real pode incluir a posição, o preço, a metragem e etc. Quanto ao número de

banheiros poderiam ser aplicados apenas a certos tipos de imóveis. Se este for o caso, uma estrutura do tipo Generalização-Especialização pode ser desenvolvida.

- **Serviços sempre aplicáveis** - identificar serviços sempre aplicáveis - isto é, comportamento (processamento) que se aplica a cada objeto em uma classe. Os serviços podem ser simples como (criar, conectar, acessar e liberar) ou complexos como (calcular, iniciar, monitorizar, terminar). Se os serviços forem os mesmos para cada objeto na classe, não há problemas; entretanto, se os serviços variarem para diferentes objetos, esta variação indica que uma estrutura Generalização-Especialização pode ser adicionada.

- **Papéis executados** - que papel ou papéis as pessoas executam. Existem usuários que interagem direto com o sistema e outros que não interagem, mas sobre os quais o sistema mantém informações. Por exemplo, que papéis um supervisor e um escriturário executam no sistema. A investigação subseqüente de atributos e serviços irá capturar a distinção entre eles em detalhes. A figura 4.3 mostra como Pessoa e Empresa participam de associação trabalha-para. Uma pessoa assume o papel de empregado e a empresa o papel de empregador.

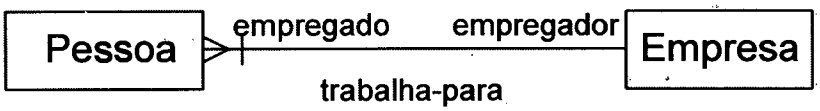


Figura 4.3 Papéis executados

- **Interação** - a interação humana acontece praticamente em todo o modelo AOO. Pode-se interagir com o modelo, chamando serviços que farão parte do problema - por exemplo: criar, conectar, acessar e liberar. A interação humana com o problema ajuda a encontrar as classes e objetos, pois a interação acontece a nível de classe e objeto, a nível de estruturas, de atributos e de serviços. Um ícone pode ser usado para mostrar a interação de pessoas com o sistema.

- **Agrupar as classes específicas do domínio do problema.** Freqüentemente, uma classe é introduzida para manter juntas classes do domínio do problema. Por

exemplo, uma classe "raiz" pode ser introduzida para manter juntas classes de uma biblioteca de classes. A figura 4.4 mostra a inclusão da classe "AOORaiz".

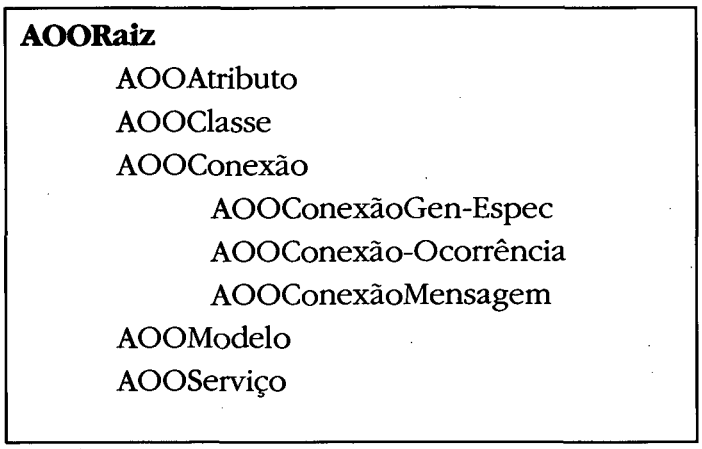


Figura 4.4 Agrupar classes específicas

- **Mais de um objeto em uma classe** - verificar classes com apenas um objeto elas podem estar mal definidas. Entretanto, se uma classe com apenas um objeto refletir a realidade, ela deve ser mantida. Pode haver uma classe objeto "Supervisor-Chefe" na raiz de uma estrutura de agregação, dependendo do contexto.

4.2.2 CONSERVAR CLASSES CORRETAS

A metodologia faz uso de vários critérios para descartar as classes desnecessárias e incorretas, conforme segue:

- **Classes redundantes** - se duas classes expressarem a mesma informação, apenas o nome mais descritivo deve ser conservado.

- **Classes irrelevantes** - se uma classe tiver pouco ou nada a ver com o domínio do problema ela deve ser eliminada. A classe pode ser importante em outro contexto.

- **Classes vagas** - algumas classes podem ter limites mal definidos ou abrangência demasiadamente ampla, estando fora do domínio do problema.

- **Atributos** - nomes que descrevem principalmente objetos isolados devem ser considerados como atributos.

- **Operações** - se um nome descreve uma operação que é aplicada a objetos e não é manipulada em si mesma, ela não deve ser uma classe.

- **Papéis** - o nome de uma classe deve refletir sua natureza intrínseca e não o papel que ela desempenha em uma associação. Uma entidade física corresponde a diversas classes. Por exemplo, Pessoa e Empregado podem ser classes distintas em algumas circunstâncias e redundantes em outras.

- **Construções de implementações** - as construções inadequadas ao mundo real devem ser eliminadas do modelo de análise. Elas podem ser necessárias mais tarde durante o projeto, mas não agora. Estruturas de dados, como listas encadeadas, árvores, vetores e tabelas são quase sempre construções de implementação e devem ser tratadas no projeto que corresponde ao capítulo 5 deste trabalho.

4.2.3 REFINAR CLASSES UTILIZANDO A HERANÇA

Esta etapa organiza as classes com a utilização de herança para compartilhar a estrutura comum. A herança pode ser acrescentada em duas direções: generalizando-se os aspectos comuns das classes existentes em uma superclasse (bottom-up) ou refinando-se as classes existentes em subclasses especializadas (top-down).

Pode-se descobrir herança partindo de baixo para cima, buscando-se classes com atributos, associações ou operações similares. Para cada generalização, define-se uma superclasse para compartilhar características comuns.

A herança múltipla pode ser utilizada para aumentar o compartilhamento, mas usá-la somente se for necessário, ela aumenta a complexidade conceitual e de implementação.

Quando o nome da mesma associação aparecer mais de uma vez com o mesmo sentido, tentar generalizar as classes associadas. A modelagem de objetos descreve a herança em detalhes na etapa 4.5 deste capítulo.

4.2.4 REGISTRAR AS CLASSES

Todas as classes devem ser registradas em um dicionário de dados. Deve-se registrar o nome e a definição da classe, como segue:

CLASSE: Contrato

DEFINIÇÃO: qualquer acordo entre a empresa e o credor-devedor que, formalmente ou não, transferem entre si algum direito ou sujeitam-se a alguma obrigação, de caráter financeiro. Por extensão, uma norma legal será tratada entre a empresa e um credor-devedor.

4.3 IDENTIFICAR ATRIBUTOS

Um atributo é uma propriedade, qualidade ou característica que pode ser atribuída a uma objeto. O atributo é um dado (informação de estado) para o qual cada objeto em uma classe tem seu próprio valor. Na AOO [COD 92] o termo atributo é definido de forma a refletir o domínio do problema e as responsabilidades do sistema.

Os atributos tornam mais claro o significado de classes e estruturas ao adicionar mais detalhes sobre o que está sendo modelado. Os atributos devem ser manipulados exclusivamente por serviços ou operações associados ao objeto a que pertencem ou a classe a ela relacionada. Existem dois tipos de atributos em um sistema OO (atributos de objetos e de classe):

1) ATRIBUTOS DE OBJETOS - Trata-se dos atributos que descrevem valores (estados) mantidos em um objeto. Diferentes objetos de uma mesma classe não compartilham os atributos de objetos, ou seja, cada um possui sua própria cópia do atributo. Um exemplo pode ser a classe empregado, com atributos de objetos como: nome, endereço, telefone, CPF, data-nascimento e valor-extra. Para o exemplo, cada funcionário deve ter seus próprios dados para cada um destes atributos. Apesar de todos estarem modelados sob uma mesma classe, os atributos são individualizados. Mesmo que os valores sejam iguais, pertencerão a objetos diferentes, identificando funcionários distintos.

2) ATRIBUTOS DE CLASSE - Algumas classes podem ter atributos compartilhados por todos os seus objetos. Eles constituem os atributos de classe. Por exemplo, um contador de objetos de uma classe. Sempre que um novo objeto é criado ou apagado, há necessidade de incrementar ou decrementar este atributo. Outro exemplo, na classe empregado pode ter um atributo de classe que é o valor da hora trabalhada, desde que todos os empregados recebam a mesma quantia.

Para definir os atributos conforme a metodologia proposta efetuar os itens:

- Como definir os atributos;
- Conservar os atributos corretos e
- Normalizar.

4.3.1 COMO DEFINIR OS ATRIBUTOS

Os mecanismos reais de identificação, por exemplo: chaves, tabelas de correlação e apontadores, são considerações de projeto e não devem ser levadas em conta agora. Os atributos de identificação oferecem um meio conveniente de referenciar um objeto e suas conexões a outros objetos. Todo objeto precisa ter identificadores (id) e identificadores de conexão (cid). Eles são utilizados para facilitar a especificação de serviços.

Fazer com que cada atributo capture um conceito atômico, ou seja, um valor único (como por exemplo: Carteira-Identidade) ou um agrupamento de valores fortemente relacionados (como por exemplo: Endereço). O conceito atômico é usado para produzir um modelo mais simples, com menos nomes de atributos e agrupamentos naturais.

Verificar resultados de AOOs anteriores em domínios iguais ou semelhantes para reutilizar atributos.

Para definir os atributos na metodologia proposta executar os seguintes itens:

- **Fazer as seguintes perguntas:**

  - O objeto em uma classe é responsável por quais informações?
  - Como o objeto é descrito no domínio do problema, que responsabilidade ele possui?
  - O que é preciso saber sobre uma classe ou um objeto?
  - Quais são suas informações e seus estados?

- **Posicionar os atributos** - colocar cada atributo na classe objeto mais adequada, verificando o domínio do problema. Para cada classe em uma estrutura Gen-Esp, colocar um atributo no ponto mais superior da estrutura, onde permaneça aplicável a cada uma de suas especializações. Se um atributo for aplicável a um nível inteiro de especializações, então ele deve ser movido para a generalização correspondente.

- **Casos especiais com atributos** - verificar para cada atributo, se existe algum valor não-aplicável, ou um valor mais significativo. Por exemplo, o atributo combustível pode ter os valores gasolina, diesel, álcool ou elétrico. Entretanto, se o objeto em uma classe for trailer, ele não se aplica. Neste caso pode-se utilizar a estrutura Gen-Esp.

- **Atributo único** - Verificar cada classe objeto com apenas um atributo. Ela pode ser uma abstração de alguma coisa na qual as responsabilidades do sistema incluem apenas um atributo ou um atributo de uma outra classe objeto, que está fora de lugar no modelo. Por exemplo a classe objeto Escriturário com apenas um atributo Nome-Legal.

- **Valores de repetição** - Verificar cada conexão de ocorrência muitos-para-muitos. Por exemplo, considerar a conexão de ocorrência entre proprietário e veículo. Os atributos Data-Hora-compra e Quantidade descrevem a interação em algum ponto no tempo entre o proprietário e um veículo. Quando este for o caso, adicionar ao modelo uma classe objeto Compra.

- **Requisitos baseados em domínio** - são aqueles que o sistema tem que ter, independentemente da tecnologia usada para elaborar o sistema.

- **Resultados derivados** - evitar resultados derivados como por exemplo: a "idade do cliente" em um sistema que contém a data de nascimento do cliente. Evitar o uso de atributos com conteúdos de cálculos, deve-se fazer uma comparação entre o tempo de CPU versus memória para armazenar o atributo para verificar a melhor opção.

- **Procedimentos operacionais** - que procedimentos o sistema deve manter com o tempo. As classes podem ter atributos como o nome do procedimento operacional, nível de autorização de acesso requerida e passos sobre o procedimento.

- **Locais** - qual a posição física, repartição ou local que o sistema precisa conhecer. Por exemplo um almoxarifado poderia ter um nome e atributos de localização.

- **Unidades organizacionais** - a que unidades organizacionais pertencem as pessoas. O sistema precisa manter registros sobre o local da repartição, o nome do responsável e o endereço.

4.3.2 CONSERVAR OS ATRIBUTOS CORRETOS

Esta metodologia propõe uma revisão, eliminando os atributos incorretos conforme segue:

- **Objetos** - se a existência de uma entidade independente for importante, e não apenas seu valor, então ela é um objeto. Por exemplo, Cidade para mala postal pode ser um atributo enquanto que Cidade para um censo poderia ser um objeto.

- **Qualificadores** - se o valor de um atributo depender de um determinado contexto, considerar a redefinição do atributo como um qualificador.

- **Identificadores** - as linguagens de programação baseadas em objetos incorporam identificadores de objetos para fazer referência a um objeto sem ambigüidade. Não deve-se indicar esses objetos identificadores nos modelos de objetos, uma vez que eles já estão implícitos naqueles objetos.

- **Atributos de ligação** - se uma propriedade depende da presença de uma ligação, então a propriedade é um atributo da ligação e não um objeto relacionado (veja maiores detalhes em "associações", adiante).

- **Valores internos** - se um atributo descreve o estado interno de um objeto que é invisível fora do objeto, então acrescentá-lo apenas durante o projeto.

- **Refinar os detalhes** - omitir os atributos menores que não vão afetar a maioria das operações.

- **Atributos discordantes** - um atributo que fica completamente diferente e não-relacionado a todos os outros atributos pode indicar uma classe que deve ser dividida em duas classes distintas.

- **Especificar os atributos** - usar nomes padronizados e legíveis (evitar prefixos, sufixos e palavras maiúsculas. Procurar ser abrangente, escolher nomes mais específicos como por exemplo, combustível ao invés de álcool (os nomes mais específicos podem ser alterados com mais facilidade). A escolha do nome de um atributo, pode ser caracterizado pelo seu primeiro identificador mantendo um padrão de nomenclatura. As palavras descritas na tabela 4.1 podem ser utilizadas como um exemplo de padrão.

Tabela 4.1 Palavras chaves para iniciar o nome de um atributo

Palavra	Abreviatura	Palavra	Abreviatura
código	cod	matrícula	mat
valor	vlr	sigla	
número	num	data	
quantidade	qtde	dia	
texto	txt	mês	
percentual	perc	ano	
descrição	descr	hora	
indicador	indic	nome	
índice	ind	taxa	

Exemplo de identificação de atributos: nome-pessoa-empregado, cod-fornecedor, num-matricula-empregado, indic-validade-código, descr-mercadoria.

4.3.3 NORMALIZAR

As tabelas e colunas em um banco de dados relacional podem ser reorganizadas para reduzir a redundância de dados e conseqüentemente reduzir o número de etapas necessárias para modificar os dados. A disciplina para fazer isso é chamada de normalização. O grau de eliminação da redundância de dados é definido pelo termo "formas normais". A normalização deve ser aplicada se a implementação dos dados ocorrer em um banco de dados relacional.

Para normalizar uma classe objeto são aplicadas três etapas de normalização [COD 92]:

Primeira Forma Normal (1FN) - Na primeira forma normal o valor do atributo deve ser atômico. Isso quer dizer que para cada atributo não há repetição de valores.

Segunda Forma Normal (2FN) - Na segunda forma normal os atributos não chave descrevem algo identificável somente pela chave toda.

A 2FN e 3FN tratam do relacionamento entre campos Chave e Não Chave. Elas são definidas em termos de dependências funcionais.

A 2FN é violada quando um item não chave é um fato sobre uma parte da chave. Pressupõe que a chave primária seja composta por mais de um atributo.

Terceira Forma Normal (3FN) - Na terceira forma normal cada atributo não chave depende somente da chave. É violada quando um item não chave é um fato sobre outro item não chave.

4.3.4 REGISTRAR OS ATRIBUTOS

Os atributos devem ser registrados em um dicionário de dados, indicando a definição, tamanho e formato de cada um.

4.4 PROJETAR O MODELO DE OBJETOS

A modelagem de objetos é uma das etapas mais importantes da análise pois ela está mais próxima ao mundo real. A modelagem de objetos estabelece uma linguagem gráfica que facilita a comunicação com o usuário. Esta metodologia divide o modelo de objetos em duas partes e é baseado nas especificações de Rumbaugh [RUN 94]:

- Ligações e associações;
- Conservar as associações corretas.

4.4.1 LIGAÇÕES E ASSOCIAÇÕES

São formas para estabelecer um relacionamento entre objetos e classes. Uma ligação é uma conexão física ou conceitual entre instâncias de objetos que estabelecem um relacionamento entre objetos e classes. Uma associação descreve um grupo de ligações com estrutura e semânticas comuns.

LIGAÇÕES - mostram o relacionamento entre as classes. A figura 4.5 mostra uma ligação um-para-muitos (obrigatória) entre a classe Fornecedor e Produto, uma ligação um-

para-muitos (opcional) entre Almoxarifado e Produto e uma ligação um-para-um entre Empresa e Almoxarifado (sendo Empresa obrigatório e Almoxarifado opcional).

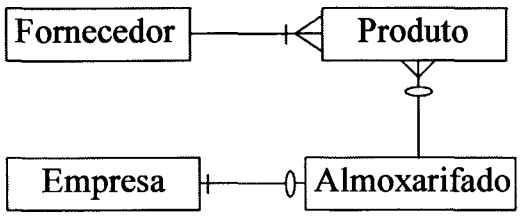


Figura 4.5 Exemplo de associações

ASSOCIAÇÕES - uma associação mostra os objetos e como eles se relacionam. A figura 4.6 descreve atributos de ligação para uma associação entre três classes. Um fornecedor fornece muitos materiais. Cada projeto pode utilizar vários materiais e de vários fornecedores. O material usado em cada projeto e com a indicação de quem foi o fornecedor é descrito em uma associação (uma nova classe) chamada material-consumido.

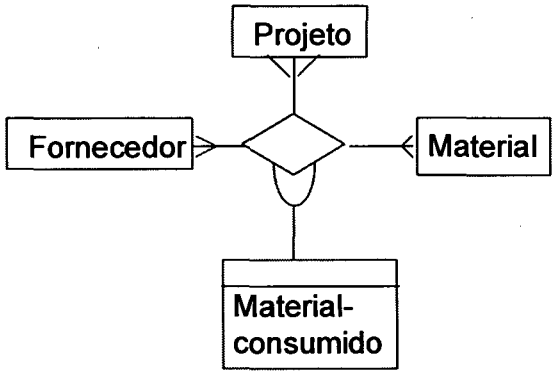


Figura 4.6 Associação ternária

A figura 4.7 mostra que é possível juntar atributos de ligação um-para-um e um-para-muitos na classe oposta a "um". A primeira parte da figura expressa um-para-muitos com os atributos de associação separados em uma nova classe (salário e cargo). Na segunda parte os atributos da associação foram colocados junto a classe pessoa. Os atributos de

ligação, no entanto, devem permanecer separados para não comprometer a flexibilidade dos sistemas no futuro.

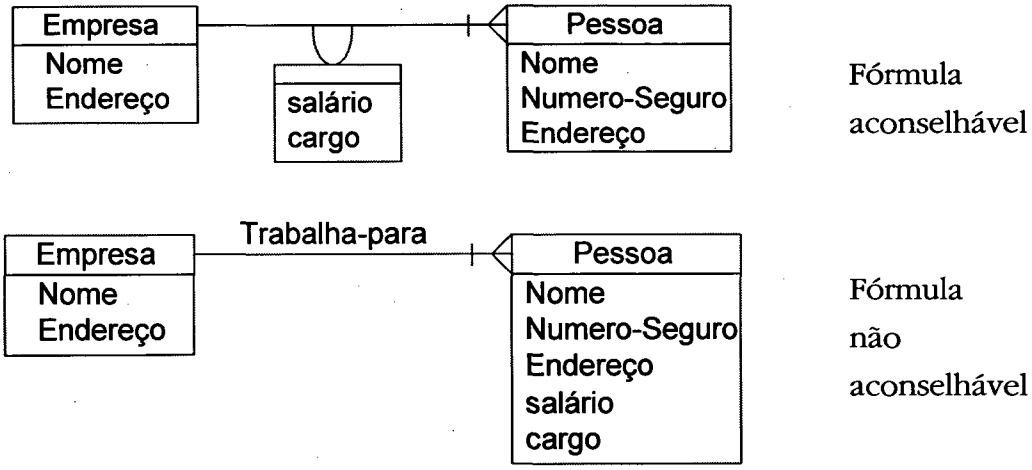


Figura 4.7 Atributos de ligação versus atributos de objeto

É importante, por vezes, modelar uma associação como uma classe. Cada ligação torna-se uma instância de classe. A figura 4.8 mostra a informação de autorização para usuários em estações de trabalho. Os usuários podem ser autorizados a utilizar muitas estações de trabalho. Cada usuário tem um diretório para cada estação de trabalho autorizada, este diretório é compartilhado por diversas estações ou diversos usuários.

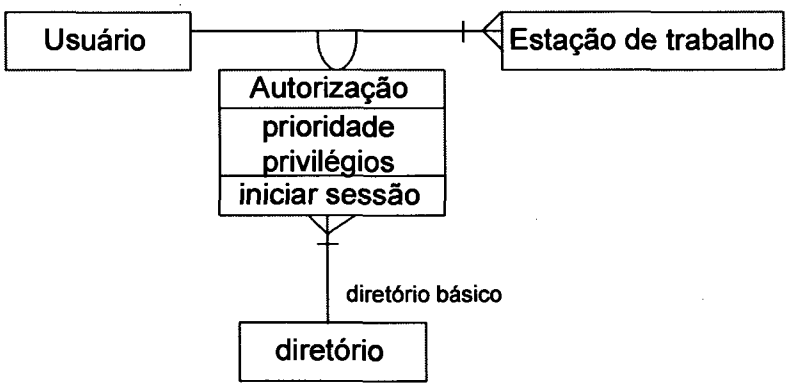


Figura 4.8 Modelagem de uma associação de classe

As associações possuem várias propriedades que tornam o modelo mais claro, tais como:

PAPÉIS - Um papel é uma extremidade de uma associação. Uma associação binária tem dois papéis, um em cada extremidade. Esta utilização permite percorrer associações a

partir de um objeto em uma extremidade, sem mencionar a associação. A figura 4.9 mostra como Pessoa e Empresa participam da associação trabalha-para. A Pessoa assume o papel de empregado e a Empresa o papel de empregador.

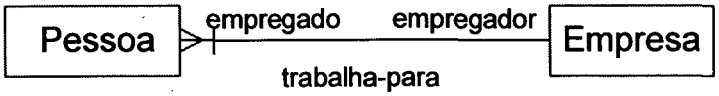


Figura 4.9 Nomes de papéis para uma associação

QUALIFICAÇÃO - Uma associação qualificada inter-relaciona duas classes de objetos e um qualificador com objetivo de reduzir a multiplicidade. É aplicado a um-para-muitos e muitos-para-muitos. A qualificação melhora a exatidão semântica e aumenta a visibilidade dos caminhos de navegação. A figura 4.10 mostra um diretório com muitos arquivos e estes pertencem a um só diretório. "Nome do arquivo" é o qualificador entre os objetos diretório e arquivo.

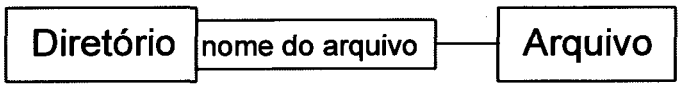


Figura 4.10 Uma associação qualificada

CONTROLE - É um tipo de relacionamento onde a classe de controle deve ser excluída se a classe a que ela se conecta for excluída. O exemplo da figura 4.11 mostra um produto que é fabricado por um único fabricante. Se o produto deixar de existir, isto deve acontecer também com o fabricante.

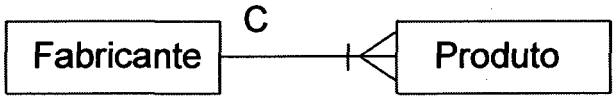


Figura 4.11 Relacionamento de controle

DEPENDENTE - Quando uma classe é dependente de outra. No exemplo da figura 4.12 os parágrafos são dependentes do texto. Se o texto for eliminado seus parágrafos também o serão.

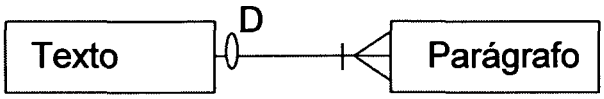


Figura 4.12 Relacionamento dependente

4.4.2 CONSERVAR AS ASSOCIAÇÕES CORRETAS

Nesta etapa a metodologia estabelece formas para eliminar as associações desnecessárias ou incorretas, usando os seguintes critérios:

- **Associação entre classes eliminada** - se uma das classes de uma associação tiver sido eliminada, a associação também deve ser eliminada ou restabelecida em relação a outras classes.

- **Associações irrelevantes ou de implementação** - eliminar qualquer associação que esteja fora do domínio do problema ou que lide com construções de implementação.

- **Ações** - uma associação deve descrever uma propriedade estrutural do domínio da aplicação e não um evento transiente.

- **Associações ternárias** - as associações entre três ou mais classes podem, em sua maioria, serem decompostas em associações binárias.

- **Associações derivadas** - pode-se omitir as associações que podem ser definidas em termos de outras associações porque são redundantes. Tanto quanto possível, classes, atributos e associações do modelo de objetos devem representar informações independentes. Caminhos múltiplos entre classes muitas vezes indicam associações derivadas que são composições de associações primitivas. Nem todas as associações que formam caminhos múltiplos entre classes indicam redundância. A figura 4.13 mostra que uma empresa emprega muitas pessoas e possui muitos computadores. A cada Pessoa estão alocados zero ou mais computadores para uso pessoal daquele empregado; alguns computadores são para uso público e não estão alocados a ninguém. A

multiplicidade da associação Alocado não pode ser deduzida das associações Emprega e Possui.

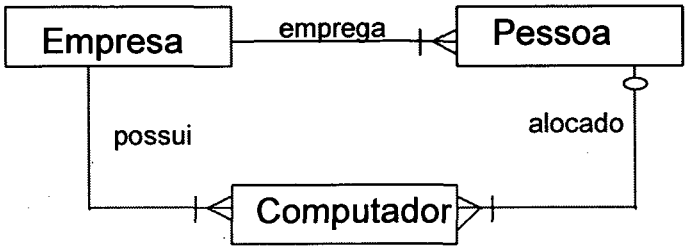


Figura 4.13 Associações não-redundantes

- **Associações com nomes inadequados** - não é necessário dizer como nem por que uma situação ocorreu, dizer apenas o que ela faz. Por exemplo, *Computador do banco mantém contas* pode ser refinado em *banco mantém conta.*

- **Nome de papéis** - colocar o nome de papéis onde forem adequados. O nome descreve o papel que uma classe da associação desempenha do ponto de vista de outra classe. Por exemplo, nas associações Trabalha-para, a Empresa tem o papel de empregador e Pessoa tem o papel de empregado.

- **Associações qualificadas** - geralmente um nome identifica um objeto dentro de algum contexto, o nome deve identificar inequivocamente o objeto. Um qualificador descreve objetos do lado "muitos" de uma associação.

- **Multiplicidade** - especificar a multiplicidade mas não de forma exagerada, visto que ela se modifica durante a análise.

- **Associações ausentes** - acrescentar qualquer associação ausente que seja descoberta.

4.4.3 REGISTRAR O MODELO

O modelo de objetos deve incorporar a documentação da análise.

4.5 IDENTIFICAR ESTRUTURAS

As estruturas facilitam a visualização existente entre as classes e mostram uma hierarquia de funções. "Uma estrutura consiste em uma forma de ligação entre duas classes, onde as classes estão relacionadas" [COA 92] . Estrutura é um termo global, podendo ser usado para a estrutura Generalização-Especialização (Gen-Esp) e estrutura de Agregação (também chamada de estrutura Todo-Parte). Este estudo sobre estruturas está baseado nas especificações de Rumbaugh [RUM 94].

A estrutura Gen-Esp [RUM 94] pode ser usada para "distinção entre as classes" como um método básico de organização dos seres humanos. Um exemplo é a generalização da classe Microcomputador mostrado na figura 4.14, e a especialização PCs. Uma estrutura Gen-Esp pode ser pensada como uma estrutura "é um" ou "é um tipo de". Por exemplo, Unix é um (é um tipo de) Microcomputador. Pode-se aplicar a herança nas estruturas Gen-Esp, com uma representação explícita de atributos e serviços mais gerais seguidos por especializações pertinentes.

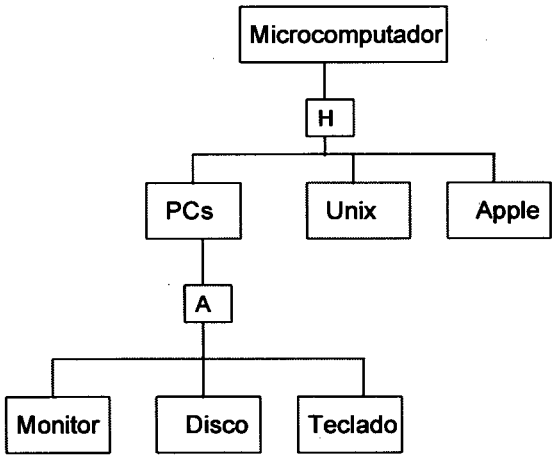


Figura 4.14 Exemplo de Estruturas

A estrutura de Agregação é outro método básico de organização dos seres humanos. Um exemplo é a agregação de PCs, e os elementos Monitor, Disco e Teclado, exibidos na figura 4.14. A Agregação pode ser pensada como uma estrutura "tem um", por exemplo, PCs tem um Monitor. As estruturas Gen-Esp e Agregação concentram a atenção do analista na complexidade de um grande número de classes para descobrir classes adicionais. A herança pode ser aplicada à estrutura Gen-Esp para que os atributos e serviços generalizados sejam identificados apenas uma vez, e depois especializados. Os próximos itens desta metodologia descrevem as estruturas de generalização-especialização e de agregação.

Para identificar as estruturas de acordo com a metodologia proposta efetuar:

- Generalização-Especialização;
- Como aplicar a generalização
- Agregação;
- Generalização versus herança;
- Agregação versus associação;
- Agregação versus generalização e
- Generalização como extensão e restrição.

4.5.1 GENERALIZAÇÃO-ESPECIALIZAÇÃO

A estrutura Gen-Esp é mostrada [COA 92] com uma classe de generalização no topo e classes de especialização abaixo. A Gen-Esp surge quando uma classe genérica divide-se em subclasses mais especializadas. À medida que se sobe na árvore hierárquica, passa-se por níveis progressivos de generalização e quando se desce, está-se especializando, conforme demonstra a figura 4.15.

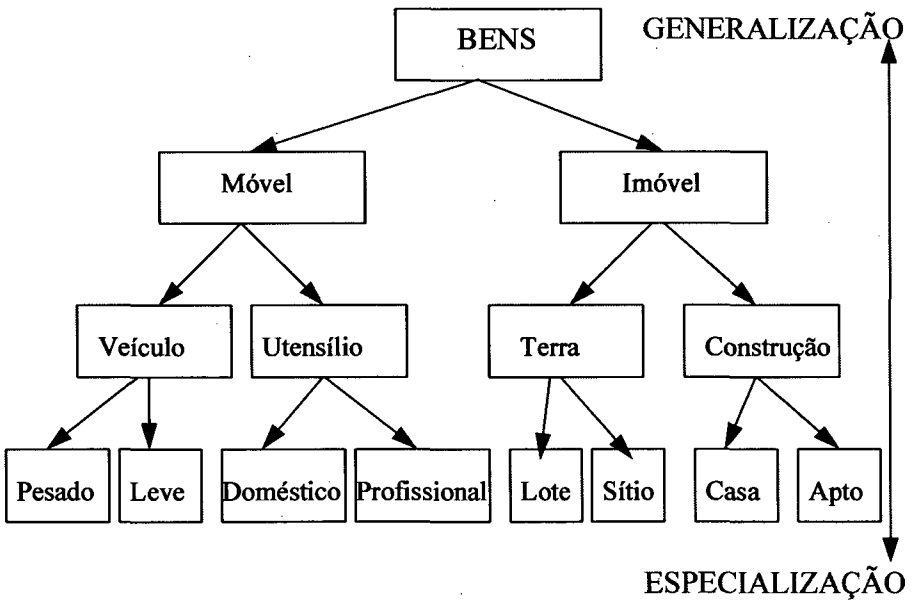


Figura 4.15 Hierarquia com Generalização/Especialização

A principal vantagem ao se definir a estrutura Gen-Esp está na otimização da definição das classes. Serviços e atributos definidos acima de uma classe, sempre que possível, devem ser compartilhados por classes derivadas, evitando a duplicação de código.

Pode-se denominar as classes de especialização atribuindo nomes semelhantes à classe de generalização, de modo a identificar tanto sua classe-pai (superclasse) como sua diferença em relação a ela. Exemplo, supondo a classe de generalização Telefone e uma especialização que consiste de "telefone móvel" e "telefone fixo", conforme mostra a figura 4.16.

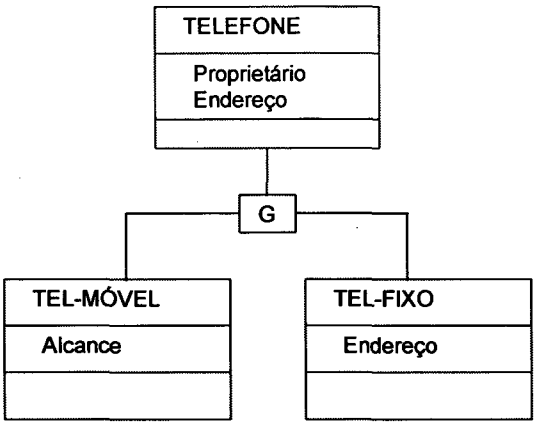


Figura 4.16 Estrutura Gráfica Generalização-Especialização

4.5.2 COMO APLICAR A GENERALIZAÇÃO

Para saber se uma estrutura pode ser generalizada aplicar os seguintes itens:

Perguntas - Considerar cada classe como uma generalização. Para suas especializações potenciais, perguntar: A classe está no domínio do problema? A classe faz parte das responsabilidades do sistema? Pode-se implementar a herança?

Estruturas - para cada Classe Objeto, examinar a classe com Gen-Esp e para cada Objeto com Agregação.

Reuso - de uma forma similar, considerar cada classe como uma especialização. Verificar os resultados de AOOs anteriores. Algumas estruturas Gen-Esp podem ser reutilizadas.

Especialização - se muitas especializações forem possíveis, é útil considerar primeiro a especialização mais simples e a mais elaborada, e depois prosseguir com as intermediárias, considerando-se os seguintes itens:

- **Sentido** - as especializações "fazem sentido" no domínio do problema, se não estiverem no domínio, então as especializações não são necessárias.

- **O que conhecer** - o sistema precisa reconhecer a diferença ente uma e outra especialização no sistema que está sendo analisado? Se não existir diferença, as especializações não são necessárias.

- **Haverá herança** - isto é, alguns atributos e serviços aplicam-se a todos as especializações.

- **Critérios** - verificar se as classes atendem aos seguintes critérios?:

  1. Se a única distinção entre as especializações for o tipo da especialização, então usar um atributo. A estrutura Gen-Esp não é necessária neste caso.

  2. Se a única distinção entre as especializações for a responsabilidade do sistema, então usar um atributo. Não há necessidade de estrutura Gen-Esp.

  3. Se uma classe de especialização não adicionar nenhum atributo ou serviço, ela não é necessária.

Hierarquia e entrelaçamento - cada estrutura Gen-Esp podem formar uma hierarquia ou um entrelaçamento [COA 92]. Certas redundâncias aparecem nas especializações. No exemplo abaixo a figura 4.17 mostra um indivíduo que é um proprietário, um escriturário, ou um escriturário e proprietário.

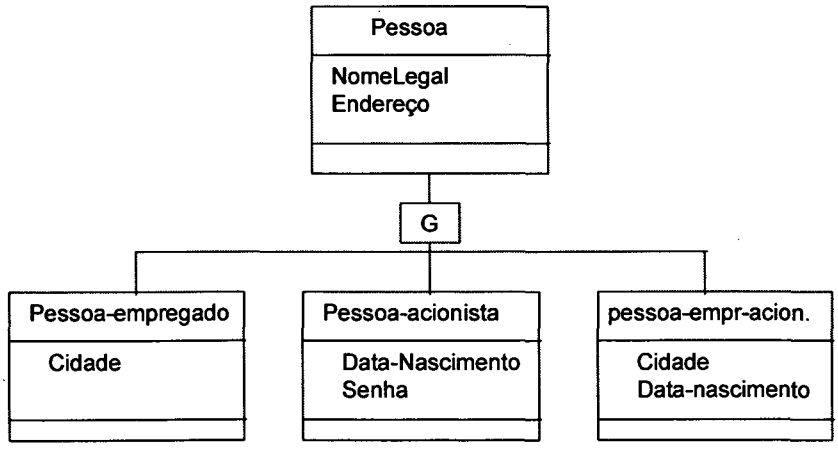


Figura 4.17 Estrutura Gen-Esp de Indivíduos, como hierarquia

Com um entrelaçamento Gen-Esp, algumas especializações adicionais do domínio do problema podem ser determinadas; além disso, outros aspectos comuns de atributo e serviço podem ser representados. A figura 4.18 é um refinamento da figura anterior (4.17), ela mostra o domínio dos indivíduos, indicando que cada indivíduo determinado no domínio é um proprietário, um escriturário, ou ambos.

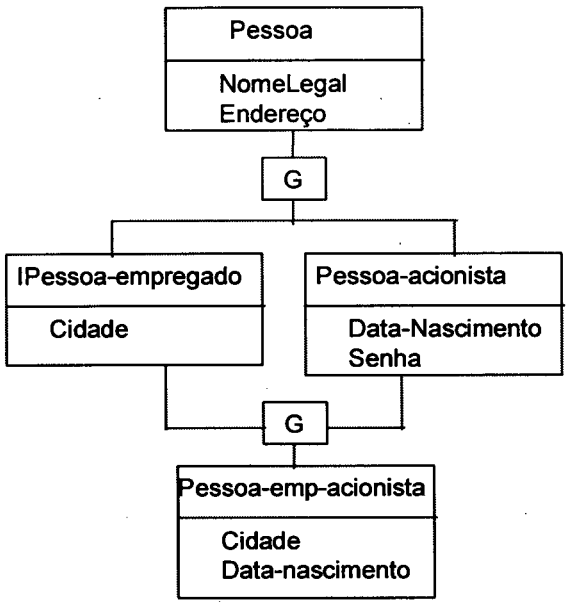


Figura 4.18 Gen-Esp de Indivíduos, na forma de um entrelaçamento

O entrelaçamento:

- Realça especializações adicionais;
- Captura explicitamente elementos comuns;
- Aumenta a complexidade do modelo.

4.5.3 AGREGAÇÃO

"É um relacionamento "todo-parte" ou "uma-parte-de" no qual os objetos que representam os componentes de alguma coisa são associados a um objeto que representa a estrutura inteira" [RUM 94]. A estrutura de agregação é mostrada como um objeto no topo ou a esquerda, e depois um ou mais objetos e abaixo ou na direita. A figura 4.19 mostra parte de um modelo de objetos para um programa de processamento de texto, mostrando a agregação.

Figura 4.19 Agregação

A figura 4.20 demonstra que a agregação pode ter um número arbitrário de níveis. A árvore de agregação é apenas uma notação simples de traçar muitas linhas interligando componentes a uma estrutura.

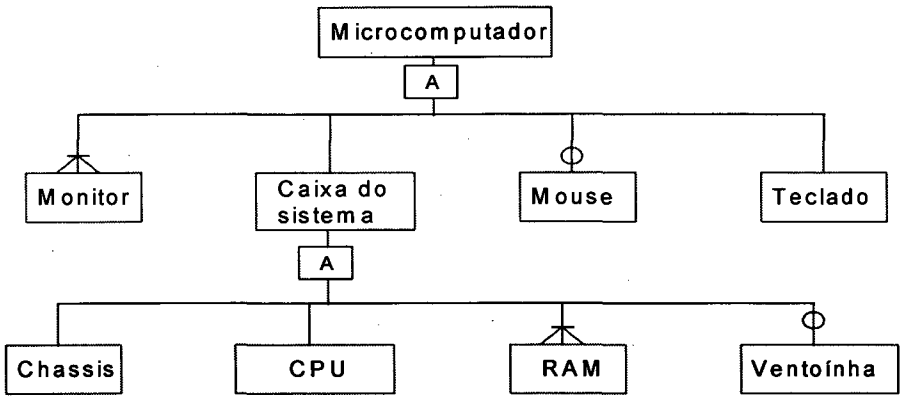


Figura 4.20 Agregação multinivelada

4.5.4 GENERALIZAÇÃO VERSUS HERANÇA

A generalização e a herança [RUM 94] são abstrações poderosas para o compartilhamento de classes similares ao mesmo tempo em que suas diferenças são preservadas. A generalização é conhecida como "é-um" cada subclasse é uma instância da superclasse. Generalização é o relacionamento entre uma classe e uma ou mais versões refinadas delas. A herança é uma implementação de uma estrutura de generalização. A classe que está sendo refinada é chamada superclasse e cada versão refinada é chamada de subclasse. A subclasse herda as características da superclasse. A figura 4.21 mostra a classe empregado com a operação "calcular pagto" que é herdada por todas as subclasses.

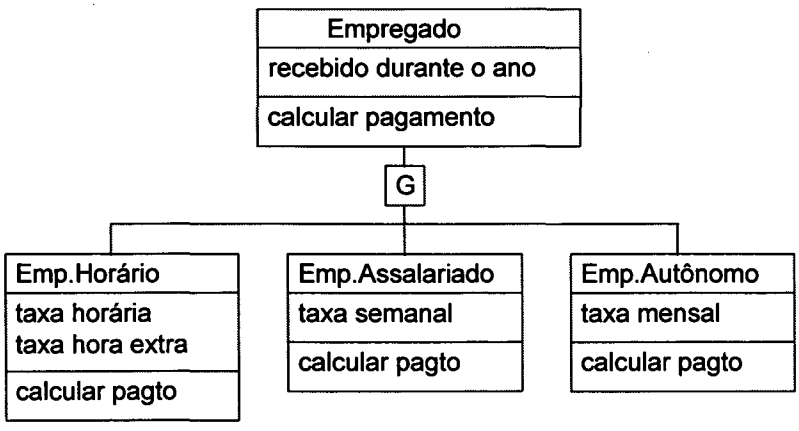


Figura 4.21 Herança a partir da generalização

Criar níveis de subclasses em excesso, pode dificultar o entendimento. A generalização é uma construção útil para a modelagem conceitual e para a implementação. Ela facilita a modelagem pela estruturação de classes e incorpora resumidamente o que é semelhante e o que é diferente em relação a elas.

Uma subclasse pode cancelar uma característica de uma superclasse pela definição de uma característica com o mesmo nome. A característica substituta refina e sobrepõe à característica a ser eliminada. Pode-se cancelar valores default, atributos e métodos de operações. Não deve-se cancelar a assinatura, ou a forma de uma característica. Um cancelamento ou substituição deve preservar o tipo, o número de atributos, o tipo de argumentos de uma operação e o tipo de retorno de uma operação. Uma característica não deve ser cancelada de modo que se torne inconsistente com a assinatura ou com a semântica da característica (herdada) original.

4.5.5 AGREGAÇÃO VERSUS ASSOCIAÇÃO

A agregação [RUM 94] é uma forma especial de associação e não um conceito independente. A agregação acrescenta conotações semânticas em certos casos. Se dois objetos forem estreitamente ligados por um relacionamento parte-todo, eles formam uma agregação. Se os dois objetos forem habitualmente considerados como independentes, embora possam muitas vezes estar ligados, eles formam uma associação. A figura 4.22 mostra a agregação de divisões e seus empregados.

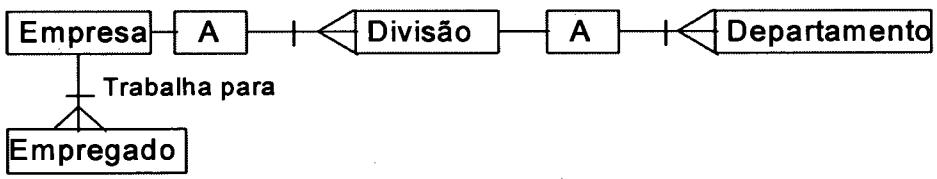


Figura 4.22 Agregação e associação

4.5.6 AGREGAÇÃO VERSUS GENERALIZAÇÃO

Agregação não é a mesma coisa que generalização [RUM 94]. Agregação relaciona-se a instâncias. A generalização é relativa a classes e é uma forma de se estruturar a descrição de um objeto isolado. Tanto a superclasse como a subclasse refere-se às propriedades de um único objeto. Uma árvore de agregação é composta por instâncias de objetos que

fazem parte de um objeto composto; uma árvore de generalização é composta por classes que descrevem um objeto. A figura 4.23 mostra a agregação e a generalização para o caso de uma lâmpada de mesa.

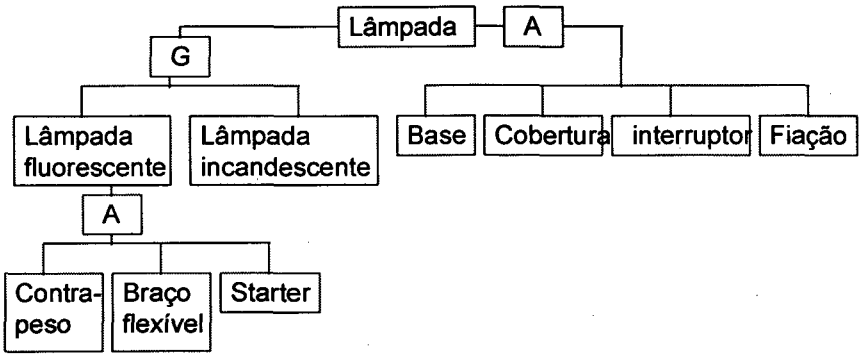


Figura 4.23 Agregação e generalização [RUM 94]

A agregação chama-se:

parte-de
relacionamento-e
todo-parte
uma-parte-de.

A generalização chama-se:

tipo-de
é-um
relacionamento-ou.

4.5.7 GENERALIZAÇÃO COMO EXTENSÃO E RESTRIÇÃO

Uma instância [RUM 94] de uma classe é uma instância de todos os ancestrais dessa classe. Uma classe descendente não pode omitir ou suprimir um atributo de um ancestral porque assim este não seria realmente uma instância ancestral. Uma subclasse pode reimplantar uma operação por questões de eficiência, mas não pode alterar o protocolo externo. Uma subclasse pode acrescentar novas características chamadas de extensão. Uma subclasse pode restringir os atributos de um ancestral chamado de restrição. As características herdadas podem receber outro nome em uma restrição.

A inclusão de uma classe pode ser feita de duas maneiras: Implicitamente, por definição ou explicitamente por enumeração. Em uma definição explícita de inclusão em uma classe, as operações que invalidariam as restrições de inclusão devem ser desativadas em termos semânticos. Uma restrição implica que uma subclasse não pode herdar todas as operações de seus ancestrais. Algumas regras para que se possa ter a herança são:

- Todas as operações de consulta e atualização são herdadas por todas as subclasses.
- As operações que podem ser refinadas pela adição de novo comportamento.
- As operações não podem ser substituídas para se fazer com que se comportem de forma diferente das operações herdadas.

EXEMPLO DE MODELO DE OBJETOS - A figura 4.24 mostra um modelo E-R para objetos de um sistema de produção onde a Empresa se relaciona com Pessoa, a qual é composta por Trabalhador ou Gerente. O Trabalhador obrigatoriamente trabalha em um Projeto e no projeto existem muitos trabalhadores. Projeto pode ter Gerente responsável. Gerente pode gerenciar um Departamento. O Departamentos é de uma Empresa e tem muitos Produtos.

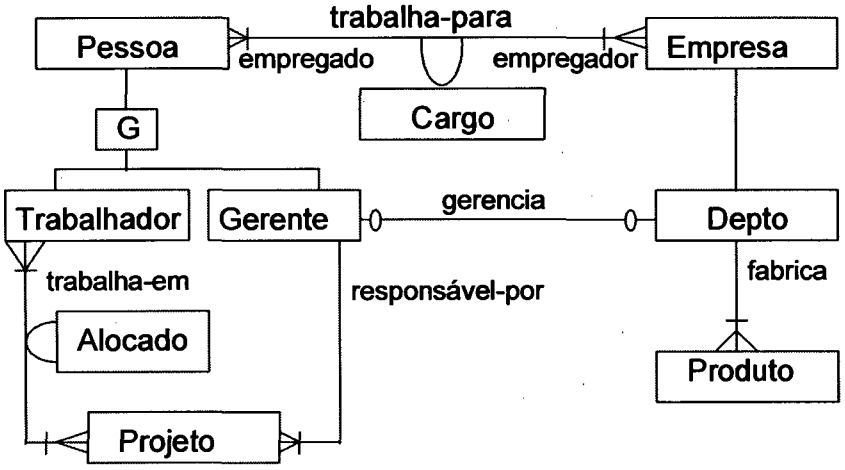


Figura 4.24 Modelo para objetos

4.5.8 REGISTRAR AS ESTRUTURAS

O modelo resultante desta etapa deve servir como parte da documentação da análise.

4.6 IDENTIFICAR SERVIÇOS

Este item descreve os serviços (também chamados de métodos). Para especificar os serviços foram escolhidas duas técnicas: A primeira, proposta por Coad e Yourdon [COA 92] "identificar serviços" e "conexões de mensagens". A segunda é "Diagrama de Fluxo de dados" proposta por Rumbaugh [RUM 94].

Serviços são as operações efetuadas pelos objetos que definem como ele deve agir quando exigido. Serviço é um comportamento específico, residente no objeto. Por exemplo contratar, despedir, pagar-dividendos são operações da classe empresa. Abrir, fechar, ocultar e reexibir são operações na classe janela.

Cada objeto passa por diferentes estados, desde a hora em que é criado até ser liberado. O estado de um objeto é representado pelos valores de seus atributos. Toda alteração nos valores de atributo reflete uma alteração de estado.

Um estado de objeto é a identificação do valor de atributo que reflete uma alteração no comportamento. Para identificar estados examinar os valores potenciais para os atributos e determinar se as responsabilidades do sistema incluem um comportamento diferente para estes valores.

Os objetos de estado apresentam os estados de um objeto com o tempo. O diagrama identifica os estados e transições de um estado para o outro. Por exemplo, no sistema Sensor (com atributos: modelo, seqüencial, intervalo, estado e valor) o valor para estado pode ser desligado, disponível e ligado. Cada estado tem seus próprios valores.

Para identificar os serviços de acordo com a metodologia proposta efetuar:

- Identificar serviços;
- Identificar conexões de mensagens;
- Diagrama de fluxo de dados (DFD);
- Como construir o DFD;

4.6.1 IDENTIFICAR SERVIÇOS

Os serviços básicos para cada objeto podem ser:

- **Criar**. Este serviço cria e inicia um novo objeto em uma classe.
- **Conectar**. Conectar e desconectar um objeto a outro.
- **Acessar**. Obter ou estabelecer os valores de atributo de um objeto.
- **Liberar**. Liberar (desconectar e eliminar) um objeto.

- **Calcular**. Calcular um resultado a partir dos valores de atributo de um objeto.

- **Monitorizar**. Monitorizar um sistema ou dispositivo externo; lida com entradas e saídas externas ao sistema, ou com controle e aquisição de dados de dispositivo.

Outros serviços específicos de cada objeto devem ser especificados.

4.6.2 IDENTIFICAR CONEXÕES DE MENSAGENS

Uma mensagem [COA 92] é qualquer comunicação feita entre duas pessoas. Para a AOO o termo define uma forma de refletir o domínio do problema e as responsabilidades do sistema. Uma conexão de mensagem modela a dependência de processamento de um objeto, indicando quais serviços ele precisa para cumprir suas responsabilidades.

Uma conexão de mensagem é um mapeamento de um objeto para um outro objeto (ou ocasionalmente para uma classe, para a criação de um novo objeto), no qual um emissor envia uma mensagem para um receptor, para a realização de algum processamento.

As conexões de mensagens existem somente em função dos serviços. Contudo, examinar as conexões de mensagens antes de especificar os serviços orienta o analista. Uma análise inicial das conexões possibilita um entendimento de muitas dependências de processamento.

Uma conexão de mensagem combinam o fluxo de dados com o evento-resposta: ou seja, cada conexão representa valores enviados no contexto de uma necessidade particular de serviço, e uma resposta recebida como resultado.

A figura 4.25 mostra um exemplo de conexão de mensagem. A seta aponta do emissor para o receptor, indicando o envio de uma mensagem. As mensagens ajudam a verificar o modelo inteiro e entendê-lo melhor onde a classe pedido-serviço envia uma mensagem para o equipamento.

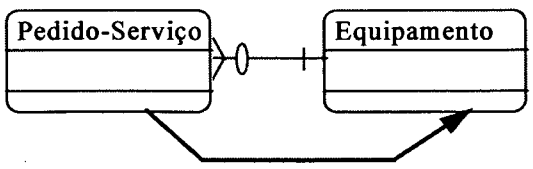


Figura 4.25 Representação de mensagem

Para identificar as conexões, perguntar para cada objeto:

- Ele precisa dos serviços de quais outros objetos?
- Quais outros objetos precisam de seus serviços?

4.6.3 DIAGRAMA DE FLUXO DE DADOS

O modelo funcional [RUM 94] consiste de múltiplos diagramas de fluxo de dados (DFD) que mostram o fluxo de valores provenientes das entradas externas, que passam pelas operações e depósitos internos de dados, a caminho das saídas externas. Os modelos funcionais especificam os resultados de um processamento sem especificar como ou quando eles são processados. A figura 4.26 mostra um DFD para a exibição de um ícone de um sistema de janelas.

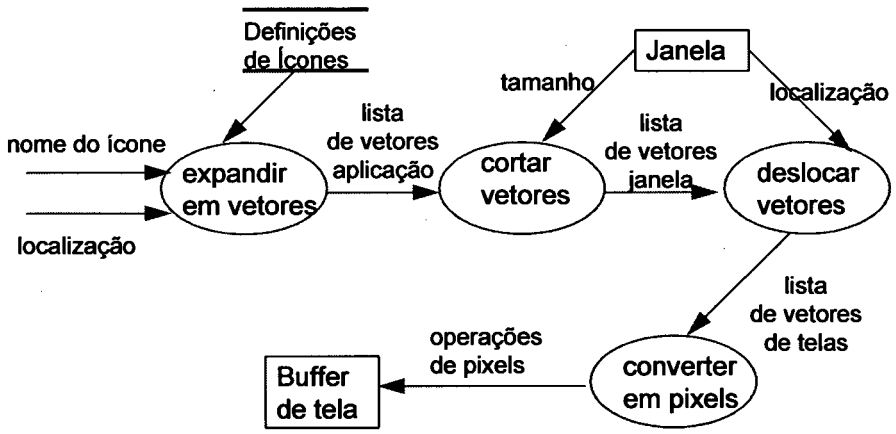


Figura 4.26 Diagrama de fluxo de dados para exibição de gráficos em janelas

O DFD é um gráfico que mostra o fluxo dos valores de dados desde suas origens nos objetos, através dos processos que os transformam, até seus destinos em outros objetos. Ele não mostra informações de controle, como o momento em que os processos são executados, ou decisões entre vias alternativas de dados.

4.6.4 COMO CONSTRUIR O DFD

Inicialmente, examinar o enunciado do problema para encontrar quaisquer valores de entrada e saída. Construir um DFD de nível mais elevado descrevendo os principais

eventos. A figura 4.27 mostra um exemplo de valores de entrada e saída de um sistema de atendimento automático (ATM).

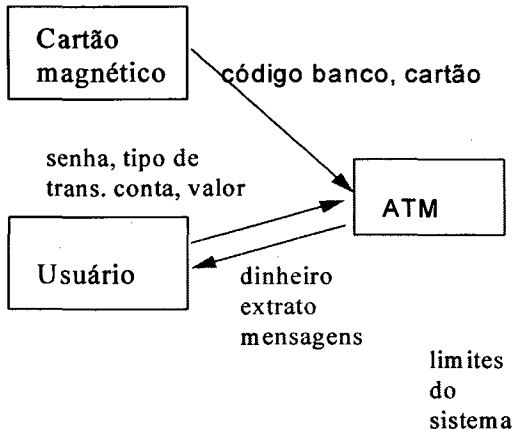


Figura 4.27 Valores de Entrada e Saída

Após, construir um diagrama de fluxos de dados mostrando como cada valor é processado a partir dos valores de entrada. Um DFD é habitualmente construído em camadas. A camada mais elevada pode ser composta por um único processo, ou talvez um processo para obter entradas, outro para computar valores e outro para gerar saídas. O DFD da figura 4.28 mostra o nível mais elevado para um sistema de caixas de atendimento automático.

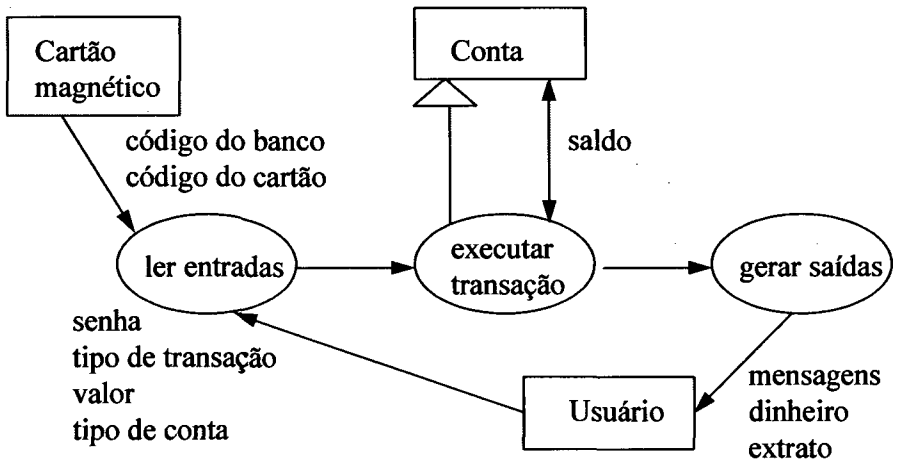


Figura 4.28 DFD de nível mais elevado para ATM

Dentro de cada camada do DFD, seguir no sentido inverso a partir de cada valor de saída para determinar a função que a processa. Se as entradas para as operações forem entradas para todo o diagrama, então o DFD terminou. Caso contrário, algumas das entradas de operações podem ser valores intermediários, que devem por sua vez serem rastreados em ordem inversa.

Desenvolver cada processo complexo do diagrama de nível mais elevado em um DFD de nível mais baixo. Se o DFD de segundo nível ainda for complexo, ele poderá ser expandido recursivamente. A figura 4.29 é um exemplo de expansão da figura 4.29.

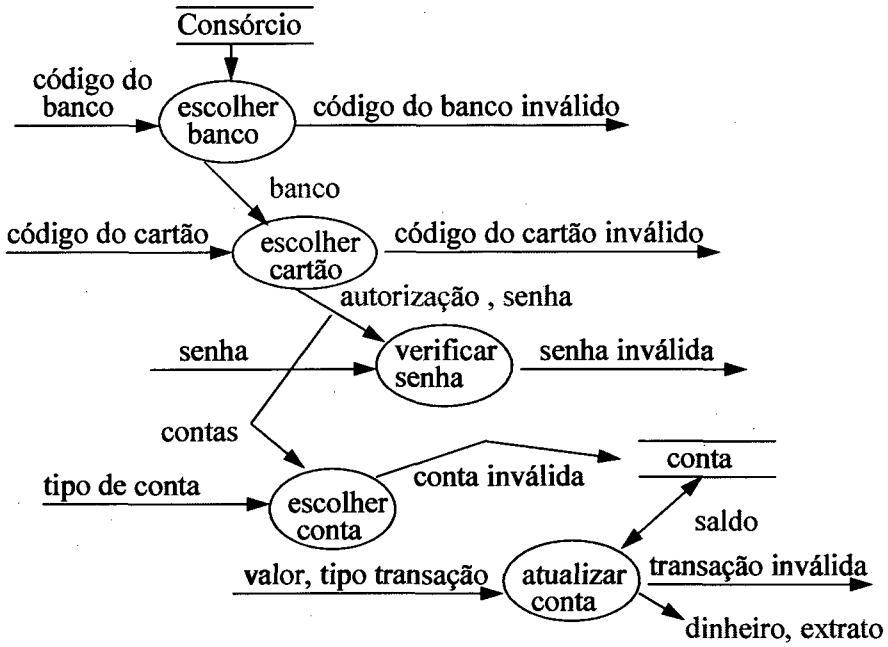


Figura 4.29 DFD do processo executar transação para ATM

A maioria dos sistemas contém objetos de armazenamento interno, que retêm valores entre interações (na figura 4.29 é utilizado o objeto Conta). Um depósito interno pode ser distinguido de um fluxo de dados ou de um processo, porque recebe valores que não resultam em saídas imediatas, e que são usados em algum momento distante.

Os DFDs especificam apenas dependências entre operações sem mostrar decisões ou seqüências de operação, eles são compostos por:

1. DESCREVER FUNÇÕES - Quando o DFD tiver sido suficientemente refinado, escrever a descrição de cada função. A descrição pode ser feita em linguagem natural. Concentrar o foco no que a função faz, não em como implementá-la. A descrição pode ser declarativa que especifica os relacionamentos entre os valores de entrada e saída. A

figura 4.30 mostra a descrição da função Atualizar Conta.

Atualizar conta (conta, valor, transação-tipo) ⇒ dinheiro, extrato, mensagem:
Se o valor de uma retirada exceder o saldo atual da conta, rejeitar a transação.
Se o valor da retirada não exceder o saldo, debitar a conta e entregar o dinheiro.
Se a transação for um depósito creditar o valor e não entregar o dinheiro.
Se a transação for uma solicitação de posição...

Figura 4.30 Descrição da função atualizar conta

2. RESTRIÇÕES ENTRE OBJETOS - As restrições são dependências funcionais entre objetos que não são relacionadas por uma dependência de entrada/saída. As restrições podem ser em dois objetos ao mesmo tempo, entre instâncias do mesmo objeto em momentos diferentes. Determinar os momentos, ou as condições sob as quais as restrições serão mantidas. Uma restrição no problema da ATM pode ser "nenhum saldo de conta pode ser negativo ou exceder o limite de crédito".

3. CRITÉRIOS DE OTIMIZAÇÃO - Especificar os valores a serem otimizados como número de mensagens físicas enviadas entre diferentes pontos; diminuir o tempo em que uma conta é bloqueada por motivos de concorrência.

4. PROCESSOS - Um processo é desenhado como uma elipse contendo uma descrição da transformação. Cada processo tem um número de setas de entrada e saída transportando valores. A figura 4.31 mostra o processo exibir ícone que representa o DFD da figura 4.27 em um nível mais elevado de abstração.

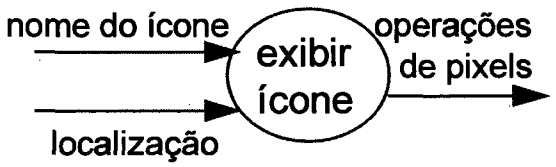


Figura 4.31 Processos

5. FLUXOS DE DADOS - Um fluxo de dados interliga a saída de um objeto ou processo à saída de outro objeto ou processo. As vezes um valor agregado de dados é subdividido

em seus componentes conforme mostra a figura 4.32.

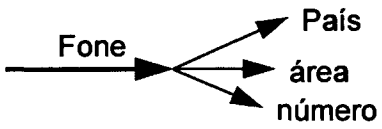


Figura 4.32 Fluxo de dados para subdividir um valor

6. ATORES - É um objeto ativo que dirige o diagrama produzindo ou consumindo os valores, eles se localizam nas extremidades. O ator é desenhado com um retângulo para mostrar que ele é um objeto. O "cliente" da figura 4.33 é um exemplo de ator.

7. DEPÓSITOS DE DADOS - É um objeto passivo que armazena dados para uso no futuro. O depósito permite que o acesso aos dados seja feito de forma diferente daquela em que foram gerados. Os depósitos são desenhados como um par de linhas paralelas contendo o nome do depósito. A figura 4.33 mostra um depósitos de dados chamado "conta".

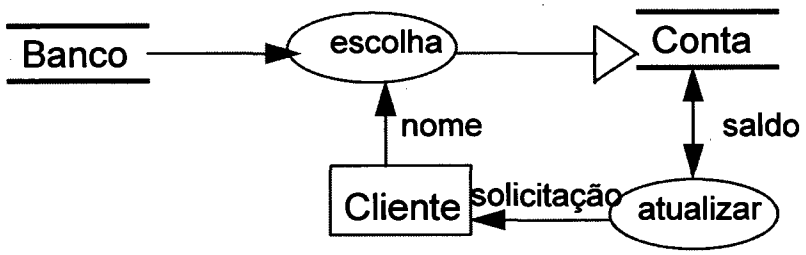


Figura 4.33 Depósitos de dados

8. DFD MULTINIVELADOS - Um DFD é especialmente útil para mostrar a funcionalidade de alto nível de um sistema e sua subdivisão em unidades funcionais menores. O processo pode ser expandido em outro diagrama de fluxo de dados. O novo diagrama pode ter depósitos que não aparecem no anterior. A decomposição em níveis deve ser feita até atingir o nível de decomposição desejado.

9. FLUXOS DE CONTROLE - Um DFD mostra todos os caminhos possíveis de valores; não são mostrados quais caminhos são executados e em que ordem. A figura 4.34 mostra um DFD para a retirada de uma conta bancária. O cliente informa a senha e a quantia, o

processo de atualização pode ser expandido com um fluxo de controle semelhante para impedir retiradas excessivas.

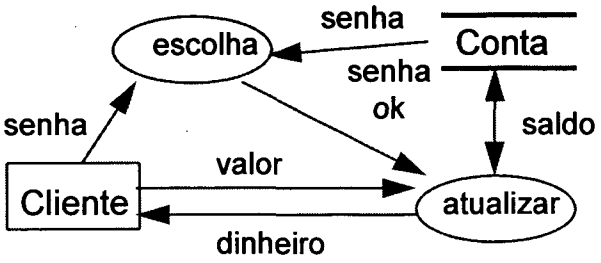


Figura 4.34 Fluxo de controle

10. ESPECIFICAR OPERAÇÕES - Os processos dos diagramas de fluxo de dados devem em algum momento ser implementados como operações sobre objetos. Cada processo do nível mais baixo, ou atômico, é uma operação. Cada operação inclui uma assinatura e uma transformação. A assinatura define a interface para a operação: os argumentos que ela exige e os valores que ela retorna. A transformação define o efeito de uma operação: os valores de saída como funções dos valores de entrada e os efeitos da operação em seus objetos.

A especificação externa de uma operação descreve somente as modificações visíveis de fora da operação. Durante a implementação podem ser criados valores internos para facilitar ou para fins de otimização.

Operações de acesso são operações que lêem ou escrevem os atributos ou ligações de um objeto. Durante o projeto, é necessário observar quais operações de acesso serão públicas e quais serão privativas da classe de objetos.

4.6.5 REGISTRAR OS SERVIÇOS

Os DFDs produzidos devem integrar a documentação do sistema.

4.7 A ITERAÇÃO DA ANÁLISE

A etapa da análise exige mais do que uma iteração para ficar completa. Os enunciados de problemas, em sua maioria, contêm falhas e a maior parte das aplicações não pode ser abordada de uma forma completamente linear, porque partes diferentes do problema interagem entre si. Para conhecer um problema em todas as suas implicações, pode-se atacar a análise iterativamente, preparando uma primeira aproximação para o modelo e depois repetir a análise à medida que o seu conhecimento se amplia. Não existe uma linha nítida entre análise e projeto.

Quando a análise está completa, o modelo serve como base para os requisitos e define o escopo do raciocínio subseqüente. Alguns requisitos especificam restrições de desempenho e métodos de soluções.

4.8 DOCUMENTAÇÃO DA ANÁLISE

Reunir a documentação completa do modelo de Análise:

- A especificação do problema;
- Classe e Objetos;
- Atributos;
- Modelo de Objetos;
- Estruturas e
- Serviços;

A meta da análise é especificar integralmente o problema e o domínio da aplicação sem introduzir desvios. O modelo de análise final serve como base para a arquitetura, o projeto e a implementação do sistema.

5. PROJETO

No projeto de objetos deve-se executar a estratégia escolhida na análise de sistema acrescentando novos detalhes. Os objetos descobertos na análise, servem como base para o projeto, mas deve-se escolher entre diferentes maneiras de implementá-los buscando a minimização do tempo de execução, memória e outras medidas de custo como por exemplo, o armazenamento de dados em disco e o tráfego de dados na rede.

As operações identificadas durante a análise devem ser expressas como algoritmos. As classes, atributos e associações da análise devem ser implementadas como estruturas específicas de dados. Novas classes de objetos podem ser introduzidas para armazenar resultados intermediários durante a execução do programa ou evitar a necessidade de recomputação. A otimização do projeto deve levar em conta as facilidades de implementação e manutenção.

Esta metodologia divide as fases do projeto em:

- Projeto de objetos;
- Interface humana - GUI;
- Distribuição de processos e
- Distribuição de dados.

5.1 PROJETO DE OBJETOS

Na fase de projeto são elaborados, refinados e otimizados os modelos de análise para a produção de um sistema. Durante a análise constrói-se o modelo de objetos sobre o qual é construído o projeto. O modelo de objetos da análise não pode mostrar operações. O analista deve converter as ações e atividades em operações vinculadas às classes do mo-

delo de objetos. Ao fazer essa conversão, inicia-se o processo de mapeamento da estrutura lógica do modelo de análise na organização física de um programa.

Esta fase da metodologia está baseada nas especificações de Rumbaugh [RUM 94] e compõem-se nas seguintes itens:

- Projetar os algoritmos;
- Otimizar o projeto;
- Implementar o controle;
- Ajustar a herança;
- Projetar as associações;
- Representação de objetos e
- Empacotamento físico.

5.1.1 PROJETAR OS ALGORITMOS

Cada serviço especificado durante a análise deve ser formulado como um algoritmo. A especificação da análise diz o que a operação faz do ponto de vista dos seus clientes, mas o algoritmo mostra como é feito. O projetista deve se preocupar com:

- Escolher algoritmos;
- Escolher estruturas de dados;
- Definir classes e operações;
- Atribuir responsabilidade por operações.

ESCOLHER ALGORITMOS - A escolha de algoritmos deve ser feita com a finalidade de minimizar os custos de implementação. A figura 5.1 mostra um objeto "Quadro de Classe" contendo uma "Lista de Operações", que por sua vez contém um conjunto de objetos "Entrada de Operações". Não há necessidade de se escrever um algoritmo para achar o

quadro de classe contendo uma determinada entrada de operação, porque o valor é encontrado por um simples caminho de ligações únicas.

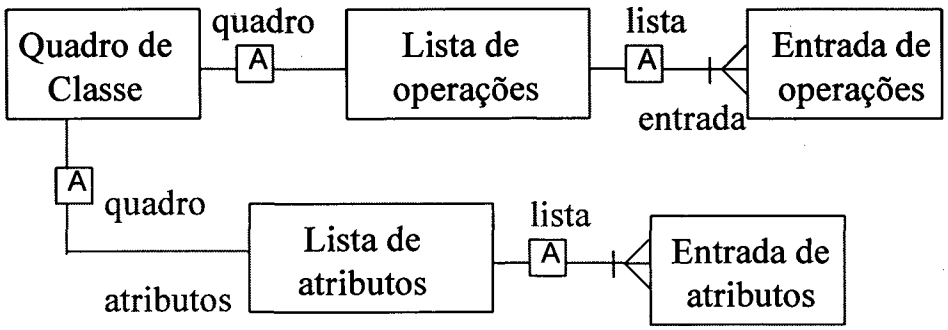


Figura 5.1 Fragmento de modelo

O nível de abstração dos algoritmos não deve ficar abaixo do nível de granularidade dos objetos no modelo de objetos. Por exemplo, ao esboçar o algoritmo recursivo de desenho para cada quadro da classe da figura 5.1, não é adequado preocupar-se com as chamadas de gráficos de baixo nível, que desenharão o ícone de quadro. As considerações na escolha entre os algoritmos alternativos incluem:

- **Complexidade computacional** - Não se preocupar com pequenos fatores de eficiência. Por exemplo, um nível a mais é insignificante se ajudar a melhorar a clareza. É essencial, entretanto, considerar a complexidade do algoritmo, isto é, como aumenta o tempo de execução (ou memória) com o número de valores de entrada.

- **Facilidade de implementação e compreensão** - pode-se desistir de um melhor desempenho em operações de pouca importância se elas puderem ser implementadas rapidamente e com um algoritmo simples.

- **Flexibilidade** - a maioria dos programas será alterada com o tempo. Um algoritmo altamente otimizado muitas vezes sacrifica a legibilidade e a facilidade de modificações.

ESCOLHER ESTRUTURAS DE DADOS - A escolha de algoritmos envolve a escolha de estruturas de dados em que eles atuam. Durante a análise, as atividades ocorrem na estrutura lógica das informações do sistema, mas durante o projeto de objetos deve-se escolher a forma das estruturas de dados que permitirão algoritmos eficientes. As estruturas de dados não acrescentam informações ao modelo de análise, mas o organizam sob uma

forma conveniente para os algoritmos que o utilizam. Muitas estruturas de dados de implementação são instâncias de classes contenedoras. Tais estruturas de dados incluem arranjos, lista, filas, pilhas, conjuntos, dicionários, associações e árvores.

DEFINIR CLASSES E OPERAÇÕES - Durante a expansão dos algoritmos, novas classes de objetos podem ser necessárias para guardarem resultados intermediários. Novas operações de baixo nível podem ser criadas durante a decomposição de operações de alto nível.

Uma operação complexa pode ser definida em termos de operações de nível mais baixo sobre objetos mais simples. Essas operações de baixo nível devem ser definidas durante o projeto de objetos porque a maioria delas não é visível externamente. Geralmente, as classes de implementação de baixo nível são colocadas em um módulo distinto.

ATRIBUIR RESPONSABILIDADE POR OPERAÇÕES - Quando uma classe é significativa, geralmente as operações sobre ela são claras. Durante a implantação, entretanto, são introduzidas classes internas que não correspondem a objetos do mundo real, porem, apenas, a algum aspecto deles. Uma vez que as classes internas são criadas para a implementação, elas são um tanto arbitrárias, e seus limites são mais uma questão de conveniência do que necessidade lógica.

Uma operação pertence a uma classe se ela tiver apenas um objeto. Se a operação tiver mais de um objeto envolvido, as seguintes perguntas podem determinar a qual objeto a operação pertence:

- Um objeto é influenciado enquanto o outro objeto executa a ação? Em geral, é melhor associar a operação com o alvo da operação, em vez de com o iniciador.

- Um objeto é modificado pela operação, enquanto outros objetos são apenas consultados pelas informações que contêm? O objeto que for modificado é o alvo da operação.

- Examinando as classes e associações que estão envolvidas na operação, qual das classes está localizada mais centralmente na sub-rede do modelo de objetos? Se as

classes e as associações formam uma estrela em torno de uma classe central única, ela é o alvo da operação.

É comum movimentar-se uma operação para cima e para baixo na hierarquia durante o projeto, à medida que seu escopo é ajustado.

5.1.2 OTIMIZAR O PROJETO

A otimização do projeto consiste em efetuar as seguintes etapas:

- Acrescentar associações redundantes.
- Reorganizar o processamento.
- Salvar atributos derivados.

ACRESCENTAR ASSOCIAÇÕES REDUNDANTES - Para a eficiência do projeto, não é desejável a existência de redundâncias na rede de associações, porque as associações redundantes não acrescentam quaisquer informações. Durante o projeto, entretanto, avaliar a estrutura do modelo de objetos para implementação. Avaliar se existe alguma organização de rede que pode otimizar os aspectos críticos do sistema. Verificar se a rede deve ser reestruturada, acrescentando novas associações.

Uma parte do modelo de objeto da fase de análise é mostrado na figura 5.2. A operação "tem-habilidade" retorna um conjunto de pessoas da empresa com uma determinada habilidade. Por exemplo, pode-se pedir todos os empregados que falam japonês.

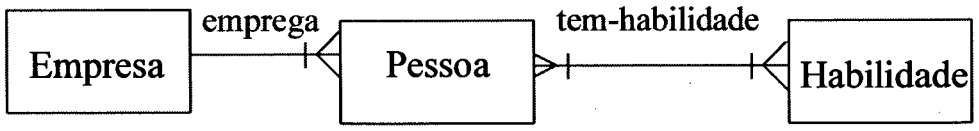


Figura 5.2 Cadeia de associações

Diversos melhoramentos são possíveis. Primeiro "Tem-habilidade" não precisa ser implementada ordenada, e sim, um conjunto misto. Nos casos em que uma pesquisa recupera muitas informações para satisfazer a um teste pode-se construir um índice para melhorar o acesso aos objetos que devem ser recuperados com freqüência. Por exemplo, pode-se acrescentar uma associação qualificada "Fala-Linguagem" proveniente de Empresa por

Empregado, em que o qualificador é a língua falada conforme mostra a (figura 5.3).

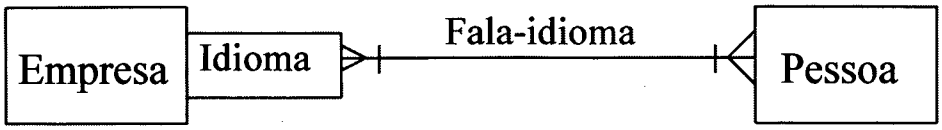


Figura 5.3 Índice para banco de dados

Fala-idioma é uma associação derivada, definida em termos das associações básicas subjacentes. A associação derivada não acrescenta qualquer informação à rede, mas permite o acesso às informações do modelo de forma mais eficiente.

Examinar cada operação para ver quais associações ela deve percorrer para obter suas informações:

- Quantas vezes a operação é chamada?
- Quanto custa sua execução?

Preparar índices para operações freqüentes e dispendiosas com baixa taxa de acertos, porque tais operações não são eficientes para implantar a utilização de loops multinivelados para percorrer um caminho na rede.

REORGANIZAR O PROCESSAMENTO - Depois de ajustar a estrutura do modelo de objetos para otimizar travessias freqüentes, o próximo item a ser otimizado para melhorar a eficiência é o algoritmo. As estruturas de dados e algoritmos são diretamente relacionados entre si. A chave para a eliminar algoritmos é suprimir caminhos inúteis.

SALVAR ATRIBUTOS DERIVADOS - O dado redundante, que pode ser derivado de outros dados, pode ser "cached" ou armazenado em sua forma computada para evitar a sobrecarga de trabalho de reprocessamento. Novos objetos ou novas classes podem ser definidas para reter essa informação.

A utilização de uma associação cache é mostrada na figura 5.4. Uma folha contém uma lista de prioridades de elementos parcialmente sobrepostos. Se um elemento for movimentado ou eliminado, os elementos sob ele devem ser redesenhados. Elementos sobrepostos podem ser encontrados pelo exame de todos os elementos que estiverem à frente do elemento eliminado. Se o número de elementos for grande, esse algarismo cresce linearmente com o número de elementos. A associação "sobrepõe" armazena os elementos que se sobrepõem a um objeto e o precedem na lista. Essa associação deve ser atualizada

quando um novo elemento for acrescentado, mas o teste de sobreposição com utilização de associação, pode ser mais eficiente.

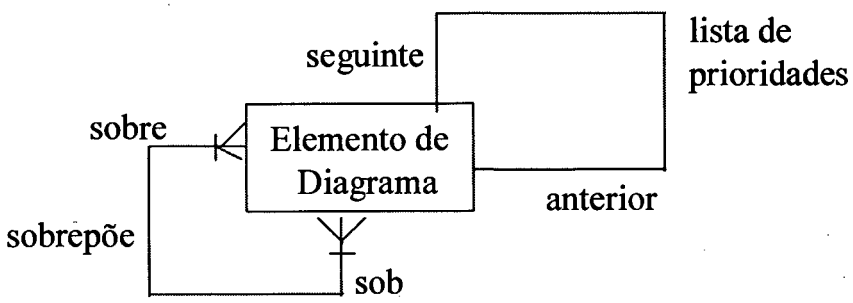


Figura 5.4 A associação como um cache

5.1.3 IMPLEMENTAR O CONTROLE

A localização do controle dentro de um programa implicitamente define o estado do programa. Qualquer máquina de estados finitos pode ser implementada como um programa. Cada transação de estado corresponde a um comando de entrada. Depois que a entrada é lida, o programa se desvia dependendo dos eventos de entrada recebidos. Cada instrução de entrada precisa manipular qualquer valor de entrada que possa ser recebido naquele momento. Em programas procedurais extremamente divididos em níveis, os procedimentos de baixo nível devem ser receber entradas, das quais nada conhecem, e passá-las através de muitos níveis de chamadas de procedimentos até que algum procedimento esteja preparado para manipulá-las. A falta de modularidade é a maior desvantagem dessa abordagem. Uma técnica de conversão de um diagrama de estados em um programa pode ser:

1. Identificar o principal caminho de controle. Começando pelo estado inicial, identificar um caminho através do diagrama que corresponda à seqüência de eventos esperado. Escrever os nomes dos estados ao longo desse caminho em seqüência linear. Isso se tornará uma seqüência de comandos de um programa.

2. Identificar os caminhos alternativos que se desviam do caminho principal e retornam a ele posteriormente. Eles tornar-se-ão os comandos condicionais do programa.

3. Identificar os caminhos retroativos que se desviam do loop principal e que se reúnem a ele no início. Eles se tornam loops no programa. Se existirem múltiplos caminhos re-

troativos que não se cruzam, eles se tornarão loops aninhados no programa. Caminhos retroativos que se cruzam não se aninham e podem ser implementados com *gotos*.

4. Os estados e transições que permanecem correspondem a condições de exceção. Eles podem ser manipulados por diversas técnicas, incluindo sub-rotinas de erros, tratamento de exceções suportado pela linguagem, e estabelecimento de teste e flags de status. O tratamento de exceções é um emprego legítimo para *gotos* em uma linguagem de programação, porque sua utilização freqüentemente simplifica a subdivisão de uma estrutura multinivelada, não utilizá-la a menos que seja necessário. Para sistemas com aquisição de dados e responsabilidades de controle sobre dispositivos locais, são necessárias múltiplas tarefas:

- Para interfaces com muitas janelas;
- Para sistemas multiusuário, muitas cópias das tarefas de usuário;
- Para arquiteturas de software com subsistemas (coordenar a comunicação);
- Para muitas tarefas em um único processador (comunicação entre as tarefas);
- Para arquiteturas de hardware com múltiplos processadores, tarefas são necessárias para cada processador;

Para sistemas que se comunicam com outros sistemas. Cada tarefa deve ser projetada, analisada e justificada. Diferentes tarefas separam comportamentos que devem ocorrer concorrentemente. Esse comportamento concorrente pode ser implementado em processadores diferentes ou ser simulado quando um único processador é usado em conjunto com um sistemas operacional multitarefa.

A seleção e justificativa das tarefas deve ser de acordo as seguintes estratégias:

- TAREFAS POR EVENTOS - Certas tarefas são dirigidas por eventos. Tais tarefas podem ser responsáveis pela comunicação com um dispositivo com uma ou mais janelas numa tela com outra tarefa, com um subsistema, com outro processador ou com outro sistema. Uma tarefa pode ser projetada para disparar um evento, geralmente assinalando o surgimento de algum dado. Esse dado pode vir de uma linha de dados de entrada ou de um buffer de dado, escrito por outra tarefa. A tarefa está adormecida (não consumindo tempo de processamento), esperando por uma interrupção a partir de uma linha de dados ou de alguma outra fonte. Na recepção da interrupção a tarefa desperta, captura os dados, coloca-os em um buffer na memória ou em algum outro destino, notifica e volta a dormir.

- TAREFAS POR TEMPO - Essas tarefas são disparadas para fazer algum processamento num intervalo de tempo especificado. Certos dispositivos podem necessitar periodicamente de aquisição e controle de dados. Certas interfaces humanas, subsistemas, tarefas, processadores ou outros sistemas podem precisar de comunicação periódica. Esse é o uso freqüente de uma tarefa dirigida por tempo. A tarefa estabelece uma hora de acordar e vai dormir. A tarefa aguarda uma interrupção do sistema para acordar e fazer o serviço, notificar e voltar a dormir.

- TAREFAS PRIORITÁRIAS - Uma tarefa prioritária acomoda tanto as necessidades de processamento de alta como de baixa prioridade. Uma tarefa crítica é usada para isolar o processamento para o sucesso ou falha do sistema em consideração, este tipo de processamento geralmente tem restrições de segurança.

- IDENTIFICAR UM COORDENADOR - Com três ou mais tarefas, considerar a adição de outra tarefa, que poderia atuar como um coordenador. Essa tarefa acrescenta custos gerais; o tempo de comutação de contexto pode desencorajar essa abordagem de projeto. Apesar disso, tal tarefa pode trazer o benefício do encapsulamento da coordenação intertarefas.

- OBSERVAR CADA TAREFA - Reduzir ao mínimo o número de tarefas. Quanto menor o número de tarefas, melhor será a compreensibilidade durante o desenvolvimento e a manutenção.

- DEFINIR CADA TAREFA - Começar pela especificação do que é a tarefa: dar um nome à tarefa; descrevê-la resumidamente, conforme a figura 5.5. Acrescentar restrições a cada tarefa ou serviço nos componentes do projeto. Atribuir a essa restrição um valor para cada serviço nos componentes do projeto orientado a objetos. Definir como cada tarefa coordena. Indicar se a tarefa é dirigida por evento ou por tempo. Definir como cada tarefa se comunica. De onde ela obtém seus valores.

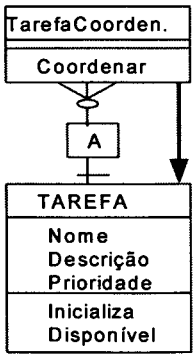


Figura 5.5 Gabarito para componentes de gerenciamento de tarefas

5.1.4 AJUSTAR A HERANÇA

À medida que o projeto de objetos progride, as definições de classes e operações podem, muitas vezes, ser ajustadas para aumentar a herança. A herança permite as seguintes facilidades:

- Classificar Objetos. Por exemplo, pode-se categorizar círculos, retângulos e triângulos porque tipos diferentes de formas compartilham tudo que possuem em comum.

- Expressar diferenças entre classes relacionadas enquanto compartilha funções e variáveis membros que implementa recursos comuns.

- Reutilizar código existente de uma ou mais classes simplesmente derivando uma nova classe a partir dela.

- Estender a classe existente, adicionando novos membros a ela.

Para ajustar a herança e fazer uso das facilidades acima, deve-se:

- Reorganizar e ajustar classes e operações.
- Abstrair o comportamento comum entre as classes.
- Utilizar a delegação.
- Nível de herança

.**REORGANIZAR E AJUSTAR CLASSES E OPERAÇÕES** - A finalidade desta reorganização deve ser feita para aumentar a utilização da herança. Algumas vezes, a mesma operação é definida através de diversas classes e pode, facilmente, ser herdada de um ancestral comum, porém, mais freqüentemente, as operações em diferentes classes são semelhantes, mas não idênticas. Modificando-se ligeiramente as definições das operações ou das classes, pode-se muitas vezes, fazer com que as operações coincidam para que possam ser cobertas por uma única operação herdada.

Para que a herança possa se usada, cada operação deve ter a mesma interface e a mesma semântica. Todas as operações devem ter a mesma assinatura, isto é, o mesmo número e tipos de argumentos e resultados. Se as assinaturas coincidirem, as operações devem ser examinadas para verificar se a semântica é a mesma. Os seguintes tipos de ajustamentos podem ser usados para aumentar a possibilidade da herança.

- Algumas operações podem ter menos argumentos do que outras. Os argumentos ausentes podem ser acrescentados, mas ignorados. Por exemplo, uma operação de desenho em um monitor monocromático não necessita de um parâmetro de cor, mas o parâmetro pode ser recebido e ignorado por uma questão de consistência com monitores coloridos.

- Algumas operações podem ter menos argumentos, porque são casos especiais de argumentos genéricos. Implementar as operações especiais pela chamada da operação geral com valores de parâmetros apropriados.

- Atributos semelhantes em classes diferentes podem ter nomes diferentes. Dar aos atributos o mesmo nome e movimentá-los para uma classe ancestral comum.

- Uma operação pode ser definida em diversas classes diferentes de um grupo, porém indefinida nas outras classes. Ela deve ser definida na classe mais ancestral e ser declarada "Não Operacional" nas classes que não se preocupam com ela.

ABSTRAIR O COMPORTAMENTO COMUM ENTRE CLASSES - As oportunidades para se usar a herança não são sempre reconhecidas durante a fase da análise do desenvolvimento, por causa disso é conveniente reexaminar o modelo de objetos em busca de coisas comuns entre classes. Além disso, novas classes e operações são sempre acrescentadas durante o projeto. Se um conjunto de operações e/ou atributos aparenta repetir-se em duas classes, é possível que as duas classes sejam na realidade variações especializadas da mesma coisa quando vistas de um nível mais alto de abstração.

Quando um comportamento comum for reconhecido, poderá ser criada uma superclasse comum, que implemente as características compartilhadas, deixando somente as características especializadas nas classes. Essa transformação do modelo de objetos é denominada abstração de uma superclasse comum, que implementa as características compartilhadas, deixando somente as características especializadas nas subclasses. Essa transformação do modelo de objetos é denominada abstração de uma superclasse comum ou de um comportamento comum. Habitualmente, a superclasse resultante abstrata, significando que não existem instâncias diretas dela, mas o comportamento que ela define pertence a todas as instâncias de suas subclasses. Por exemplo, a operação de desenho de uma figura geométrica em uma tela de monitor exige preparação e execução de geometria. A execução varia entre figuras diferentes, como círculos, linhas e estrias, mas a preparação, assim como o ajuste da cor, da espessura das linhas e de outros parâmetros podem ser herdadas por todas as classes de figuras.

As vezes vale a pena abstrair uma superclasse mesmo quando existe apenas uma subclasse que herdou dela no projeto. Embora isso não resulte em qualquer compartilhamento de comportamento no projeto imediato, as superclasses abstratas assim criadas podem ser reaproveitadas em futuros projetos.

As superclasses abstratas têm outros benefícios além do compartilhamento e da reutilização. A subdivisão de uma classe em duas, que separam os aspectos específicos dos aspectos mais genéricos, é uma forma de modularidade. Cada classe é um componente mantido separadamente com interface bem documentada.

UTILIZAR A DELEGAÇÃO - Com a finalidade de compartilhar o comportamento quando a herança for semanticamente inválida. A herança é um mecanismo para implementar a generalização, em que o comportamento de uma superclasse é compartilhado por todas as suas subclasses. O compartilhamento do comportamento só é justificável quando ocorre um verdadeiro relacionamento de generalização, isto é, somente quando pode ser dito que a subclasse é uma forma da superclasse. As operações da subclasse que cancelam a operação correspondente da superclasse têm a responsabilidade de oferecer os mesmos serviços fornecidos pela superclasse, e possivelmente mais. Quando a classe B herda a especificação da classe A, podemos admitir que cada instância da classe B é uma instância da classe A, porque ela se comporta da mesma maneira.

Muitas vezes, quando se utiliza a herança como uma técnica de implementação, pode-se alcançar a mesma meta de forma mais segura fazendo de uma classe um atributo, ou associada de outras classes. Desse modo, um objeto poderá chamar seletivamente as funções desejadas de outra classe, utilizando delegação em vez de herança. A delegação consiste em capturar uma operação de um objeto e enviá-la para outro objeto que faça parte ou seja relacionado ao primeiro objeto. Apenas as operações significativas são delegadas ao segundo objeto, de modo que não existe perigo de serem herdadas operações sem significado.

Em geral, é melhor não utilizar a herança estritamente para fins de implementação. Reservar o uso da herança para casos onde se pode dizer que a instância de uma classe também é instância de alguma outra classe.

NÍVEL DE HERANÇA - Acomodar o nível de herança suportado pela linguagem de programação. Se a herança múltipla não for suportada, pode-se modificar a estrutura para que utilize Herança Única ou Herança Zero como segue:

1) Herança Única. Em uma linguagem de herança única podem ser aplicadas duas abordagens para sair de uma estrutura de herança múltipla para uma estrutura de herança única: Decompor em múltiplas hierarquias e/ou Nivelar dentro de uma hierarquia única.

- **Decompor em múltiplas hierarquias**, com mapeamentos entre elas. Nessa abordagem, divide-se o padrão de herança múltipla em duas hierarquias, com um mapeamento, ou seja, uma estrutura de agregação ou uma conexão de ocorrência, entre elas. As figuras 5.6 e 5.7 apresentam um exemplo de decomposição.

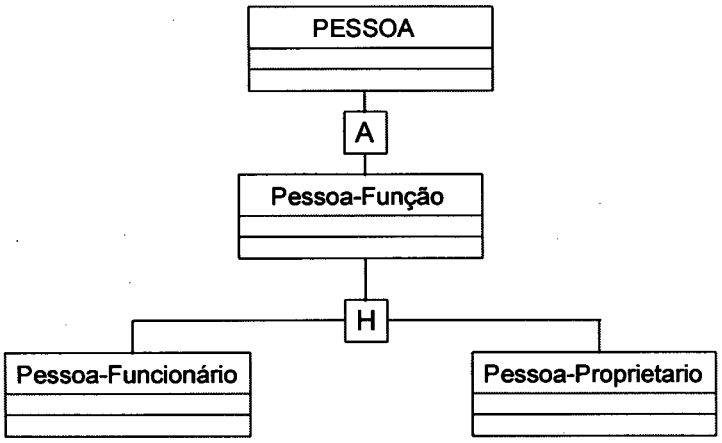


Figura 5.6 Acomodar herança única (1)

Alternativamente isso pode ser modelado com uma conexão de ocorrência, escolhendo a opção que melhor preserva a compreensão do domínio do problema, demonstrado na figura 5.7.

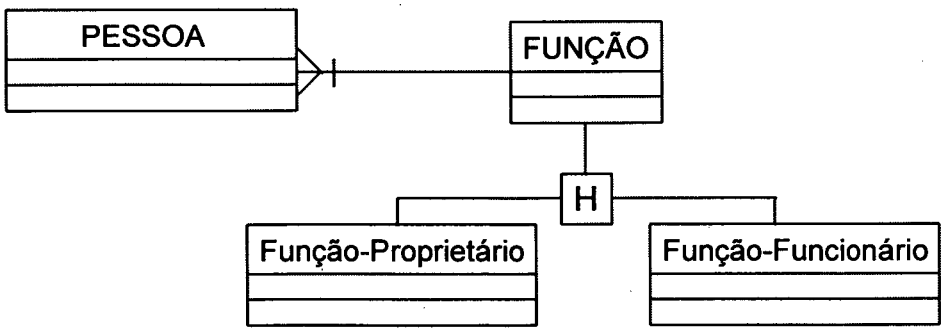


Figura 5.7 Acomodar herança única (2)

- **Nivelar dentro de uma hierarquia única.** Nessa abordagem nivela-se uma hierarquia de herança múltipla numa hierarquia de herança única. Isso quer dizer que uma ou mais generalizações-especializações já não estão explícitas no projeto e que alguns atributos e serviços serão repetidos nas classes de especialização exemplificado na figura 5.8.

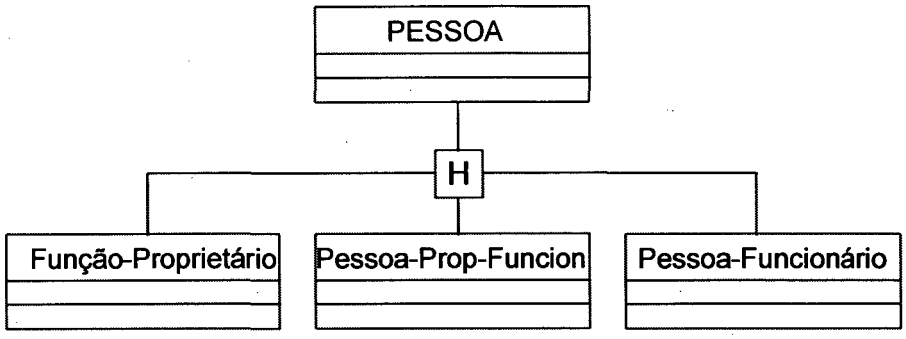


Figura 5.8 Herança

2) Acomodar uma linguagem de herança zero. A herança em uma linguagem de programação proporciona uma sintaxe para captar a semântica do domínio do problema, referente à generalização-especialização; ela expressa claramente atributos e serviços comuns e proporciona uma base para a reutilização através de extensibilidade.

Porém, por uma série de razões organizacionais, alguns projetos pode optar passar da AOO e DOO (Projeto Orientado a Objetos) para uma linguagem de herança zero. Para uma linguagem de herança zero, cada estrutura Gen-Espec necessitará nivelar a hierarquia num agrupamento de classes objetos - embora isso possa ser feito na própria linguagem de programação, usando as convenções de nomenclatura para agrupá-las, em vez de fazê-lo explicitamente no modelo do DOO, nesse ponto do projeto, conforme a mostra a figura 5.9.

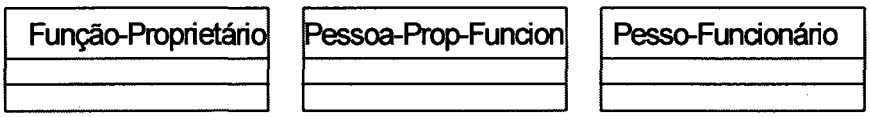


Figura 5.9 Acomodar herança zero

5.1.5 PROJETAR AS ASSOCIAÇÕES

As associações providenciam caminhos de acesso entre os objetos. São entidades conceituais úteis para a modelagem e para a análise. Durante a fase do projeto de ambos é

preciso formular uma estratégia para a implementação das associações no modelo de objetos. Esta estratégia consiste em:

- Análise das associações;
- Associações de sentido único;
- Associações de dupla direção e
- Atributos de ligação.

ANÁLISE DAS ASSOCIAÇÕES - Até agora foi presumido que as associações são intrinsecamente bidirecionais, o que certamente é verdadeiro em sentido abstrato. Mas se algumas associações da aplicação são percorridas apenas em uma direção, sua implementação pode ser simplificada. Contudo, os requisitos da aplicação podem mudar, e mais tarde pode ser acrescentada uma nova operação que necessite percorrer a associação em direção contrária.

Para serviços de prototipagem, utilizar as associações bidirecionais para acrescentar um novo comportamento e expandir ou modificar rapidamente a aplicação. Para serviços de implementação otimizar algumas associações. Qualquer que seja a estratégia de implementação escolhida, ocultar a implementação utilizando operações de acesso para percorrer e atualizar as associações. Isso permitirá modificar a decisão com esforço mínimo.

ASSOCIAÇÕES DE SENTIDO ÚNICO - Se uma associação for percorrida em uma só direção, ela poderá ser implementada como um ponteiro - um atributo que contém uma referência a um objeto. Se a multiplicidade for "um", ela é um simples ponteiro; se a multiplicidade for "muitos", então ela é um conjunto de ponteiros. Se a extremidade "muitos" for ordenada, poderá ser usada uma lista no lugar do conjunto. A primeira parte da figura 5.10 mostra muitas pessoas que "trabalha para" a empresa. A segunda parte mostra uma forma de implementar esta associação, um ponteiro "empregador" apontando para "empregados".

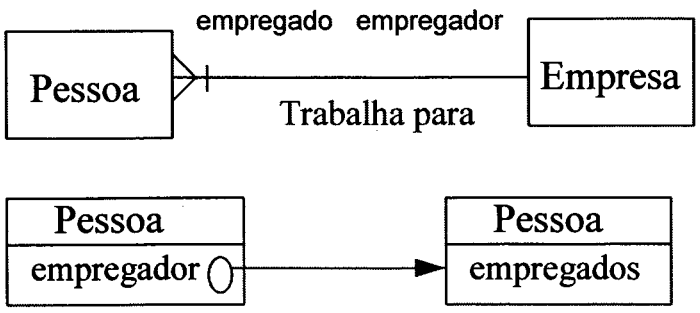


Figura 5.10 Implementação de associação

ASSOCIAÇÕES DE DUPLA DIREÇÃO - Muitas associações são percorridas em ambas as direções, embora, habitualmente, sem a mesma freqüência. Existem três abordagens para a sua implementação.

- Como um atributo em uma só direção, e executar uma pesquisa quando for exigida uma travessia inversa.

- Implementar como atributos em ambas as direções, utilizando técnicas como a da figura 5.11. Essa abordagem permite acesso rápido, mas há o overhead da atualização.

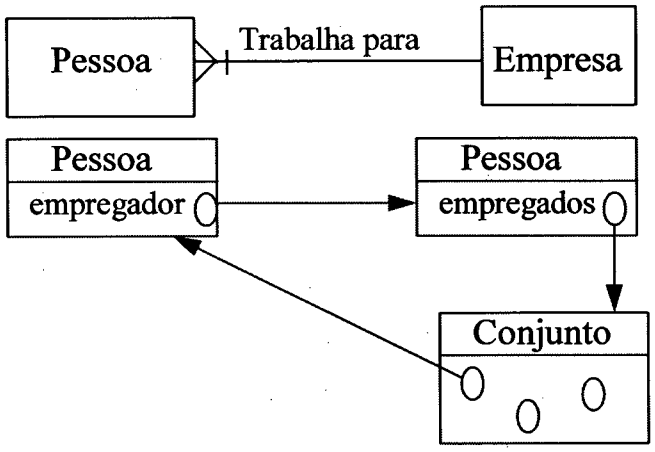


Figura 5.11 Associação de dupla direção

ATRIBUTOS DE LIGAÇÃO - Se uma associação possui atributos de ligação, então sua implementação vai depender da multiplicidade. Se a associação for um-para-um, os atributos de ligação podem ser armazenados como atributos de qualquer um dos objetos. Se a associação for muitos-para-um, os atributos de ligação podem ser armazenados como atributos do objeto "muitos", uma vez que cada objeto "muitos" só aparece uma vez na associação. Se a associação for muitos-para-muitos, os atributos de ligação não podem ser associados com qualquer objeto; a melhor abordagem costuma ser implementar a associação como uma classe distinta, em que cada instância representa uma ligação e seus atributos.

5.1.6 EMPACOTAMENTO FÍSICO

Os programas são constituídos de unidades físicas discretas que podem ser editadas, compiladas, importadas ou manipuladas de alguma outra forma. Na linguagem Ada, por exemplo, "pacote" é uma construção explícita da linguagem para modularidade. As linguagens baseadas em objetos têm vários graus de empacotamento, em qualquer projeto de grande porte, o cuidadoso particionamento de uma implementação em pacotes é importante para permitir que diferentes pessoas trabalhem cooperativamente em um programa. O empacotamento envolve as seguintes questões:

- Ocultamento de informações internas da visão externa.
- Coerência de entidades.
- Construção de módulos.

OCULTAMENTO DE INFORMAÇÕES - Uma das metas do projeto é considerar as classes como "caixas pretas", cujas interfaces externas são públicas, mas cujos detalhes internos permanecem ocultos à visão. O ocultamento de informações internas permite que a implementação de uma classe seja modificada sem exigir que qualquer cliente da classe modificar o código.

Do ponto de vista do empacotamento procurar minimizar as dependências, enquanto a otimização beneficia-se dos detalhes e pode conduzir a componentes e associações redundantes. O projetista deve equilibrar essas exigências conflitantes.

Durante a análise, não nos interessa o ocultamento de informações. Durante o projeto, entretanto, a interface pública de cada classe deve ser cuidadosamente definida. O projetista deve decidir quais atributos podem ser acessíveis de fora da classe. Essas decisões devem ser registradas no modelo de objetos pelo acréscimo de anotações (primitivas) após os atributos que devem ficar ocultos, ou pela divisão da lista dos atributos em duas partes.

Cada operação deve ter um conhecimento limitado do modelo completo, incluindo a estrutura de classes, associações e operações. Quanto menos itens uma operação conhecer, menos provável será que ela seja afetada por quaisquer modificações. De maneira inversa, quanto menos operações conhecerem detalhes sobre uma classe, mais fácil será modificar a classe, se isso vier a ser necessário. Os seguintes princípios de projeto ajudam a limitar o escopo do conhecimento de qualquer operação:

- Atribuir a cada classe a responsabilidade de executar operações e oferecer informações que pertençam a ela.

- Chamar uma operação para ter acesso aos atributos pertencentes a um objeto de uma outra classe.

- Evitar percorrer associações que não estejam interligadas à classe corrente.

- Definir interfaces em um nível de abstração tão alto quanto possível.

- Ocultar objetos externos nos limites do sistema pela definição de classes abstratas de interfaces, isto é, classes intermediárias entre o sistema e objetos externos não trabalhados.

- Evitar aplicação de um método ao resultado de outro método, a menos que a classe resultante seja fornecedora de métodos para o chamador, em vez disso, considerar a escrita como um método para combinar as duas operações.

COERÊNCIA DE ENTIDADES - Um importante princípio de projeto é a coerência de entidades. Uma entidade, assim como uma classe, uma operação ou um módulo, é coerente se estiver organizada de acordo com um plano consistente, e todas as suas partes se ajustarem em direção a uma meta comum. Uma entidade deve ter um único tema principal: ela deverá ser uma coleção de partes relacionadas.

Um método deve fazer bem alguma coisa. Um único método não deve conter política e implementação. Política é a tomada de decisões dependentes do contexto. Implementação é a execução de algoritmos inteiramente especificados. A política envolve a tomada de decisões, a reunião de informações globais, a interação com o mundo exterior e a interpretação de casos especiais. Um método político contém declarações de I/O, condicionais, e tem acesso a depósito de dados. Um método político não contém algoritmos complicados mas, em vez disso, chama diversos métodos de implementação. Um método de implementação executa exatamente uma operação sem tomar quaisquer decisões, sem fazer suposições ou desvios.

A separação da política de implementação aumenta muito a possibilidades de reusabilidade. Os métodos de implementação não contêm quaisquer dependências de contexto, de modo que eles provavelmente estão disponíveis para serem reutilizados. Os métodos

políticos geralmente devem ser reescritos em uma nova aplicação, porém eles são muitas vezes simples e consistem principalmente em decisões de alto nível e em chamadas em métodos de baixo nível.
Por exemplo, considerar uma operação para creditar juros em uma conta de cheques. Os juros são compostos diariamente com base no balanço diário, mas os juros de todo o mês são perdidos se a conta for fechada. O crédito de juros deve ser dividido em duas partes: um método de implementação que calcule os juros devidos em um par de dias, sem considerar quaisquer perdas ou outras provisões; e um método político que decide se e para que intervalo o método de implementação é chamado. Essa separação permite que tanto o processo político quanto o de implementação sejam modificados independentemente e aumenta muito a oportunidade de reutilização do método de implementação, que é provavelmente o mais complicado. Os métodos políticos são menos prováveis de serem utilizados, mas não costumam ser tão complicados, porque não contêm algoritmos computacionais.

Uma classe não deve servir para muitos propósitos ao mesmo tempo. Se ela for muito complicada, poderá ser subdividida utilizando-se generalização ou agregação. As unidades menores têm mais possibilidades de reutilização do que as maiores e mais complicadas.

A CONSTRUÇÃO DE MÓDULOS - Durante as fases de análise e de projeto de sistema, particionar o modelo de objetos em módulos (e por causa da limitações da tela ou do tamanho do papel os módulos podem ter sido ainda mais subdivididos em folhas). Essa organização inicial pode não ser adequada ou desejável para o empacotamento final da implementação do sistema. As novas classes que foram acrescentadas durante o projeto ou se incluem em um módulo ou camada existente ou podem ser organizadas em um módulo separado ou camada que não existia na análise.

Os módulos devem ser definidos de modo que suas interfaces sejam mínimas e bem definidas. A interface entre dois módulos consiste em associações que relacionam as classes de um módulo com as classes de outro e em operação que acessam classes através das fronteiras dos módulos.

Certamente, existem outros aspectos a serem considerados. Os módulos devem ter alguma coesão funcional ou unidade de propósito. As classes de um módulo devem representar espécies semelhantes de coisas na aplicação ou devem ser componentes do mesmo objeto composto.

O número de diferentes operações que percorrem uma determinada associação, é uma boa avaliação da firmeza do acoplamento. Esse número expressa a quantidade de diferentes maneiras pelas quais a associação é utilizada, e não a freqüência da travessia. Tentar encapsular acoplamentos fortes dentro de um único módulo.

5.2 PROJETAR A INTERFACE HUMANA - GUI

Uma interface [AID 91] com o usuário é um conjunto de técnicas e mecanismos que a pessoa usa para interagir com um objeto. Qualquer tipo de objeto tem uma interface com o usuário, e uma interface de objeto é desenvolvida de acordo com as necessidades do usuário e as razões para o uso do objeto.

A interface com o usuário pode ser um conjunto de botões, tais como um telefone ou um gravador de vídeo. No caso de um computador, a interface com o usuário pode incluir o teclado, um dispositivo apontador (mouse) e os itens que aparecem na tela. A interface permite ao usuário comunicar-se com o computador e vice-versa. As interfaces incluem:

- Linhas de comandos, entrada de comandos;
- Menus de navegação, hierárquico;
- Gráfico no qual o usuário aponta e interage com elementos visíveis.

No mundo real, um objeto é um item que uma pessoa possui para executar um trabalho como por exemplo uma máquina de calcular ou uma régua. Interfaces com usuários orientadas a objetos permitem ao desenvolvedor um meio de trabalho coerente no qual cada elemento é um objeto, os quais podem interagir com muitos outros elementos. Um objeto é qualquer componente visual de uma interface que o usuário pode trabalhar como se fosse uma unidade de trabalho, independentemente de outros itens, para executar uma tarefa. Cada objeto pode ser representado por uma ou mais imagens gráficas chamadas de ícones, com o qual o usuário interage. Entretanto, um objeto nem sempre é representado por um ícone, ele pode ser, por exemplo, uma janela para mostrar informações.

5.2.1 A IMPORTÂNCIA DE UMA INTERFACE

Para Chede [CHE 95], uma interface deve ser planejada e padronizada para todas as aplicações. Uma interface natural e intuitiva deve apresentar objetos que sejam

imediatamente reconhecidos pelo usuário e se comportem de maneira similar aos objetos do mundo real. Por exemplo, um botão é associado pelo usuário a uma função de controle, como ligar e desligar um aparelho qualquer. Os ícones devem ser escolhidos adequadamente e sua imagem deve lembrar ao usuário sua função. O projeto de interfaces gráficas é um aspecto de grande importância para o sucesso da aplicação. Ela deve ser simples, evitar os "arco-íris" nas telas e o uso de ícones obscuros. Os usuários devem ser envolvidos no projeto e nos padrões adotados.

É difícil [RUM 94] avaliar a interface com o usuário sem testá-la. Muitas vezes a interface pode ser simulada, para que os usuários possam experimentá-la.

A estratégia para desenvolver a interface de acordo com esta metodologia consiste em:

- A importância de uma interface;
- Análise e projeto da interface;
- A influência de cores e fontes;
- O sistema visual composto por janela, botões, menus e listas.
- O projeto de ícones;
- Mensagens;
- Multimídia.

5.2.2 ANÁLISE E PROJETO DA INTERFACE

As principais metas do projeto da interface são ajudar o usuário a transferir conhecimento dos produtos, incrementar a produtividade, aumentar a satisfação com o produto e reduzir a taxa de erros. Os princípios do projeto são colocar o usuário no controle da interface, reduzir a carga de memória do usuário e fazer uma interface consistente.

A interação humana [COA 93] precisa de exame detalhado, tanto na análise como no projeto. Na AOO, esse exame detalhado é feito de modo que os atributos e serviços requeridos - o conteúdo requerido - seja especificado. A prototipação pode ser usada para auxiliar durante a extração e especificação dos requisitos.

Na DOO, o componente de interação humana (CIH) acrescenta a esses resultados o projeto da interação humana e os detalhes dessa interação. Isso inclui o formato de ícones, botões, menus, listas, janelas e relatórios. A prototipação pode ser utilizada para auxiliar no desenvolvimento e seleção dos mecanismos reais de interação. A aplicação de uma estratégia sistemática suportada pela prototipação é importante para o sucesso da inter-

face. Esse componente capta como uma pessoa comandará o sistema e como o sistema apresentará as informações ao usuário.

Algumas organizações podem achar útil - até mesmo essencial - projetar alguma parte da interface em paralelo com a AOO. O Uso do modelo multicamadas, multicomponentes ajuda a separar a interface dependente de implementação a partir do trabalho de análise e requisitos da AOO; e essa separação reduz o impacto de mudanças devido a alterações na tecnologia de construção usada para implementar a própria interação humana.

As decisões de projeto afetam as pessoas. As emoções e percepções mentais de um indivíduo podem ser afetadas positiva ou negativamente. Além disso, o comportamento organizacional (isto é, a cultura empresarial) também pode mudar. A influência [COA 93] pode ser de largo alcance, comportando muitas respostas como:

- Pavor, raiva, exasperação e constrangimento;
- Tédio;
- Criatividade e satisfação.

Os analistas estudam os usuários com a finalidade de obter o contexto e o conteúdo corretos durante a AOO. O analista precisa estudar os usuários, projetando a interação específica, com o uso das tecnologias de interação disponíveis para um determinado sistema.

A estratégia para projetar esse componente consiste em:

- Classificar os usuários;
- Descrever os usuários em seus cenários de trabalho;
- Projetar a hierarquia de comando;
- Aprimorar a hierarquia de comando;
- Projetar a interação detalhada;
- Continuar no sentido da prototipação;
- Projetar as classes da interface e
- Projetar os ícones, botões, janelas, menus e listas.

1) CLASSIFICAR OS USUÁRIOS - Estudar os usuários que utilizarão o sistemas, para isso o analista pode fazer entrevistas e/ou observar suas atividades. Estudar como o sistema afetará a atividades das pessoas envolvidas. Perguntar o que elas desejam realizar, de que tarefas elas precisam? Que ferramentas podem ser elaboradas para suportar suas atividades? Como as ferramentas podem ser elaboradas para serem mais eficientes?

Começar pela classificação dos indivíduos em categorias diferentes. Para iniciar essa classificação, se uma estrutura Gen-Espec "pessoa" estiver presente no componente do domínio, usar o padrão de Gen-Espec como ponto de partida.

Após considerar subconjuntos adicionais de pessoas que irão interagir com o sistema que está sendo considerado. Examinar a classificação por uma ou mais dessas ordenações:

Por nível de conhecimento: ⇒ Iniciante/Ocasional/Intermediário/Avançado.
Por nível organizacional: ⇒ Executivo/Diretor/Funcionário/Supervisor/Auxiliar.
Pela associação em diferentes grupos: ⇒ Funcionário/Cliente/Acionista.

2) DESCREVER OS USUÁRIOS EM SEUS CENÁRIOS DE TRABALHO - Para cada categoria de pessoa definida na etapa anterior, considerar e tabular o seguinte:

Quem é o usuário;
Quais são seus propósitos;
Suas características (idade, instrução e limitações);
Fatores críticos para o sucesso;
    Necessidades/Desejos;
    Gostos/Empatias/Inclinações;
Nível de conhecimento;
Cenários de trabalho.

Exemplo:
Quem: Eu sou um executivo.
Propósito: Eu quero usar os resultados da análise para entender meu negócio, ajudar-me a reformulá-lo e obter vantagens estratégicas.
Características: 52 anos de idade, Curso superior, não gosto de letras pequenas.
Fatores: Eu desejo ver só os itens que me dizem respeito e quero que utilize a minha terminologia.
Cenários: Minha equipe de auxiliares prepara um modelo filtrando as informações significativas. Eu observo o modelo procurando vantagens competitivas.

3) PROJETAR A HIERARQUIA DE COMANDO - Projetar a hierarquia considerando os seguintes aspectos:

3.1 Estudar as maneiras de interação existentes com o usuário. Se a hierarquia de comando deve existir dentro de um sistema de interação estabelecido, começar pelo

estudo das maneiras de interação humana existentes. Isso é útil para interfaces gráficas com o usuário.

3.2 Estabelecer uma hierarquia de comando inicial que pode ser apresentada de várias maneiras:

- Uma série de telas de menu;
- Uma barra de menu, conforme a figura 5.12;
- Uma série de ícones que executam ações quando é colocado sobre eles.
- Começar com essa abstração básica de procedimentos dos serviços, e modificá-la para corresponder às necessidades específicas.

Arquivo	Editar	Inicializar	Estado	Estilo	Janela

Figura 5.12 Esboço de menu inicial

4) APRIMORAR A HIERARQUIA DE COMANDO - Para refinar a hierarquia mostrada na figura 05.19, levar em consideração: a ordenação, o agrupamento em blocos de agregação, a amplitude versus a profundidade e os passos mínimos de navegação.

- Ordenação. À medida que desenvolver tal hierarquia, selecionar cuidadosamente diferentes nomes de serviços. Depois, ordenar os nomes de serviços dentro de cada parte da hierarquia com o serviço mais freqüentemente usado aparecendo primeiro na lista, e na ordem das etapas de trabalho habitual.

- Agrupamento em blocos de agregação. Procurar pelos padrões de agregação através dos próprios serviços, para ajudar na organização e no agrupamento de serviços dentro da hierarquia.

- Agrupamento em blocos conforme a amplitude e a profundidade mostrado na figura 5.13.

- Trabalhar dentro das diretrizes de prevenção de sobrecarga da memória das pessoas.

Arquivo	**Editar**	**Inicializar**	**Estado**	**Estilo**
Abrir...	Adicionar...		Desligado...	Fonte...
Fechar	Alterar...		Disponível...	Tamanho...
Encontrar...	Eliminar...		Ligado...	

Figura 5.13 Agrupar em blocos

5) PROJETAR A INTERAÇÃO DETALHADA - A interação humana pode ser projetada (e depois avaliada) com base em uma variedade de critérios estabelecidos nesta metodologia:

- **Controle do usuário**. Desenvolver uma interface onde o usuário possa configurar janelas, cores, localização de ferramentas, menus, suprimir mensagens ou opções.

- **Clareza**. Colocar os elementos em uma disposição que permita destacar determinadas partes. A interface deve ser fácil de ler (a tendência é ler a tela de cima para baixo e da esquerda para a direita). A colocação de itens na tela deve ser conforme a sua importância levando em conta esta tendência. Deve-se também, evitar textos e/ou mensagens com erros gramaticais.

- **Feedback**. Para cada ação do usuário deve ter um retorno. Enviar mensagens esclarecedoras. Por exemplo, indicar o que o computador está fazendo, ou se a operação for demorada, exibir a quantidade de trabalho efetuada.

- **Consistência**. Usar termos, passos e ações consistentes com o assunto. A interface deve ser baseada no mundo real. Um comando de atalho deve ter o mesmo comportamento em todas as interfaces projetadas, mantendo um padrão.

- **Poucos passos**. Minimizar o número de digitações ou cliques do mouse e mesmo a distância no menu suspenso necessária para conseguir a execução de alguma tarefa. Minimizar o tempo necessário para obter resultados significativos nos diferentes níveis de conhecimento - iniciante, intermediário e avançado.

- **Reconhecimento X chamada**. Os humanos lembram de 7 elementos, (podendo aumentar ou diminuir 2 elementos). Neste aspecto é importante indicar ao usuários os quais foram os últimos objetos utilizados (ou arquivos). Os usuários não preci-

sam lembrar-se ou escrever informações de uma janela, para então aplicá-las em outras janelas. Indicar o caminho de acesso a diretórios, antecipando a ação do usuário. Agrupar as informações em grupos relacionados.

- **Desfazer tarefas - educação.** As pessoas podem cometer erros. Possibilitar que o usuário possa desfazer alguma tarefa. Pedir a confirmação de determinadas atividades como a deleção de informações.

- **Tempo de esforço para aprender**. Manter o esforço reduzido. Não esperar que as pessoas leiam a documentação impressa. Fornecer uma referência de ajuda on-line para os recursos mais avançados (desta forma os usuários podem descobrir detalhes conforme a sua necessidade).

- **Atratividade, Prazer e Encanto (aspecto e sensação)**. As pessoas usam o software quando ele é agradável, a menos que elas sejam obrigadas a utilizá-lo. Escolher um bom visual, com dimencionamento adequado de fontes e cores.

6) CONTINUAR NO SENTIDO DA PROTOTIPAÇÃO - A interação homem-computador é essencial para o projeto da interface. Os usuários precisam experimentar e aprimorar a interação proposta num padrão consistente. Fazer diversas alternativas de protótipos. Deixar o usuário experimentar cada uma delas. Consultar as pessoas. Observar suas reações e repetir gradualmente o trabalho até chegar a uma interface cada vez mais eficiente. A qualidade da interface leva a uma satisfação do usuário. Esta satisfação vem da estreita sintonia de cada peça da interação humana. Fazer perguntas aos usuários tais como:

- Como o usuário sente o produto?
- O que o usuário realmente gostaria de ter em seguida?
- Que seqüência de comandos poderia facilitar o seu trabalho?
- De que outras ferramentas ele precisa para ser mais eficiente?

7) PROJETAR CLASSES DA INTERFACE - As classes da interface variarão um pouco, dependendo da interface gráfica com o usuário que está sendo usada. Para projetar as

classes CIH, começar pela organização do projeto de interação humana, pelas janelas e componentes conforme a figura 5.14.

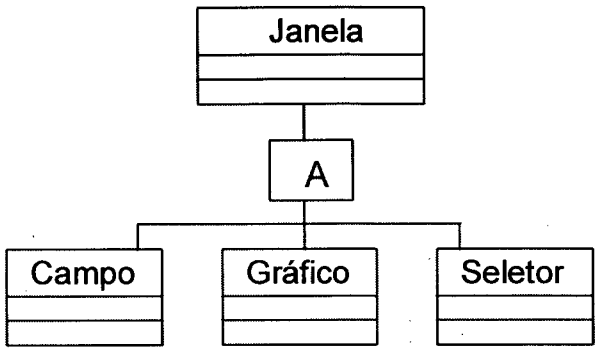


Figura 5.14 Classes de interação humana

Cada classe contém a definição da barra de menu, do menu suspenso e dos menus instantâneos para uma janela. Cada classe também define os serviços necessários para criar os menus, destacar um item selecionado e invocar a resposta correspondente. Cada classe é também responsável pela apresentação real das informações dentro de uma janela. Além disso, cada classe encapsula todas as suas considerações de diálogo físico.

8) PROJETAR ÍCONES, BOTÕES, MENUS, LISTAS E JANELAS - Levando em conta o GUI descrito nesta metodologia.

5.2.3 CORES E FONTES

Esta metodologia considera os seguintes aspectos importantes para definir as cores e fontes de uma interface gráfica:

1) O poder de comunicação visual - As pessoas organizam [MSF 93] o que vêem e como pensam sobre informações por agrupamento espacial. Na verdade, elas lêem um texto examinando minuciosamente as formas dos grupos de letras. Outro aspecto quanto a comunicação visual é que ele inclue sugestões (insinuações, palpites) emocionais que motivam, dirigem (conduzem, apontam, controlam, indicam) ou distraem.

2) Composição e organização - A pessoas tendem a ler uma tela da esquerda para a direita e do cima para baixo, conhecida como linha de processamento visual. Os olhos estão sempre atraímos por elementos coloridos sobre o preto e o branco, para isolar elementos sobre um grupo, e para isolar elementos gráficos sobre o texto. Todo visual é influenciado por outro visual próximo a ele. Um bom projeto depende completamente do texto e da interface gráfica.

É importante iniciar o projeto gráfico antes de desenvolver o sistema. O projeto gráfico pode ajudar a visualizar elementos do sistema e contribuir com idéias que ajudarão a motivar o usuário e organizar a informação de forma mais clara.

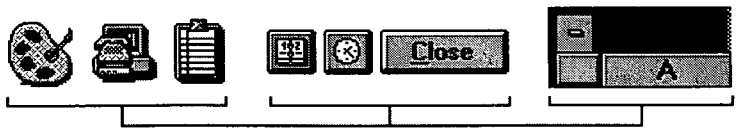
3) Princípios de interfaces (de impressionar e chamar a atenção) - alguns princípios visuais unificam a forma básica do projeto de interface visual. Eles envolvem o uso de propriedades visuais táticas, contrastes, cor e fontes.

4) O Poder da 3-D - a terceira dimensão (3-D) é usada para enfatizar uma função e para ter feedback de ações do mundo real, como mostra a figura 5.15. Quando o botão é pressionado, ele é "empurrado", dando ao usuário feedback de força. Elementos que aparecem para recuar são estáticos ou inativos.

 Acionando um botão tem-se a ilusão de que ele está pressionado.

Figura 5.15 Botões em 3-D

5) Contorno em preto para dar contraste em baixa resolução - Devido aos computadores com resolução VGA, com sua limitação em 16 cores, deve-se usar linhas pretas para definir margens (extremidades, bordas). As duas tonalidades de cinza disponíveis em VGA não possuem contrastes fazendo com que simples pontos de linhas em cinza sejam difíceis de focar e causam esforço nos olhos e dificuldade de leitura. Os ícones possuem contorno em preto para enfatizar a forma e para facilitar o visual, conforme a figura 5.16.



Ícones, botões e molduras de janelas, todos tem contorno em preto para definir suas bordas.

Figura 5.16 Contorno em preto

Quando existir apenas duas tonalidades de cinza, deve-se usar o preto e o branco para realçar as cores e a sombra deve ser escura para produzir o efeito 3-D. Para criar a terceira dimensão as cores necessárias são: branco, cinza claro, cinza escuro e preto.

6) Como as pessoas vêem a cor - Os olhos podem distinguir milhões de cores diferentes, mas a visão colorida pode ser fraca (debilitada). A seguir são apresentados conceitos e técnicas sobre a utilização de cores baseados em [MSF 93]:

- **TRICHROMATIC** (composto de três cores) - A retina dos olhos tem cones que percebem cores como vermelho, verde ou azul. Um dos princípio das três cores é que temos poucos cones azuis em nossos olhos e poucos deles ocorrem em fóvea (depressão da retina do olho), focando o centro dos olhos. Os olhos tem aproximadamente o mesmo número de cones vermelhos e verdes. A figura abaixo 5.17 de [MIN 94] mostra como o olho percebe a cor. O azul é focalizado na frente da retina dando a impressão de estar longe. O verde é focalizado sobre a retina e o vermelho atrás da retina dando a impressão de estar mais próximo. Por este motivo o azul funciona bem como uma cor de fundo e o vermelho é a primeira cor a ser notada.

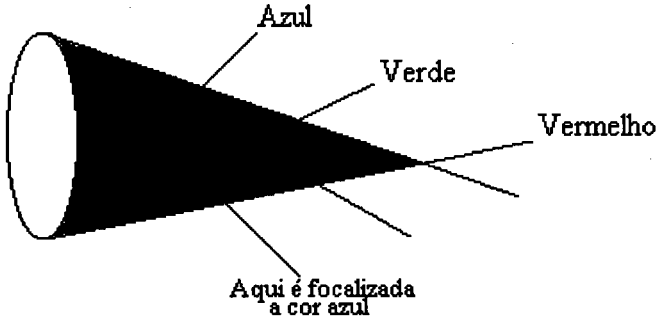


Figura 5.17 Onde as cores são focadas

- **PROPRIEDADES DAS CORES** - As cores são classificadas de acordo com três propriedades: coloração(nuança), saturação(pureza) e brilho(vivacidade, valor). Coloração é o nome da cor, saturação é a intensidade e brilho é onde se diminui a escala de escuro para claro, conforme mostra a figura 5.18.

Figura 5.18 Propriedades das cores

- **CORES OPOSTAS CRIAM ESPAÇOS** (superfícies) - Cores também possuem propriedades espaciais: tende-se a ver a borda de objetos com seus receptores (barras) em preto e branco e então colocá-los em margens com cores. As pessoas vêem cores com afinidade com outras cores ao redor delas. Uma moldura altera a ondulação quando se visualiza outra cor. As cores opostas como a vermelho/verde contribuem a tendência para ver cores em oposto (ao contrário) porque tem-se dificuldade de focalizar o vermelho e o verde ao mesmo tempo.

- **PROPRIEDADES DA COR** - As cores introduzem propriedades emocionais e/ou psicológicas. Por exemplo, o vermelho excita os olhos e o azul acalma. Algumas pessoas amam o rosa, outras podem odiá-lo. As pessoas gostam da cor. As cores são uma fonte de escolha, elas melhoram o marketing e dão a impressão de linhas de amizade. Se a escolha das cores for pobre, pode afetar severamente a usabilidade e criar uma aparência de circo. É importante lembrar que [MSF 93] cerca de 10% dos adultos masculinos tem alguma forma de confusão com a cor (como por exemplo o daltonismo), e uma pequena fração dessas, não vêem o brilho fiel do vermelho/verde. As pessoas idosas tem dificuldade de distinguir o azul e o banco.

- **USAR CORES CONFORME O TIPO DE MONITOR** - Existem 16 cores no sistema VGA, 20 cores no 8514 e 256 cores no Super VGA. Essas cores sempre permanecem iguais na tela, mesmo que elas sejam altamente coloridas. As 16 cores são mostradas na tabela 5.1, onde a numeração que aparece após o nome da cor corresponde a coloração, saturação e brilho.

Tabela 5.1 As cores do VGA

	Ciano 0-255-255 Ciano escuro 0-128-128 Verde 0-255-0 Verde escuro 0-128-0		Cinza 192-192-192 Cinza escuro 128-128-128 Branco 255-255-255 Preto 0-0-0
	Amarelo 255-255- 0 Amarelo escuro 128-128-0 Vermelho 255-0-0 Vermelho escuro 128-0-0		Magenta 255-0-255 Magenta escuro 128-0-128 Azul 0-0-255 Azul escuro 0-0-128

As quatro cores extra do 8514 e Super VGA são: Amarelo claro, verde claro, azul claro e cinza médio.

- **ESPECIFICAÇÃO DE RESOLUÇÃO** - Interfaces gráficas devem estar em escala correta para cada tipo de resolução de monitor, conforme a tabela 5.2.

Tabela 5.2 A resolução de acordo com o monitor

Tipo de Monitor	Número de Pontos	Número de cores
VGA	640 x 480	16 cores
Super VGA	640 x 480	256 cores
	720 x 540	16 e 256 cores
	800 x 600	16 e 256 cores
	1024 x 768	16 e 256 cores
8514	1024 x 768	256 cores
EGA	640 x 350	16 cores sem cinza claro
Monocromático	640 x 480	preto e branco

- **RECOMENDAÇÃO COM CORES** - As cores atraem os olhos, assim é bom usá-las para conduzir a atenção, entretanto muitas cores podem tornarem-se confusas. O azul é uma cor pobre para objetos ou textos pequenos, porque ela é difícil de focar. Mas o azul é uma boa opção para o segundo plano (a cor de fundo).

  - Usar cores para mostrar agrupamentos ou relacionamentos - por exemplo, tendemos a ver todas as coisas vermelhas na tela como parte de um conjunto. As pessoas são lentas para associar a cor com um significado, como por exemplo, se for vermelho o usuário está em modo de edição, se for verde o usuário está no modo de leitura.

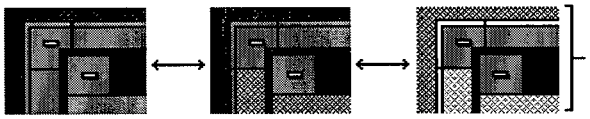
  - Cores opostas (tal como o vermelho e verde) não devem ser usados juntas porque eles apresentam vibração nos olhos na tentativa de focá-las. As cores brilhantes tendem virar o oposto às pós-imagens na retina. As pessoas devem ficar distante de grandes áreas com cores brilhantes.

  - Por causa da confusão com a cor (daltonismo), procurar usar cores com redundância de brilho - não confiar em cores sozinhas (exclusivas) para indicar uma propriedade ou função.

- **USAR CORES DISCRETAS E NÃO "CIRCOS" COLORIDOS** - As cores padrões devem ser deliberadamente discretas. Cores devem ser usadas somente para indicar atividade ou seleção, (se está ativo ou selecionado).

- Cores devem ser usadas em ícones com muito cuidado. Muitas cores distraem. As pessoas gostam da cor, e parecem entender imagens icônicas melhor quando elas são coloridas, semelhantes a partes do mundo real.

- A cor também é um aspecto pessoal. O Windows por exemplo, tenta dar ao usuário o controle sobre a escolha de cores, conforme a figura 5.19, permitindo que ele altere qualquer elemento colorido, ou escolha em um conjunto prédefinido de esquemas que são designados para serem usados, mas ajustar-se a uma variedades de gostos.



O usuário pode selecionar cores na interface com o Windows

Figura 5.19 Tipos de cores no Windows

- **CONCLUSÃO** - Uma interface deve ser específica para uma finalidade e dirigida a um único propósito. Procurar não utilizar cores opostas e brilhantes. Evitar que a interface tenha um aspecto desagradável, dando a impressão de um "circo" ou o colorido de um "arco-íris".

7) FONTES - As fontes tem outras funções além de disporem formas de letras. Esta metodologia propõe o uso dos seguintes critério:

- **A importância** - Semelhante a outros elementos visuais, criar fontes tem um modo (uma disposição), motivando pessoas e organizando as informações.

- **Fontes organizam a informação** - elas podem criar uma hierarquia de informações. Variando o tamanho e a textura (quantidade de pontos) de uma fonte, pode-se ver textos com maior ou menor importância e perceber a ordem na qual eles devem ser lidos.

- **A Legibilidade das fontes e dos tamanhos influência** - (dá valor ou ônus, demostrado na figura 5.20) - pela natureza (qualidade) da tela de um computador, fontes são normalmente menos legíveis em linha (on-line, na tela) do que impressas. Geralmente, fontes do tipo itálico e Serif são difíceis de serem lidas por causa do enta-

lhado que resulta de muitos (excessivos) pontos por caracter. Similarmente, fontes menores (um ponto de largura) são difíceis de ler por pessoas idosas.

Resolution	O entalhe nas margens e o tamanho deste exemplo de fontes causam maior esforço visual e não são recomendadas para fontes de telas em interfaces de computador.
Resolution Resolution	A lisura e claridade das fontes do exemplo são fáceis de olhar e permitem um rápido reconhecimento.

Figura 5.20 Tipos e tamanhos de fontes

5.2.4 SISTEMA VISUAL PADRÃO

O sistema visual padrão é composto por várias técnicas, as principais estão descritas nesta metodologia, como por exemplo janelas, botões, listas e menus.

BALLON HELP é uma técnica para exibir o nome da função quando o cursor passa sobre o ícone. É uma técnica que deve ser utilizada com cuidado. O usuário deve ter a condição de ativá-la e desativá-la.

WAVE FILES ou arquivos com extensão .WAV são sons emitidos quando se usa uma função, quando ocorre um erro ou associados a uma mensagem. Esta técnica pode ser utilizada dependendo do som associado. Por exemplo, o som de um cavalo relinchando quando se comete um erro é desaconselhável.

1) BOTÕES - Existem três grupos principais de botões 3-D: botões de texto, botões gráficos e botões de controle, exibidos na figura 5.21. Esses botões compartilham o olhar dimensional e percepção, possuem vários estados de aparência quando são pressionados.

Figura 5.21 Tipos de botões

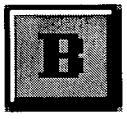
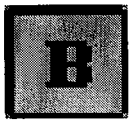
1.1) BOTÕES DE TEXTO - Os botões de texto são geralmente localizados em caixas de diálogos, figuras ou em barras de ferramentas. Existem três estados de aparência de botões de texto exibidos na tabela 5.3: subir (up), descer com mouse (mouse down) ou desabilitar (disable).

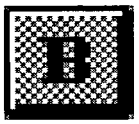
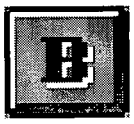
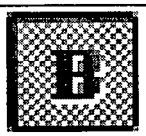
Tabela 5.3 Aparências de botões

	Subir (up)
	Descer com mouse (mouse down)
	Desabilitar (disable)

1.2) BOTÕES GRÁFICOS - São localizados em tiras (ribbons), réguas (rulers), barras de ferramentas (toolbars), caixas de ferramentas (toolboxes), superfície própria e controles de navegação.
Existem seis estados de aparência de botões gráficos, subir (up), pressionado com mouse (mouse down), indeterminado (indeterminate), desativado (disabled) e desativado para baixo (down disabled). Todos os botões possuem borda com uma linha em preto, conforme a tabela 5.4.

Tabela 5.4 Seis estados de um botão

Botão	Composição das cores
	O estado subir (up) usa: uma linha de um ponto em preto na borda e branco em cima e esquerda. Dois pontos em cinza escuro de sombra na base e a direita. Cinza claro na superfície do botão.
	O estado (mouse down) usa: sombra cinza em cima e esquerda, não possui realce. Usa um cinza claro em segundo plano. A mudança dos gráficos para baixo (pressionado) e ponto à direita. O estado acionado acontece quando é pressionado pelo mouse.
	O estado pressionado (down) do botão usa uma linha de pontos cinza escuro como sombra em cima e a esquerda. Não tem realce e usa o exemplo de tabuleiro com cinza claro e branco na superfície. O gráfico se desloca para baixo e para a direita um ponto. O estado visual ocorre com funções de botões Rádio/Checkbox pela ação do mouse.

	O estado indeterminado é igual ao estado acima (não acionado) com exceção de que o botão usa a superfície de tabuleiro com cinza claro e branco, e cor preto na imagem gráfica alterada para cinza escuro.
	O estado desabilitado (disabled) é parecido com o acima (não acionado) com exceção da imagem no gráfico no botão de superfície. O contorno branco é substituído por um ponto para a esquerda e para baixo contrastando com o cinza escuro do contorno.
	O botão desabilitado (down disabled) é similar ao pressionado com exceção da imagem gráfica na superfície. O contorno branco é substituído por um ponto para a esquerda e para baixo contrastando com o cinza escuro. A imagem parece embaçada.

Originalmente, botões gráficos podem incluir texto na imagem. O tamanho do botão aumenta podendo ser dependente do tamanho do texto. Gráficos e textos juntos devem ser usados quando o produto necessita de estética gráfica e quando o gráfico sozinho não é claro. Existem dois estilos recomendados. As figuras 5.22 e 5.23 exibem botões gráficos.

Layout vertical do tamanho do texto.
O gráfico fica acima do texto.

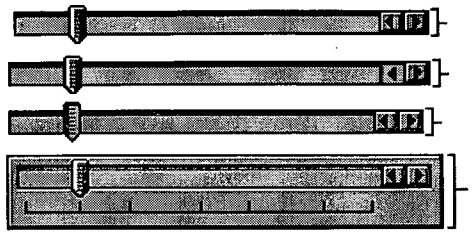
Figura 5.22 Botões gráfico com texto em baixo

O gráfico fica sempre a esquerda do texto.

Figura 5.23 Botões gráficos com texto ao lado

As figuras 5.24 a 5.28 mostram diversos tipos de botões de controle que podem ser utilizados em uma interface gráfica.

1.3) BOTÕES de DESLIZAR (slides) - São utilizados para indicar por exemplo o percentual de conclusão de uma tarefa. Ele também pode ser usado para mostrar o nível de volume de som.



nenhuma seleção
seleção à esquerda
punho (puxador) pressionado
deslizador com réguas abaixo

Figura 5.24 Botão de deslizar

1.4) BOTÕES DE ROTAÇÃO (spin) - São utilizados para aumentar ou diminuir valores de um item.

10	Botões normais
11	Pressionando para aumentar o valor 11, 12, 13...
09	Pressionando para diminuir o valor 09, 08, 07...

Figura 5.25 Botão de rotação

1.5) BOTÕES DE RÁDIO (radio button)

Define... ◉ Constant ○ Variable	Eles são utilizados para mostrar coisas mutuamente excludentes, em situações onde somente uma opção deve ser selecionada. Pelo menos dois botões devem aparecer.

Figura 5.26 Botão de rádio

1.6) BOTÕES DE MÚLTIPLA ESCOLHA (check box)

Estilo ☒ Negrito ☐ Itálico ☐ Tachado ☒ Oculto ☐ Caixa alta ☒ Maiúsculas	Os botões de selecionar (check box) são usados para um conjunto de coisas não mutuamente excludentes. Por exemplo, selecionar as características de uma determinada palavra ou caracter.

Figura 5.27 Botão de múltipla escolha.

1.7) PUSH BUTTON (pressionar) DE CONTROLE

OK Cancel Help	Este botão pode ser usado para mostrar uma ação que o usuário deve tomar. Ao pressionar o botão tem-se a impressão de que ele foi empurrado.

Figura 5.28 Botão push button

2) MENUS - Uma boa estrutura de menus torna a interface mais fácil de aprender e usar. Quando uma opção do menu não puder ser utilizada ela pode ficar desbotada (com uma cor fraca). Quando abrir um menu e nele aparecer um triângulo apontando para a direita tem-se um menu em cascata e se aparecer reticências (...) teremos uma caixa de diálogo. Os tipos de lista para menu são exibidos nas figura 5.29 e 5.30, incluem:

- Pull-down menu;
- Menu em Cascata;
- Pop-up menu (flutuante);
- Tear-off menu(destacável)

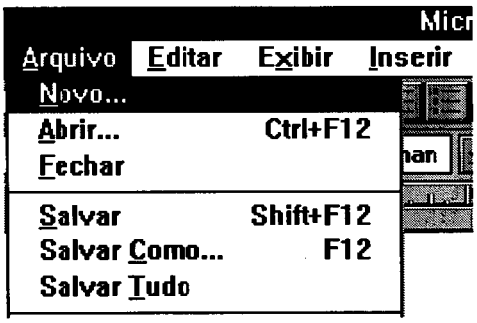
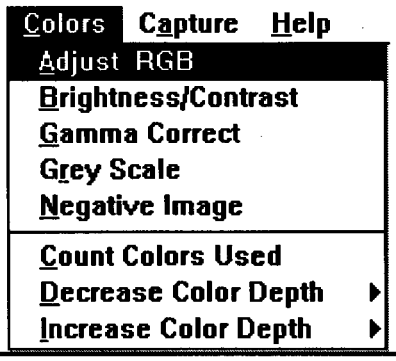
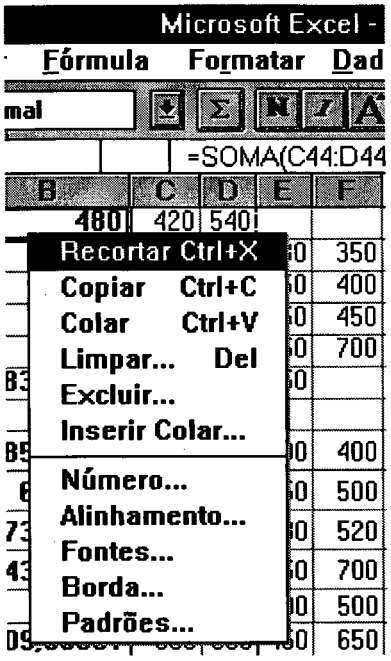
2.1) O menu de escolha Pull-down é utilizado para mostrar um conjunto de submenus.	2.2) Menu em cascata, apresenta um triângulo apontando para a direita. É utilizado quando existe outro nível de seleção.
	

Figura 5.29 Menu Pull-down e menu em cascata

2.3) Menu Pop-up (flutuante) este exemplo vem do Microsoft Excel. O menu "Recortar" é acionado com o botão direito do mouse.



2.4) Menu Tear-off (destacável) usado no Microsoft PowerPoint, onde o menu de AutoFormas pode ser colocado em vários lugares.

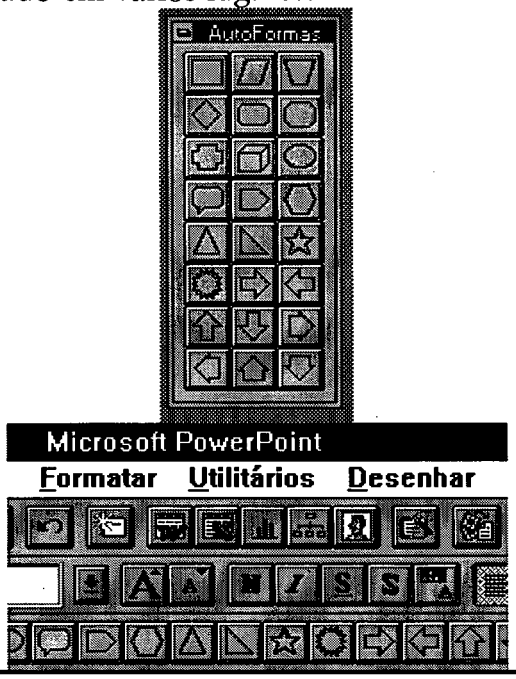
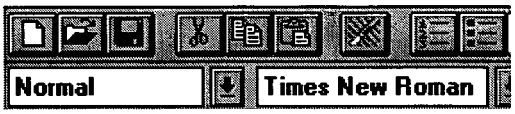


Figura 5.30 Menu flutuante e menu destacável

3) BARRA E CAIXA DE FERRAMENTAS - A barra de ferramentas (toolbars) e a caixa de ferramentas (toolboxes) são similares na sua função mas são diferentes na sua aparência exibida na figura 5.31. Elas são usadas para agrupar botões.

Barras de ferramentas são (faixas) horizontais localizadas na parte superior da janela que contém as ferramentas (tools). Caixas de ferramentas (de instrumentos) são barras verticais com seus títulos em miniatura e botões de menu de sistema. Elas podem estar flutuando ou vinculadas (fixadas) no lado esquerdo da janela. Ambas as variações são recomendadas, mas a barra de ferramentas horizontal é geralmente, a opção padrão (default).



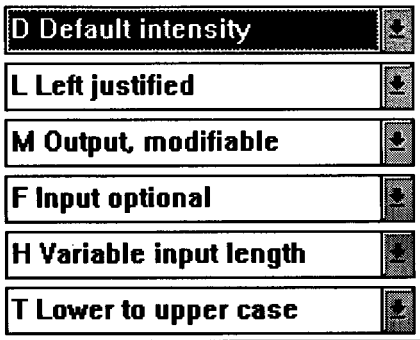
3.1) Barra de ferramentas (Toolbar)

Figura 5.31 Barra de ferramentas

4) LISTAS - São técnica para exibir informações. As figuras 5.32 a 5.34 mostram tipos de lista utilizadas nas interfaces.

4.1) LISTA DROP-DOWN CONTROL - São utilizadas para selecionar um item em uma janela.

Esta lista é chamada Drop-Down Control e pode trabalhar em conjunto com List-Box. A seta indica que existem várias opções e se for pressionada resulta no exemplo ao lado.



Através de uma List-Box pode-se selecionar um ou mais itens.

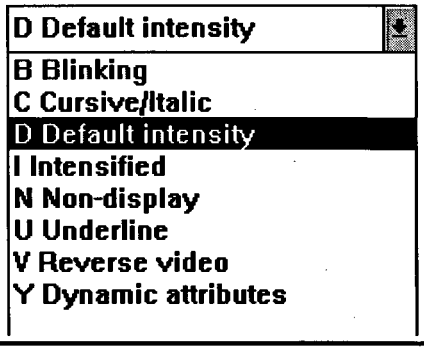
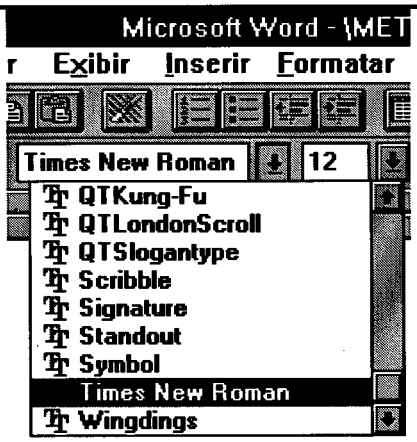


Figura 5.32 Lista drop-down e lista de controle

4.2) LISTA DROP-DOWN



É semelhante a drop-down control. Esta lista é usada para escolher uma opção entre várias. Uma delas já está setada como default. Utiliza a list-box com a barra de rolar, pois é uma técnica usada quando existem pelo menos oito opções.

Figura 5.33 Lista drop-down

5) CAIXA DE TEXTO

São controles de edição onde o usuário digita a informação desejada. Pode ser uma pesquisa em um texto, ou uma palavra a ser encontrada no dicionário. No exemplo abaixo à direita usando a caixa de texto em conjunto com drop-down a palavra desejada é "windows", o usuário não terminou de digitá-la e já havia uma palavra selecionada.

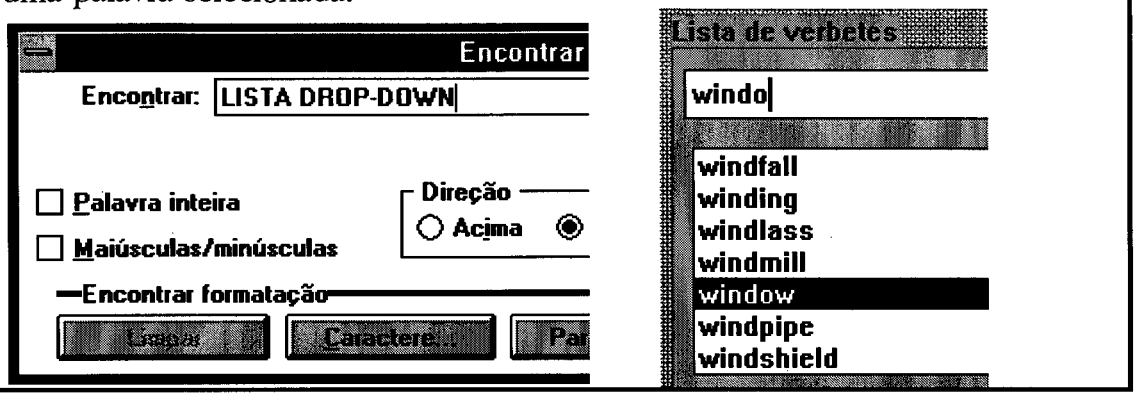


Figura 5.34 Caixa de texto

6) CURSORES - Os cursores também são conhecidos como apontadores. Essas imagens tem um potencial de feedback imediato para o usuário sobre a ação disponível na interface. Um apontador bem projetado puxa (dirige) os olhos do usuário naturalmente para o ponto alvo onde o usuário pode apontar de maneira fácil e correta. A flecha do cursor acrescenta maior feedback durante a movimentação. Localizando o objeto com a extremidade (com a ponta) da flecha. Exemplos na figura 5.35.

Copiar	Mover
Aumenta o tamanho da janela	Aumenta a janela na vertical
Posição onde está o cursor	Para onde está apontando o cursor
Os cursores são acionados com o dispositivo mouse que contém um ou mais botões que permitem ao usuário interagir com a interface. O botão do mouse é usado para efetuar uma dentre suas três funções: Selecionar, Manipular e Mostrar o menu Pop-up.	

Figura 5.35 Tipos de cursores

5.2.5 PROJETO DE ÍCONES

Os ícones são imagens que lembram [MSF 93] uma função no computador. Quando colocada no desktop, ele ajuda usuários navegar através da interface. Um sistema baseado em ícones é um eficiente meio de acessar informações porque usuários podem ver a tela inteira de uma vez, em lugar de procurar informações em menus hierárquicos.

USO DE FIGURAS - O maior sucesso dos ícones está baseado em figuras que são comunicações com conceitos abstratos através de associações concretas pelo uso de objetos do mundo real para representar idéias abstratas. Os usuários aprendem mais facilmente, pois associam ao ícone uma função como mostra a figura 5.36.

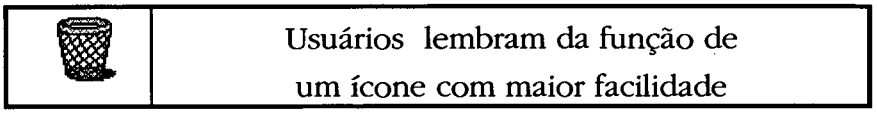


Figura 5.36 Tipo de ícone para apagar informações

COMO DESENHAR OS ÍCONES - Ícones que aparecem juntos devem ter um estilo similar. Estabelecendo uma aparência geral que minimiza as distrações e permite ao usuário focar o conteúdo. As seguintes regras ajudam a construir ícones:

- **IDÉIA** - Primeiro, deve-se ter entendido claramente as finalidades do ícone e identificar onde ele será exibido.

- **TAMANHOS** - Existem dois tamanhos de ícones: padrão com tamanho de 32 X 32 pontos e pequeno em 16 x 16 pontos.

- **DETALHES** - Ícones ilustrativos comunicam-se com conceitos de metáforas melhores do que símbolos abstratos. Isto porque o usuário efetua funções básicas sobre ícones baseados em objetos do mundo real.

- **INICIANDO COM PRETO E BRANCO** - Iniciar sempre os desenhos com preto e branco, como mostra a figura 5.37. Introduzir cores mais tarde, elas podem enfraquecer o projeto. Se um ícone fica bonito em preto e branco, ele ficará melhor colorido. Iniciar com um esboço de ícone com contorno em preto. O contorno auxilia o

aspecto da figura e ajuda a assegurar que a imagem será visível com uma variedade de fundos (segundo plano).

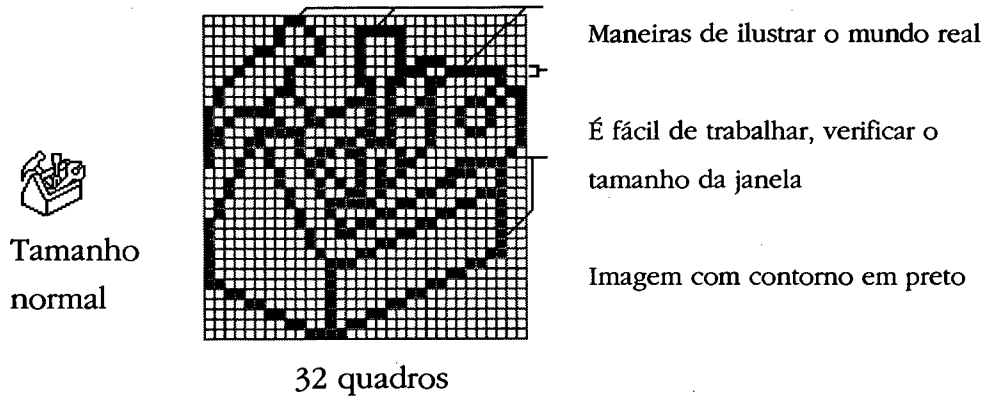


Figura 5.37 O desenho de um ícone, iniciando em preto e branco

- **OBJETOS EM 3-D** - Quando for possível, pintar ícones como objetos em terceira dimensão. Esta estratégia aumenta a identificação e melhora a comunicação. Uma maneira de utilizar a 3-D é pintar objetos em ângulos oblíquos.

- **CORES** - Após ter estabelecido a forma geral, cores podem ser aplicadas, conforme a figura 5.38. Cuidar para aplicar cores apropriadamente e com algumas restrições. Cores em ícones devem aumentar, não distrair do propósito do ícone.

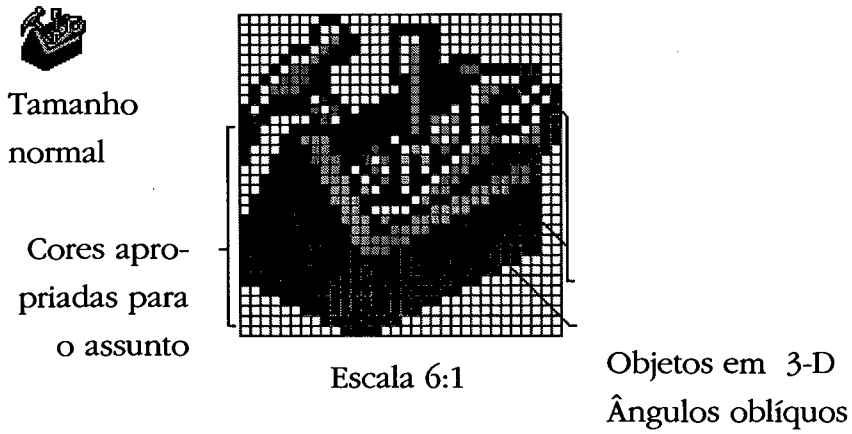


Figura 5.38 Cores em ícones

- **SENTIDO E ÂNGULO DA LUZ** - Para produzir a interface em 3-D, colocar uma fonte de luz vinda do canto superior esquerdo da tela. Os ícones podem também concordar

com este padrão. A orientação de ícones em 3-D é melhor à esquerda do que à direita como mostra a figura 5.39.

	Fonte e orientação de luz corretas
	Fonte e orientação de luz incorretas

Figura 5.39 Origem da luz

- **PROPORÇÕES** - Tentar usar proporções em ícones que sejam apropriados para visualizar a distância e de acordo com a resolução do vídeo.

- **ANTIALIAS ("anti-aliasing")** - É uma ilusão de ótica que consiste em reduzir os degraus (a borda irregular ou serrilha) de uma imagem. Esta técnica consiste na colocação de sombras de cor próximas a cor original para dar uma aparência limpa. Ao invés de usar o preto sobre o branco usar tonalidades que vão suavizando: branco, depois cinza-claro, cinza-escuro e finalmente o preto. Quando esta técnica é usada na margem externa de um ícone, ela deve ser feita com moderação (suave), podendo a imagem chocar-se com a cor de fundo. Procurar desenhar o ícone com várias cores de fundo. As figuras 5.40 e 5.41 mostram esta técnica.

	As bordas internas podem ser melhoradas com esta técnica
	Cuidado para não aplicar o antialias em demasia na borda externa

Figura 5.40 A utilização do antialias

Sem antialias	Com antialias	Resultado	
			Bom
			Ruim

Figura 5.41 Os resultados do antialias

5.2.6 MENSAGENS

As mensagens suprem o usuário com informações ou descrições de problemas. Elas podem estar associadas a botões de pressão (push buttons) que ajudarão o usuário a decidir como prosseguir. As mensagens [IBM 92] podem ser de informação, aviso ou de ação. Evitar o uso da palavra "erro", o usuário pode sentir-se incompetente. Utilizar símbolos padrões junto a mensagem como "stop", "?" e "!", exemplificados na figura 5.42.

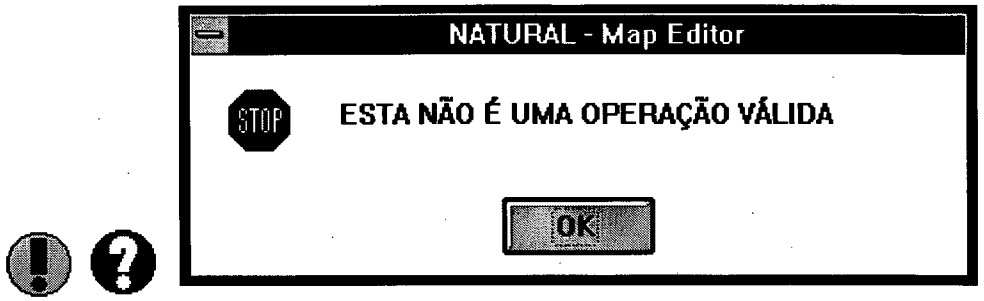


Figura 5.42 Exemplo de mensagem

5.2.7 MULTIMÍDIA

As informações aqui descritas não são de como projetar componentes de multimídia; são observações de uso geral para os projetistas.

CONSIDERAÇÕES PARA O PROJETO - Interfaces com usuário convencionais permitem ao usuário trabalhar com informações em texto ou com informações gráficas. A multimídia permite ao usuário trabalhar com tipos mais complexos, como audio e vídeo (som e imagem) permitindo interfaces com o usuário usando:

- Imagens, estáticas ou em movimento, gravadas ou sintetizadas;
- Som, gravado ou sintetizado.

A utilização da multimídia, permite:

- Acrescentar sugestões visíveis, como ícones animados;
- Adicionar sugestões audíveis, como conversas ou música;
- Animar objetos; como mapas, desenhos e gráficos;
- Imagens em vídeo, como produtos para demonstração.

Informações em multimídia podem enriquecer e aumentar a comunicação entre pessoas e entre usuários e seus computadores. Entretanto, o projeto de produtos que usam inform-

ações multimídia são mais desafiantes do que projetar produtos que usam apenas textos e gráficos.

MOSTRAR E MANIPULAR INFORMAÇÕES - Objetos multimídia possuem certas características que afetam a maneira em que o projetista deve mostrar o objeto de um produto. Os objetos multimídia são tipicamente uma coleção de dados apresentados em sucessão ao usuário. Para mostrar corretamente os dados em objetos multimídia eles devem ser apresentados ao usuário em uma seqüência correta e em tempo correto. Por exemplo, uma seqüência de animação é apresentada para o usuário como uma série de imagens como se fosse um simples quadro. Na seqüência correta, cada quadro deve aparecer em uma ordem relativa, e o intervalo de tempo entre cada quadro deve ser constante.

Objetos multimídia podem combinar um objeto com som. Para realizar o efeito desejado, o projetista deve sincronizar os objetos. Isto é, ligar segmentos de dados entre cada objeto, fazendo com que os segmentos sejam mostrados no mesmo ponto em determinado instante no tempo.

TRATANDO DIFERENTES CONFIGURAÇÕES - A interface multimídia tem considerável demanda em computadores. Entretanto existem fatores que afetam a habilidade do usuário para gerenciar e manipular objetos tais como:

- Tamanho, cor e resolução do monitor de vídeo.
- Adaptar placas e periféricos, como vídeo, audio e gravadores.
- Memória para o sistema e capacidade de armazenamento.

O projetista deve estar atento de que alguns usuários podem não ter todos os componentes de hardware necessários para utilizar os objetos multimídia.

5.3 DISTRIBUIÇÃO DE PROCESSOS

A distribuição de processos é uma etapa da metodologia proposta que fornece subsídios sobre a distribuição de carga de processamento entre clientes e servidores. Para efetuar a distribuição de processos entre clientes e servidores é importante manter um padrão no ambiente. Segundo [CER 91] os padrões necessários num ambiente distribuído são: Hardware, Lógica de sistemas, Programas de aplicação, Definição de dados, Protocolos de comunicação, Documentação, Segurança, Preços e distribuição de custos, Administração de dados e Administração de Banco de Dados.

Esta etapa da metodologia estabelece as regras para o projeto levando em conta onde deve ocorrer o processamento. Para executar esta etapa a metodologia define:

- Tipos de processos;
- Equilibrar o processamento;
- Distribuir a carga de processamento.

5.3.1 TIPOS DE PROCESSOS

Os processos executados no cliente ou no servidor podem ser definidos em cinco categorias, exemplificados na figura 5.43 e descritos a seguir.

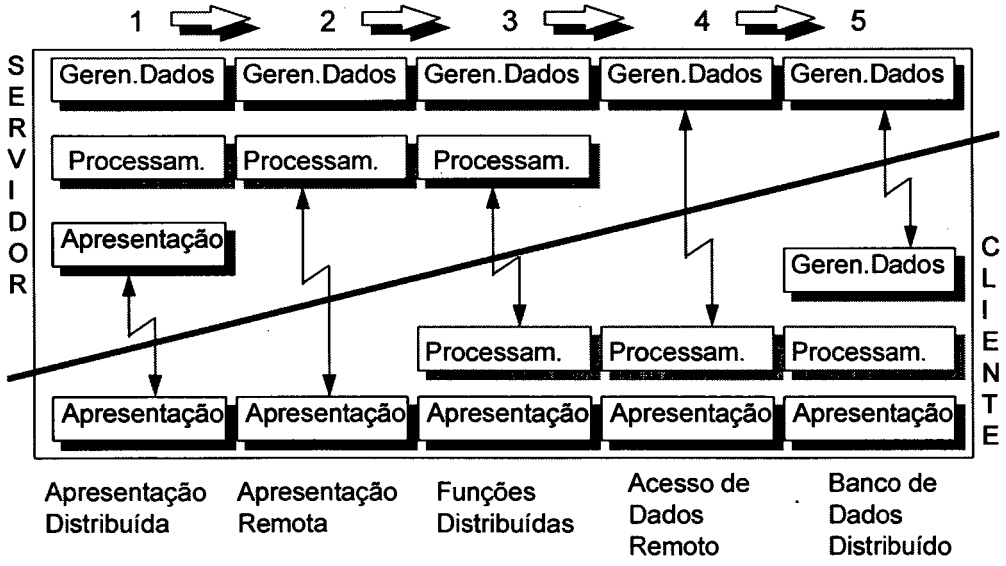


Figura 5.43 Formas de distribuição de processos

1. Apresentação Distribuída - corresponde a divisão das tarefas entre o cliente e o servidor da seguinte maneira: a apresentação da interface pode rodar tanto no cliente como no servidor, as demais tarefas como a gerência de banco de dados e o processamento da aplicação rodam no servidor.

2. Apresentação Remota - corresponde a divisão das tarefas entre o cliente e o servidor da seguinte maneira: a apresentação da interface roda no cliente, as demais tarefas como a gerência de banco de dados e o processamento da aplicação rodam no servidor.

3. Funções Distribuídas - corresponde a divisão das tarefas entre o cliente e o servidor da seguinte maneira: a apresentação da interface rodar no cliente, O processamento da aplicação pode rodar tanto no cliente quanto no servidor. A gerência de banco de dados roda no servidor.

4. Acesso de Dados Remoto - corresponde a divisão das tarefas entre o cliente e o servidor da seguinte maneira: a apresentação da Interface e o processamento da aplicação rodam no cliente, A gerência de Banco de dados roda no servidor.

5. Banco de dados Distribuído - corresponde a divisão das tarefas entre o cliente e o servidor da seguinte maneira: a apresentação da interface e o processamento da aplicação rodam no cliente, a gerência de banco de dados pode rodar tanto no cliente como no servidor.

5.3.2 EQUILIBRAR O PROCESSAMENTO

O equilíbrio de processamento é feito através da quantidade de trabalho realizado em cada cliente ou servidor. A figura 5.44 extraída de Renaud [REN 94] ilustra uma divisão de trabalho entre o cliente e o servidor, levando em conta a atividade realizada. A figura mostra que o Processamento Interativo deve ser realizado no cliente, o Processamento de

Banco de Dados deve ser realizado no servidor. O Processamento da Aplicação pode ser realizado no cliente ou no servidor, dependendo do contexto.

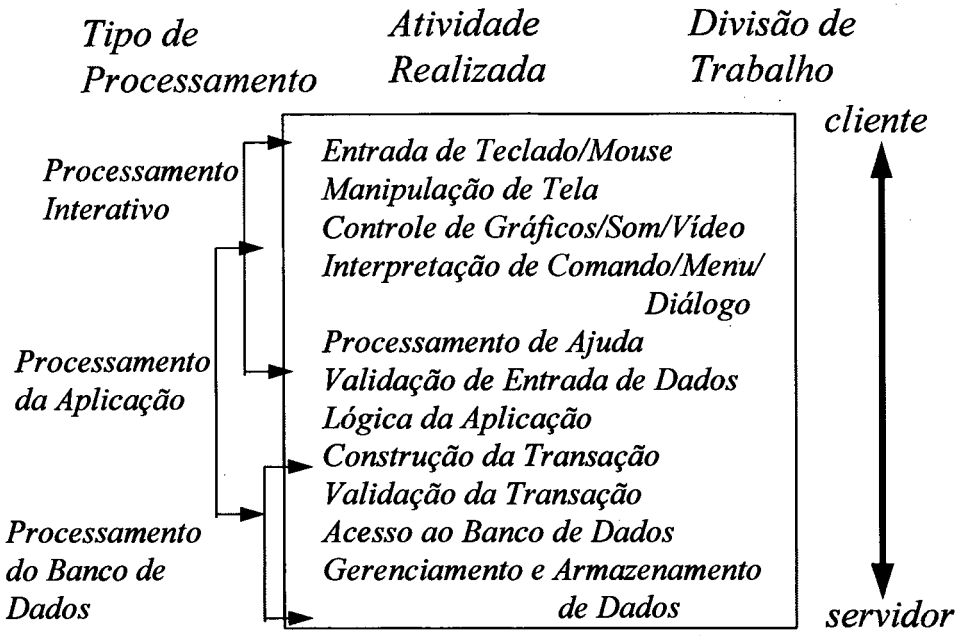


Figura 5.44 Equilíbrio de Processamento [REN 94]

Para definir onde deve ocorrer o processamento esta metodologia estabelece um conjunto de regras práticas. A decisão porém, só pode acontecer de acordo com uma análise sobre o contexto da aplicação, a divisão proposta por Renaud na figura acima (5.53) é apenas uma sugestão. Para ter uma visão geral do ambiente de rede e C/S esta metodologia coloca algumas regras que o analista deve levar em conta para elaborar o projeto. Elas baseiam-se nas seguintes questões:

- Onde deve ser executadas a lógica da aplicação, a manipulação de erros, a preparação da transação e a validação da transação; no cliente ou no servidor?

- Em algumas situações a lógica da aplicação exige muita manipulação de dados, seria apropriado transferir esses dados pela rede?

- O analista deve fazer uma análise sobre o custo beneficio da execução da manipulação de erros. Decidir se o processamento deve ficar no cliente ou no servidor?

- O processamento da transação pode ser mais eficiente num servidor do que no cliente?

As regras são:

1. Colocar o máximo de processamento no cliente

O processamento do cliente é dedicado a um único usuário, enquanto o processamento do servidor é compartilhado entre vários usuários. Isso pode significar que em alguns casos o processamento no cliente é mais rápido. Mesmo que o servidor tenha uma CPU mais rápida do que a dos clientes, qualquer trabalho realizado no servidor diminui sua produção (throughput).

2. Realizar atividades de cálculo no cliente

As arquiteturas de computadores estão cada vez mais especializadas permitindo que muitos cálculos sejam realizados no cliente. Estas atividades podem incluir:

- Classificação e pesquisa de tabelas na memória;
- Aritmética de ponto flutuante e fixo;
- Compressão e descompressão de dados;
- Inferência e outras manipulações de regras;
- Criptografia e decodificação de dados;
- Captura ou execução de vídeo;
- Compilação de código fonte;
- Processamento de voz;
- Análise estatística e
- Processamento de textos.

3. Separar o processamento por usuário e multiusuário

Se qualquer parte da lógica de uma aplicação for baseada em um único usuário, implementar essa lógica no cliente. Por exemplo, validar uma tela de entrada de dados no contexto de um único usuário, pois as edições são baseadas nos dados inseridos por esse usuário. O processamento de ajuda é outro exemplo baseado em um único usuário. Já a validação de uma transação pode exigir várias consultas ao banco de dados para assegurar que certas regras não sejam violadas. Se a quantidade de trabalho realizado como parte dessas consultas for menor no servidor do que o trabalho realizado para validar a transação no cliente então, a validação deverá ocorrer no servidor.

4. Gerenciar recursos compartilhados no servidor

Dados e periféricos compartilhados entre vários usuários devem ser gerenciados por um servidor, para que eles sejam acessíveis por todos os usuários. Algumas poucas exceções podem ser arquivos mantidos para uso individual, mas assim mesmo estes poderiam ser mantidos num servidor para facilitar as tarefas de backups. Os recursos mantidos tanto no cliente como no servidor devem ser gerenciados conforme mostra a figura 5.45.

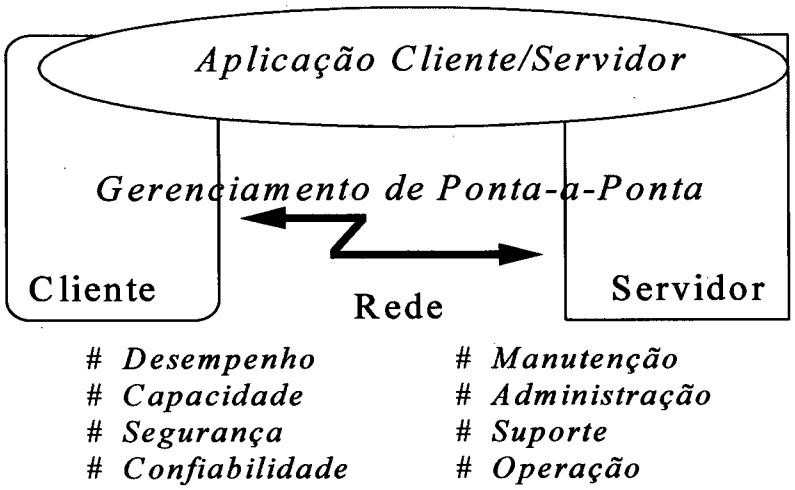


Figura 5.45 Gerenciamento de Sistemas Ponta-a-Ponta

5. Evitar a centralização de serviços

Gerenciar recursos compartilhados usando o processamento no servidor não significa usar um único servidor. Dados regionais são mais acessados pela região onde eles foram gerados e, portanto, o banco de dados pode ser particionado por regiões, instalando-se servidores em locais estratégicos. Os sistemas distribuídos normalmente oferecem maior throughput e maior disponibilidade do que os sistemas centralizados.

6. Assegurar O gerenciamento dos dados localmente

Quando for utilizado o particionamento de dados, deve-se medir o número de interações para cada objeto. A localidade deve ser explorada ao máximo, entretanto a maioria dos sistemas não possui localidade de referência total, um ponto de equilíbrio deve ser encontrado. Onde o dado for colocado deve ter gerenciamento.

7. Planejar transferências de dados

Cuidar para que os servidores são fiquem sobrecarregados com transferências de dados desnecessárias ou feitas em horários inconvenientes. Evitar que os clientes decidam unilateralmente quando iniciar a transferência.

8. Reduzir os dados transferidos entre clientes e servidores

Quanto maior for a quantidade de dados transferidos, maior a oportunidade de problemas e recuperação de erros. Além disso o canal de comunicação é um recurso compartilhado. Qualquer aumento de tráfego pode reduzir a produção (throughput) da rede.

9. Comprimir grandes transferências de dados

Se a transmissão for grande, considerar sua compressão para reduzir o impacto da largura de banda da rede e agilizar o envio.

10. Criar checkpoints em grandes transferências de dados

As grandes transferências devem ter checkpoints (pontos de verificação) para que elas sejam reinicializadas quando houver problemas, a partir do ponto onde ocorreu o erro.

11. Projetar o sistema pensando nas interrupções

Alguma parte de uma grande rede pode estar fora do ar. Por isso, projetar a aplicação para trabalhar com estas interrupções. Projetar aplicações envolvendo um número mínimo de nós da rede. Evitar transações que atualizam várias bases de dados em nós diferentes.

12. Centralizar a administração e distribuir recursos

Grande parte do custo/benefício dos sistemas cliente/servidor pode ser afetado com custos operacionais altos, a menos que o sistema seja projetado visando a facilidade de

administração e operação. Isso significa oferecer um ponto central do qual poderá administrar os recursos distribuídos, mostrado na figura 5.46 de [DAT 91].

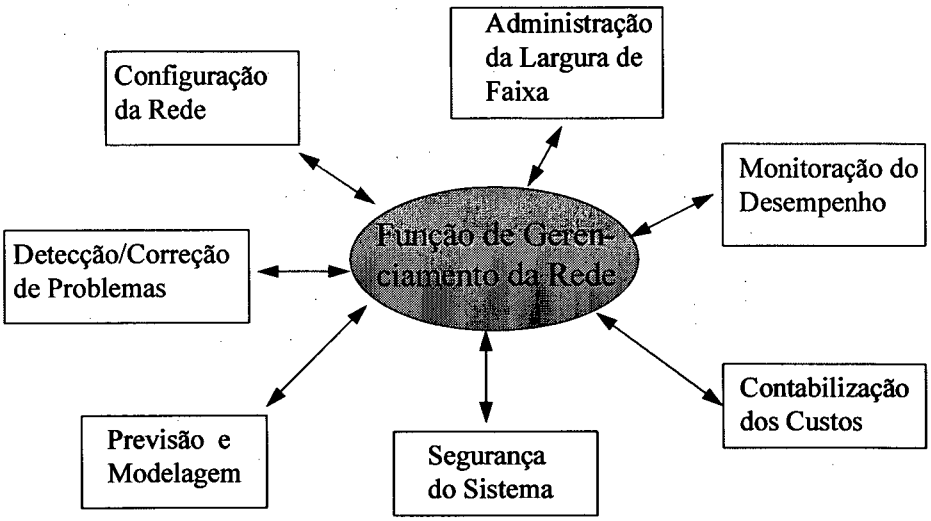


Figura 5.46 Funções de Gerenciamento de Rede

13. Construir perímetros de segurança para sistemas e aplicações

A segurança deve ser planejada com antecedência, usando os perímetros de segurança necessários para a aplicação, sistemas, dados, rede e acesso físico à instalação, conforme mostra a figura 5.47. A segurança deve levar em conta os clientes os servidores e a rede.

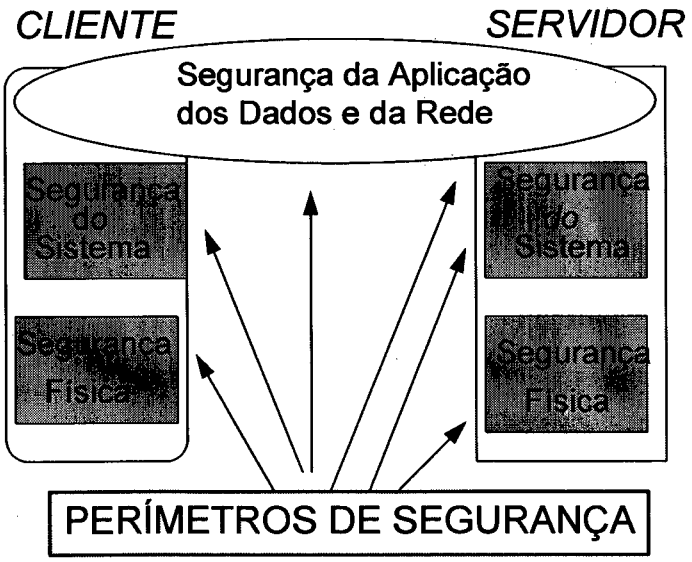


Figura 5.47 Perímetros de Segurança

14. Analisar a confiabilidade da arquitetura

Analisar as vulnerabilidades na arquitetura da rede. Tentar explorar as redundâncias que ocorrem naturalmente na arquitetura em camadas. Rotas alternativas da topologia da rede podem ser utilizadas quando houver falhas. Os sistemas podem ser projetados visando as falhas na rede, escolhendo rotas alternativas.

15. Medir a demanda independentemente da capacidade oferecida

Normalmente há milhares de combinações possíveis durante a decisão sobre aplicações em uma rede. É importante uma visão clara dos vetores de carga da aplicação, independentemente da capacidade em potencial das diferentes máquinas.

16. Utilizar chamadas de procedimentos remotos (RPC)

O RPC é um mecanismo para a implementação de processamento distribuído. Uma chamada de "procedure" remota é semelhante a chamar uma procedure local (ou seja, uma sub-rotina). O RPC é fácil de usar e pode reduzir a quantidade de código de programação. O funcionamento da RPC está ilustrado na figura 5.48.

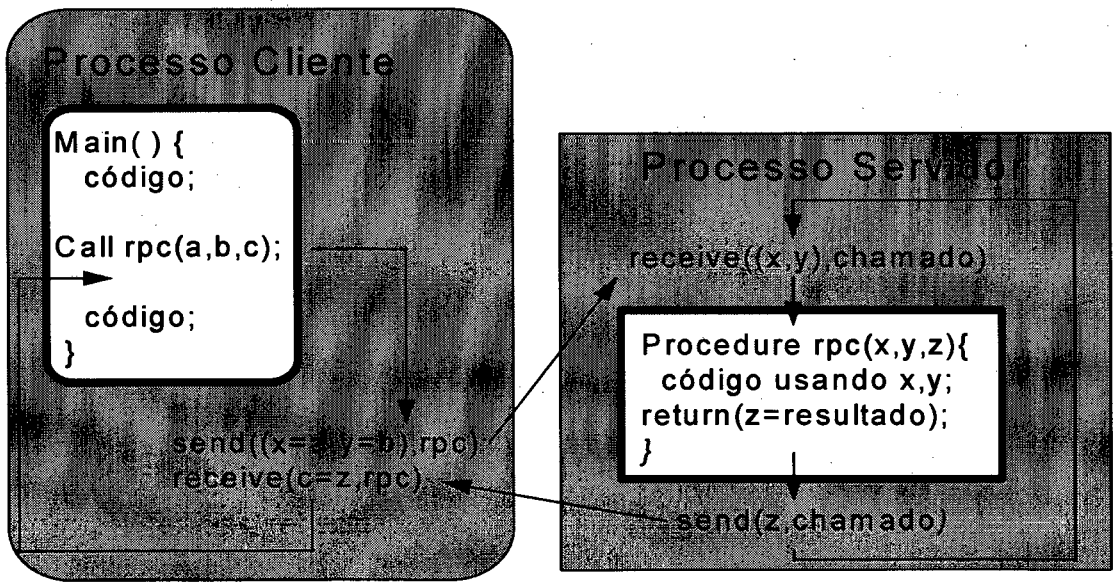


Figura 5.48 A Chamada de Procedimento Remoto

17. Utilizar transações adequadas

As funções de gerência de transação é implementar o conceito de transação, gerenciar as atividades e recursos do sistema e monitorar o início e término das atividades do sistema.

A transação é uma maneira [GRA 95] de assegurar a integridade do banco de dados. A aplicação deve determinar quando a transação inicia e termina. Se o término for anormal uma transação deve ser desfeita. O tamanho de uma transação pode determinar a eficiência do gerenciador de banco de dados. A transação tem as seguintes característica (ACID):

ATOMICIDADE - É uma exigência para que a unidade de trabalho seja indivisível. O sistema deve garantir que, ou todos os comandos sejam executados, ou o banco não reflete o resultado da execução de nenhum deles. O gerente de transações transforma-a em uma unidade atômica de trabalho, implementando os comandos de iniciar, migrar, terminar, cancelar, e reiniciar transações.

CONSISTÊNCIA - É uma exigência para que uma aplicação mova o banco de dados de um estado inconsistente para um estado consistente. Quando uma transação é comprometida, a aplicação deve garantir que todas as alterações nos dados sejam desfeitas.

ISOLAÇÃO - É a exigência de que uma transação não deve ser afetada por outra. Uma transação é executada isoladamente de outras transações.

DURABILIDADE - É uma exigência de que, uma vez que uma transação está comprometida, seu efeito no banco de dados é durável ou permanente. Nenhuma ação subseqüente ou falha do banco de dados pode fazer com que as mudanças afetadas pelas transações comprometidas sejam perdidas.

Em um sistema de gerenciamento de banco de dados distribuído, a implementação destes comandos é mais complexo, pois uma transação pode modificar dados armazenados em mais de um nó, o que significa que vários nós têm que cooperar no processo. Esta cooperação entre nós e o tamanho de cada transação podem comprometer a performance da rede.

5.3.3 DISTRIBUIR A CARGA DE PROCESSAMENTO

Em tempo de desenvolvimento é importante distinguir um servidor lógico e o computador em que ele rodará. O projeto cliente/servidor [REN 94] pode fazer uso de uma perspectiva virtual, durante os primeiros estágios do projeto pode-se pensar em servidores lógicos, e não físicos, oferecendo serviços lógicos dentro de um computador para poder simular a sua carga de trabalho.

Esta fase da metodologia procura efetuar um equilíbrio no processamento sem levar em conta se o processamento deve acontecer no cliente ou no servidor mas levando em conta a carga de cada servidor. A figura 5.49 mostra o ciclo que um projeto pode ter para equilibrar o processamento.

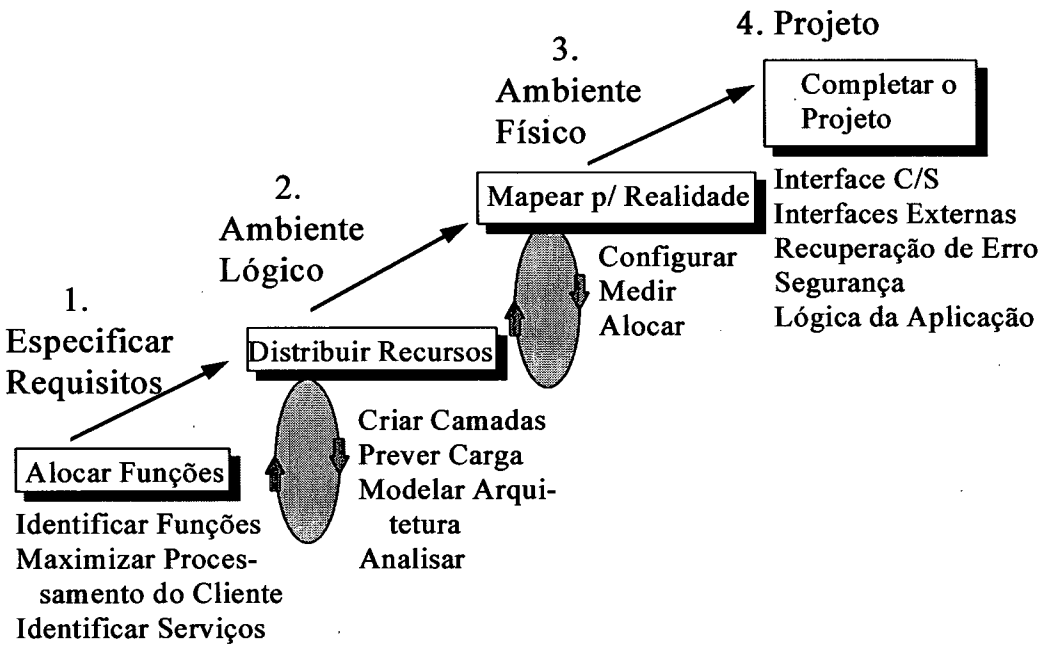


Figura 5.49 O Ciclo da análise de carga

As etapas propostas neste ciclo possuem as seguintes características:

1. Especificar os requisitos funcionais entre clientes e servidores

- Identificar as funções de equilíbrio de processamento.
- Práticas de mono usuário x multiusuário, alocando o máximo de processamento no cliente.
- Identificar servidores em potencial tratando cada função compartilhada ou recurso como um servidor virtual.

2. Distribuir recursos entre servidores virtuais

- Distribuir dados e outros recursos entre servidores virtuais, criando camadas de processamento.
- Prever a carga da aplicação resultante da distribuição de recursos.
- Modelar as implicações de interação, confiabilidade e desempenho dessa distribuição.
- Analisar o resultado da modelagem e repetir o ciclo.

3. Mapear os servidores virtuais em configurações físicas

- Selecionar a tecnologia e os produtos a serem configurados.
- Integrar estes produtos para assegurar a interoperabilidade.
- Mapear a demanda de capacidade na configuração real e repetir o ciclo até obter a melhor alocação.

4. Completar o projeto fazendo o seguinte:

Completar o projeto de distribuição de processos verificando as seguintes tarefas:

- Terminar a interface entre cliente e servidor;
- Detalhar interfaces com outros sistemas;
- Completar o projeto da aplicação.

5.4. DISTRIBUIÇÃO DE DADOS

Um dos maiores desafios quando se distribui os dados [GIL 95] em um ambiente de rede, está no gerenciamento dos conjuntos e subconjuntos de dados (cópias, réplicas, extratos), pois além dos fatores de segurança e integridade deve-se observar a performance e o volume de dados transmitidos. Neste contexto a localização dos arquivos é essencial.

A decisão de distribuir ou não os dados para um determinado sistema pode ser tomada através de argumentos técnicos que visem melhorar a performance. Contudo, pode-se decidir em manter os dados centralizados devido a questões como segurança, administração e backups.

A distribuição dos recursos ao longo da rede e em particular a localização dos dados não é simples. Esta etapa apresenta um estudo sobre algumas das técnicas mais utilizadas

para se obter uma distribuição dos dados que atenda aos requisitos de performance e volume de dados transmitidos.

5.4.1 MOVIMENTAÇÃO DE DADOS

Um dos objetivos da distribuição de dados é reduzir seu movimento pela rede, colocando os dados em locais fisicamente mais próximos dos usuários. Uma técnica para verificar o movimento é a Análise de Afinidade(AF) que pode ajudar a identificar quais dados exibem interações localizadas. A afinidade de uma entidade é a freqüência (F) de qualquer interação com essa entidade.

Em [REN 94] é apresentada uma fórmula para definir um fator de afinidade (FA). Nela são considerados:

A freqüência de acesso entre um cliente x (Cx) e um servidor y (Sy), representada por F(Cx,Sy);
O número total de acessos do cliente x, considerando todos os servidores (T(Cx));
O número total de acessos efetuados em um servidor y considerando todos os clientes (T(Sy));

O fator de afinidade é dado por FA(Cx,Sy) = F(Cx,Sy) / T(Sy) + T(Cx). Observando os dados constantes na tabela 5.5, obtém-se um fator de afinidade entre o cliente 1 e o servidor 3 igual a:

FA (C1, S3) = F(C1,S3) / T(S3) + T(C1).
= 100 / 850 + 330
= 0,084

Desta tabela, pode-se concluir que existe uma maior afinidade entre o cliente 1 e o servidor 2, pois FA (C1,S2) é o mais alto. Isso indica que cliente 1 e servidor 2 devem estar localizados o mais próximo possível. O mesmo acontece com o cliente 3 e servidor 3.

Tabela 5.5 Freqüência de acesso e fatores de afinidade

FREQÜÊNCIA DE ACESSO CLIENTE/SERVIDOR (F)					FATORES DE AFINIDADE (FA)		
(x) C(y)	**Servidor 1**	**Servidor 2**	**Servidor 3**	**T(c(x))**	**Servidor 1**	**Servidor 2**	**Servidor 3**
Cliente 1	30	200	100	330	0,037	**0,233**	0,084
Cliente 2	50	100	175	325	0,064	0,118	0,149
Cliente 3	150	100	300	550	0,149	0,093	**0,214**
Cliente 4	100	75	150	325	0,127	0,088	0,128
Cliente 5	130	50	125	305	0,170	0,060	0,108
T(s(y))	460	525	850				

Além da análise de afinidade entre clientes e servidores, o canal de comunicação deve ser utilizado da forma mais eficiente possível, evitando-se a transmissão de dados desnecessários e limitando-se a transferência de dados.

A realização de processamento no servidor reduz a quantidade de dados transferidos. Uma técnica para reduzir a quantidade de dados que trafega na rede é comprimir os dados antes de enviá-los. Outro item diz respeito a reinício de transmissões por motivos de falhas. Devem existir mecanismos que permitam a reinicialização a partir do ponto onde a falha ocorreu.

Outro aspecto a ser considerado na movimentação dos dados diz respeito à eficiência da rede. É uma medida de quantos dados do usuário são transferidos em proporção ao overhead enviado na rede. Uma medida aproximada da eficiência da rede, ignorando o efeito de erros na rede, pode ser calculada por: E = M / (M + O) [REN 94]. Onde:

M = tamanho da mensagem em bytes.
O = overhead necessário para enviar a mensagem: bytes de header + tamanho do ACK + (número de pacotes enviados x atraso da rede x velocidade da rede) .

A aplicação deve prever a melhor utilização dos blocos de dados a serem transmitidos, calculando o tamanho do bloco ideal. Maiores detalhes sobre eficiência da rede podem ser encontrados em [TAN 94].

5.4.2 FORMAS DE DISTRIBUIÇÃO DE DADOS

1) DADOS CENTRALIZADOS

Em geral, faz sentido centralizar os dados [GIL 95] se grandes quantidades forem freqüentemente atualizadas ou se todos os usuários acessam todos os dados por igual e precisam ver os valores de dados mais atualizados. Um exemplo clássico é um sistema de reserva de passagens aéreas. Outro motivo para centralizar os dados é se forem constantemente manipulados como um todo (por exemplo, pesquisados, classificados, ou resumidos). Os dados são mais fáceis de gerenciar se forem centralizados.

CENTRALIZADOS - C - O banco de dados está localizado num ponto central sem distribuição. Esta normalmente é a melhor técnica, porque é a mais simples e evita cópias de dados. O controle, definição e manutenção são fáceis de implementar. A base principal também é chamada de primária. A figura 5.50 mostra um exemplo de dados centralizados.

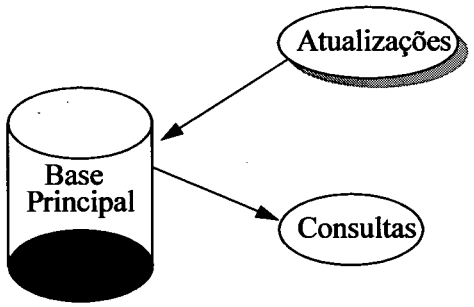


Figura 5.50 Arquivos centralizados

2) DADOS DISTRIBUÍDOS

Na definição de Martin, [MAR 89] os dados podem ser distribuídos de maneiras diferentes, conforme mostra a figura 5.51.

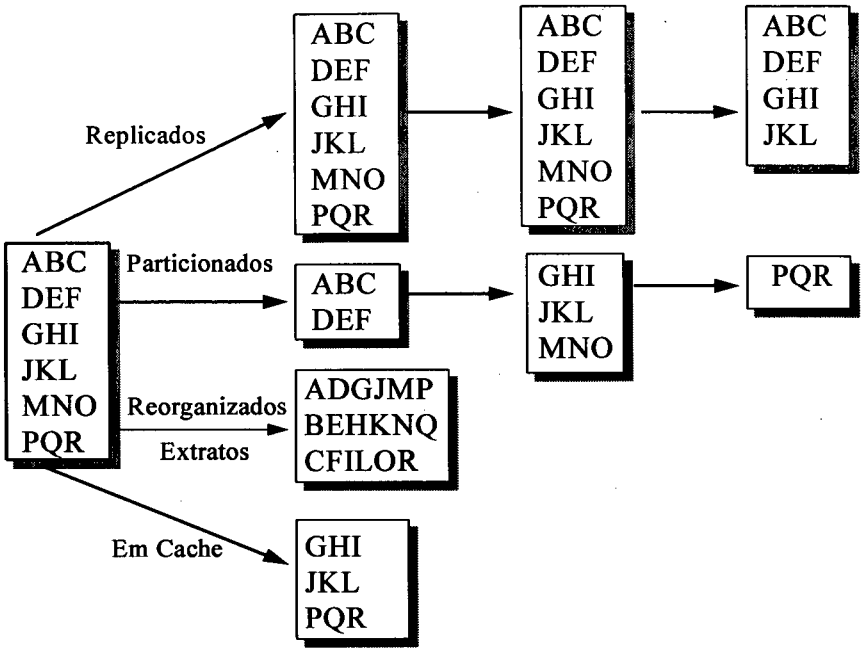


Figura 5.51 Tipo de Dados Distribuídos

Distribuir dados explora o fato de que os acessos aos dados costumam refletir uma localidade de referência. Um nó da rede, em particular, provavelmente acessará alguns dados mais freqüentemente do que outros. Por exemplo, os dados regionais normalmente serão acessados com mais freqüência pelos clientes dentro da região do que pelos clientes em outras regiões. Se os dados puderem ser distribuídos entre os servidores, de modo que a maior parte destes acessos agrupados seja satisfeita localmente, uma hierarquia de processamento poderá ser criada. A figura 5.52 apresenta um possível cenário de hierarquia de processamento onde A, B e C são servidores estrategicamente situados.

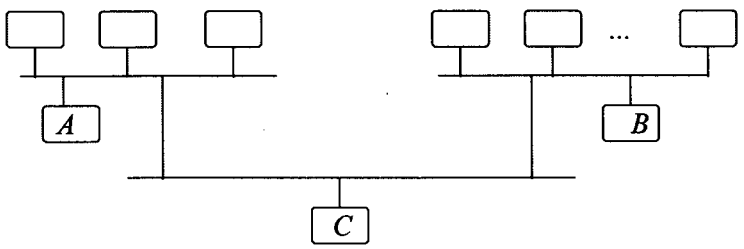


Figura 5.52 Sistemas cliente/servidor de duas camadas

3) DADOS REPLICADOS

Num esquema replicado, os servidores A e B (figura 5.52) contém os mesmos dados do servidor C. O motivo mais comum para replicar dados é distribuir o overhead do processamento da consulta entre as várias máquinas. Muitas aplicações realizam muito mais consultas do que atualizações. Replicar os dados em outros servidores permite que essas consultas sejam feitas por muitos servidores. Entretanto, qualquer atualização que ocorra em um workgroup deve ser propagada a todos os outros workgroups.

A desvantagem é a maior complexidade no tratamento de atualizações. Se as atualizações puderem ser acessadas por qualquer servidor, o bloqueio de dados e o sincronismo do servidor poderão gerar problemas.

RÉPLICAS - As réplicas são cópias de um banco de dados principal que podem receber consultas e atualizações, neste caso, os dados do banco de dados principal tem que ser reatualizados à partir das réplicas distribuídas, exemplificado na figura 5.53. Existem três tipos de réplicas segundo [HAC 93]: periódicas, contínua e finalização. A tabela 5.6 mostra as vantagens e desvantagens das réplicas.

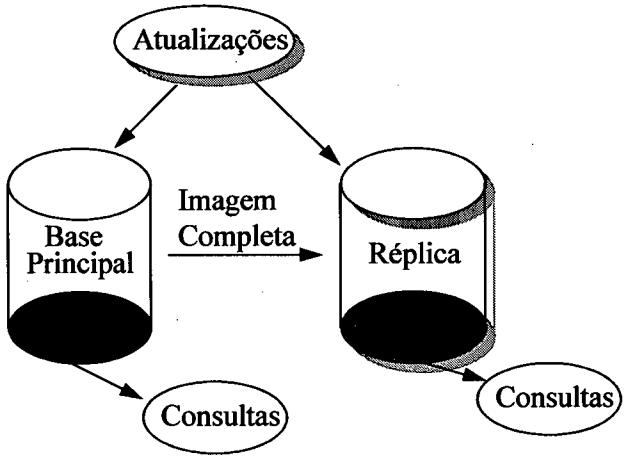


Figura 5.53 Arquivos distribuídos em forma de réplica

Tabela 5.6 Vantagens e desvantagens das réplicas

VANTAGENS	DESVANTAGENS
Minimiza as interrupções devido a queda do sistema (rede, luz)	Necessidade de recursos extras (disco)
Melhora a performance	Sincronização dos dados
Podem ter "views" diferentes	Overhead no processamento
Consistência	
Tráfego na rede	

RÉPLICA PERIÓDICA - RP - É um cópia que pode ser atualizada e que retorna as modificações para o banco principal com uma certa freqüência. Esta técnica muda o local de atualização, passando da base principal para a réplica. A base principal permitirá apenas consulta. O aspecto crítico é que as alterações na réplica devem ser enviadas a base principal para efetivar a resincronização. Intervalos (períodos) pré-definidos vão alimentar a base principal, esta técnica está demonstrada na figura 5.54.

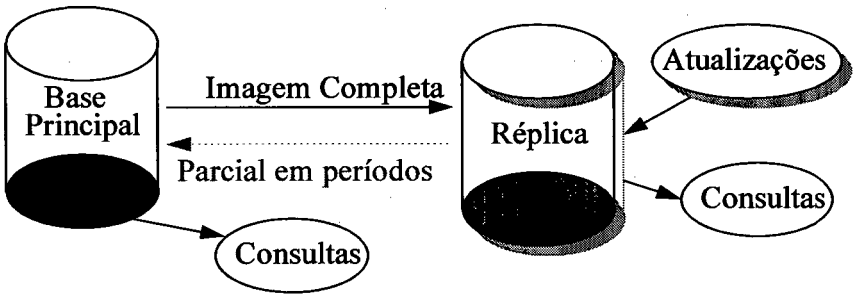


Figura 5.54 Arquivos distribuídos em forma de réplica periódica

RÉPLICA CONTÍNUA - RC - É um tipo de réplica que envia os dados alterados para o banco principal sempre que sofrer uma modificação. Isto possui a vantagem de manter a consistência com o banco de dados primário, sujeito ao atraso em aplicar cada mudança. A desvantagem da réplica contínua é que podem acontecer falhas e a base principal estaria inconsistente. A recuperação e reinício da transação é crítica para manter a

sincronização entre a réplica e o banco principal. A figura 5.55 mostra a réplica contínua.

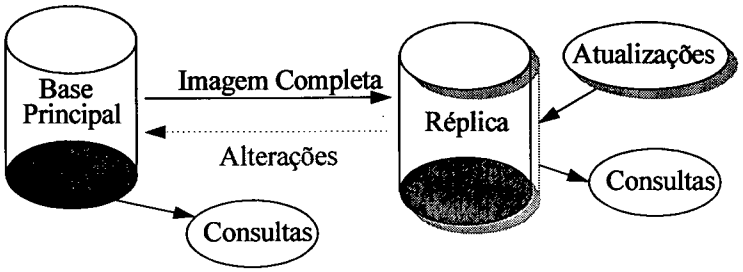


Figura 5.55 Arquivos distribuídos em forma de réplica contínua

RÉPLICA FINALIZAÇÃO - RF - A réplica (figura 5.56) não só atualiza o banco principal como distribui a atualização para as outras réplicas. Esta técnica também é chamada de réplica de finalização porque o sítio de atualização pode mudar entre banco de dados de acordo com quem deseja atualizar o objeto específico. É particularmente útil quando alguém não pode pré-determinar quais objetos de banco de dados devem ser copiados para os bancos secundários. O primeiro passo é copiar uma imagem completa para o sítio requisitante. O próximo passo é resincronizar periodicamente a réplica com a base primária.

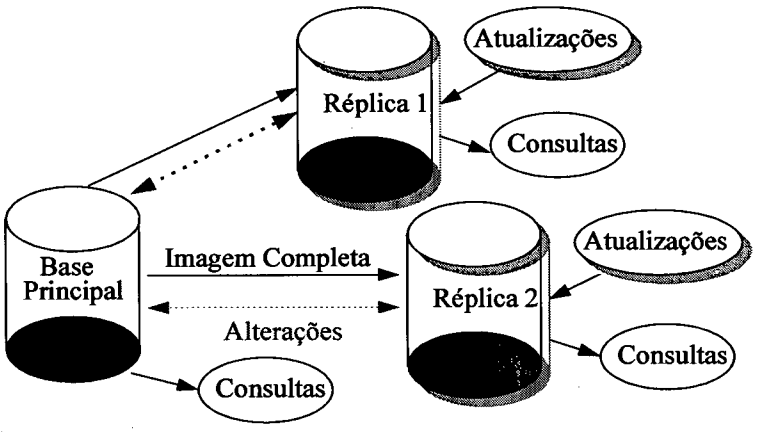


Figura 5.56 Arquivos distribuídos - réplicas atualizadas

4) DADOS PARTICIONADOS

Num esquema particionado, os servidores do nó A e B (ver figura 5.52) não possuem dados em comum, e cada um contém seus dados localmente. Se um cliente no nó A solicitar dados não mantidos no seu local, o servidor A poderá passar o pedido à camada seguinte, gerenciada pelo servidor C. O servidor C não possui dados - só sabe quem possui quais dados. O servidor C atua como corretor de informações, localizando os dados so-

licitados no servidor B e atuando como cliente substituto para o servidor B. Quando os dados chegam de B, são repassados para o servidor A. Como essa interação é feita apenas entre servidores, é transparente ao cliente original.

PARTICIONAMENTO - P - São segmentos sem "overlap" do banco principal, mostrados na figura 5.57. Há partições verticais e horizontais. Só é possível atualizar em um local ou sítio. Nesta técnica o banco de dados primário é separado sem qualquer sobreposição ou duplicação nos dados. Uma dificuldade desta técnica é que se um cliente requisita serviços de uma outra localidade que não está no seu sítio primário, então esta requisição deve ser submetida remotamente ao escritório adequado. Outro problema refere-se a uma determinada chave de um sítio que deve ser movimentada para outro local (uma mudança de endereço).

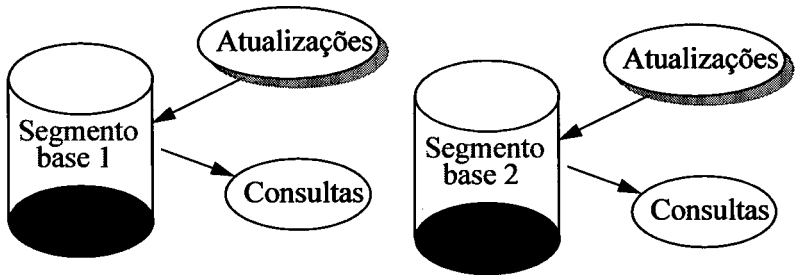


Figura 5.57 Arquivos particionados

5) DADOS REORGANIZADOS OU EXTRATOS

Em um esquema reorganizado, os servidores do nó mantêm dados num nível detalhado de granularidade. Outros grupos de trabalho podem precisar ver estes dados, mas somente num nível mais alto - talvez resumidos de alguma maneira. O servidor C (da figura 5.52) mantém a visão de resumo de todos os dados nos grupos de trabalho. Se um cliente no nó A solicitar dados resumidos, o servidor A encaminhará o pedido ao servidor C. Se os dados em um nó forem atualizados, o servidor do nó deverá atualizar a informação de resumo no servidor C.

EXTRATOS - São cópias de um banco de dados principal que só podem receber consultas. Esta técnica é importante pois reduz o tráfego de dados na rede, mas em contrapartida, precisa de área em disco adicional. Cópias de arquivos voláteis perdem sua importância rapidamente. Os extratos são classificados por: extrato simples, por tempo e atualizado.

EXTRATO SIMPLES - ES - É um fragmento ou uma cópia do banco principal (figura 5.58) que não pode receber atualizações. É uma formula usada para dar suporte à outras atividades, levando este extrato para estações inteligentes. É excelente para dados históricos e para dados pouco voláteis (que não se alteram). Os extratos devem ter uma política de gerenciamento especial. Devem ser evitados extratos em demasia, onde cada usuário tenha uma. A documentação é importante. A extração (criação do extrato) deve ser feita por pessoas específicas e em horários especiais.

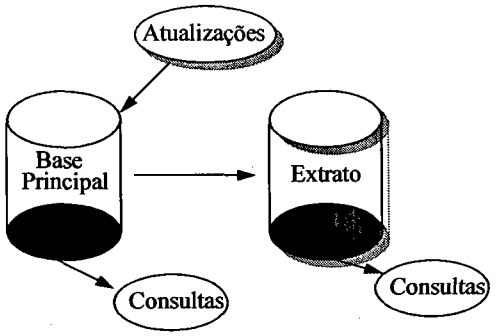


Figura 5.58 Arquivos distribuídos em forma de extrato simples

EXTRATOS POR TEMPO - ET - É um tipo de extrato simples (figura 5.59) onde a aplicação ou o usuário decidem se os dados são válidos ou não, (baseados no dia e hora que foram gerados). Se a aplicação julgar o extrato desatualizado ela pode iniciar uma nova extração. No entanto, esta técnica pode causar impacto no tráfego da rede.

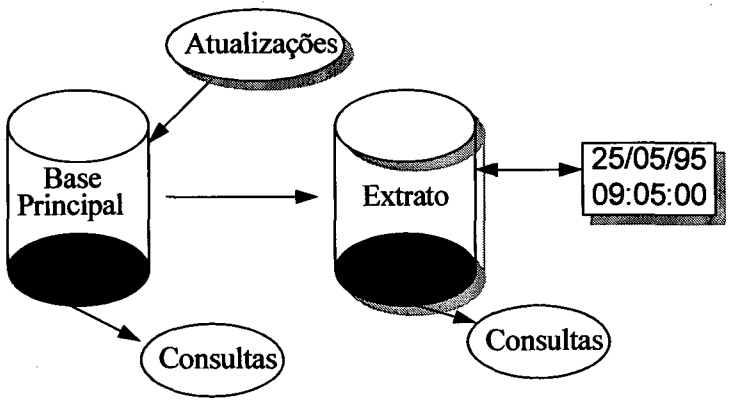


Figura 5.59 Arquivos distribuídos em forma de extrato por tempo

EXTRATOS ATUALIZADO - EA - É um caso de extrato simples o que é executado em intervalos específicos de tempo e de forma automática. O usuário sabe que após o intervalo o arquivo estará atualizado. A melhor técnica para criar este extrato é fazer uma

cópia total e depois a cada intervalo, enviar o que foi modificado. A figura 5.60 ilustra esta técnica.

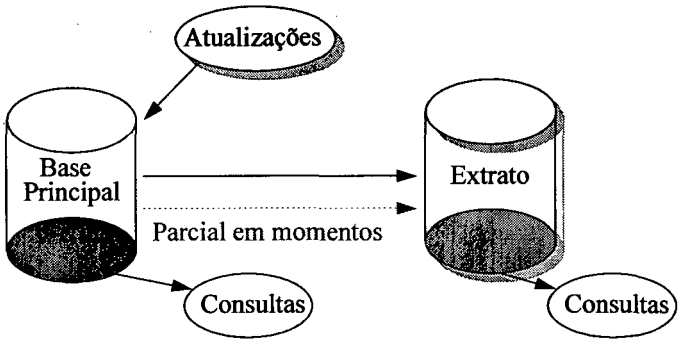


Figura 5.60 Arquivos distribuídos - extrato por intervalos

6) DADOS EM CACHE

Num esquema em cache, os servidores do nó "pegam emprestado", temporariamente, os dados do servidor central. Normalmente isso é feito para explorar a localidade de referência e reduzir o tráfego da rede entre os grupos de trabalho e o servidor central. Todos os caches reduzem a necessidade de consultas com o servidor central. Entretanto, o modo como o depósito central é atualizado depende da política de cache.

Se o esquema de caching for *read-only*, todas as atualizações são enviadas ao servidor central, que as propaga a todos os nós que possuem os mesmos dados em cache (incluindo o local que originou a atualização). Assim, somente o servidor central pode realizar uma atualização. Se o esquema de caching for *write-through*, o servidor do nó atualiza seu cache e envia a atualização para um servidor central. Se o esquema de caching for *write-back*, o servidor do nó atualiza seu cache e notifica o servidor central de que a sua versão é inválida. O depósito de dados é atualizado apenas quando outro servidor de nó referenciar os mesmos dados. Logo, a política de atualização *write-back* é "preguiçosa".

Embora os caches *write-back* sejam ligeiramente mais difíceis de se projetar, eles reduzem a quantidade de tráfego de atualização entre o cache e o depósito de dados em comparação a outros esquemas de caching.

5.4.3 FATORES QUE INFLUENCIAM A DISTRIBUIÇÃO

Os principais fatores que influenciam na distribuição dos dados são: o volume de dados, a fragmentação, a volatilidade e o tempo de transmissão pela rede, conforme a tabela 5.7. As seguintes tabelas [GIL 95] ajudam a verificar qual o tipo de arquivo a ser utilizado. Após responder as questões pode-se ter um resultado: distribuído ou centralizado.

Tabela 5.7 Distribuir ou Centralizar

QUESTÕES DE DISTRIBUIÇÃO DO DADO	DISTRIBUÍDO	CENTRALIZ.
Volume de Dados Qual é o volume de dados envolvido?	Baixo	Alto
Qual a facilidade para transferir os dados?	Fácil	Difícil
Fragmentação Lugares distribuídos precisam de todo o arquivo ou apenas de uma parte?	Parte	Todo
Qual é o grau de dificuldade em particionar o banco de dados em segmentos?	Fácil	Difícil
Atualização A atualização dos dados ocorre em múltiplos lugares distribuídos?	Um só lugar	Vários lugares
Volatilidade Os dados são alterados com freqüência?	Pequena	Grande
Qual é o impacto se a distribuição não for sincronizada com o banco principal?	Pequena	Grande
Tempo O tempo de transmissão dos dados é importante?	Pouco	Muito

1) QUANDO DESCENTRALIZAR E COMO

Se a opção encontrado no item acima foi distribuída, então deve-se verificar qual o tipo de distribuição. Nas tabelas abaixo (tabelas 5.8 a 5.15) são feitas afirmações. Responder para cada uma delas V/F (verdadeiro ou falso). Dar notas de 1 a 7 da seguinte maneira. Se a resposta for verdadeira colocar MAIS 5,6,7. Se a resposta for falsa colocar MENOS 1,2,3. Na dúvida anotar 4. Se as respostas forem próximas ao 7, o arquivo é Centralizado(C). Se as respostas ficarem abaixo do 5 tentar a próxima tabela Particionado (P). Algumas perguntas questionam itens quanto a quantidades ou ao tempo. Cada usuário deve determinar o que é por exemplo, um arquivo grande.

Tabela 5.8 Pesos para avaliar a utilização da opção de arquivos centralizados

CHECKLIST	**1=Menos C, 7=Mais C**						
DADOS CENTRALIZADOS **C**	1	2	3	4	5	6	7
Dados são alterados muitas vezes.							
O número de colunas é muito grande.							
Os dados são muito coesos(ligados) e não permitem fragmentação por arranjos de chaves ou segmentos de dados.							
Segurança é a chave do negócio e vital para a integridade dos dados.							
O dado é a chave vital para a aplicação empresarial							

Tabela 5.9 Pesos para avaliar a utilização da opção de arquivos particionados

CHECKLIST	**1=Menos P, 7=Mais P**						
DADOS PARTICIONADOS - **P**	1	2	3	4	5	6	7
O dado é conhecido em apenas um local geográfico (departamento) e não é compartilhado pela empresa.							
O dado é propriedade local e a maior parte do fluxo de trabalho localiza-se ali.							
Uma área descentralizada é um setor para os dados e pré-processo ou posto de processo de outra área.							
Dados são alterados freqüentemente							

Tabela 5.10 Pesos para avaliar a utilização da opção de arquivos de extrato simples

CHECKLIST	**1=Menos S, 7=Mais S**						
EXTRATO SIMPLES **S**	1	2	3	4	5	6	7
Os dados raramente se alteram, o local copia-os para leitura.							
Dados são usados para dar suporte à decisão e não são identificados por seqüência de data ou hora.							
Dados não são a chave para aplicações empresariais.							

Tabela 5.11 Pesos para avaliar a utilização da opção de arquivos de extrato por tempo

CHECKLIST	**1=Menos ET, 7=Mais ET**						
EXTRATO TEMPO **ET**	1	2	3	4	5	6	7
Os dados são alterados periodicamente e o local os copia para leitura.							
Os dados são usados para dar suporte à decisão e não são identificados por seqüência de data ou hora.							
Dados não são chave para nenhuma aplicação empresarial							

Tabela 5.12 Pesos para avaliar a utilização da opção de extrato por intervalo

CHECKLIST	**1=Menos EA, 7=Mais EA**						
EXTRATO ATUALIZADO **EA**	1	2	3	4	5	6	7
Os dados são alterados periodicamente e o local os copia para leitura.							
A quantidade de dados é grande e as alterações podem ser identificadas em análise de período de tempo, chaves alteradas ou por atributos modelo.							
Dados não são chave para nenhuma aplicação empresarial.							

Tabela 5.13 Pesos para avaliar a utilização da opção de arquivos de réplica periódica

CHECKLIST	**1=Menos RP, 7=Mais RP**						
RÉPLICA PERIÓDICA **RP**	1	2	3	4	5	6	7
Os dados alteram-se lentamente							
Dados são específicos para poucas áreas ou divisões							
Dados são fáceis de identificar e segmentar							

Tabela 5.14 Pesos para avaliar a utilização da opção de arquivos de réplica contínua

CHECKLIST	**1=Menos RC, 7=Mais RC**						
RÉPLICA CONTÍNUA **RC**	1	2	3	4	5	6	7
Os dados alteram-se até 50% num período de tempo.							
Dados alterados são fáceis de serem reconhecidos.							
Dados são específicos para várias áreas geográficas .							
Dados são fáceis de identificar e segmentar.							

Tabela 5.15 Pesos para avaliar a utilização da opção de arquivos de réplica atualizada

CHECKLIST	**1=Menos RF, 7=Mais RF**						
RÉPLICA FINALIZAÇÃO **RF**	1	2	3	4	5	6	7
Os dados alteram-se até 30% num período de tempo.							
Dados não são específicos de uma área geográfica.							
É difícil identificar que entidades terão alterações.							

2) RESUMO DA DISTRIBUIÇÃO

Levando em consideração as técnicas citadas e os fatores de fragmentação, volatilidade, atualização, volume de dados e tempo de transmissão na rede, pode-se ter algumas considerações sobre a utilização de uma técnica em lugar de outra:

- Se a estrutura de dados é difícil de se separar, a técnica é centralizada;
- Se os dados são voláteis, usar centralização ou partição;
- Os extratos e réplicas são aplicáveis quando existe pequena volatilidade.

A tabela 5.16 resume alguns casos e pode ser usada como um guia.

Tabela 5.16 Resumo das técnicas para descentralizar dados

ESTRATÉGIA DE DISTRIBUIÇÃO (resumo)				
FRAGMENTAÇÃO	**VOLATILIDADE**	**ATUALIZAÇÃO**	**MÉTODO**	
Difícil	N/A	N/A	Dado centralizado	**C**
Fácil	Alta	1 local	Partição	**P**
Fácil	Nenhuma	Não	Extrato simples	**ES**
Fácil	Baixa	Não	Extrato tempo	**ET**
Fácil	Média	Não	Extrato atualizado	**EA**
Fácil	Baixa	1 local	Réplica periódica	**RP**
Fácil	Média	1 local	Réplica contínua	**RC**
Fácil	Baixa	Multilocal	Réplica finalização	**RF**
Fácil	Alta	Multilocal	Centralizado	**C**
Fácil	Média	Multilocal	Partição	**P**

A tabela 5.17 é usada para registrar o resultado da distribuição dos dados.

Tabela 5.17 Resultado de arquivo para cada local da empresa

RESULTADO			
	Local 1	**Local 2**	**Local 3**
Entidade 1	C	ET	ES
Entidade 2	P	P	P
Entidade 3	RF	RC	C
C = CENTRALIZADO		P = PARTIÇÃO	
ES = EXTRATO SIMPLES		RP = RÉPLICA PERIÓDICA	
ET = EXTRATO TEMPO		RC = RÉPLICA CONTÍNUA	
EA = EXTRATO ATUALIZADO		RF = RÉPLICA FINALIZAÇÃO	

3) MATRIZ CRUD

Esta tabela analisa a quantidade de dados que cada entidade fará uso em um processo. Ela pode ser preenchida no início do processo como base para responder as perguntas sobre centralizar ou não. Ela também é útil como parâmetro para a definição do banco de dados, dimensionadas as quantidades. As quantidades podem ser expressas em regis-

tros ou em transações. A tabela deve ser aplicada para cada local onde o dado será utilizado. As quantidades podem ser expressas para cada item (C,R,U,D) ou apenas como um Total, demonstrado na tabela 5.18.

Tabela 5.18 Freqüência de operações

MONTAGEM DE MATRIZ CRUD (local abc)			
AÇÃO **OBJETO**	CRIAR SÓCIO	CRIAR PEDIDO	ATUALIZAR PRODUTO
SÓCIO	C	R(200)	R
PEDIDO		C (50)	R,U
PRODUTO		R(200)	U
FREQ. DIÁRIA	500	450	250
FREQ. PICO HORA	100	150	100
C = Create	**R = Read**	**U = Update**	**D = Delete**

4) CONCLUSÃO

As técnicas aqui mencionadas não são exatas. Elas constituem um método que ajudará o usuário sobre a forma de manter seus dados quanto a aspectos de distribuição.

Muitas vezes as decisões de manter um arquivo distribuído ou centralizado não são aquelas caracterizadas pela performance ou redução de tráfego na rede. Elas podem basear-se em itens como segurança e integridade dos dados não abordados nesta metodologia. Além disso, custos ou outras formas de administração de dados ou de desenvolvimento de sistemas podem determinar como os dados devem ser mantidos.

5.5 A DOCUMENTAÇÃO DO PROJETO

As decisões de projeto discutidas nesta fase devem ser documentadas ao serem tomadas. O documento do projeto deve ser uma extensão do documento de análise dos requisitos. Assim, o documento do projeto incluirá uma descrição revista e muito mais detalhada do modelo de objetos, tanto na forma gráfica (diagrama de modelos de objetos) como na textual (descrição de classes, serviços e atributos).

6. CONCLUSÃO

Para especificar uma metodologia voltada ao desenvolvimento de sistemas com o uso da Orientação a Objetos e a arquitetura cliente/servidor foi utilizada uma combinação da técnica de modelagem estruturada de objetos com extensão nos conceitos de entidade-relacionamento (E-R), e técnicas de modelagem do ciclo de vida dos objetos descritos em detalhes por Rumbaugh, Coad/Yourdon e Yourdon. O projeto voltado para a arquitetura C/S se baseia nas especificações de Renaud. O projeto de interface gráfica é uma compilação entre as mais importantes interfaces GUI.

As técnicas aqui utilizadas fazem uso de ferramentas já conhecidas pelos analistas e programadores. Esta semelhança facilitará a introdução da tecnologia OO.

As empresas estão trabalhando interligadas em rede. Isto significa que uma aplicação comercial para a organização envolve o uso de conceitos de distribuição de objetos. Nenhuma das metodologias populares publicadas descrevem as interfaces gráficas (GUI) ou a distribuição de dados e processos para a arquitetura cliente/servidor. E é exatamente neste paradigma onde a tecnologia OO será mais aplicada.

A introdução da tecnologia OO [GRE 93] nas empresas que estão utilizando plataformas de desenvolvimento normais (orientação funcional) devem levar em conta os investimentos a serem feitos com ferramentas para suportar o novo paradigma. Estas ferramentas, além de linguagens de programação podem incluir banco de dados, ferramentas CASE e rotinas de uso geral da organização.

Além dos custos e a necessidade de treinamento para adaptar-se as mudanças tecnológicas algumas organizações podem ter razões válidas para adiar ou mesmo evitar a transição de metodologias tradicionais para metodologias orientadas a objetos. No entanto, para o desenvolvimento de aplicações em larga escala, o potencial de benefícios de produtividade e ganho de qualidade fazem com que a decisão seja inevitável. Para muitas organizações a questão não é "se" elas devem adotar a tecnologia de objetos mas simplesmente "quando" adotá-la.

Esses fatos fazem com que este trabalho apresente uma metodologia de segunda geração. Ela integra em um único trabalho as técnicas OO, C/S, distribuição de dados e distribui-

ção de processos. Por isso, acredita-se que este trabalho representa uma metodologia que orientará o desenvolvimento dos sistemas atuais e futuros.

Trabalhos Futuros - esta metodologia não aborda alguns aspectos do ciclo de vida dos objetos e/ou de um sistema de informação. Para atender a este apertos podem surgir outros trabalhos abordando os seguintes tópico:

- Implementação - as classes e objetos e os relacionamentos desenvolvidos durante a análise e aperfeiçoados no projeto são traduzidos para uma linguagem de programação baseada normalmente em objetos.

- Planejamento de Sistemas - identificar e implementar sistemas de informações necessários para suportar as funções requeridas pela organização, bem como um planejamento das classes e objetos existentes.

- Qualidade - implementar pontos de verificação ao longo da metodologia para garantir a qualidade de cada etapa executada. Este trabalho teria como meta principal efetuar a análise e produzir software com qualidade e taxa reduzida de erros.

BIBLIOGRAFIA

[ADC 91] System Application Architecture, AD/Cycle Concepts, IBM, San Jose, 1991.

[AID 91] Advanced Interface Design Reference (sc34-4290-00), IBM, Cary-Carolina do Norte - USA, 1991.

[BAR 94] BARKAKATI, Nabajyoti & Peter D. Hipson, Visual C++ , Rio de Janeiro, Editora Berkeley, 1994.

[BOO 94] BOOCH, Grady, Object-Oriented Analysis and Design with Applications, Redwood City, CA: The Benjamin/Cummings Publishing Company, 1994.

[CAS 85] CASANOVA, Marco Antônio, Princípios de Sistemas de Gerência de Banco de Dados Distribuídos, Rio de Janeiro, Editora Campus, 1985.

[CER 91] CERÍCOLA, Osvaldo Vicente, Banco de Dados Relacional e Distribuído, Rio de Janeiro, Editora Livros Técnicos e Científicos, 1991.

[CHE 95] CHEDE, Cesar Taurion, NOVOS DESAFIOS, Computerworld, Pp 18-20, 20 de fevereiro de 1995.

[COD 92] COAD, Peter e Edward Yourdon, Análise Baseado em Objetos, Editora Campos, Rio de Janeiro, 1995.

[COD 93] COAD, Peter e Edward Yourdon, Projeto Baseado em Objetos, Editora Campos, Rio de Janeiro, 1995.

[COG 94] COGE,(Comitê de Gestão Empresarial do Setor Elétrico), Relatório: Processamento Distribuído em Multiplataforma, Rio de Janeiro, 1995.

[CUA 91] System Application Architecture - Common User Access - Guide to User Interface Design (sc34-4289-00), IBM, Cary-Carolina do Norte USA, 1991.

[EDW 90] YOURDON, Edward, Análise Estruturada Moderna, Editora Campus, Rio de Janeiro, 1990.

[DRI 90] Centrais Elétricas do Sul do Brasil - ELETROSUL, Departamento de Recursos da Informação (DRI), Metodologia de Desenvolvimento de Sistemas, Florianópolis, 1990.

[DAT 91] DATAPRO Reports on International Telecommunications, 1991.

[DRU 86] DRUCKER, P, F., Inovação e o Espírito Empreendedor, Editora Nova Fronteira, São Paulo, 1986.

[JAC 92] JACOBSON, Ivar et al, Objetct-Oriented Software Engineering: A Use Case Driven Approach, Reading, MA: Addison-Wesley, 1992.

[JAM 91] JAMES, Martin e Carma McClure, Técnicas Estruturadas e CASE, Editora Makron, São Paulo, 1991.

[GRA 95] GRANDI,Gilberto, Análise e Distribuição de Dados em Ambiente Cliente/servidor, Curso de Pós-Graduação em Ciências da Computação, UFSC, Florianópolis, 1995.

[GIL 95] ----------, Técnicas de Distribuição de Dados, II Congresso de Informática e Telecomunicações de Mato Grosso, UFMT, Cuiabá, 1995.

[GRE 93] GRECHENIG, T. et al, Where Tool-Oriented System Development will Fail OO-Software Engineering Requires Serious Efforts in Analysis and Design, Univ.of Technol., Vienna, Austrália - Proceed. of 6th Internat. Conference on Software Eng. and its Applications, Paris, France. 15 Nov 93, Pp 639-649.

[JAM 91] JAMES, Martin e Carma McClure, Técnicas Estruturadas e CASE, Editora Makron, São Paulo, 1991.

[HAC 93] HACKATHORN, Richard D., Conectividade de Banco de Dados Empresariais, Rio de Janeiro, Editora Infobook, 1995.

[HAR 95] HARMON, Paul, Object-Oriented Strategies, Volume V, Número 7, Pp 9, San Francisco, CA, July, 1995.

[IBM 92] IBM the Official Guide, Object-Oriented Interface Design, Published by Que Corporation, Carmel, IN, 1992.

[MAR 89] MARTIN, James e Clive Filkelstein, Engenharia da Informação, São Paulo, Compucenter Sistemas, 1989.

[MET 94] ELETROSUL, CDE, Metodologia do Ciclo do Planejamento Empresarial, Florianópolis, 1995.

[MET 91] ELETROSUL, DRI/DIOM, Metodologia Para Planejamento de Sistema de In formação, Florianópolis, 1991.

[MIC 93] MICROSOFT Corporation, Analysis and Design of CLient-Server Systems, Pp 73-74, USA, 1995.

[MSF 93] MICROSOFT, Visual Design Guide, Microsoft Corporation, USA, 1995.

[MIN 94] MINASI, Mark, Interface Gráfica Com o Usuário, Rio de Janeiro, Editora Infobook, 1995.

[MON 94] MONTENEGRO, Fernando e Roberto Pacheco, Orientação a Objetos em C++, Rio de Janeiro, Editora Ciência Moderna, 1995.

[PER 94] PERRY, Greg, Programação Orientada para Objetos com Turbo C++, Rio de Janeiro, Editora Berkeley, 1994.

[REN 94] RENAUD, Pauel E., Introdução aos Sistemas Cliente/servidor, Rio de Janeiro, Editora Infobook, 1995.

[RUM 94] RUMBAUGH, James, et al, Modelagem e Projeto Baseados em Objetos, Rio de Janeiro, Editora Campus, 1995.

[SAL 93] SALEMI, Joe, Banco de Dados Cliente/Servidor, Rio de Janeiro, Editora Infobook, 1995.

[TAN 94] TANENBAUM, Andradw S., Rede de Computadores , Rio de Janeiro, Editora Campus, 1995.

[YOU 95] YOURDON, Ed et al, Mainstream Objects, New York, Prentice-Hall Inc, 1995.