UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

FERRAMENTA DE APOIO AO PROCESSO DE REUTILIZAÇÃO DE ESPECIFICAÇÃO ESTRUTURADA

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

GILVAN JUSTINO

BLUMENAU, JUNHO/1999

1999/1-17

FERRAMENTA DE APOIO AO PROCESSO DE REUTILIZAÇÃO DE ESPECIFICAÇÃO ESTRUTURADA

GILVAN JUSTINO

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Everaldo Artur Grahl — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Everaldo Artur Grahl

_____

Prof. Oscar Dalfovo

_____

Prof. Marcel Hugo

AGRADECIMENTOS

Agradeço ao professor Everaldo Artur Grahl que me orientou no desenvolvimento deste trabalho.

Aos meus colegas pela colaboração.

Aos meus pais, pelo apoio que ofereceram durante todo o decorrer do curso.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A reutilização de software é uma das formas existentes para se aumentar a produtividade e qualidade na produção de software. Um dos componentes a serem reusados é a especificação de sistemas. Este trabalho objetiva criar um processo prático de reutilização de especificação estruturada com o auxílio de uma ferramenta. A ferramenta desenvolvida está baseada na técnica da Analogia e permite a reutilização de diagramas de fluxo de dados, modelos entidade-relacionamento e dicionários de dados.

ABSTRACT

The reusing of software is one of the existing forms to increase productivity and quality in the production of software. One of the components to be reused is the system specification. This work creating a pratical of reusing of aims at structured specification with the help of a tool. The developd tool is based on the Analogy technique and allows the reusing of data flow diagram, relationship-entity models and data dictionary.

1 INTRODUÇÃO

1.1 ORIGEM

Reutilizar software consiste em reaproveitar partes, que foram utilizadas na especificação ou desenvolvimento de um software, na construção de um novo software. Conforme [RAM98], a prática de reutilização de software é considerada como um dos fatores de maior importância na amenização dos problemas resultantes da crescente complexidade e da abrangência que os sistemas estão atingindo. [MCC92] cita os seguintes benefícios adquiridos com a aplicação de componentes reutilizáveis: redução do tempo e do custo do desenvolvimento, melhoria na qualidade de software, aumento da produtividade e compartilhamento do conhecimento sobre o sistema.

Segundo [MCC92], qualquer informação utilizada na construção de um novo software pode ser reutilizada: o código de programação, especificações de projeto, estrutura de dados, dados, documentação, protótipos, interfaces com o usuário, testes, etc.

O conceito de reutilização de software em nível de especificação é um dos métodos mais interessantes de aproveitamento de software, pois permite eliminar completamente o esforço envolvido em projetar, codificar e testar uma implementação da especificação reutilizada [GRA97].

A especificação estruturada é uma metodologia utilizada para modelar o controle e o fluxo da informação num sistema. Esta metodologia compreende três técnicas: diagrama de fluxo de dados (DFD), modelo entidade-relacionamento (MER) e dicionário de dados.

[PRE95] descreve o DFD como uma técnica gráfica que representa o fluxo de informação e as transformações que são aplicadas à medida que os dados movimentam-se no sistema.

O MER permite a visualização, através de um gráfico, dos objetos que compõe o sistema e seus relacionamentos enquanto o dicionário de dados descreve o conteúdo destes objetos.

Os benefícios oferecidos com a reutilização de especificações são potenciais, como descreve [MAI92]: a reutilização de especificações bem sucedida permite melhorar a produtividade no desenvolvimento de software dando aos desenvolvedores um começo mais rápido durante a análise de requisitos.

O uso de ferramentas baseadas no computador que auxiliam o processo de reutilização são extremamente úteis pois permitem simplificar o processo de desenvolvimento. Visando este objetivo, a ferramenta proposta neste trabalho auxiliará o processo de reutilização de especificações estruturadas.

1.2 OBJETIVO

O objetivo deste trabalho é criar um processo prático de reutilização de especificações estruturadas com auxílio de uma ferramenta. Esta ferramenta deve permitir ao usuário reusar estruturas de diagramas de fluxo de dados, modelos entidade-relacionamento e dicionários de dados a partir de sistemas já modelados na ferramenta.

1.3 ORGANIZAÇÃO DO TEXTO

Este trabalho está dividido nos seguintes capítulos:

O primeiro capítulo apresenta uma visão geral do trabalho, destacando o reutilização, a especificação estruturada, o objetivo e a organização deste trabalho.

No segundo capítulo encontra-se os conceitos relacionados com a reutilização de software: os benefícios oferecidos, tipos de componentes reusáveis, as dificuldades do reuso de software e ferramentas de apoio à reutilização.

O terceiro capítulo tratará do estudo da especificação estruturada. A especificação estruturada compreende três ferramentas: o diagrama de fluxo de dados (DFD), o modelo entidade-relacionamento (MER) e o dicionário de dados.

No quarto capítulo encontra-se o estudo da técnica de reuso utilizada para este trabalho: Analogia.

No quinto capítulo está a especificação da ferramenta implementada.

O sexto capítulo apresenta a descrição da ferramenta com a apresentação das principais telas.

O sétimo capítulo estão as conclusões e sugestões do trabalho.

2 REUSO DE SOFTWARE

2.1 CONCEITOS INICIAIS

Desde o surgimento do computador se tem visto o crescente avanço da tecnologia. O progresso acelerado do hardware e o seu baixo custo permitiu o desenvolvimento de aplicações cada vez mais complexas e poderosas.

Conforme [MIT90], o software tem provado ser um gargalo no processo de produção de sistemas baseados no computador, já que a produção de software é lenta, cara e propensa a erros. Estes problemas são atribuídos sobretudo à baixa qualificação dos programadores e à baixa qualidade do próprio processo de produção.

As discussões sobre os problemas com a produção do software iniciaram em 1969 com o surgimento da "crise do software". Esta expressão foi utilizada para descrever as dificuldades e as conseqüentes frustrações que o desenvolvimento e a manutenção de software trouxeram para as empresas [MIT90]. A baixa produtividade, atrasos na entrega do produto, a falta de qualidade, o volume dos recursos investidos e o alto tempo despendido com a manutenção do software são, de acordo com [PIM99] e [GRA97], problemas que ainda persistem nos dias atuais.

Algumas das estratégias comentadas por [MIT90] e [GRA97] que podem promover a produtividade e a qualidade no processo de produção de software são:

a) automatizar ao máximo o processo de produção;
b) delegar a produção de software a sistemas especialistas;
c) reutilizar softwares existentes tanto quanto possível.

O reuso de software é uma das práticas mais eficazes para melhorar o processo de desenvolvimento, a qualidade e produtividade do software e é considerada um dos maiores fatores para amenizar os problemas resultantes da crescente complexidade e da abrangência que os sistemas estão atingindo [RAM98].

[RAM98] define a reutilização de software como sendo uma técnica utilizada para reaproveitar partes de um sistema que já foi construído e estão disponíveis, na construção de um novo software.

O conceito de reutilização de software abrange não somente código de programa, mas vai desde especificações de projeto, código de programa até documentação do sistema.

Para que o processo de reuso de software seja prático é necessário o suporte de ferramentas apropriadas.

2.2 BENEFÍCIOS

Um dos principais motivos que levam as empresas a praticarem a reutilização de software é o fator econômico. Os principais benefícios oferecidos com a reutilização de software é a diminuição do tempo de desenvolvimento do software e, consequentemente, a diminuição de custo. Este é um fatores que mais motivam as empresas a implantar um programa de reutilização de software. Porém estes não são os únicos benefícios oferecidos com o emprego de técnicas para a reutilização de software. [RAM98], [MCC92] e [VER99] relacionam os seguintes benefícios:

a) aumento da produtividade. O aumento da produtividade é alcançado com a redução do tempo gasto por não precisar desenvolver o item reusado;
b) aumento da qualidade. Já que os componentes reusados já foram usados em outro sistema, os mesmos já foram testados e corrigidos e por isso devem ser mais consistentes e apresentar menos erros;
c) diminuição dos custos. A produtividade e o aumento da qualidade reduzem o tempo gasto com a implementação e com a manutenção do software. Além disso, os processos de documentação e testes podem ser otimizados com o reuso;
d) permite o compartilhamento de conhecimentos entre os diversos grupos de desenvolvimento;
e) facilita o estudo sobre a arquitetura de sistemas e como construir bons sistemas.

O aumento da produtividade é justificado por [KUT99] devido a três fatores:

a) reuso de software amplia a capacidade de programação. O programador tem menos código para escrever e pode se concentrar na estrutura do programa;
b) reuso de software reduz a quantidade de documentação e de testes;
c) uso de componentes reusáveis no desenvolvimento de softwares torna a manutenção mais fácil, pois seus desenvolvedores estão mais familiarizados com

os blocos que foram reutilizados e por isso entendem mais rapidamente o sistema completo.

E a qualidade é atribuída às seguintes características dos software reusáveis:

a) softwares reusáveis são bem projetados, isto é, são projetados para o reuso;
b) softwares reusáveis são bem documentados - conforme um padrão preestabelecido;
c) softwares reusáveis são exaustivamente testados pois foram garantidos para reuso. Quanto mais o software for reusado menor será a chance de se encontrar erros pois, se existiam, provavelmente usuários anteriores perceberam e estes já foram corrigidos;
d) softwares reusáveis possuem funções fáceis de entender e por isso serão utilizados apropriadamente.

[KUT99] e [YOU95] mencionam também que a reusabilidade pode oferecer um mecanismo para a prototipação, de modo que ofereça ampla funcionalidade e é capaz de permitir que o protótipo seja tão fácil e rápido de ser construído quanto qualquer outra forma de protótipo.

2.3 FORMAS OU MODOS DE REUTILIZAÇÃO

Conforme [FRE86], [MCC92] e [MEN89], qualquer informação necessária no processo de criação de software pode ser reusada para a criação de um novo software. Isto leva a concluir que existem diversas formas de reutilização. [YOU95] classifica as formas de reutilização em quatro níveis: reuso de código, reuso de dados, reuso de projetos físicos e reuso de especificações.

2.3.1 REUSO DE CÓDIGO

O reuso de código é a forma mais conhecida e encontrada na literatura e consiste em aproveitar o código de programas.

No trabalho de [FRE86] são descritas as características desta área da reusabilidade:

a) item que é reusado é um pedaço do código executável;

b) assume-se geralmente que o fragmento a ser reusado deverá ser realizado num sistema operacional específico;
c) foco principal do reuso de código é reduzir o número de linhas de código que o programador deve escrever ao construir uma nova aplicação;
d) qualquer único fragmento de código que for reusado tem pequeno ou nenhum significado sem estar presente com outras partes de código.

De acordo com [YOU95], o reuso de código pode ser realizado através das seguintes formas:

a) cortar e colar do código fonte. Esta é a forma de reutilização mais primitiva de todas. O código para ser reusado é copiado e algumas vezes sofre algumas modificações a fim de se adaptar ao novo contexto. Este modo de reuso possibilita que o programador produza erros ao copiar ou modificar e traz principalmente problemas de gerenciamento de configuração onde uma alteração no código original provocaria a alteração em todos programas que foram reusados a partir deste;
b) inclusões a nível de fonte. Este recurso é usado para incluir as definições de uma biblioteca num programa. Isto é feito, na linguagem de programação C, por exemplo, com a declaração "include". As inclusões a nível de fonte requerem uma cópia a cada vez que a biblioteca é invocada. Assim, toda vez que se usa "include" num programa, todo o código da unidade que se está invocando é copiado para o programa;
c) vínculos binários. Tem o mesmo objetivo das inclusões a nível de software, porém é mantido apenas uma cópia no sistema das definições reusadas. Este benefício é oferecido pelas linguagens de programação com o procedimento de linkedição;
d) invocação em tempo de execução. Ao contrário das duas últimas formas de reutilização de código, o reuso por invocação em tempo de execução não requer uma cópia junto ao módulo executável. Assim, o programador não precisa definir qual componente será chamado. Isto possibilita uma certa flexibilidade, pois o sistema poderá definir em tempo de execução qual componente chamar.

O reuso ao nível de código, segundo [MCC92] é limitado à linguagem, sistema operacional e aplicação. Mas o maior problema do reuso de código é que antes mesmo de ser

reusado, o código geralmente precisa ser alterado. Estas alterações permitem a introdução de erros e outros efeitos colaterais imprevistos e não desejados. O perigo que pode ser gerado com estas alterações no código tem sido o maior motivo das empresas em não usar esta técnica de desenvolvimento de sistemas.

Apesar do reuso de código ser o mais conhecido ele é o que proporciona menos impacto sobre a produtividade global, pois representa apenas de 10% a 15% sobre a criação do software [YOU95].

2.3.2 REUSO DE DADOS

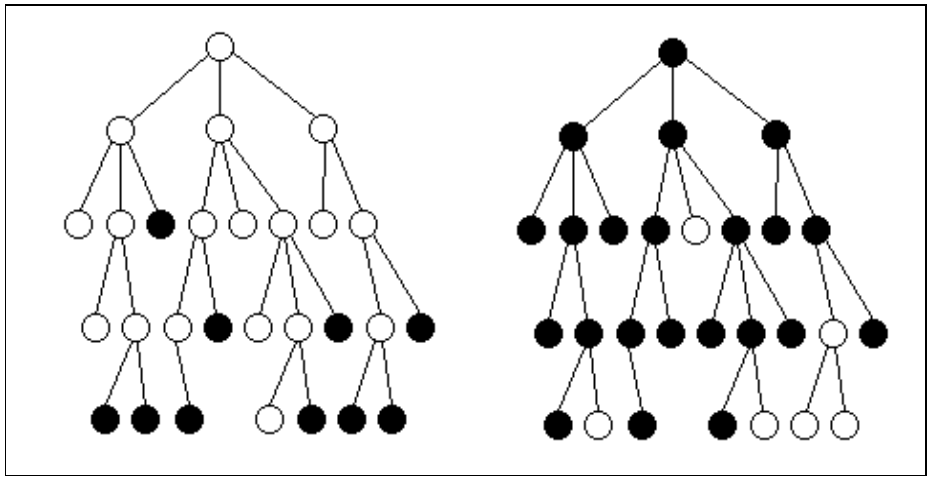
O reuso de dados é realizado com uma das formas de reuso de código, através de inclusões a nível de software. Com o este tipo de recurso pode-se reusar, por exemplo, estruturas de tabelas, registros e parâmetros globais.

É possível reusar todos os dados de um repositório CASE, segundo [YOU95].

2.3.3 REUSO DE PROJETOS FÍSICOS

Segundo [MEN89], resultados mais efetivos do uso da reutilização podem ser alcançados com o reuso de projeto físico e especificação, pois estas fases do desenvolvimento do software são as mais difíceis e consomem muito mais tempo.

O trabalho de [YOU95] apresenta uma ilustração (Figura 1) para demonstrar a diferença entre o reuso de código e o reuso de projeto, onde o reuso de código normalmente ocorre em níveis inferiores da hierarquia, enquanto a reusabilidade de projeto abrange vários níveis.



Fonte: [YOU95]

Figura 1 - Reuso de Código X Reuso de Projeto

2.3.4 REUSO DE ESPECIFICAÇÕES

A reusabilidade de especificações é considerada a mais importante entre todas as outras formas de reuso de software. Pois, segundo [YOU95], ela nos permite eliminar completamente o esforço envolvido em projetar, codificar e testar uma implementação da especificação reutilizada.

A reutilização de especificação compreende a reutilização de modelos de especificação, isto é, diagramas de fluxo de dados, diagramas entidade-relacionamento, diagramas de transição de estado, etc. Segundo [YOU95], o reuso de especificações é possível graças ao repositório de dados das ferramentas CASE, onde todos estes modelos estão mantidos no repositório.

Apesar disso, [KUT99] diz que especificações de programas podem ser reusáveis mas num contexto bem limitado pois é extremamente difícil extrapolar os limites de domínio.

2.4 ABORDAGENS TECNOLÓGICAS

Os principais mecanismos utilizados para reutilizar software, conforme [YOU95] são: biblioteca de componentes e técnicas de desenvolvimento orientadas a objetos.

2.4.1 BIBLIOTECA DE COMPONENTES

Segundo [MCC92], um pré-requisito para promover o reuso de software é uma biblioteca de componentes reusáveis.

Uma biblioteca de componentes geralmente é encontrado nas empresas como um conjunto composto geralmente de código-fonte de funções, sub-rotinas ou classes. Porém [MCC97] relaciona os tipos de componentes que podem fazer parte de uma biblioteca de componentes conforme pode ser visto na Tabela 1.

Alguns pontos relatados por [YOU95] são usado para verificar o grau de comprometimento da empresa com o reuso de software. A empresa deve ter respostas satisfatórias para as seguintes perguntas:

a) os componentes da biblioteca são bem documentados ?
b) que tipo de gerenciamento de configuração é usado para acompanhar o curso das versões atual e anterior dos componentes da biblioteca ?
c) os componentes são bem testados ?
d) que tipo de mecanismos de consulta e procura estão disponíveis para ajudar os engenheiros de software a localizar componentes úteis ?
e) quem tem permissão para inserir novos componentes na biblioteca ?

Segundo [YOU95], uma biblioteca de componentes deve conter entre duzentos e quatrocentos componentes. Uma biblioteca com muitos componentes dificultará a pesquisa dos mesmos e provavelmente o programador deixará de utilizá-lo. Além disso o componente devem ser de preferência bem pequeno afim de executar uma tarefa coesa e funcionalmente bem definida, evitando uma sobrecarga no desempenho ou memória do sistema.

Tabela 1 - Tipos de Componentes Reusáveis

Tipos de componentes reusáveis	
Diagrama de ações	"Esqueleto" de documentação (manual do usuário, documentações técnicas, documentação de operações)
Algoritmo	
Pacotes de aplicação	
Modelos de aplicação	
Regras de negócio	Definições de tipos de entidades
Componentes	Sistema de herança existente
Definição de elementos de dados	"Esqueleto" de planos (plano de projeto, plano de testes, plano de implementação)
Modelo de dados	
Estrutura de dados	
Modelos de processo	Modelos de processo
Esqueleto de funções de programa	Protótipo
Módulo de código do software	Arquitetura de sistema
Interfaces do usuário	Módulos utilitários
Modelos de processo	

Fonte: [MCC97]

2.4.2 TÉCNICAS DE DESENVOLVIMENTO ORIENTADAS A OBJETOS

As técnicas de desenvolvimento orientadas a objetos são metodologias encontradas nas linguagens de programação orientadas a objetos. Esta metodologia oferece alguns benefícios, como por exemplo encapsulamento de dados e funções. O encapsulamento consiste em esconder dados e funções que são necessários apenas para o próprio componente, não sendo de interesse para o programador.

Mas uma das características mais interessantes é a herança. Nestas linguagens, um novo componente (geralmente chamado de classe por estas linguagens) pode ter todas as características e funcionalidades obtidas de um outro componente. Esta característica é chamada de herança, onde um componente “filho” herda de um outro componente, chamado de “pai”, toda a sua definição, isto é, características e funções. Além disso, pode-se facilmente acrescentar novas funções ou alterar aquelas existentes a fim de adaptar o novo componente ao novo contexto.

2.5 O PROCESSO DE REUSO

Na análise e projeto de novos sistemas a possibilidade de reuso precisa ser considerada e os componentes de biblioteca apropriados precisam ser incorporados.

De acordo com [MIT90], a integração do reuso com o tradicional ciclo de vida de software é necessário, por isso o modelo de ciclo de vida tradicional de desenvolvimento de software requer uma dimensão a mais para acomodar a possibilidade de reuso de software em suas várias fases.

De acordo com [KUT99], o reuso pode ser inicializado em qualquer fase do ciclo de vida do software, seja durante o projeto, especificação ou a implementação, dependendo do tipo de componente que se deseja reutilizar. O reuso também pode ser realizado durante a revisão de cada fase do ciclo de vida do software pois torna-se mais fácil encontrar os objetos comuns entre as aplicações. O processo de reuso também poderá ser realizado numa nova fase denominada de Análise de Domínio, conforme [KUT99] comenta. Nesta fase, os requisitos são analisados para uma aplicação genérica e não para uma aplicação específica. Dessa forma, processos comuns em diversas aplicações poderão ser identificados.

Conforme [MCC92], um ambiente composto de ferramentas integradas é necessário para gerenciar os componentes. Dois aspectos do reuso de software devem ser considerados:

a) construir componentes reusáveis;
b) usar componentes reusáveis.

De acordo com [MCC92], para construir componentes reusáveis necessita-se de ferramentas capazes de:

a) identificar componentes que podem ser amplamente reusados;
b) definir componentes e adaptá-los para o reuso;
c) classificar e armazenar os componentes numa biblioteca;
d) representar e descrever componentes reusáveis.

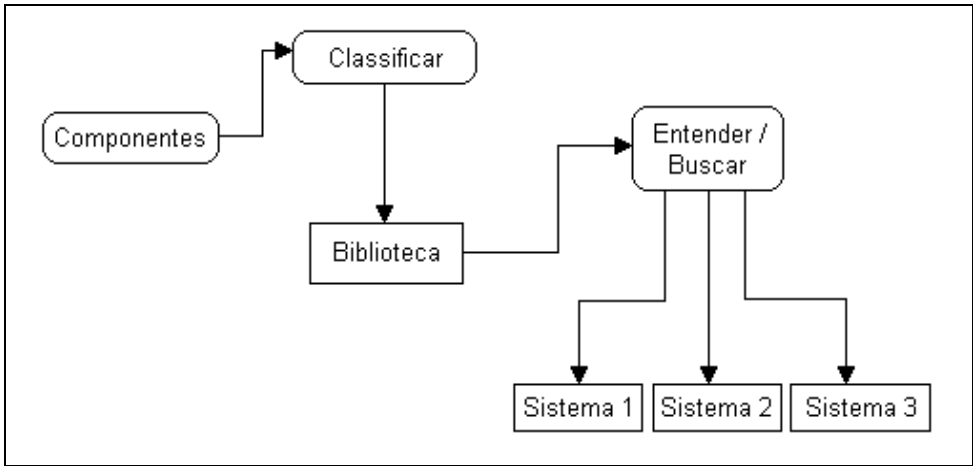
E para usar componentes reutilizáveis efetivamente é necessário ferramentas que:

a) localizam os componentes reusáveis;
b) permitem entender componentes reusáveis;
c) permitem alterar componentes reusáveis;

d) permitam combinar e incorporar componentes reusáveis.

O ambiente de reutilização apresentado por [RAM98] é composto por duas ferramentas: uma ferramenta classifica os componentes e os coloca em um repositório e uma outra ferramenta é responsável pela busca de um determinado componente neste repositório. Este ambiente está ilustrado na Figura 2.

Um dos métodos mais conhecidos de classificação de componentes é o esquema de classificação de facetas. Este método, segundo [MCC92], baseia-se num conjunto de identificadores, ou segundo a terminologia, facetas. Cada faceta representa uma característica de um componente. Assim cada componente é, de acordo com [RAM98], descrito em termos de descrições e classificações fixas. Por exemplo, nome do componente, tipo, área de aplicação, mídia e tecnologia são facetas de um esquema de classificação. Desta forma todos os componentes reusáveis tem características que identificam as facetas. A busca por um componente que atenda determinada necessidade se dá pela comparação das características fornecidas pela pessoa que deseja reutilizar um componente com as características de cada componente.



Fonte: [RAM98]

Figura 2 - Ambiente de Reuso

Para [MCC97] o ambiente de reuso deve ser composto de ferramentas que sejam capazes de armazenar, organizar e gerenciar componentes reusáveis e ter suporte a múltiplas versões de um componente. Além disso, os componentes precisam ser rápidos e fáceis de serem encontrados e a ferramenta deve ser fácil de entender para que os desenvolvedores

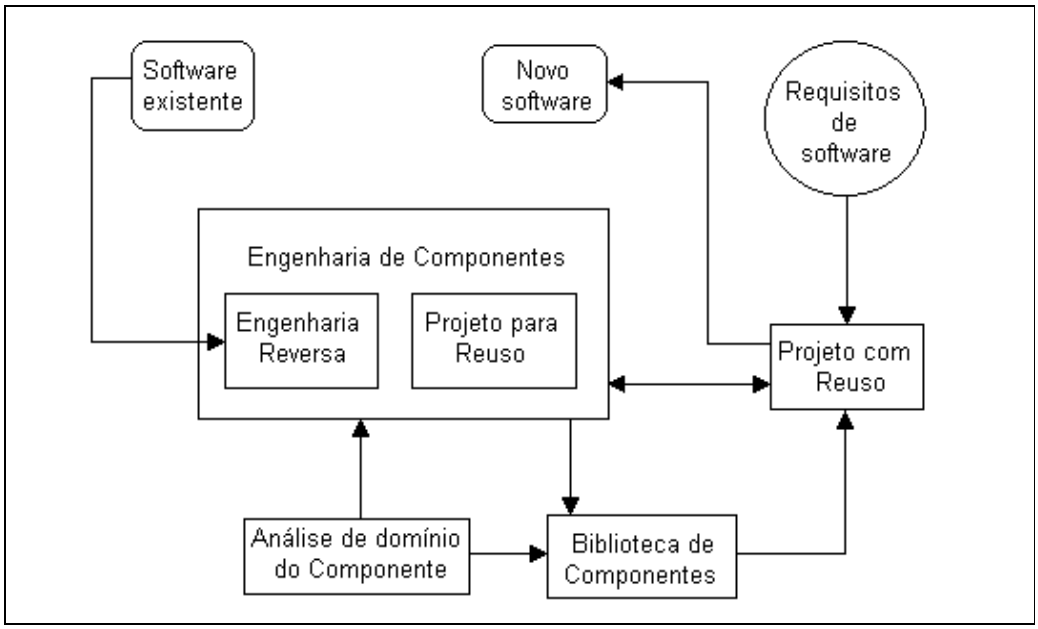
usem os componentes já prontos, catalogados na ferramentas, ao invés de construí-los novamente. São recomendadas cinco ferramentas para dar suporte a criação e uso de uma biblioteca de componentes, conforme pode ser visto na Tabela 2.

Um outro ambiente de reuso baseado na engenharia reversa é descrito por [RAM98]. Neste ambiente a engenharia reversa é utilizada para extrair e reconhecer componentes de software já prontos. Logo após os componentes são armazenados na biblioteca de componentes através de uma análise de domínio do componente. A partir daí, o desenvolvimento de um novo projeto poderá buscar o componente na biblioteca e reutilizá-lo na formação de um novo sistema. Este ambiente pode ser visto na Figura 3.

Tabela 2 - Ferramentas para suporte à biblioteca reusável

Ferramenta	Descrição	Uso
Repositório	Armazena componentes reusáveis	Desenvolvedores usam os componentes do repositório para reuso e incluem novos componentes
Navegador	Examina o repositório a procura de componentes disponíveis	Desenvolvedores e gerentes usam para examinar o conteúdo da biblioteca
Sistema de gerenciamento de configuração	Usado para gerenciar versões e configurações de software	Gerentes usam para controlar versões dos componentes de software
Catálogo	Armazena uma referência para componentes reutilizáveis	Potenciais reutilizadores selecionam componentes para examinar as possibilidades de reuso

Fonte: [MCC97]



Fonte: [RAM98]

Figura 3 - Ambiente de Reutilização baseado na Engenharia Reversa

No trabalho de [KUT99] é apresentado um roteiro para a criação de componentes reusáveis, baseado na DSSA (Arquiteturas de Software de Domínio Específico):

a) definir o escopo do domínio;
b) definir/refinar os elementos do domínio;
c) definir/refinar o Projeto do Domínio e as restrições ao Requisitos de Implementação;
d) desenvolver arquiteturas/modelos de domínio;
e) produzir/coletar os produtos reusáveis.

Muitas vezes o componente que se deseja reusar não pode ser reusado tal como foi implementado, isto é, algumas modificações no componente reusável são necessárias para que ele se adapte as novas condições. Estas alterações no componente de reuso são chamadas de parametrizações ou customizações e segundo [KUT99] esta é a principal dificuldade encontrada no processo de reuso.

2.6 PROGRAMAS DE REUTILIZAÇÃO

Programas de reutilização são estratégias utilizadas para as empresas adotarem a reutilização como metodologia de desenvolvimento de sistemas.

Um programa de reutilização é necessário para promover o reuso na empresa, pois existem vários obstáculos que podem impedir a empresa de utilizá-lo, conforme será visto a seguir.

2.6.1 DIFICULDADES

Existem diversos problemas que podem ocorrer durante o processo de reuso e inclusive antes mesmo que o programa de reuso seja implantado.

Segundo [YOU95], a não utilização da reusabilidade no desenvolvimento de software é devido a quatro razões principais:

a) a maneira errada de como são ensinados, através dos livros didáticos de engenharia de software e cursos universitários, os processo de projetar sistema e escrever código. Na grande maioria das vezes o estudante é levado a construir um sistema a partir ‘do zero’, conforme diz [YOU95], e a reusabilidade não é promovida ou

mesmo discutida. Esse modo de desenvolver sistemas, sem reusar software, permanece com o estudante ao iniciar a carreira profissional;

b) engenheiro de software novato tem grande prazer intelectual em resolver problemas que já foram parcialmente ou totalmente resolvidos, a fim de achar uma solução melhor;
c) as experiências malsucedidas com a reusabilidade são um motivo para o engenheiro de software não apoiar a reutilização de software;
d) não há recompensas pelo desenvolvimento de componentes reusáveis. Tomando como exemplo a implementação propriamente dita do software, normalmente as empresas medem a produtividade dos programadores pela quantidade de linhas de código novo que é escrito.

As dificuldades que podem ocorrer antes mesmo que o programa inicie são chamados por [KUT99] de Inibidores do Reuso.

Para [RAM98], os inibidores da utilização do reuso envolvem questões técnicas, organizacionais, políticas e psicológicas e são atribuídos aos programadores e aos administradores de software.

As principais dificuldades relatadas por programadores no momento de reutilizar código e projetos reusáveis são, conforme [KUT99]:

a) programador considera mais fácil escrever seus próprios códigos do que entender readaptar códigos criados por outras pessoas;
b) programador resiste em utilizar códigos reusáveis pois sente a segurança do seu emprego ameaçada;
c) programador reclama da falta de ferramentas que o ajudem localizar componentes ou sistemas que possuam partes reusáveis;
d) programador reclama da inexistência de métodos de desenvolvimento de software que enfatizam códigos reusáveis, geralmente, ele se sente só na hora de reusar projetos e especificações;
e) programador briga com o analista de sistema pois este não especifica as partes que podem ser reusáveis;
f) faltam explicações sobre vantagens e técnicas de reuso nos cursos que preparam programadores;

g) outras reclamações dos programadores são: tamanho, velocidade, falta de padrão, diferença de linguagem que os códigos reusáveis possuem.

As reclamações técnicas são a respeito da falta de componentes ou bibliotecas reusáveis bem definidas, úteis e confiáveis e a falta de um ambiente integrado. As reclamações de caráter psicológico ocorrem quando os programadores resistem em utilizar os componentes reusáveis que foram construídos por outra pessoa, pois isso lhes causa insegurança.

O fracasso do programa de reutilização pode acontecer quando os administradores de software deixam de utilizar os componentes reutilizáveis. [KUT99] descrevem os seguintes motivos que podem fazer isso acontecer:

a) a existência de ferramentas que tornem os programadores mais produtivos permitindo que os projetos sejam realizados com um número menor de pessoas. Isso aumenta o risco do administrador de software perder o seu emprego;
b) a reutilização de componentes de software criados por outras pessoas pode trazer problemas legais. Por exemplo, se um problema aparece num componente que foi reutilizado, de quem é a responsabilidade ?
c) gasta-se tempo e dinheiro no ajuste de componentes de software que serão reusáveis;
d) durante a produção de um software, criar componentes reusáveis exige tempo e aumento de custo que não fazem parte do orçamento do projeto.

Alguns obstáculos que ocorrem na implementação do programa de reuso, segundo [RAM98], são:

a) pânico criado pela falta de compreensão sobre porque se deve praticar a reusabilidade;
b) a falta de compromisso a longo prazo;
c) a falta de apoio da administração da empresa;
d) a pouca disposição para mudar o atual funcionamento;
e) a prática de reusabilidade sem um prévio treinamento;
f) nenhum apoio à metodologia da reusabilidade;
g) a realização de experimentos utilizando a reusabilidade;

h) a cultura incorporada pela política de desenvolvimento de sistemas da empresa, principalmente se a mesma exercer programas de “recompensa” aos funcionários;
i) a administração da empresa não convencida do valor empresarial da reusabilidade;
j) a não existência de ferramentas de apoio à prática do reuso de software;
k) a não existência de algum componente para ser reutilizado, ou seja, a não existência de uma biblioteca com componentes a serem reutilizados.

As dificuldades com a reusabilidade são responsáveis pelo fracasso do programa de reuso e de acordo com [YOU95] estão relacionadas com falhas gerenciais, falhas técnicas ou a combinação de ambas. As causas mais comuns são:

a) recursos ou investimento inadequados;
b) não criar um grupo separado para criar componentes de software reusáveis;
c) não recompensar os engenheiros de software pela reusabilidade;
d) controle de configuração inadequado;
e) mecanismos de procura/consulta/pesquisa inadequados;
f) muito pouco controle sobre os componentes que são colocados na biblioteca;
g) incluir características técnicas que sejam grandes demais e/ou tenham muitos efeitos colaterais;
h) interfaces não documentadas;
i) nenhuma facilidade para exceções e sobreposições.

Alguns problemas que podem ocorrer durante a execução do programa de reuso na empresa, conforme visto em [MCC92], são:

a) dificuldade de encontrar componentes comuns em software já prontos;
b) falta de padronização nos programas;
c) dependência de linguagens de programação;
d) decidir que componentes devem fazer parte da biblioteca;
e) entender os requerimentos internos de um componente;
f) entender o que uma alteração em um determinado componente pode causar;
g) descrever e classificar os componentes de software;
h) falta de suporte a reusabilidade;
i) aumento (em média de 25 %) de custo para desenvolver um componente de software projetando-o para a reusabilidade;

j) desestímulo causado pelos benefícios da reusabilidade serem de longo prazo;
k) a falta de prática de readaptação de componentes já existentes à reutilização.

[GRA97] classifica os problemas em três grupos:

a) problemas de população: como obter componentes reusáveis de bibliotecas ?
b) problemas de carga: como obter componentes apropriados das bibliotecas ?
c) problemas de construção: como construir novas aplicações com os componentes encontrados ?

2.6.2 SOLUÇÕES ÀS DIFICULDADES NO REUSO

[KUT99] apresenta algumas tecnologias para solucionar alguns problemas, conforme exposto na Tabela 3.

De acordo com [VER99], são aspectos fundamentais para a adoção da reutilização:

a) a criação de um repositório contendo os componentes de software para a reutilização;
b) inclusão de atividades relativas ao reaproveitamento de software na metodologia de desenvolvimento de sistemas estabelecida;
c) seleção de ferramentas automatizadas para suportar o desenvolvimento baseado na reutilização de componentes de software;
d) compromisso gerencial e aceitação dos profissionais de sistemas na utilização dessa prática.

Segundo [RAM98], um dos fatores de sucesso da implantação da reutilização é o comprometimento da administração. A administração da empresa deve estar ciente de que a reusabilidade exige que haja na empresa uma mudança na cultura e no comportamento da mesma no que se refere a desenvolver softwares e que os resultados do programa de reuso não são imediatos. Conforme [RAM98], os benefícios da reusabilidade em uma empresa, em geral levam de um a quatro anos para serem percebidos.

Tabela 3 - Soluções Tecnológicas aos Inibidores do Reuso

Inibidores de Reuso	Tecnologia
Falta de Componentes reusáveis	Análise de Domínio aquisição de conhecimento/representação padronização de construtores componentes com domínio específico multimídia
Custo para construir componentes reusáveis	ambiente de reengenharia linguagens de prototipação geradores de aplicação
Falta de Qualidade dos componentes	interface de especificação formais verificação de implementação terminologia padrão
Dificuldade de modificar componentes reusáveis	parametrização poliformismo
componentes em geral são ineficientes	sistemas operacionais linguagens extensíveis/compiladores
dificuldade para integrar /compor componentes	linguagens de prototipação captura de conhecimento/manipulação arquiteturas em camadas ferramentas de integração verificação
Possível violação de direitos autorais	criptografia autenticação segurança de software

Fonte: [KUT99]

Alguns requisitos básicos devem ser seguidos pela empresa para que seja obtido sucesso na implantação do programa de reuso, conforme [KUT99] descreve:

a) conseguir suporte da administração. A administração deve conhecer os benefícios que podem se adquiridos com a implantação do reuso bem como os problemas que podem vir a acontecer. É fundamental que administração apoie as mudanças que irão ocorrer. Alguns procedimentos para esclarecer a administração são sugeridos:
   - mostrar casos bem sucedidos da implantação do reuso em outras empresas;

- apresentar uma análise dos benefícios a longo prazo;
- avaliar se a empresa poderá continuar competitiva sem utilizar o reuso.

b) modificar o processo de desenvolvimento. O processo de desenvolvimento deve ser modificado para que incorpore o processo de reuso;
c) sobrepor inibidores não técnicos. Alguns problemas culturais devem ser combatidos através da motivação da prática do reuso;
d) criar programas de incentivos. Um programa que incentive o desenvolvedor a criar e a usar componentes reusáveis mostrou-se ser um grande fator do sucesso do reuso;
e) estabelecer métricas do reuso. A quantidade de reuso de uma organização deve ser conhecido (medido) para saber se o processo está causando impacto;
f) desenvolver uma linha guia. A transição para uma nova tecnologia é complicada e deve sempre ser acompanhada por treinamentos. A empresa deve orientar seus desenvolvedores através de linhas guias e padrões;
g) focalizar um único domínio. A empresa deve definir se deseja concentrar-se em componentes reusáveis em vários domínios ou num único domínio. Segundo [KUT99], as empresas conseguiram taxas de reuso muito mais altas quando focalizaram um único domínio.

Conforme [RAM98], o programa de reutilização deve abordar desde a introdução da reutilização na empresa, até a fase de avaliação dos projetos desenvolvidos.

Para os problemas expostos por [GRA97], as seguintes soluções são propostas:

a) para o problema de população: análise de domínio, bibliotecas de construção, geração de componentes pelo domínio de linguagens;
b) para o problema de carga: esquemas de classificação;
c) para o problema de construção: linguagens com propriedades genéricas, porém não há escopo de semântica da aplicação.

3 ESPECIFICAÇÃO ESTRUTURADA

De acordo com [ROC87] especificações de sistema são representações de um software, que descrevem um sistema desde uma visão macro até uma visão detalhada. Especificações de sistema tem quatro funções:

a) ser base para o desenvolvimento;
b) permitir o controle da qualidade do produto;
c) estabelecer a comunicação entre o pessoal envolvido no projeto;
d) auxiliar no entendimento do problema a ser resolvido.

Existem alguns enfoques para especificações, conforme [ROC87] relata. A metodologia mais utilizada é a Análise Estruturada.

Segundo [PRE95], a Análise Estruturada é um método de análise de requisitos de software que surgiu entre o final da década de 1960 e início da década de 1970. Nesta época, surgiram os primeiros trabalhos de modelagem da análise. Foi Tom DeMarco que popularizou o termo "Análise Estruturada" quando apontou os problemas e falhas da fase de análise de sistemas e estabeleceu as principais metas de um método de análise. Neste momento, também foram propostos por Tom DeMarco símbolos gráficos que possibilitariam ao analista criar modelos de fluxo de informação, sugeriu uma heurística para o uso desses símbolos, sugeriu que um dicionário de dados e narrativas de processamento pudesse ser usado como complemento aos modelos de fluxo de informação e apresentou numerosos exemplos que ilustravam o uso desse novo método. Algumas deficiências desta técnica de análise de sistemas foram apontadas e novas extensões foram introduzidas.

De acordo com [FIS90], a Análise Estruturada é um simples conjunto de ferramentas cujo objetivo é produzir uma especificação estruturada, que é um tipo de documento dos requisitos funcionais.

No trabalho de [ROC87] descreve-se a Análise Estruturada como um conjunto de técnicas e ferramentas cujo objetivo é auxiliar na análise e definição de sistemas, através da construção de modelos utilizando técnicas gráficas.

De acordo com [FIS90], as especificações estruturadas, construídas segundo as técnicas da Análise Estruturada, dão ao desenvolvedor de software e ao usuário final um veículo comum de comunicação.

A análise estruturada, ao contrário de outras metodologias, concentra-se na modelagem de dados e no fluxo de informações.

A especificação estruturada é classificada, segundo [MEN98] como uma especificação semi-formal, ou "formatada" [ROC87], pois a especificação estruturada utiliza uma linguagem que não tem uma semântica bem precisa. Isto pode ser considerado como um problema, pois uma especificação semi-formal possibilita a inclusão de erros. Além disso [MEN89] aponta que a ausência de formalidade dificulta o relacionamento da especificação com a implementação, uma vez que para fazer-se este relacionamento os elementos da linguagens de especificação e implementação têm que ser bem conhecidos. Por outro lado, [MEN89] comenta que sobre as especificações formais ainda há muito a ser realizado, em pesquisa e educação, até que estes métodos formais sejam bem aceitos e além disso especificações semi-formais são muito mais fáceis de entender que especificações formais. Para [MEN89], tanto as linguagens semi-formais, como as formais, têm seu campo de aplicação mais adequado, sendo que o uso de especificações formais é mais recomendado quando a inexistência de erros for uma necessidade. De acordo com [PRE95], os métodos semi-formais, especialmente a Análise Estruturada, são os mais pesquisados e utilizados nas empresas de todo o mundo, pois requerem menos esforços que as modelos formais.

A Análise Estruturada visa resolver alguns dos problemas existentes na área de desenvolvimento de sistema, conforme [MEN89] relata, atendendo aos seguintes objetivos:

a) facilitar a comunicação entre usuários e analistas;
b) permitir a visualização do global e do detalhe separadamente;
c) servir de base ao projeto físico.

Com base nestes objetivos a Análise Estruturada foi desenvolvida com as seguintes características:

a) utiliza recursos gráficos, o que facilita a compreensão da especificação;
b) é concisa, ou seja, permite a exposição das idéias de forma resumida e precisa;

c) utiliza a abordagem *top-down*, permitindo que o sistema a ser especificado seja particionado em parte tão independentes quando possível, o que facilita o desenvolvimento e o entendimento da especificação;
d) gera uma especificação em alto nível, que apresenta as funções do sistema sem a representação de como o sistema será implementado para uma máquina particular.

Baseado em alguns comentários de pessoas que utilizaram a Análise Estruturada, [MEN89] relaciona as suas conclusões a respeito dos modelos gerados por esta técnica:

a) a descrição imprecisa da Análise Estruturada dá margem a interpretações diversas. E, por este motivo, não se pode garantir que a especificação esteja correta como também é difícil a sua utilização sem a assistência de um usuário experiente. No entanto, uma vez compreendido, este método não é difícil de ser utilizado. A facilidade de leitura e escrita da especificação permite que o usuário possa compreender a especificação, aumentando as chances de que ela esteja de acordo com os seus requisitos;
b) apesar das linguagens utilizadas pela Análise Estruturada não serem formais, apresentam recursos que permitem a verificação de que a especificação está completa ou que suas informações estão coerentes entre si. Entretanto este processo é muito trabalhoso quando realizado sem apoio automatizado.

O desenvolvimento de software de boa qualidade depende da qualidade de suas especificações, pois [RAC87] afirma que muitas vezes os problemas com sistemas estão relacionados a erros e inadequações em suas especificações.

3.1 MODELOS

A Análise Estruturada expõe o que é feito e o que vai ser feito através do uso de gráficos, o que torna a visualização e entendimento muito mais claros e objetivos [JUN97].

Conforme [GAN90], o objetivo da modelagem é fornecer aos usuários e projetistas de um sistema uma forma razoavelmente rápida de expressar, trocar e refinar suas idéias iniciais sobre o alcance e o conteúdo do sistema, usando diagramas para mostrar os dados, os processos (funções) e suas inter-relações. Os principais modelos utilizados são:

a) diagramas de fluxo de dados (DFD) que mostram os processos, os depósitos de dados e os fluxos de dados que entram, que contornam e saem do sistema;
b) dicionário de dados, que mostram o conteúdo de cada componente do DFD;
c) modelos entidade-relacionamento (MER) que mostram as entidades de dados no sistema e a natureza de suas associações.

3.1.1 DIAGRAMA DE FLUXO DE DADOS

Um diagrama de fluxo de dados, ou DFD, é uma ferramenta que descreve o fluxo de informação e as transformações que são aplicadas à medida que os dados se movimentam no sistema.

O DFD é descrito por [GAN90] como uma ferramenta cujo objetivo é mostrar um sistema ou parte de um sistema ao setor comercial, de onde vêm os dados, para onde os dados vão quando deixam o sistema, onde eles são armazenados, que processo os transforma e as interações entre os depósitos de dados e os processos.

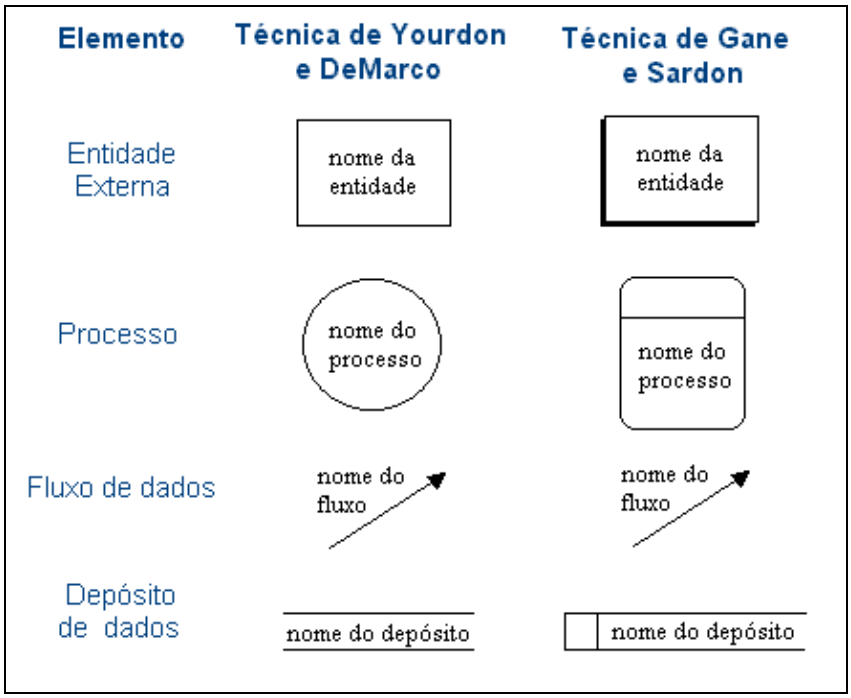
O DFD é um diagrama composto por quatro elementos, conforme apresenta [PRE95], [GAN90] e [YOU95]:

a) entidade externa: é um componente que o sistema se comunica. É um produtor ou consumidor de informações que fica fora dos limites do sistema a ser modelado. Pode ser, um hardware, uma pessoa, um grupo de pessoas, uma empresa ou até mesmo um outro sistema, que introduza informações no sistema ou que receba informações transformadas pelo sistema. As entidades externas também são conhecidas por fontes, coletores de dados ou terminadores;
b) processo: é um transformador de informações que fica dentro dos limites do sistema a ser modelado. Uma ou mais entidades externas ou depósitos de dados fornecem dados para um processo e este processo modifica os dados e os retorna transformados para uma ou mais entidades externas ou depósitos de dados. Geralmente o nome do processo é a descrição da função do processo, como por exemplo “gerar transações de vendas”;
c) depósito de dados: é um repositório de dados onde são armazenados os dados para serem usados em um ou mais processos. Normalmente é um arquivo ou uma tabela de um banco de dados;

d) fluxo de dados: é um item de dado ou coleção de itens de dados. O fluxo de dados é utilizado para mostrar o movimento de dados de um ponto do sistema ao outro. O símbolo utilizado para representar um fluxo de dados é uma seta. O sentido da seta indica da onde a informação está vindo e para onde está indo. Uma descrição breve e significativa deve ser colocada ao longo da seta.

Conforme [GAN90], existem duas técnicas principais utilizadas para representar um diagrama de fluxo de dados, porém muito parecidas: a técnica de Gane e Sardon, e a técnica de Yourdon e DeMarco.

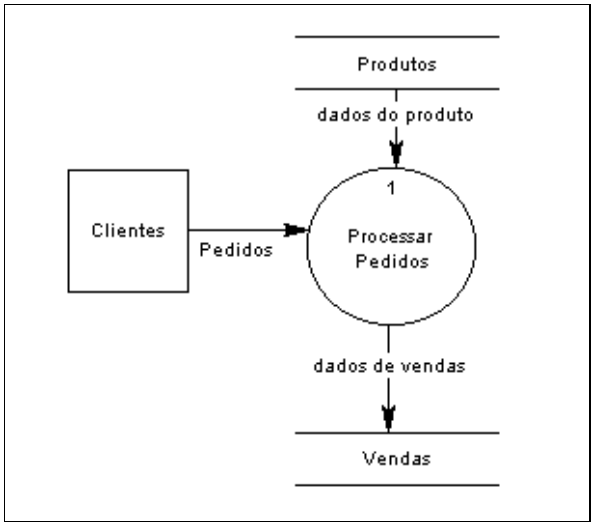
A simbologia utilizada para representar estes elementos no modelo de Gane e Sardon e no modelo de Yourdon e DeMarco estão representadas na Figura 4:



Fonte: [GAN90]

Figura 4 - Elementos do DFD

A Figura 5 exemplifica o diagrama de fluxo de dados apresentando um subsistema onde uma série de pedidos é feita pelo cliente e o processo “Processar pedidos”, usando informações do depósito de dados “Produtos”, encarrega-se de atualizar as informações de venda no depósito de dados chamado “Vendas”.



Fonte: [GAN90]

Figura 5 - Exemplo de DFD

As características do DFD descritas por [GAN90] são:

a) DFD estabelece os limites da área do sistema e da área comercial coberta pelo sistema. As coisas que são representada pelo símbolo de entidade externa estão, por definição, fora do sistema. Os processos que não aparecem no DFD não fazem parte do projeto;
b) DFD é não-técnico. Não há nada no DFD que não seja facilmente entendido pelo pessoal da área comercial, familiarizado com a área comercial enquadrada, tenham ele ou não qualquer conhecimento sobre computadores;
c) DFD mostra tanto os dados armazenados no sistema quanto os processos que transformam aqueles dados. Ele mostra a relação entre os dados no sistema e os processos no sistema;
d) DFD não contém o elemento tempo, isto é, não há indicação de seqüência no processamento.

[GAN90] aconselha os analistas de sistema a não denotar erros ou exceções do sistema ao desenhar os DFDs, a não ser que sejam muito importantes para os usuários.

Uma descrição de um ou dois parágrafos para cada processo pode ser muito útil como uma fonte de referência de fácil acesso quando se está fazendo um DFD, conforme [GAN90] explica.

A técnica de Yourdon/DeMarco sugere que a diagramação de um sistema deve começar com um "diagrama de contexto" que simplesmente mostre as entidades externas e as entradas e saídas do sistema ou área comercial. Depois o diagrama de contexto pode ser expandido para mostrar o subsistemas existentes (aqui chamado de diagrama "nível zero") e cada subsistema poderá ser expandido também até que os detalhes da lógica do processo possam ser escritos em uma página, conforme [GAN90] relata. Normalmente um diagrama de fluxo de dados que abrange um sistema completo conterá três níveis, isto é, níveis zero, 1 e 2; mas poderá chegar a ter mais dois níveis se o sistema for muito complexo. Ainda é recomendado não incluir mais que sete processos em cada diagrama. É por este motivo que [PRE95] diz que o diagrama de fluxo de dados pode ser usado para representar um sistema ou software em qualquer nível de abstração.

[RAC87] relaciona os passos para se construir um diagrama de fluxo de dados:

a) identificar as entidades externas envolvidas;
b) identificar as entradas e saídas esperadas (procurar descobrir grupamentos lógicos de entradas e saídas. Marcar as entradas e saídas associadas a condições de erros);
c) identificar as consultas e pedidos de informações que podem surgir. Especificar um fluxo de dados que defina a informação dada ao sistema e um fluxo de dados que defina o que se deseja do sistema;
d) desenhar o primeiro esboço do diagrama;
e) verificar se todas as entradas e saídas listadas foram incluídas (exceto aquelas que tratam de erros e exceções);
f) produzir um segundo esboço mais claro;
g) rever o segundo esboço com um representante do usuário. Anotar qualquer mudança resultante da revisão;
h) produzir uma expansão de nível inferior para cada processo definido no segundo esboço. Resolver o tratamento dos erros;
i) fazer a terceira e última revisão do diagrama.

Algumas diretrizes são mencionadas por [YOU90] afim de promover a construção de DFDs com sucesso:

a) escolher nomes significativos para os componentes do DFD. Um bom nome para nomes de processos é utilizar um verbo e um objeto. Isto é, empregar um verbo que

represente ação (um verbo transitivo, que exige um objeto) e um objeto adequado formando uma sentença descritiva do processo. Por exemplo: Calcular Trajetória do Míssil, Manter Clientes etc. Devem ser evitados verbos cujo sentido possa abranger tarefas não determinadas, como por exemplo “fazer serviço”, “manipular entrada”, “processar dados” etc. Além disso, não somente os processos mas também os outros componentes do DFD devem utilizar um vocabulário conhecido pelo usuário;

b) numerar os processos. De acordo com [YOU90] é mais fácil referenciar os processos quando ele estão numerados. Por exemplo, é mais fácil em uma discussão sobre um DFD dizer “o processo 1” ao invés de “o processo Emitir Relatório de Livros Vendidos”;

c) evitar DFD complexos demais. Conforme já visto anteriormente, o DFD pode ser dividido em vários níveis, cada nível contendo sete processos. Isto evita a criação diagramas complexos e difíceis de entender.

3.1.2 DICIONÁRIO DE DADOS

Conforme [GAN90], o dicionário de dados é a ferramenta utilizada para documentar o DFD. Através do DFD, especifica-se os elementos de dados que estão sendo mantidos e os dados que estão sendo transportados em cada fluxo de dados.

De acordo com [RAC87], conforme for ampliado os detalhes dos conteúdos dos fluxo de dados, processos e depósito de dados, um dicionário de dados torna-se necessário para manter estes dados.

Para [FIS90], o dicionário de dados consiste em catálogos, ou bancos de dados, que contém a definição de todos os elementos do diagrama de fluxo de dados. Por isso, num dicionário de dados encontra-se todos os atributos de qualquer dado.

O dicionário de dados é descrito por [PRE95] como uma listagem organizada de todos os elementos de dados que são pertinentes ao sistema, com definições precisas e rigorosas, de forma que tanto o usuário como o analista de sistemas tenham uma compreensão comum das entradas, das saídas, dos componentes dos depósitos de dados e até mesmo dos cálculos intermediários. O autor também cita algumas informações que podem estar contidas no

dicionário de dados, tais como nome, onde é usado, como é usado, tipos de dados, restrições ou limitações. Além destas informações, [RAC87] também sugere a inclusão de um exemplo.

Os trabalhos de [GAN90] e [PRE95] descrevem a notação utilizada para escrever um dicionário de dados, conforme pode ser visto na tabela 4.

Tabela 4 - Notação utilizada no Dicionário de Dados

Construção de dados	Notação	Significado
	=	é composto de
Seqüência	+	e
Seleção	[|]	ou .. ou
Repetição	{ }	repetições de
	( )	dados opcionais
	* *	comentários

Fonte: [PRE95]

A construção de dados pode ser:

a) seqüência: uma seqüência de itens de dados;

b) seleção: uma seleção entre os itens de dados;

c) agrupamento: um agrupamento repetido de itens de dados.

Para demonstrar o uso da notação, um fluxo denominado “fatura” foi definido como mostra a Figura 6.

```
Fatura =  Número da Fatura +
          Nome do Cliente +
          Endereço do Cliente +
          Data da Fatura +
          (Instruções Especiais) +
          {Código do Produto + Quantidade + Preço Unitário + Custo do Item}
```

Figura 6 - Exemplo de Dicionário de Dados

3.1.3 MODELO ENTIDADE-RELACIONAMENTO

Segundo [HUG98], o Modelo Entidade-Relacionamento é um diagrama utilizado para representar o modelo conceitual de dados de um sistema, isto é, [JUN97] apresentar uma visão dos dados sob o ponto de vista da organização. O Modelo Entidade-Relacionamento, ou MER, é portanto, um diagrama que se concentra apenas nas entidades de dados [GAN90], detalhando as associações existentes entre elas [HUG98].

De acordo com [BAR94], um Modelo Entidade-Relacionamento é composto pelos seguintes elementos:

a) entidade: são componentes reais ou abstratos que a empresa utiliza e se interessa em armazenar dados sobre eles. Para [YOU90], uma entidade de dados representa uma coleção ou um conjunto de objetos (coisas) do mundo real, materiais ou imateriais. Por exemplo: clientes, fornecedores, peças, produtos. Um entidade de dados também pode ser usada para representar acontecimentos, como por exemplo um cliente fazendo um pedido de um produto.
b) relacionamento: é uma associação entre duas entidades. Um relacionamento representa um conjunto de conexões entre objetos.
c) atributo: Um atributo caracteriza ou qualifica uma determinada propriedade de uma entidade. Conforme [JUN97], O atributo é um dos itens de dados que é armazenado sobre uma entidade. Por exemplo, "data de nascimento" poderia ser um atributo de uma entidade "cliente".

Existem também outros termos, conforme [HUG98] descreve:

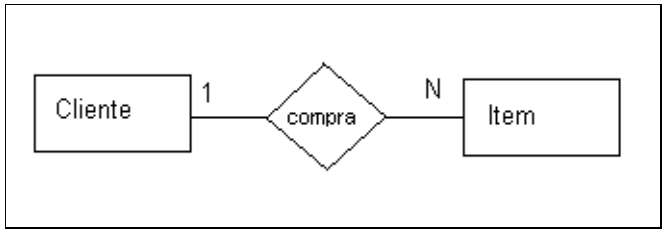
a) ocorrência: é um conjunto de atributos de uma determinada Entidade. Num banco de dados uma ocorrência é representado por um registro de dados;
b) identificador ou Atributo Determinante. Corresponde a um ou mais atributos que identificam unicamente uma ocorrência de uma entidade. De acordo com [YOU90] cada elemento de uma entidade de dados só pode ser identificado de uma única forma. Isto é, existe um modo de diferenciar as instâncias individuais da entidade. Por exemplo, numa entidade chamada Cliente, cada elemento da entidade poderia ser identificada pelo número do CPF;
c) grau de relacionamento. o grau de relacionamento identifica o número de entidades que participam de um relacionamento. De acordo com [BAR94], quando duas

entidades participam do relacionamento o grau é binário, quando há três entidades que participam de um relacionamento o grau é dito ternário;

d) classe de relacionamento ou cardinalidade: identifica o tipo de relacionamento entre duas entidades, que pode ser um-para-um (1:1), um-para-muitos (1:N) e muitos-para-muitos (M:N).

Normalmente a cardinalidade é definida, conforme [BAR94], fixando-se uma ocorrência de uma entidade e perguntando-se quantas ocorrências da outra poderão existir (se uma somente ou se "n" possíveis). O caso M:N é identificado como sendo dois relacionamentos 1:N, um em cada sentido do relacionamento.

Existem algumas notações utilizadas para desenhar um MER, uma das mais conhecidas é a de Peter Chen [BAR94], como mostra o exemplo na Figura 7.



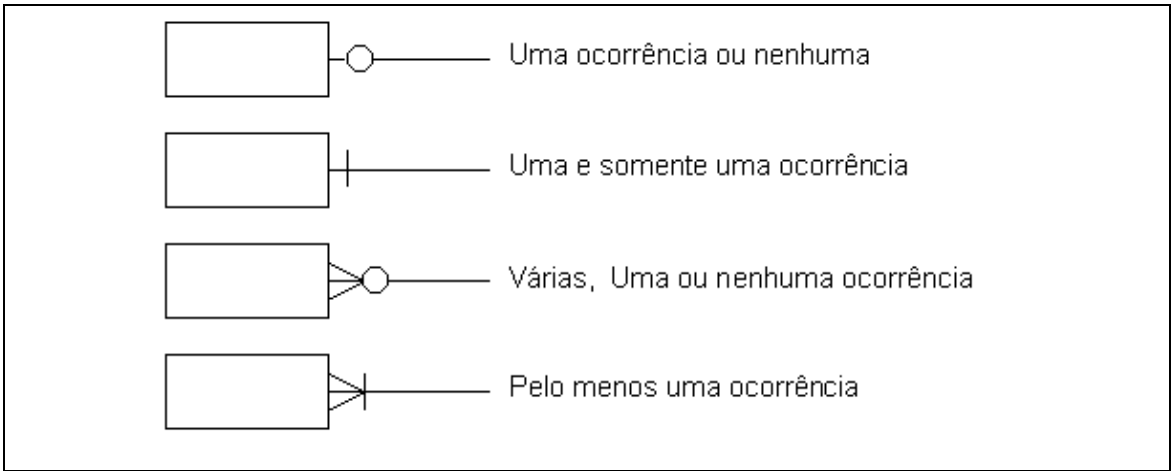
Fonte: [BAR94]

Figura 7 - Notação do MER (de Peter Chen)

Um quadrado ou retângulo é utilizado para representar uma entidade enquanto um losango representa o relacionamento entre as entidades. A cardinalidade pode ser vista próximo ao relacionamento. No exemplo da Figura 7, o relacionamento é de 1:N.

Um outro aspecto importante na hora de representar um relacionamento é se o relacionamento é obrigatório ou opcional. Conforme [GAN90], as entidades podem estar relacionadas, mas não em todos os casos, por exemplo: entidade "Funcionários" e entidade "Projetos". Um funcionário pode estar designado para um projeto, ou vários, ou nenhum; da mesma forma, um projeto pode estar autorizado, mas não haver ainda nenhum funcionário designado para ele.

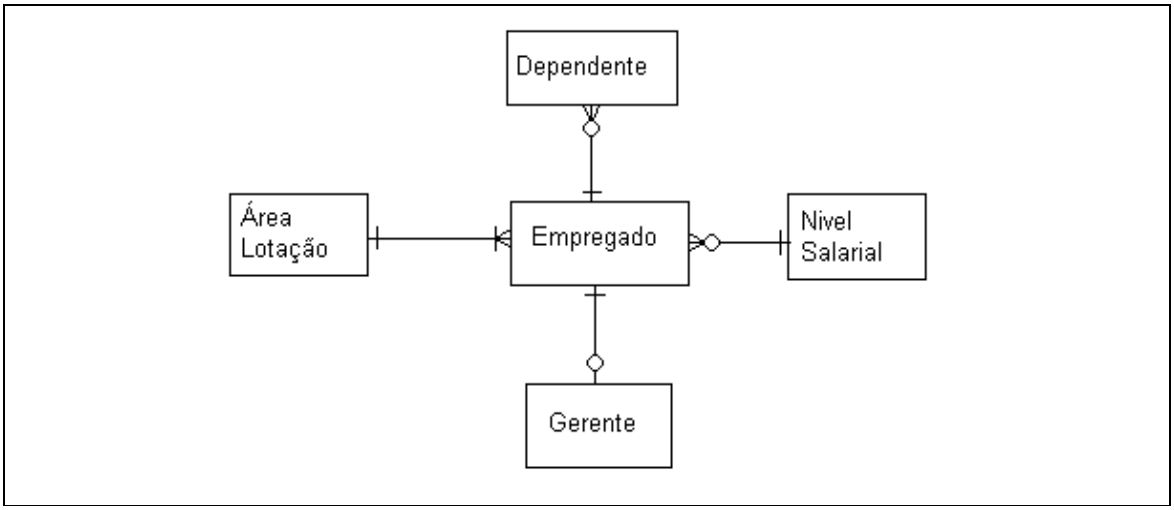
Uma outra representação do modelo entidade-relacionamento é apresentada por [JUN97] .Esta notação aborda a obrigatoriedade ou não das entidades e pode ser vista na Figura 8.



Fonte: [JUN97]

Figura 8 - Notação usada para desenhar o MER

Um exemplo do uso desta notação pode ser visto na Figura 9



Fonte: [JUN97]

Figura 9 - Exemplo de MER

Deste exemplo pode-se descrever que:

a) uma “Área Lotação” tem obrigatoriamente pelo menos 1 “Empregado”;
b) um “Empregado” está vinculado obrigatoriamente a uma “Área Lotação”;
c) um “Empregado” pode ter vários, um ou nenhum “Dependente”;
d) um “Dependente” (se existir) está obrigatoriamente vinculado a um “Empregado”;
e) um “Empregado” pode ser “Gerente”;
f) um “Gerente” é um “Empregado”;

g) um “Empregado” tem obrigatoriamente um “Nível Salarial”;

h) em um mesmo “Nível Salarial” pode-se ter vários, um ou nenhum “Empregado”.

4 ESTRATÉGIAS PARA REUSO DE ESPECIFICAÇÕES

De acordo com [HAL92], a maior causa de erros no desenvolvimento de software pode ser localizada na análise de requisitos. Além disso, a fase de análise de requisitos é conhecida como sendo uma das partes mais difíceis da engenharia de software. De acordo com [HAL92], o reuso de especificações de software pode ser uma solução para estes problemas.

A reutilização de especificações tem como objetivo minimizar ou evitar os problemas encontrados na fase de análise de requisitos através do aproveitamento de especificações de sistemas existentes.

Em seu trabalho, [MAI92] afirma que o sucesso no reuso de especificações pode auxiliar o processo de requerimentos de software possibilitando a construção de especificações mais completas, consistentes e claras.

Para [SPA99], o reuso de especificações pode levar a um aumento significativo de produtividade no desenvolvimento de software porque ele pode melhorar a perfeição e precisão de especificações, reduzindo a probabilidade de erros, reduzindo tempo e custo do projeto do software.

Segundo [MEN89] a reutilização de especificações é mais efetiva que a reutilização de código, pelo fato do grau de abstração das especificações ser maior do que o do código.

Já [YOU95] descreve que a reusabilidade de especificações permite eliminar completamente o esforço envolvido em projetar, codificar e testar uma implementação de um especificação obtida através do reuso.

De acordo com [MAI92], benefícios potenciais podem ser alcançados com o reuso bem sucedido de especificações:

a) pode ajudar a superar as dificuldades encontradas por engenheiros de software inexperientes durante os estágios iniciais de desenvolvimento de software;
b) a reutilização pode reduzir a carga mental do engenheiro de software durante este período de definição do modelo;

c) permite melhorar a qualidade e produtividade dando um começo mais rápido durante a análise de requisitos;
d) pode enriquecer a base de conhecimento própria do engenheiro de software, fornecendo a experiência necessária para entender e resolver problemas análogos.

Como pode ser visto, o reuso de especificações é uma das formas de reuso de software mais interessantes de se implementar, porém é pouco discutida na literatura. [MEN89] diz que o reuso de especificação é uma abordagem que ainda está em fase inicial de pesquisa.

Segundo [HAL92] o processo de reuso de especificações compreende duas fases:

a) recuperação. Consiste em recuperar uma especificação adequada para ser reutilizada no desenvolvimento do novo sistema, isto é, consiste em buscar modelos similares ao novo sistema para serem utilizados na análise do novo sistema;
b) customização. Especificações prévias raramente satisfazem exatamente as necessidades de uma aplicação nova, consequentemente alguma alteração na especificação que será reutilizada será necessária. Um precursor essencial da customização é entender o que a especificação representa.

Na literatura foram encontrados duas técnicas que apoiam o reuso de especificações: DRACO e Analogia.

O DRACO é uma das técnicas utilizadas para reusar especificações comentadas por [MEN89]. Segundo a autora, o DRACO é exemplo de um sistema que combina técnicas de transformação de programas, administração de bibliotecas de componentes reusáveis e técnicas de Inteligência Artificial. O DRACO, segundo [FRE86], propõe construir linguagens de especificação de alto nível para serem utilizadas na construção de sistemas. No paradigma DRACO o elemento de reuso são linguagens formais denominadas de domínio. Estas linguagens são construídas com o intuito de encapsular os objetos e operadores de um determinado domínio, ou área de conhecimento. Maiores informações sobre este paradigma podem ser encontradas em [MEN89] e [FRE86].

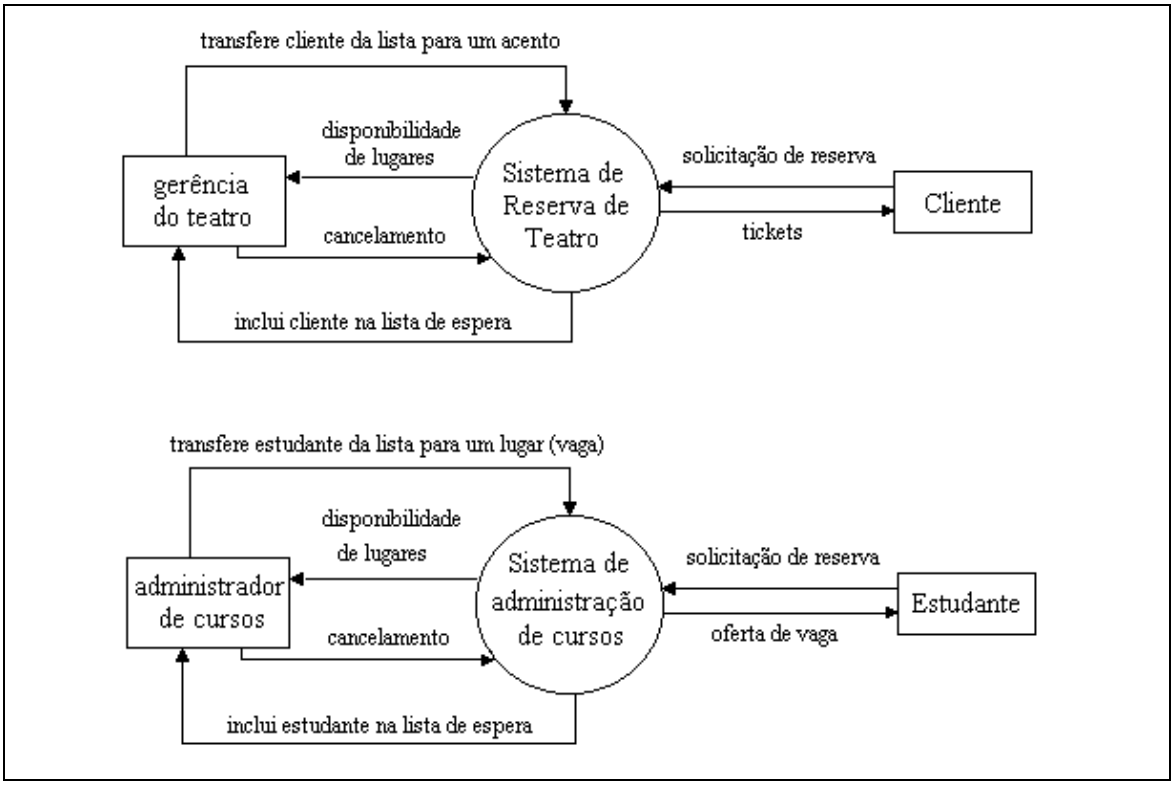
4.1 ANALOGIA

O reuso por Analogia, segundo [ROS94], tem sido inspirado no raciocínio humano de resolver novos problemas a partir do reuso de partes de problemas previamente resolvidos.

De acordo com [MAI92], a Analogia é um paradigma usado para reutilizar especificações de sistemas análogos, isto é, que compartilham conceitos similares. Para demostrar o potencial da reutilização, dois sistemas exemplos são citados por [MAI92]:

a) reservas para o Teatro. O sistema de reservas para o teatro permite que lugares sejam reservados para qualquer espetáculo. Pode-se reservar um lugar, ou blocos de lugares, e os lugares podem variar de preço. A gerência do teatro usa o sistema para responder a solicitações e gerenciar as reservas. Uma lista de espera é criada quando um espetáculo está lotado; quando há cancelamentos, pessoas da lista de espera são transferidas para os lugares liberados;
b) sistema de Administração de Cursos. O sistema de gerenciamento de cursos da universidade gerencia os cursos de período integral e meio período. O administrador dos cursos usa o sistema para responder as solicitações e o gerenciamento de vagas dos cursos. Aos candidatos são oferecidos lugares condicionais em cada curso, que possuem um número máximo de lugares. Uma lista de espera é usada para os estudantes que não conseguiram vaga imediatamente. Estes estudantes tem preferência nas vagas disponíveis por motivo de cancelamento.

Observando os DFD dos dois sistemas na Figura 10, pode-se verificar que ele são bastante similares.



Fonte: [MAI99]

Figura 10 - Exemplo de sistemas similares

Apesar dos sistemas serem de domínios diferentes os dois sistemas compartilham características semelhantes como “reservas”, “listas de espera” e “lugares”. Se o sistema de administração de cursos fosse implementado, ele poderia ser baseado no sistema de reservas de teatro, reusando tal especificação. Portanto, segundo [MAI92], a Analogia é um paradigma eficiente para reusar especificações entre sistemas que compartilham conceitos similares.

Segundo [ROS94], a biblioteca de componentes que auxiliará o processo de reuso de especificações poderá considerar sistemas completos ou apenas parte dele. [MAI92] também afirma que o reuso é possível em vários níveis de análise, entre detalhes estruturais e funcionais tais como o processo (no exemplo, os processos “reserva de assento do teatro” e “reserva de vaga no curso”), depósitos de dados (“assentos do teatro” e “vagas de curso”) e entidades externas (“gerência do teatro” e “administrador de cursos”).

Segundo [MAI92], para identificar a similaridade entre especificações devem ser avaliados o propósito e a função dos componentes.

De acordo com [MAI92] e [HAL92] uma biblioteca de especificações poderá conter especificações concretas e/ou especificações abstratas.

Especificações concretas são especificações de sistemas reais, enquanto especificações abstratas são especificações genéricas, isto é, especificações que não são de nenhum sistema em particular. As especificações abstratas existem apenas para serem reutilizadas em especificações concretas.

Uma estrutura semelhante às especificações abstratas, apresentado por [GAM95], é o *pattern*. Um *pattern* é definido pelo autor como uma descrição abstrata de um problema e sua solução. Por isso o *pattern* não representa uma situação concreta, sendo como um *template* que pode ser aplicado em muitas diferentes situações.

Nos estudos de [MAI92] foram identificados 13 abstrações de sistemas.

No trabalho de [ZIR97] é comentada a realização de um conjunto de testes para avaliar os aspectos de produtividade envolvidos no processo de elaboração de modelos de requisitos de sistemas através da analogia com modelos similares previamente construídos. Foram montados grupos compostos por estudantes e analistas experientes e foi dado a eles sistemas para serem resolvidos com e sem reutilização. Neste experimento chegou-se as seguintes conclusões:

a) os diagramas construídos mediante reutilização de modelos análogos são mais completos e corretos do que os diagramas construídos desde o início sem reutilização;
b) ganho de produtividade é maior quando o sistema a ser reutilizado tem um grau de similaridade maior. Os benefícios da reutilização caem drasticamente com a diminuição do grau de similaridade;
c) houve uma considerável redução de diversos tipos de erros na modelagem com reutilização;
d) houve diminuição do tempo para modelagem do sistema com reutilização, porém foi necessário primeiro ler e entender a especificação reutilizada. Nos testes de reuso de sistemas pouco similares o tempo para modelagem foi maior que na modelagem sem o reuso;

e) processo de especificação de sistema é mais prático quando baseado na reutilização.

Alguns experimentos foram realizados por [MAI92] sobre o comportamento dos engenheiros de software durante a compreensão da especificação e reutilização. Destes experimentos chegou-se às seguintes conclusões:

a) foi notado que os engenheiros de software inexperientes tiveram uma predisposição à preguiça, optando por copiar o modelo, sem o entendimento completo;
b) os engenheiros experientes, ao contrário dos inexperientes, evitaram copiar o modelo e se concentraram em entender os componentes análogos antes de reutilizá-los. Porém eles tiveram dificuldade em compreender a analogia por inteiro, ressaltando que alguma ajuda deve ser dada neste sentido;
c) reuso de especificações concretas mostraram-se mais proveitosas que as abstratas, uma vez que a analogia foi reconhecida. Engenheiros novatos, no entanto demonstraram a preguiça mental que a reutilização pode encorajar. [MAI92] novamente sugere que um suporte tutorial se faz necessário para encorajar a compreensão das analogias.

Para [MAI92], um ambiente de reutilização de especificação é necessário para prevenir a cópia de especificações, melhorar o entendimento da analogia e promover o desenvolvimento da especificação. A ferramenta deve conter um assistente, um módulo de explicação que deverá orientar e ensinar os engenheiros de software durante a reutilização conduzindo-os através das estratégias de reutilização que encorajam o entendimento e a transferência da analogia, auxiliado pelas descrições dos mapeamentos da especificação nova e a especificação reutilizada.

Segundo [MAI92], a reutilização de especificações é um processo que envolve a interação do engenheiro de software, tanto na fase de identificação de especificações reusáveis como na customização desta especificação.

De acordo com [MAI92], o engenheiro de software deverá ter uma mente aberta para validar a analogia através da reutilização pois as analogias poderão fornecer uma combinação

inexata com novos contextos e as técnicas tradicionais ainda serão requeridas para desenvolver a especificação de requisitos.

Durante a fase de identificação de especificações poderão se encontrados uma ou mais especificações análogas e o engenheiro de software será responsável por selecionar a especificação mais próxima e fazer as devidas adaptações, sempre que possível auxiliado pelo módulo de explicação.

A Analogia, segundo [HAL92], também pode suportar o reuso de sistemas com poucas características, isto é, que pertencem a domínios totalmente diferentes. Neste caso são sugeridos heurísticas para auxiliar no processo de identificação de especificação. As heurísticas também podem ser usadas para guiar o diálogo com o engenheiro de software a fim de extrair características sobre os novos domínios e executar procedimentos eficientes de procura e recuperação de especificações candidatas ao reuso a partir de um repositório. O autor também sugere utilizar neste caso um método de identificação de especificações melhor que os esquemas de classificação de componentes. Um método baseado na pesquisa de estruturas e propriedades de modelos conceituais são mais poderosas que esquemas de classificação que são limitados à disciplina dos desenvolvedores de software em utilizar termos padrões para classificação e subsequente recuperação.

[ROS94] sugere que os casos bem sucedidos de recuperação e os mal sucedidos fiquem armazenados para consultas posteriores.

4.2 TRABALHO CORRELATO

Um trabalho que implementava o reuso de especificações foi identificado por [MAI92]. Esta ferramenta, denominada IRA, aborda o uso de domínio e o reuso destes.

O protótipo IRA, conforme [MAI92], foi preenchido com dez abstrações e com suporte a trinta heurísticas para identificação de diferenças entre sistemas de domínios diferentes. Após a conclusão da ferramenta foi comprovado com testes a eficiência do protótipo. Estudos de usuários revelaram os benefícios e os problemas decorrentes do reuso, conforme descrito no tópico 4.2.

5 ESPECIFICAÇÃO DA FERRAMENTA

Para elaborar a especificação do protótipo utilizou-se a Análise Estruturada.

A Análise Estruturada é uma metodologia para modelagem de dados e fluxo de informações, que compreende o diagrama de fluxo de dados (DFD), dicionário de dados e modelo entidade-relacionamento (MER).

A ferramenta Power Designer DataArchitect e Power Designer Process Analist foram utilizadas para elaborar o MER e o DFD, respectivamente.

5.1 DIAGRAMA DE CONTEXTO

Segue na Figura 11 o Diagrama de Contexto da ferramenta.

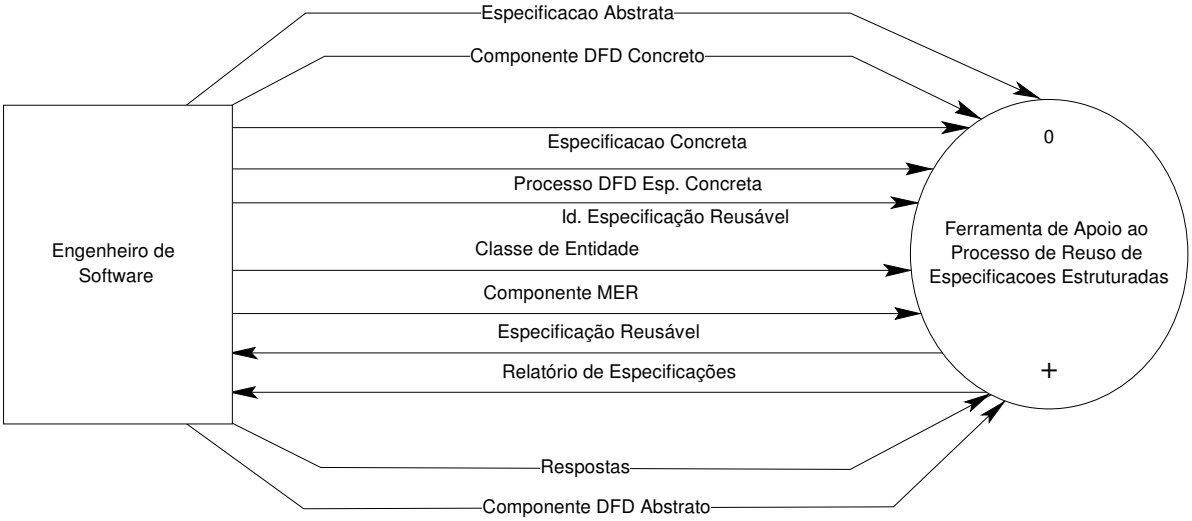


Figura 11 - Diagrama de Contexto da ferramenta

5.2 DFD NÍVEL 0

A Figura 12 apresenta o diagrama de fluxo de dados no nível 0.

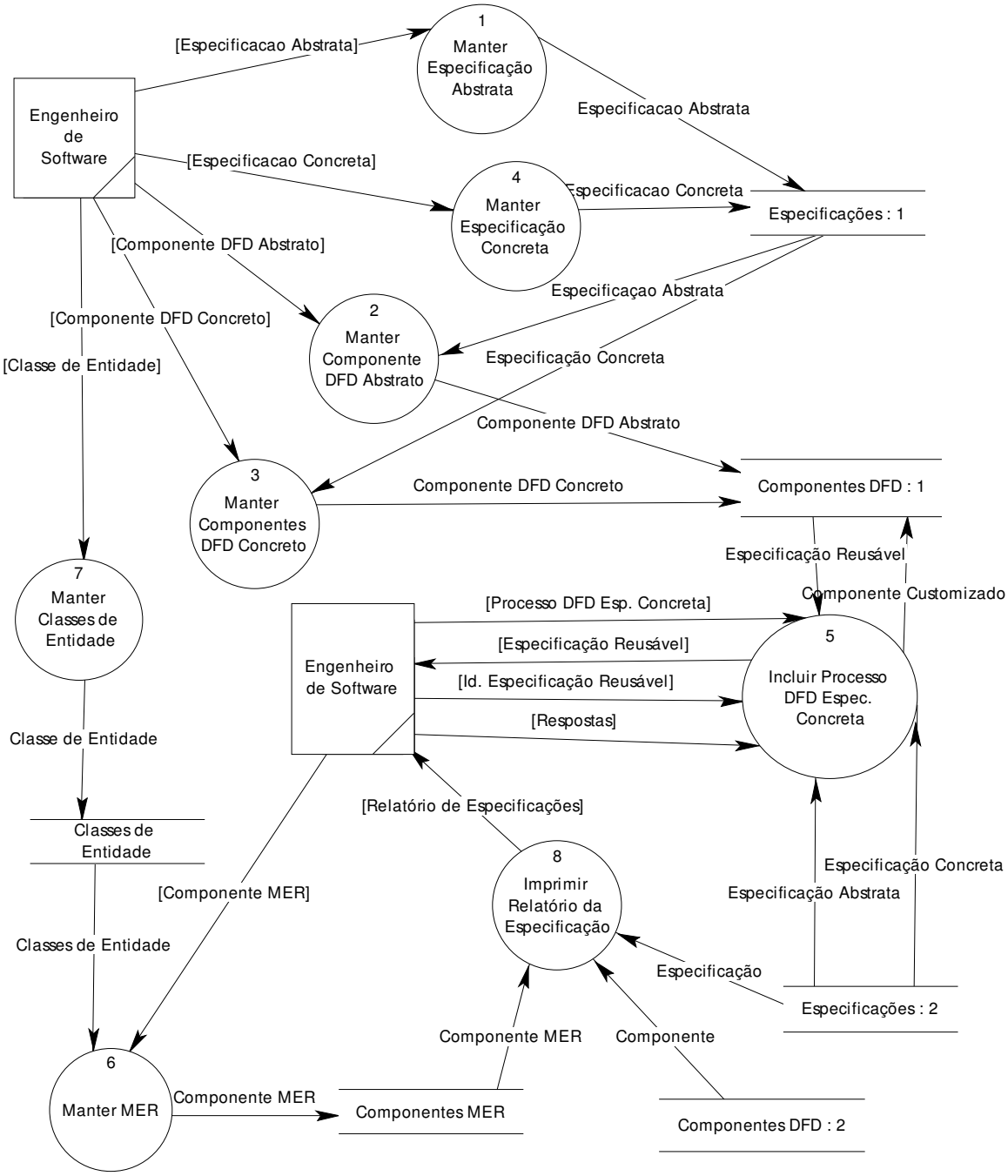


Figura 12 - DFD Nível 0

O Processo “Manter Especificação Abstrata” (Processo 1) é responsável pelo cadastramento das especificações abstratas. Neste processo apenas são definidos os parâmetros da especificação que são utilizados na identificação de componentes reusáveis. Concluindo o procedimento tem-se a especificação criada sem qualquer componente incluído.

Depois de definir os parâmetros da especificação no Processo 1, o usuário cadastra os componentes que compõem a especificação abstrata no Processo “Manter Componente DFD Abstrato” (Processo 2). Especificações abstratas são válidas apenas para o DFD.

As especificações concretas são mantidas no Processo “Manter Especificação Concreta” (Processo 4). Os componentes do DFD que compõem a especificação são mantidos pelo Processo “Manter Componentes DFD Concreto” (Processo 3), com exceção da inserção de componentes do tipo “Processo” que ocorre no Processo “Incluir Processo DFD Espec. Concreta” (Processo 5). O Processo 5 difere do Processo 3 porque esta tarefa exige que o protótipo pesquise, identifique e customize componentes de DFD reusáveis, ou seja, um procedimento diferente daqueles que ocorrem nos Processos 2 e 3.

No Processo “Manter MER” (Processo 6) são mantidos os componentes do MER. Neste procedimento poderão ser utilizados as Classes de Entidades, mantidos no Processo “Manter Classes de Entidade” (Processo 7).

O Processo “Imprimir Relatório da Especificação” (Processo 8) é responsável pela emissão do relatório da especificação.

5.3 DICIONÁRIO DE DADOS

Especificação Abstrata = Identificação +
Função +
{Operação} +
Descrição +
Nome

Especificação Concreta = Identificação +
Nome +
(Descrição)

Componente DFD Abstrato = Identificação +
Tipo [Entidade | Fluxo | Deposito | Processo] +
Pergunta

Componente DFD Concreto = Identificação +
Tipo [Entidade | Fluxo | Deposito | Processo] +
Função +
Operação +
Nome +
Coordenadas + * Coordenadas do componente na tela *
(Id. Componente Origem) + * Quando o componente é fluxo, este campo contém a identificação do componente de onde os dados partem *
(Id. Componente Destino) + * Idem ao componente anterior, porém contém a identificação do componente destino dos dados *
(Definição Dicionário) * é a definição do dicionário de dados, dos componentes fluxo e depósito *

Processo DFD Esp. Concreta = Nome
(Ação)
(Informação)
Função

Id. Especificação Reusável = Identificação

Respostas = {Resposta}

Componente Customizado = {Componente DFD Concreto}

Classe de Entidade = Nome +
{Campo}

Componente MER = Identificação Especificação +
Identificação Componente +
Tipo [Entidade | Relacionamento] +
Nome +
Coordenadas +
(Classe de Entidade) +
(Id. Componente 1 +
Id. Componente 2 +
Tipo Relacionamento [1:1 | 1:N | N:1 | M:N] +
Cmp 1 Opcional + * A entidade 1 que pertence ao relacionamento é opcional ou obrigatória (ocorrências) ? *
Cmp 2 Opcional)

Especificação Reusável ={ Função +
Operação +
Informação +
Arquivo +
Tipo Especificação: [Abstrata | Concreta]}

Relatório de Especificações = Nome +
Especificação [Especificação Abstrata | Especificação Concreta] +

{Componente: [Componente DFD Concreto | Componente DFD Abstrato)] +
{Componente MER}

5.4 MODELO ENTIDADE-RELACIONAMENTO

Segue na Figura 13 o MER da ferramenta.

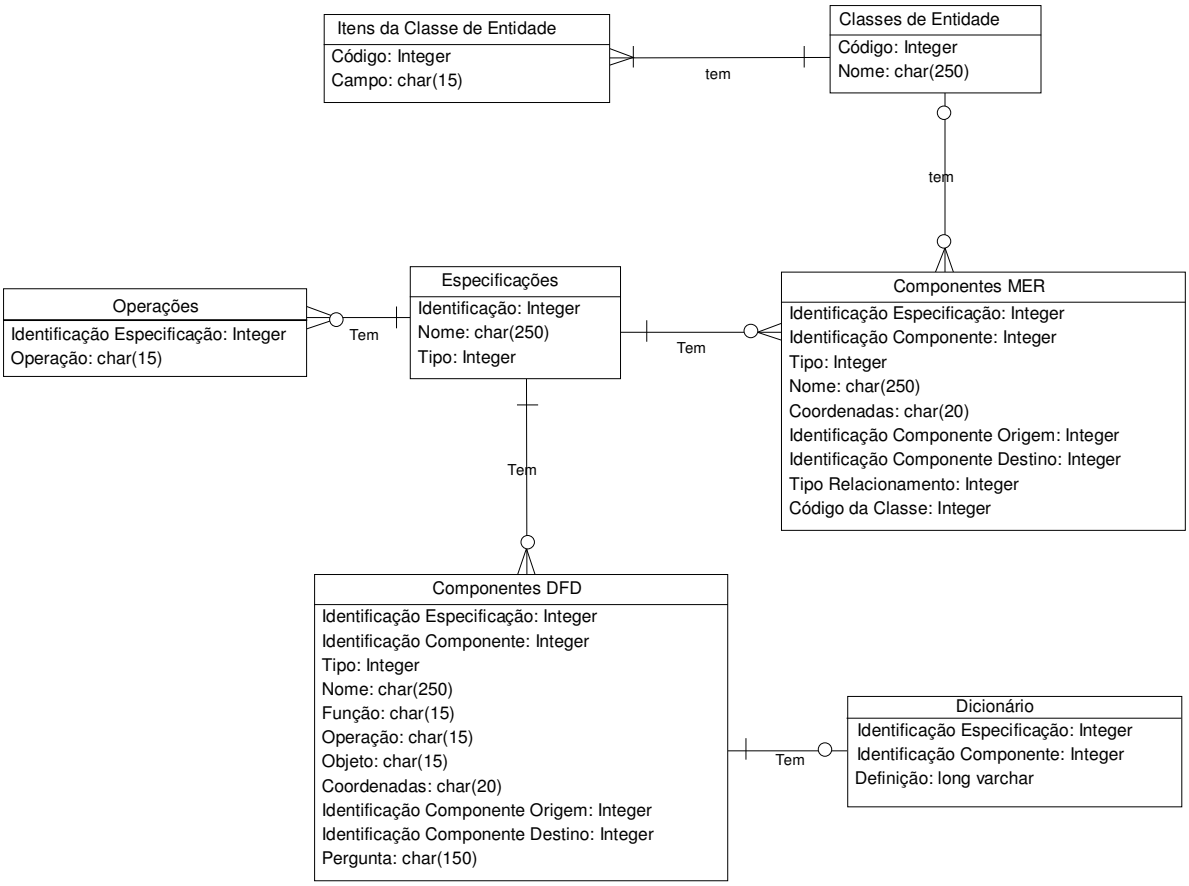


Figura 13 - MER

6 APRESENTAÇÃO DA FERRAMENTA

Este capítulo apresenta a ferramenta desenvolvida expondo suas telas e explicando o seu uso. Ao final do capítulo há um exemplo demonstrando o uso.

O software implementado foi desenvolvido no ambiente de programação Delphi 3.0 e é executado na plataforma Windows 32 bits (Windows 95,98 e NT).

6.1 VISÃO GERAL DO PROTÓTIPO

A ferramenta desenvolvida possui um editor de DFD e MER, utilizados para desenhar os respectivos diagramas. As especificações criadas são gravadas no repositório da ferramenta e estarão disponíveis para serem reusadas na construção de futuras especificações.

No DFD, as tentativas de aproveitamento de especificações de outros sistemas ocorrem ao inserir um componente do tipo processo. O reuso portanto ocorre ao nível de processo, ou seja, uma especificação reusável consiste em um processo e componentes relacionados a este processo (fluxos de dados, entidades e depósitos de dados ligados ao processo).

O reuso no DFD é possível quando a ferramenta identifica especificações reusáveis. O processo de identificação de especificações reusáveis foi baseado na técnica da Analogia, vista no capítulo anterior. Uma especificação será reusável se for análoga à especificação que se deseja fazer. Para que duas especificações sejam análogas elas devem ter a mesma função.

Para extrair os componentes do repositório foi utilizado o método de classificação de componentes de facetas, visto no capítulo 2.5. As facetas utilizadas foram:

a) tipo de especificação: Concreta ou Abstrata;
b) função da especificação;
c) objeto ou informação. É a informação que o processo irá manipular.

Através dos experimentos de [MAI92] e [ZIR97], concluiu-se que o reuso de especificações concretas são mais eficazes que o reuso de especificações abstratas, por isso na ferramenta implementada, as especificações concretas, quando identificadas, são sempre sugeridas como sendo as principais especificações a serem utilizadas. Serão então apresentadas ao usuário especificações concretas que tenham a mesma função e que

manipulem a mesma informação. Especificações abstratas que tenham a mesma função também serão apresentadas ao usuário e estarão disponíveis para reusar.

Após selecionar o tipo de especificação que será utilizada, dois procedimentos poderão acontecer, dependendo do tipo de especificação. Quando a especificação a ser reusada é concreta, o protótipo copia esta especificação para a especificação do sistema atual. É neste momento que ocorre o reuso do dicionário de dados, pois a definição do dicionário de dados dos componentes envolvidos também é copiada para especificação do sistema atual.

Se a especificação a ser reusada for abstrata, um outro procedimento é executado, onde a ferramenta auxiliará o reuso desta especificação, através de perguntas feitas ao usuário. Estas perguntas são criadas pelo usuário no momento da criação de especificações abstratas.

O reuso de componentes do MER foi baseado na Analogia e usa também o conceito de herança existente na técnica de orientação a objetos. Componentes entidades com o mesmo nome podem ser reusados por serem similares. Os componentes entidades também podem derivar de classes de entidades, que são modelos de entidades de dados, e neste caso recebem todos os atributos da classe.

6.2 A INTERFACE

A Figura 14 apresenta a interface da ferramenta.

O ambiente é composto por um menu principal cujas opções são Arquivo, Editar, Componentes e Ajuda. Abaixo do menu principal existe uma barra de ferramentas para acesso aos comandos mais comuns. Ao lado esquerdo está a barra de ferramentas de desenho, utilizados na construção de DFD e MER. A Figura 15 mostra a descrição de cada botão da barra de ferramentas.

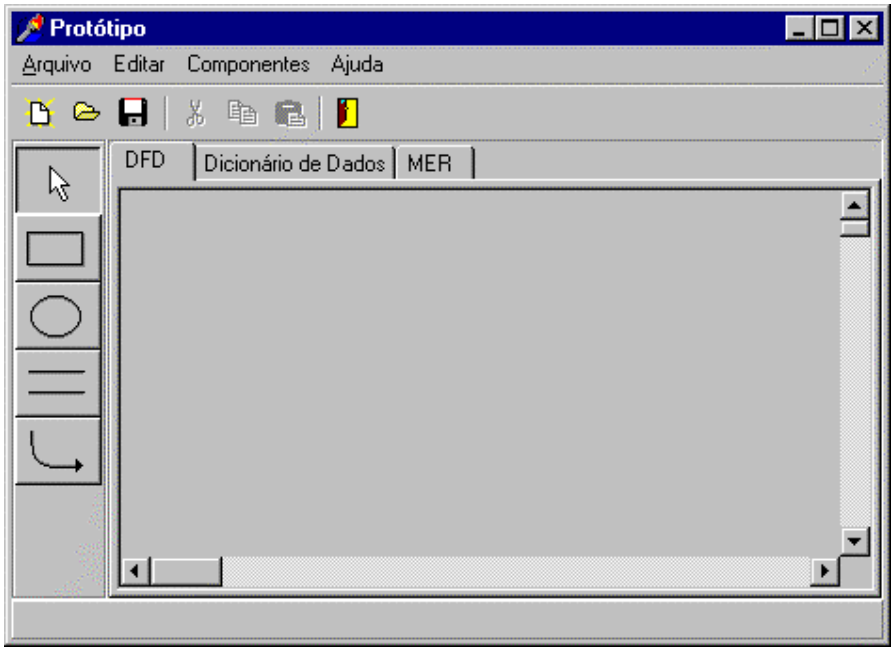


Figura 14 - Interface da Ferramenta

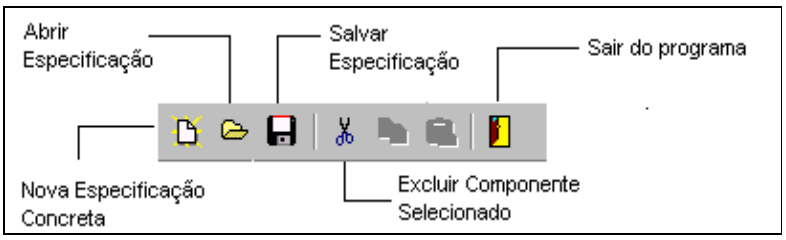


Figura 15 - Barra de Ferramentas

6.3 MENU ARQUIVO

O menu “Arquivo” é subdividido nos seguintes itens:

a) nova Especificação. Inicia o processo de criação de especificações. Este comando será detalhado mais adiante;

b) abrir. Este comando é usado para recuperar uma especificação que foi previamente criada e salvada no repositório, permitindo a edição da mesma. O comando Abrir também pode ser acessado pela barra de ferramentas;

c) salvar. O comando Salvar irá gravar a especificação atual no repositório da ferramenta. As especificações são arquivos com extensão “esp” e se encontram no

diretório do repositório. O comando Salvar também pode ser acessado através da barra de ferramentas;

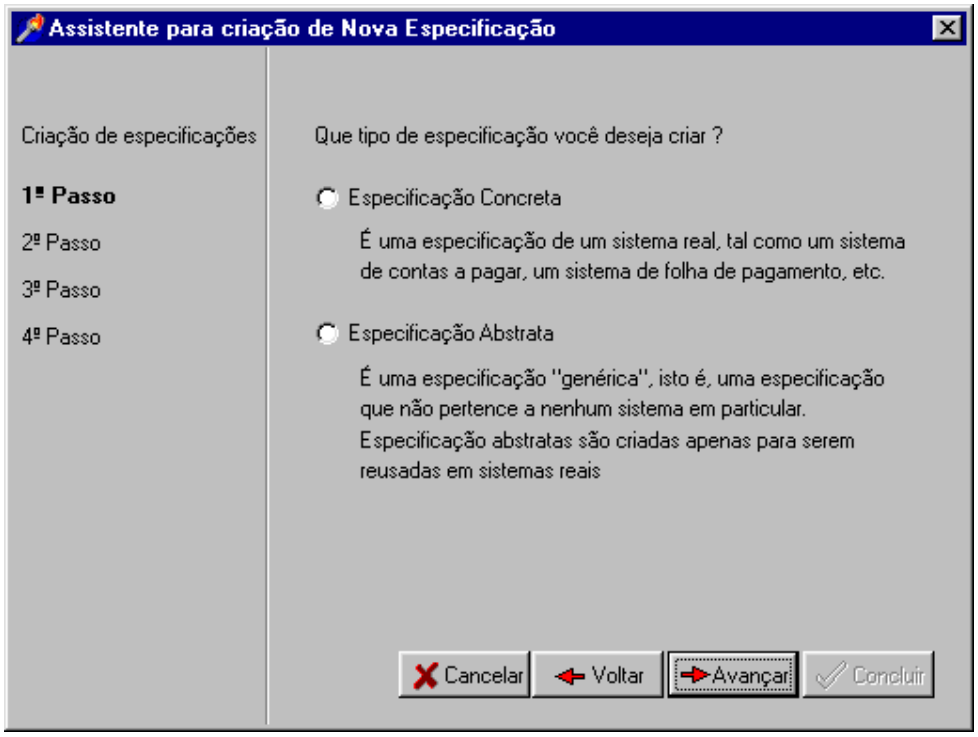
d) fechar. O comando Fechar abandona a edição da especificação corrente;
e) propriedades da especificação. Este comando permite visualizar as propriedades definidas no procedimento de criação de especificação, permitindo eventuais alterações na definição;
f) configurar. A principal finalidade do comando Configurar é identificar a localização do repositório da ferramenta, que é um diretório onde todas as especificações são salvadas. O processo de reuso se baseia nas especificações que estão armazenadas neste diretório;
g) sair. O comando Sair finaliza a execução do programa.

6.3.1 NOVA ESPECIFICAÇÃO

O comando Nova Especificação irá chamar o Assistente para criação de especificações, que é um procedimento que auxilia o usuário da ferramenta na criação de especificações abstratas e especificações concretas. Durante o processo são apresentados explicações e exemplos com a finalidade de ajudar o usuário na construção de especificações.

O processo de criação de especificações é dividido em até quatro passos, numerados de 1 a 4.

No primeiro passo (Passo 1), o usuário deve definir o tipo de especificação que deseja criar. A especificação pode ser concreta ou abstrata. Os passos seguintes serão executados de acordo com o tipo de especificação que for selecionada.



Definindo o tipo de especificação como abstrata, o usuário é questionado, no Passo 2, sobre a função da especificação a ser criada, ou seja, a finalidade da especificação. Uma especificação abstrata deve sempre se referir a um processo no diagrama de fluxo de dados, ou seja, sempre conterá um e somente um processo. A função da especificação é ponto chave na identificação de especificações reusáveis. No Passo 3, será necessário incluir palavras que descrevam o que a especificação faz. Como especificações abstratas sempre correspondem a um processo no diagrama de fluxo de dados, estas palavras deverão ser nomes comuns a processos com a mesma função. Por exemplo, numa das especificações criadas a função foi definida como “manutenção”, enquanto que as palavras que descrevem esta função foram “manter”, “incluir”, “cadastrar”, “alterar” e “excluir”. Por último (Passo 4) deverá ser incluído uma descrição mais completa da finalidade da especificação que será criada.

Outros parâmetros são solicitados quando a especificação selecionada for concreta. Neste caso, no Passo 2 o usuário deverá informar o nome da especificação. No terceiro e último passo será necessário digitar uma descrição para a especificação que será criada.

O usuário também poderá utilizar o botão Novo existente na barra de ferramentas para iniciar a criação de uma nova especificação concreta.

6.4 MENU EDITAR

O menu Editar dá acesso a dois comandos: Excluir e Propriedades do Componente. O comando excluir será utilizado para excluir o componente do DFD ou do MER que está atualmente selecionado.

O comando Propriedades do Componente permite acessar as definições de um componente do DFD ou do MER. Ao inserir um componente, o usuário define algumas propriedades do componente e estas propriedades podem ser alteradas ou consultadas a partir desta tela.

6.5 RELATÓRIOS

É possível através da ferramenta imprimir os diagramas e o dicionário de dados. Para isto existe no menu Componente, o item Relatório que dá acesso ao procedimento de geração de relatórios.

6.6 ELABORANDO ESPECIFICAÇÕES

Após identificar o tipo de especificação que se quer criar, o usuário poderá elaborar os diagramas da especificação. Para desenhar o DFD, usa-se a barra de ferramentas de desenho, conforme pode ser visto na Figura 16.

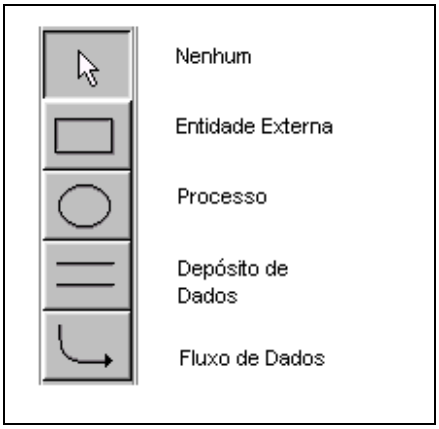


Figura 16 - Barra de ferramentas de desenho do DFD

Para desenhar o MER, tem-se acesso a barra de ferramentas que está na Figura 17.

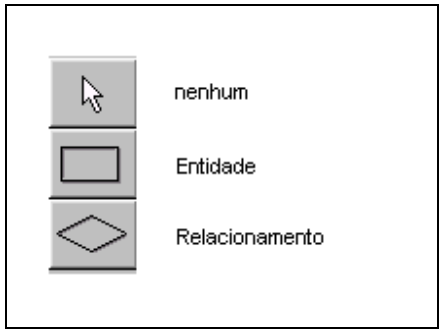


Figura 17 - Barra de Ferramentas de desenho do MER

Para incluir algum componente na tela, seja ele do DFD ou MER, deve-se dar um clique no componente correspondente e depois um clique na tela e o componente será desenhado. No caso do componente "fluxo de dados" do diagrama DFD, ao selecioná-lo na barra de ferramentas, deve-se primeiro clicar num componente origem e depois num componente destino e somente neste momento será desenhado o componente fluxo de dados. O mesmo acontece com o componente Relacionamento, existente no diagrama MER.

O processo para criar especificações abstratas e concretas são diferentes e são abordados nos itens seguintes. No caso de especificações abstratas não existe a definição de um dicionário e de um modelo entidade-relacionamento, como ocorre com as especificações concretas.

6.6.1 ESPECIFICAÇÕES ABSTRATAS

As especificações abstratas são criadas para serem reusadas em especificações concretas. Pode-se comparar uma especificação abstrata como sendo um modelo que será utilizado na construção de outras especificações.

Para criar especificações abstratas deve-se começar incluindo um componente processo. O processo incluído deverá ser o único existente na especificação. Após isto pode-se inserir os outros componentes que compõem a especificação.

Ao tentar inserir algum outro componente, o programa irá solicitar que se digite uma "pergunta". Esta pergunta que for informada será realizada ao usuário no momento em que esta especificação for reusada e a resposta corresponderá ao nome do componente.

Um exemplo pode ser visto na Figura 18. A pergunta “Um dado será mantido pelo processo. Quem vai informar ao sistema sobre este dado ?” será feita ao usuário quando ele tentar reutilizar esta especificação. E a resposta que o usuário informar será o nome do componente. Neste caso o componente era uma entidade externa.

É possível que a pergunta já tenha sido feita a um outro componente. Então neste caso seria inconveniente ter que responder a mesma pergunta duas ou mais vezes. Por isso pode-se selecionar o item “Pergunta já feita a outro componente” e escolher a pergunta.

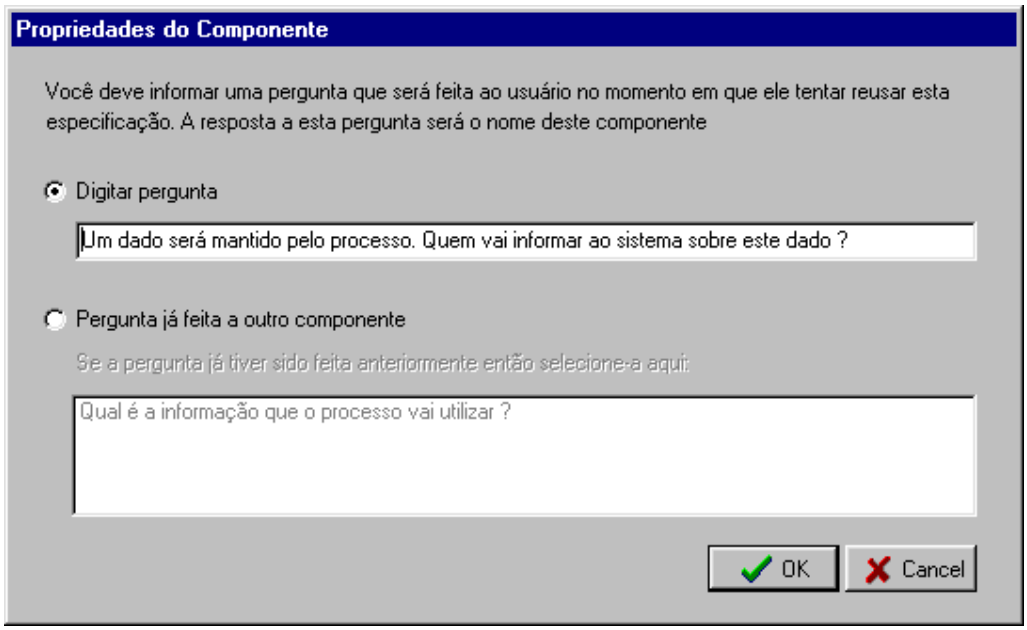


Figura 18 - Propriedades de um componente de uma especificação abstrata

Este procedimento ocorre toda vez que um componente do DFD de uma especificação abstrata é inserido na tela.

6.6.2 ESPECIFICAÇÕES CONCRETAS

As especificações concretas, isto é, especificações de sistemas reais, compreendem o DFD, dicionário de dados e MER, conforme foi visto no Capítulo 3 . A Figura 14 apresentou a interface da ferramenta e nela é possível visualizar o acesso a área de desenho destas três ferramentas da Especificação Estruturada.

6.6.2.1 DFD

Para criar especificações concretas pode-se iniciar incluindo qualquer componente. Dependendo do tipo de componente será apresentado uma tela para digitação das propriedades do componente.

A Figura 19 mostra a tela das propriedades do componente depósito de dados.

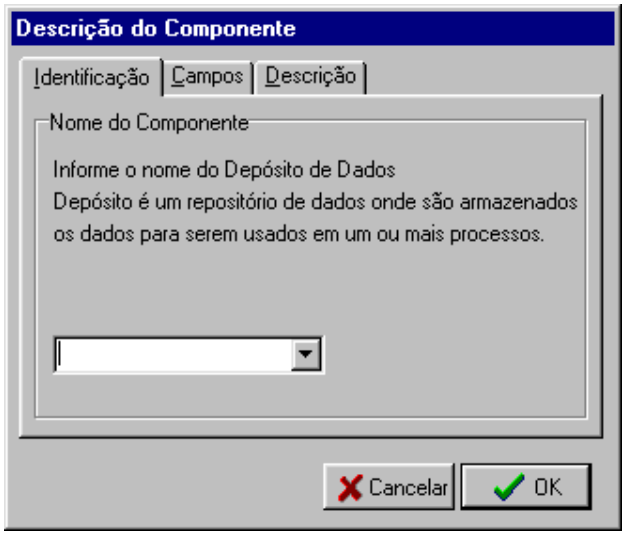


Figura 19 - Propriedades do Componente Depósito de Dados

Pode-se observar que a tela apresentada na Figura 19 apresenta uma descrição do tipo do componente, que visa auxiliar o usuário na criação do diagrama.

A "página" Campos da tela somente estará disponível para os componentes Fluxo de Dados e Depósito de Dados onde dever-se-á informar o conjunto de campos que compõem o componente. Estes dados informados no componente são integrantes do dicionário de dados.

Para o componente Processo será apresentado a tela que está na Figura 20.

Digitando-se a "ação", o programa procura automaticamente por alguma especificação abstrata que tenha esta operação e a traz no campo função.

O campo "nome do processo" também é montado automaticamente com a digitação dos campos "ação" e "informação manipulada".

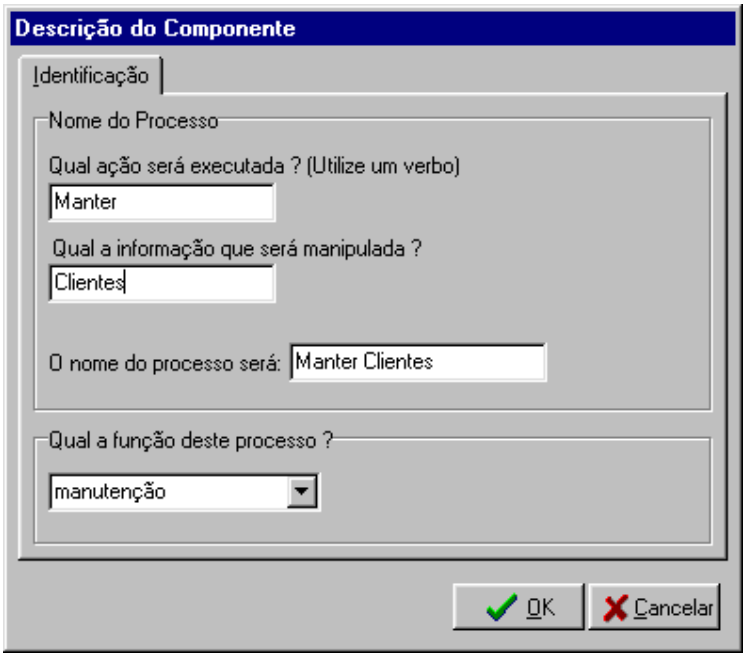


Figura 20 - Propriedades do Processo

Os dados informados nesta tela servirão de parâmetros para o procedimento que faz a identificação de especificações reusáveis. Será necessário digitar pelo menos a função do componente para que seja possível pesquisar por especificações reusáveis.

Ao clicar no botão OK, serão apresentados todas as especificações que poderão ser usadas. Se a especificação reusada for uma especificação concreta, a ferramenta irá copiá-la para a especificação do sistema atual. Se for selecionado uma especificação abstrata, o programa irá fazer as perguntas informadas durante o procedimento de criação da especificação abstrata, como vê-se na Figura 21. Este é o processo de customização do componente, onde o usuário parametriza o componente de acordo com o contexto.

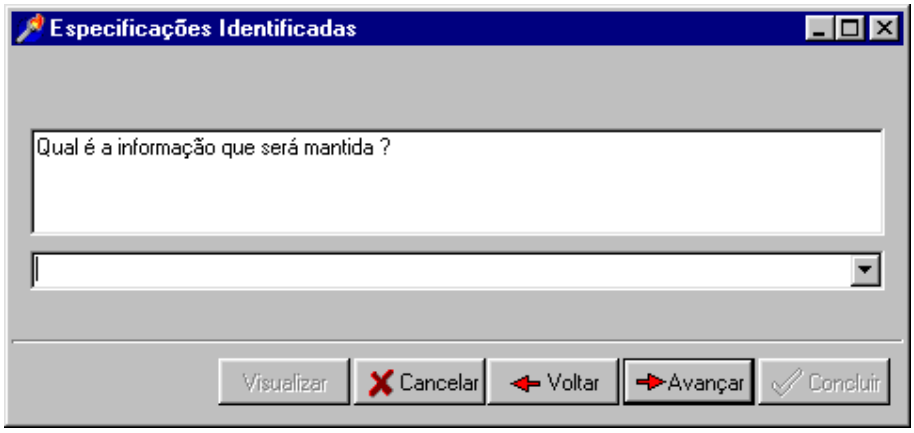


Figura 21 - Customização do Componente

6.6.2.2 DICIONÁRIO DE DADOS

O reuso de dicionário de dados ocorre no momento em que ocorre o reuso do diagrama de fluxo de dados. Além disso, o dicionário de dados só pode ser reusado a partir de especificações concretas, já que especificações abstratas não possuem dicionário de dados.

Quando uma especificação é reutilizada, o dicionário de dados dos componentes integrantes da especificação reutilizada é copiado para a especificação atual.

6.6.2.3 MER

O modelo de entidade-relacionamento poderá ser reusado a partir de duas formas: através da estrutura de classes de entidade e através de sistemas já especificados na ferramenta.

Ao clicar num componente Entidade e depois de clicar na tela, o programa pede para digitar o nome da Entidade. É possível pesquisar as entidades do repositório como mostra a Figura 22.

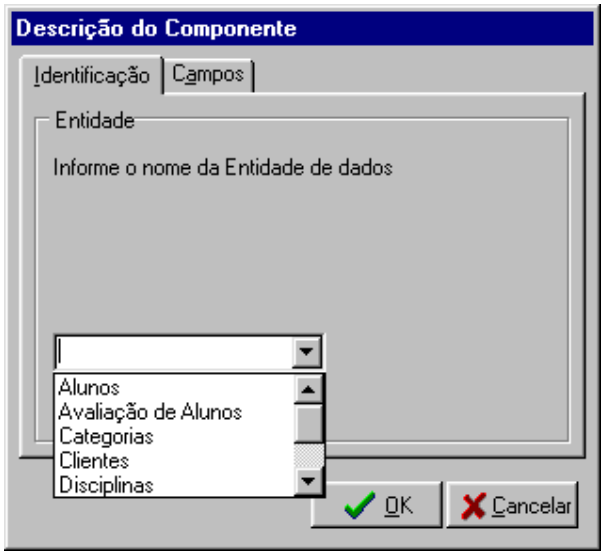


Figura 22 - Tela para digitação da Entidade

Se uma das entidades for selecionada, os atributos da entidade serão reusados na entidade da especificação atual.

No menu Componentes existe o item “Classes de Entidade (MER)”. Acessando-se esta opção tem-se acesso a tela da Figura 23.

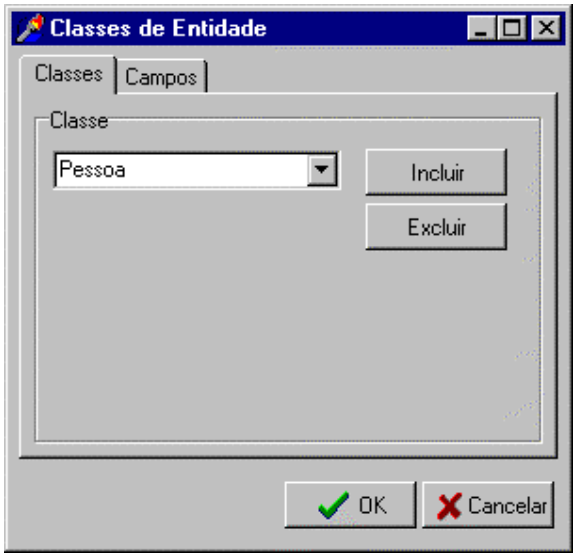


Figura 23 - Classes de Entidade

Classes de entidade são modelos de entidades. Outras entidades podem ser construídas baseando-se numa classe de entidades. Assim como ocorre com orientação a objeto onde um objeto descendente de outro herda todas as suas características, uma entidade que se origina de uma classe de entidades também herda as suas características. No caso, as características de uma entidade são os seus campos. Os campos são informados na página “Campos”.

Um das classes criadas foi a classe de pessoas. Os atributos desta classe foram cadastrados no protótipo como pode ser visto na Figura 24.

Figura 24 - Exemplo de Classe de Entidades

Para inserir um componente do tipo Relacionamento será necessário informar o nome do relacionamento, o tipo de relacionamento e a obrigatoriedade entre as entidades. A Figura 25 mostra a tela onde estes dados são digitados.

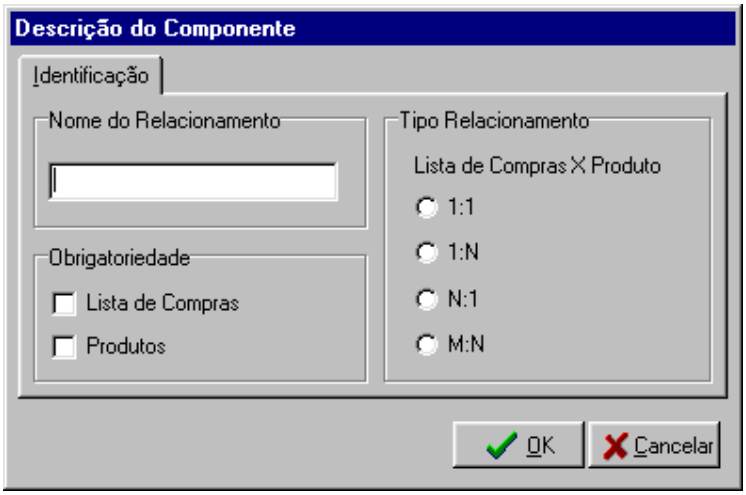


Figura 25 - Propriedades de um Relacionamento

Depois de salvar a especificação todos os componentes do MER estarão disponíveis para serem reusados em outras especificações.

6.7 EXEMPLO

Um exemplo, extraído de [POM94], será apresentado para demonstrar passo a passo o reuso de especificações.

6.7.1 REUSO DO DFD E DICIONÁRIO DE DADOS

Esta demonstração do reuso de especificações será iniciada pelo diagrama de fluxo de dados.

Uma nova especificação foi iniciada e nela serão inseridos os seguintes processos: Cadastrar Fornecedores, Emitir Lista de Demanda, Cadastrar Produtos e Cadastrar Lista de Compras.

O primeiro processo que será incluído será “Cadastrar Lista de Compras”. O componente foi selecionado e o programa pede para que se informe os parâmetros (Figura 26).

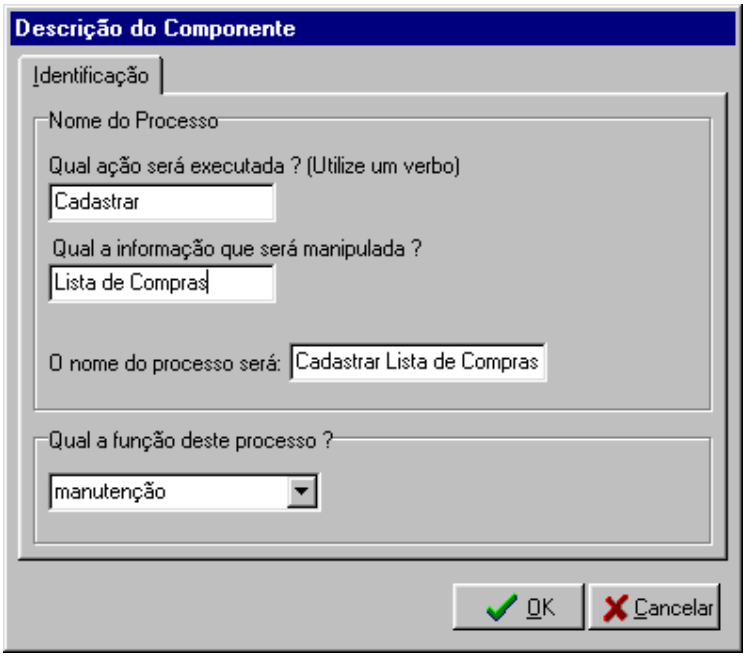


Figura 26 - Propriedades do Processo “Cadastrar Lista de Compras”

Ao clicar no botão OK, o programa irá pesquisar especificações reusáveis e solicitar ao usuário que escolha por uma especificação (Figura 27). Neste caso apenas uma especificação foi identificada.

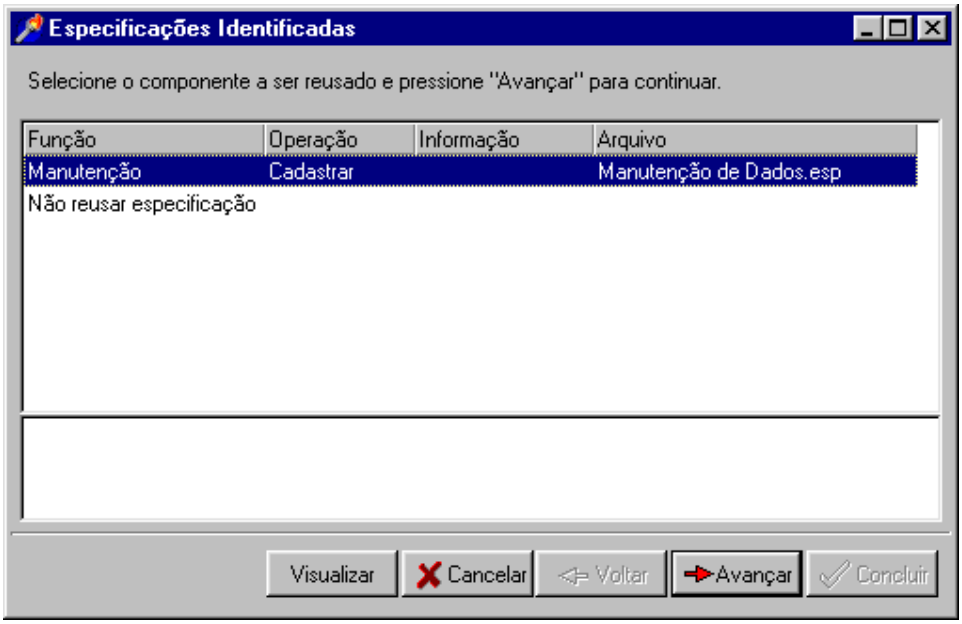


Figura 27 - Especificações Identificadas

Selecionando-se a especificação “Manutenção de Dados”, o programa irá auxiliar na customização da especificação, já que a especificação a ser reusada é abstrata. Para isso, a ferramenta fará uso das perguntas que foram cadastradas para a especificação reusável “Manutenção de Dados”.

A primeira pergunta feita ao usuário foi “Qual é a informação que será mantida ?” e a resposta foi “Lista de Compras”. Isto pode ser visto na Figura 28. Clicando-se o botão Avançar o programa fará a próxima pergunta da especificação (Figura 29).

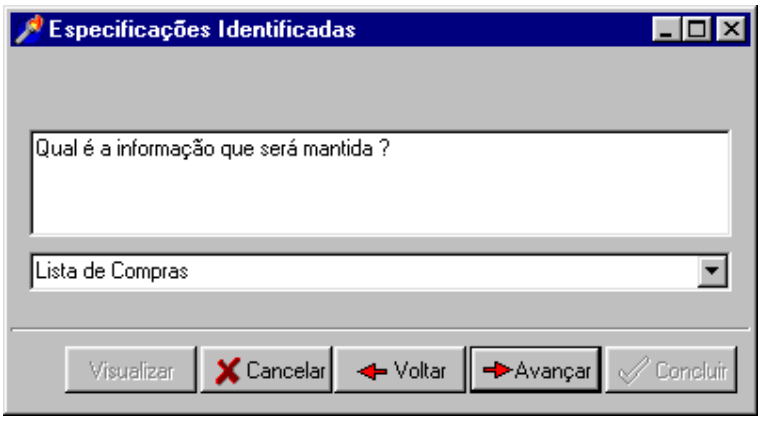


Figura 28 - Customização da Especificação

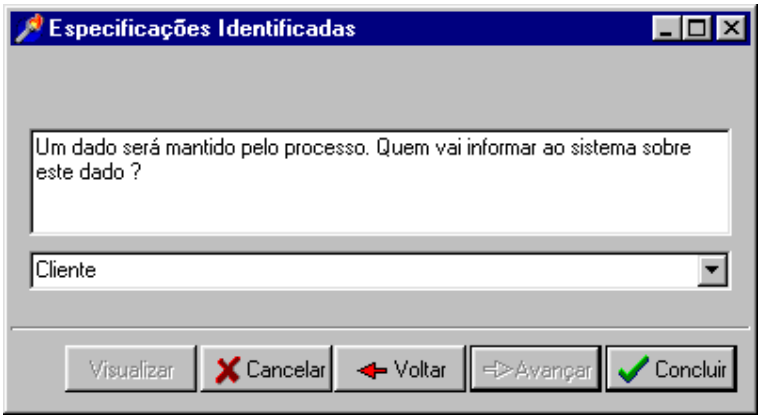


Figura 29 - Customização da Especificação

Esta foi a última pergunta realizada pela ferramenta (isto pode ser observado porque o botão Concluir neste momento está disponível, enquanto o botão Avançar não). Ao clicar no botão OK, a especificação será então desenhada (Figura 30).

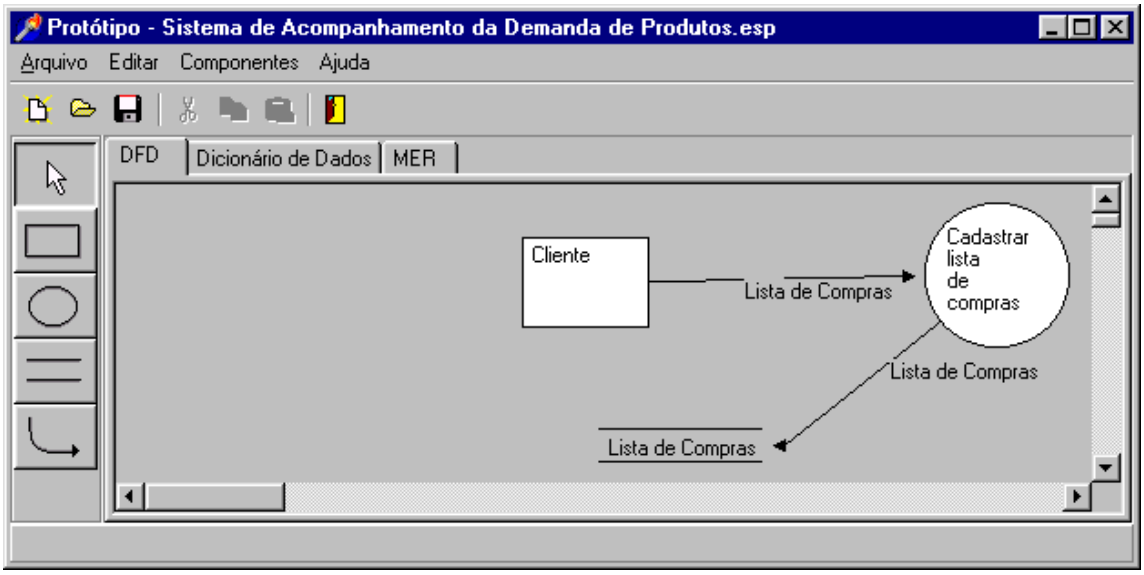


Figura 30 - Especificação reutilizada

Como esta especificação foi incluída através de uma especificação abstrata, não foi possível reutilizar o dicionário de dados. Neste caso, o dicionário de dados deverá ser informado pelo usuário. Na Figura 31 o usuário informa os dados que compõem o fluxo "Lista de Compras". Observa-se aqui que tanto um fluxo como o depósito tem o mesmo nome, além de existir mais outro fluxo com o mesmo nome. Nesta situação basta digitar as definições de "Lista de Compras" apenas uma vez, independente de qual componente for selecionado.

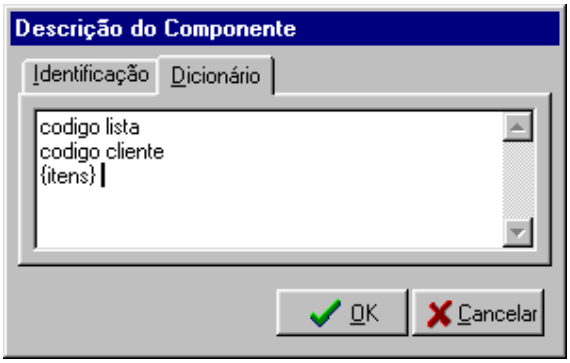


Figura 31 - Definição do Dicionário de dados de "Lista de Compras"

O próximo passo será a inclusão do processo "Cadastrar fornecedores". O componente foi selecionado e colocado na tela. Os parâmetros podem ser conferidos na Figura 32.

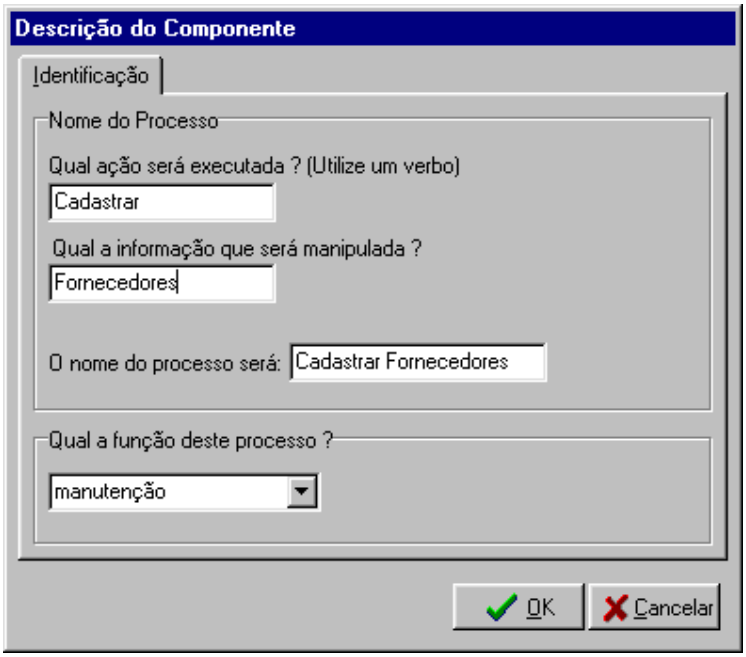


Figura 32 - Propriedades do Processo “Cadastrar Fornecedor”

Para este processo foi identificado duas especificações reusáveis, sendo uma especificação concreta, vinda de outro sistema anteriormente cadastrado, e a outra abstrata (Figura 33).

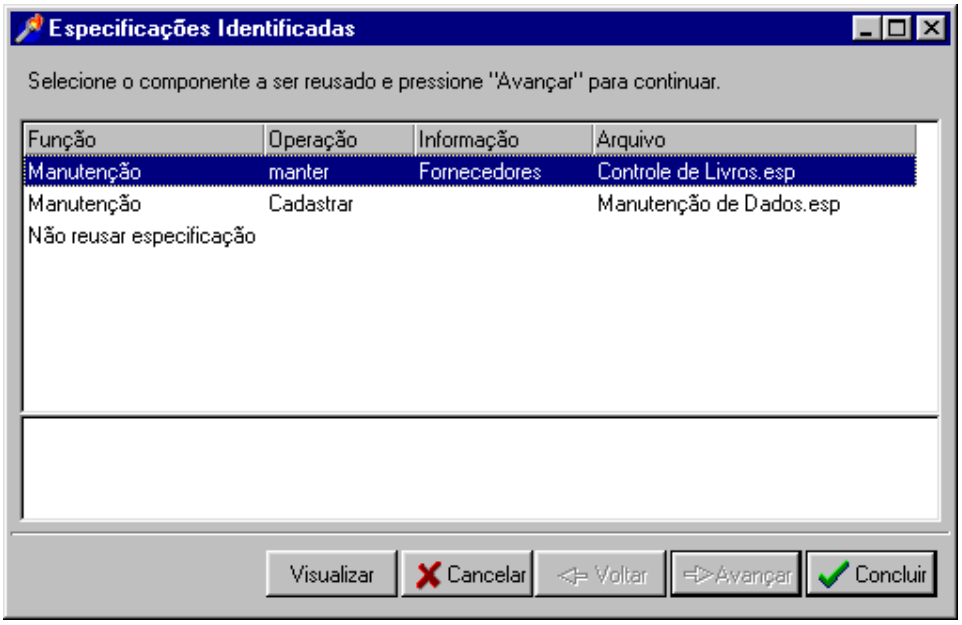


Figura 33 - Identificação de Especificações

A primeira especificação foi selecionada e a ferramenta a trouxe para a especificação atual, como mostra a Figura 34. Como a especificação reusada foi concreta, também foi possível aproveitar o dicionário de dados. Clicando-se na página “Dicionário de Dados” da tela principal, pode-se visualizar a descrição de todos os fluxos e depósitos da especificação (Figura 35).

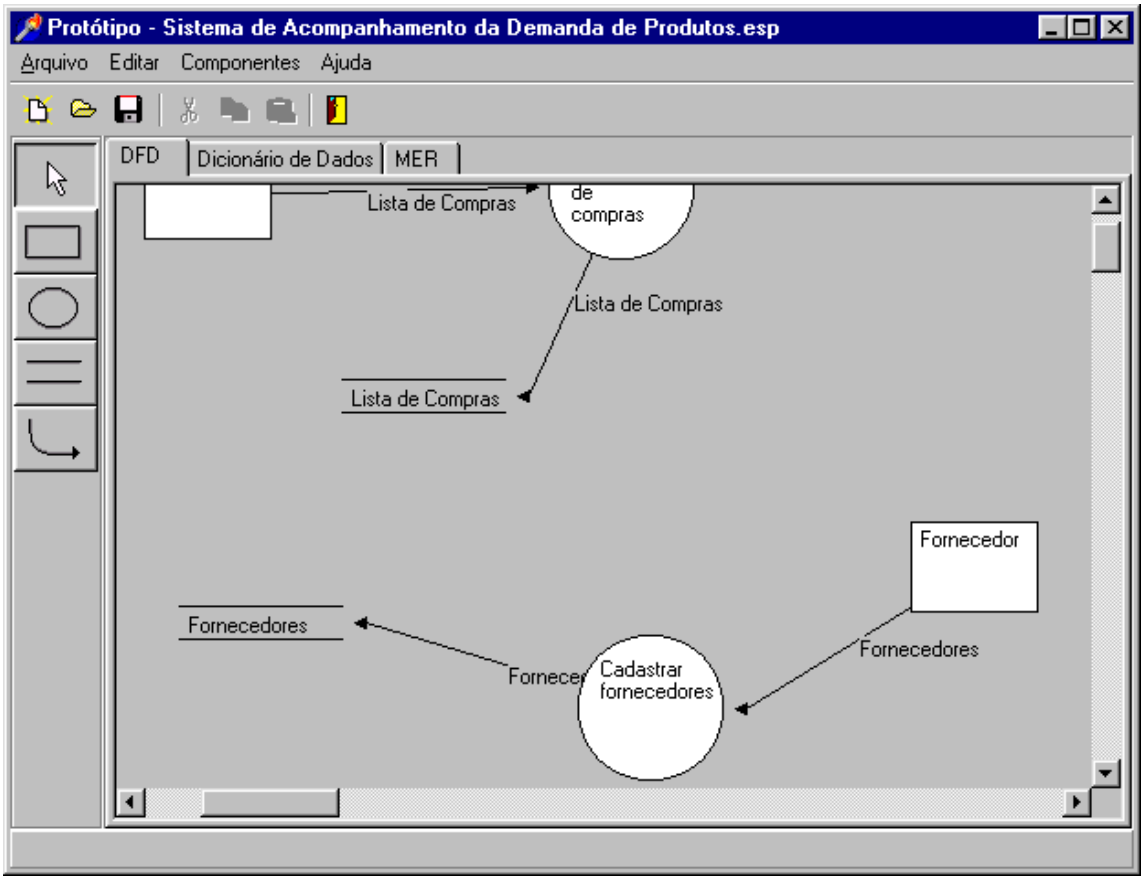


Figura 34 - Tela demonstrando especificação reusada

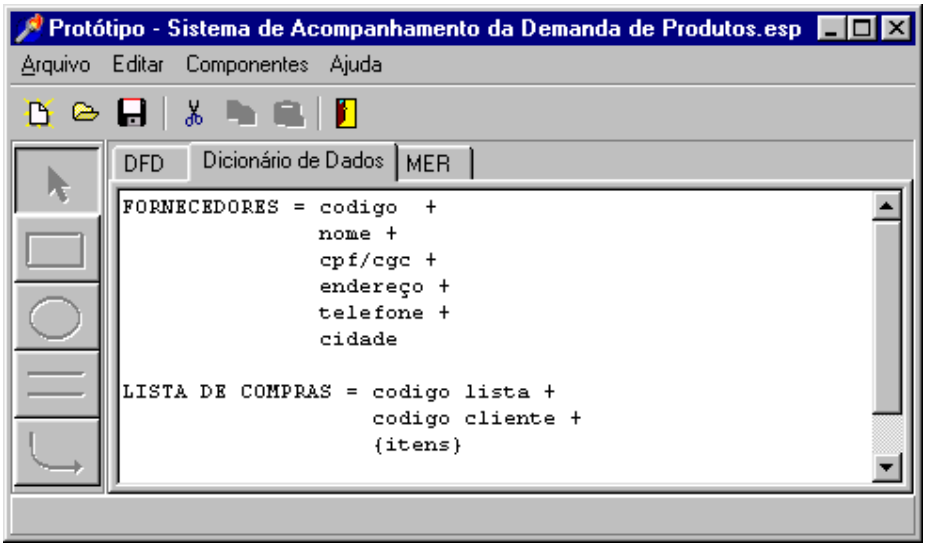


Figura 35 - Visualização do Dicionário de Dados

O próximo passo será incluir o processo “Cadastrar Produtos”, como mostra a Figura 36. Para este processo foi identificado uma especificação reusável (Figura 37).

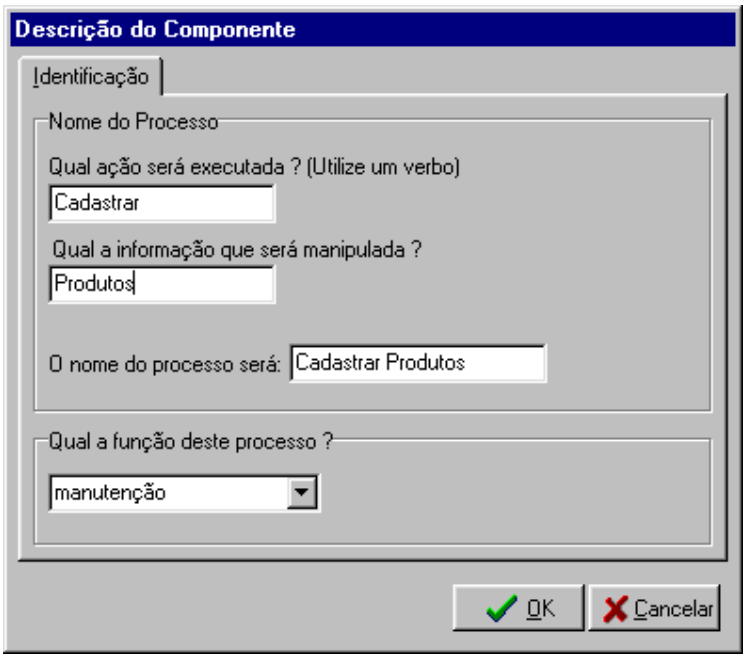


Figura 36 - Propriedades do Processo "Cadastrar Produtos"

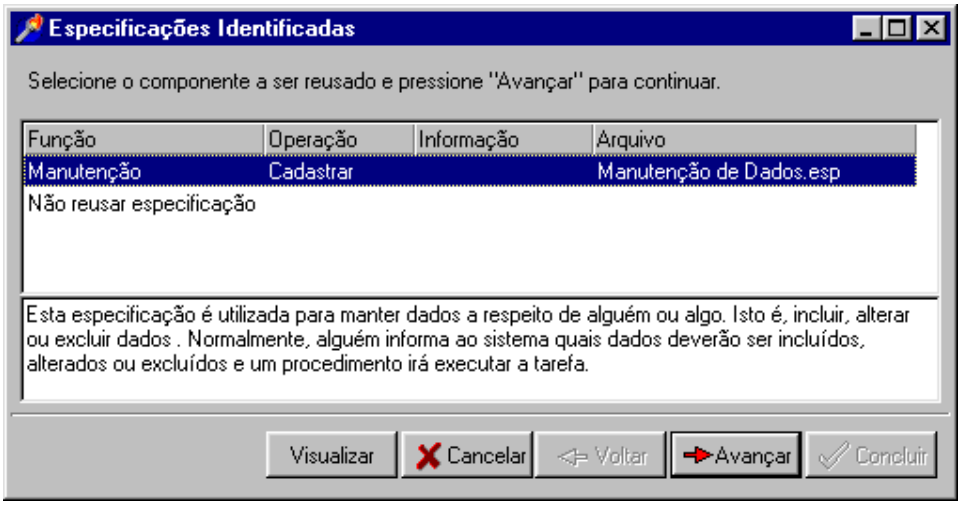


Figura 37 - Especificações Reusáveis

O processo de customização é igual ao do processo “Cadastrar Lista de Compras”, ou seja, as mesmas perguntas foram feitas. A especificação com a inclusão deste processo está na Figura 38.

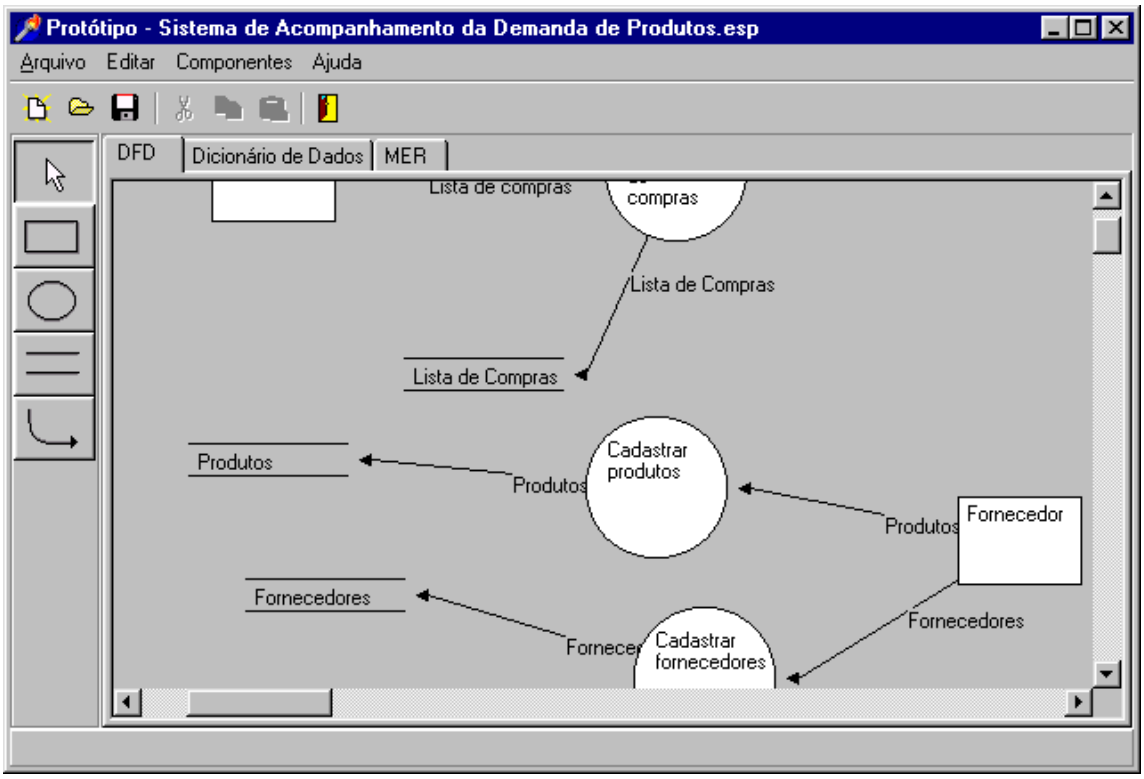


Figura 38 - Especificação reusada

O processo “Emitir Lista de Demanda” também será incluído e as propriedades podem ser vistas na Figura 39.

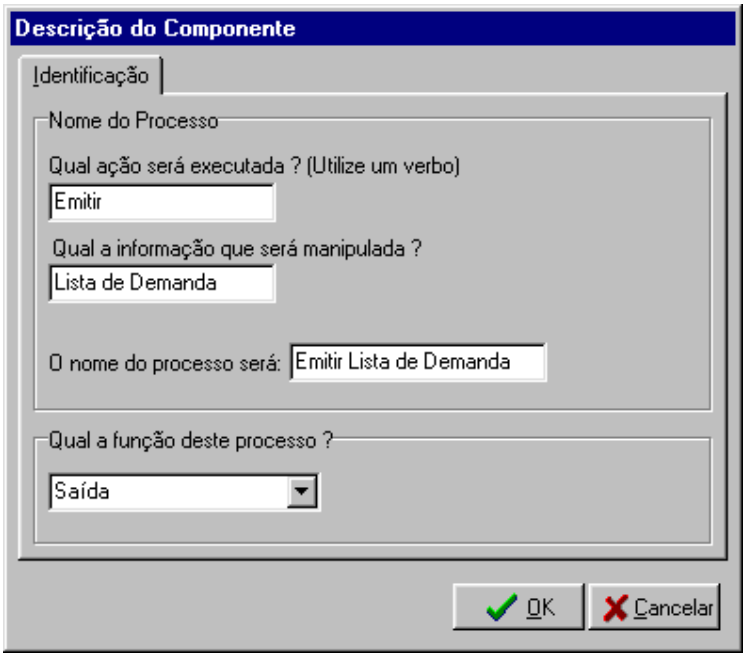


Figura 39 - Propriedades do Processo Emitir Lista de Demanda

Neste caso foi identificado uma especificação abstrata, a especificação “Saídas” (Figura 40) . A seqüência das perguntas podem ser encontradas na Figura 41, Figura 42 e na Figura 43.

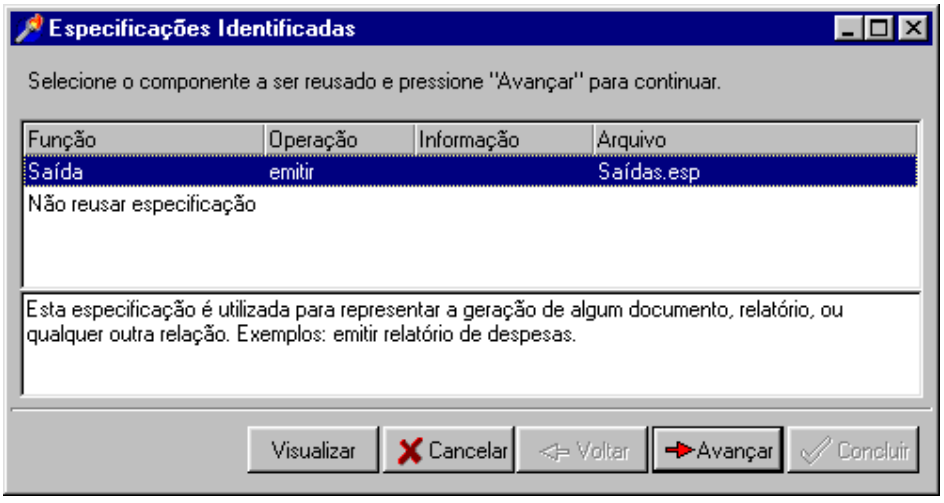


Figura 40 - Especificações Reusáveis Identificadas

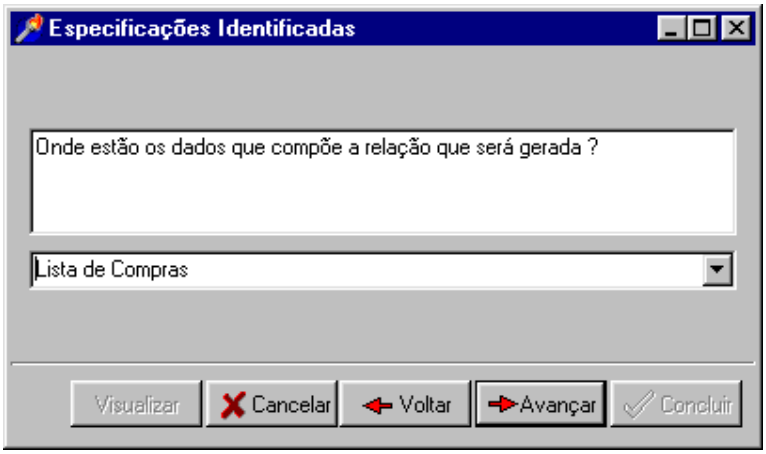


Figura 41 - Customização da especificação "Emitir Lista de Demanda"

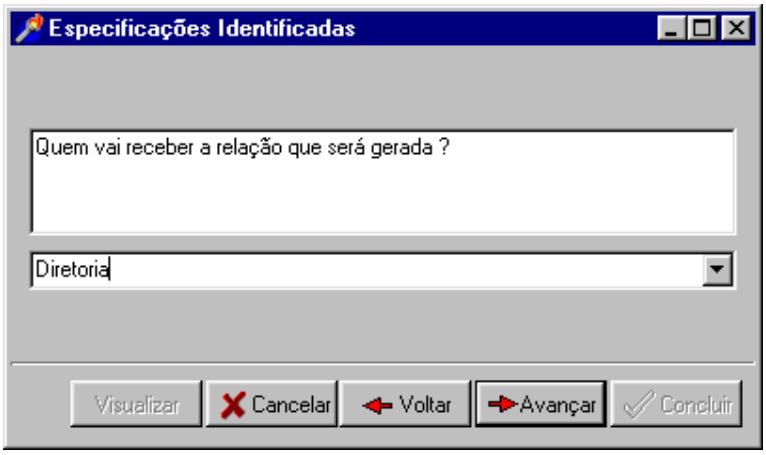


Figura 42 - Customização da Especificação “Emitir Lista de Demanda”

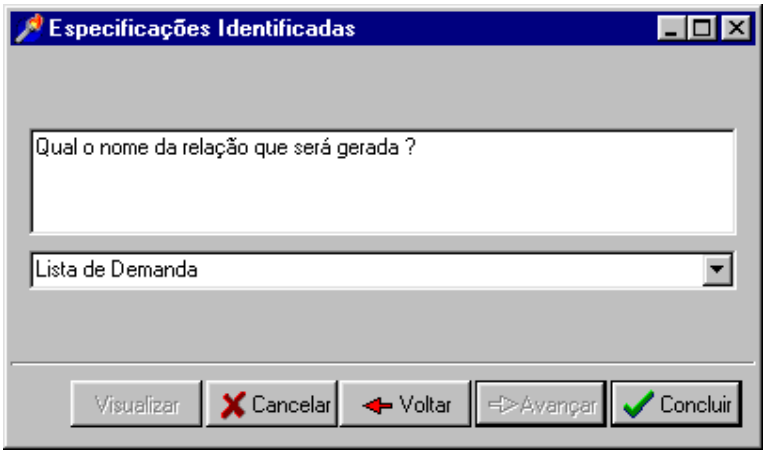


Figura 43 - Customização da Especificação “Emitir Lista de Demanda”

Concluindo-se o procedimento, a especificação será incluída na tela (Figura 44)

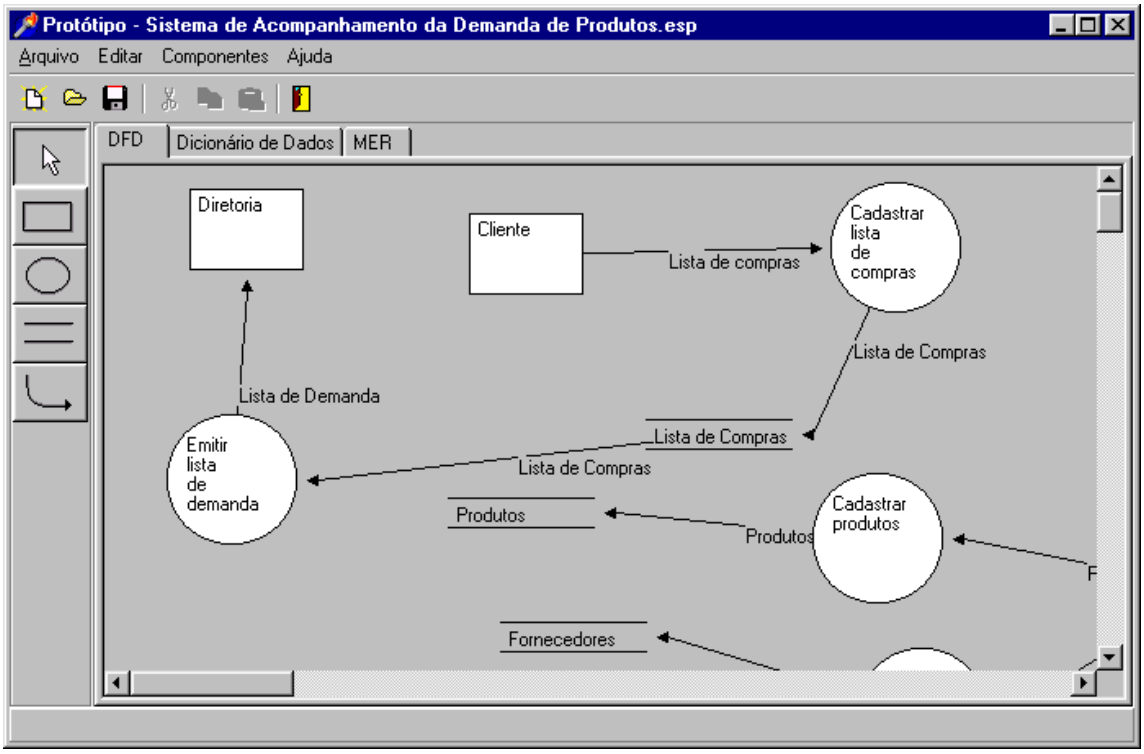


Figura 44 - Especificação após incluir o processo "Emitir lista de demanda"

A especificação reutilizada não atendeu completamente ao problema porque o processo “Emitir Lista de Demanda” busca dados das entidades Produtos e Fornecedores. O usuário deverá então incluí-los manualmente. O DFD completo está na Figura 45.

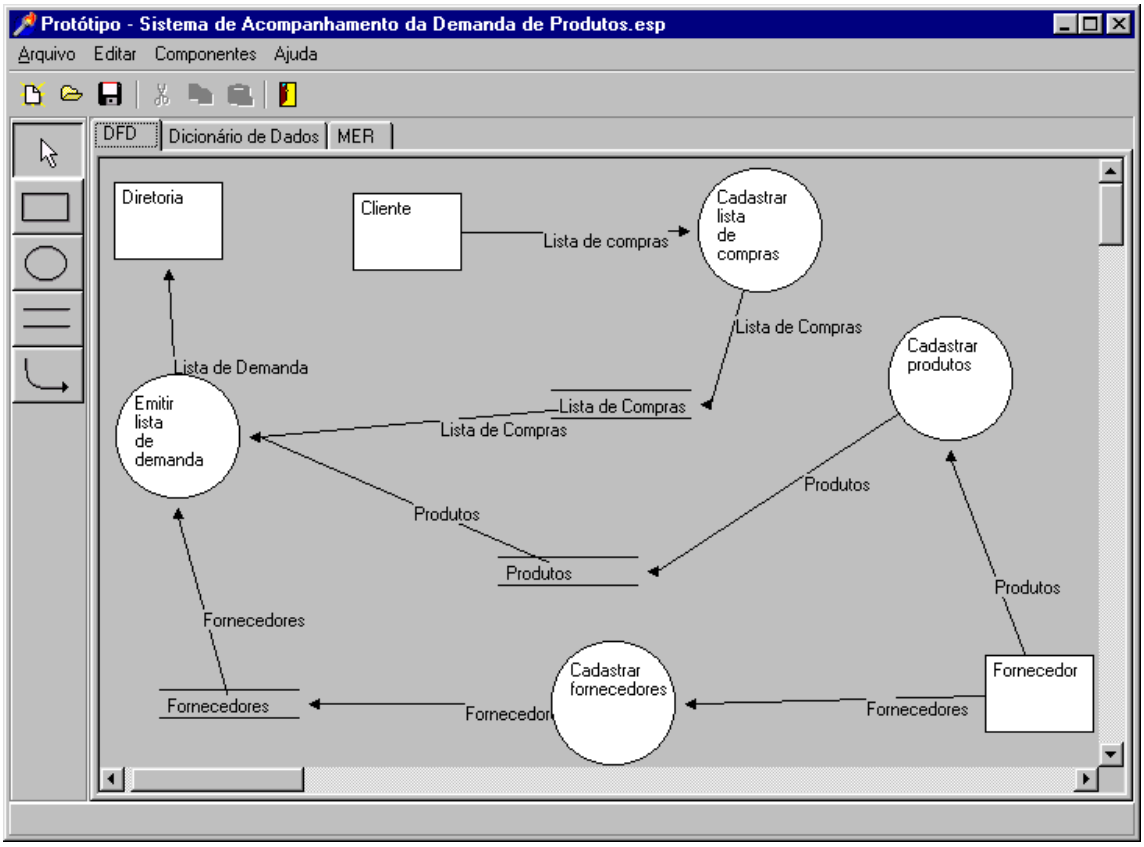


Figura 45 - DFD concluído do Sistema exemplo

6.7.2 REUSO DO MER

O passo seguinte é modelar o MER do sistema. Após selecionar o componente Entidade e depois de clicar na tela, o programa mostra a tela pedindo o nome da entidade. Como já havia um sistema que tinha a entidade Clientes, esta foi reutilizada na especificação do sistema atual (Figura 46).

Um outro componente a ser colocado é a entidade Fornecedor. A entidade Fornecedor é classificada como sendo do tipo Pessoa e por isso esta classe poderá ser reusada na construção desta entidade. Na Figura 23 foi visto a definição da classe Pessoa e agora ela poderá ser usada, bastando para isso selecioná-la e clicar no botão “Usar” (Figura 47).

As outras entidades foram colocadas da mesma forma que a entidade Fornecedor, porém sem reuso.

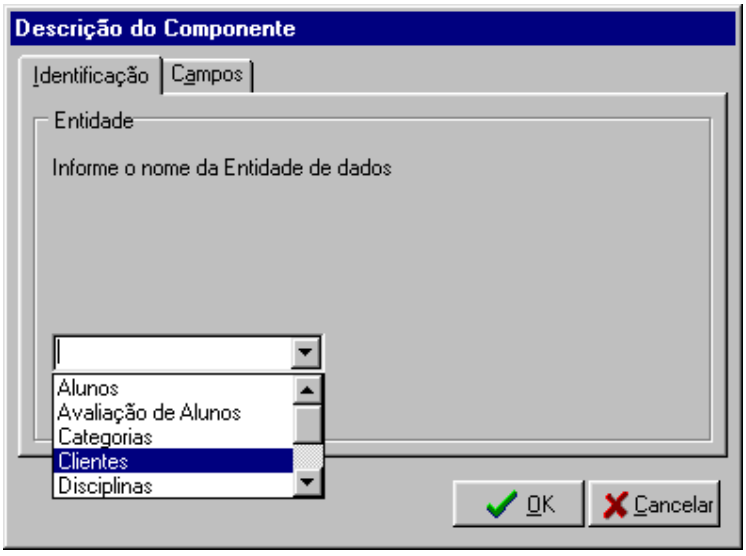


Figura 46 - Reuso no MER

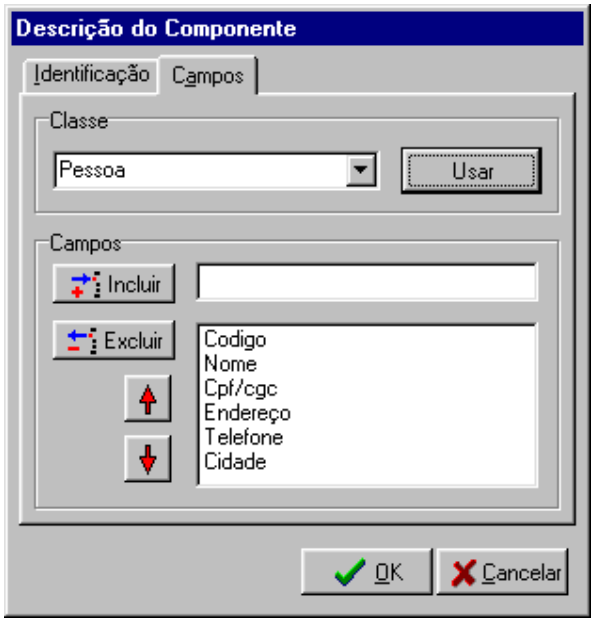


Figura 47 - Entidade Fornecedor derivando da classe "Pessoa"

Depois disso foi incluído o relacionamento entre as entidades, como por exemplo as Entidades Lista de Compras e Produtos. O relacionamento entre estas entidades está na figura 48

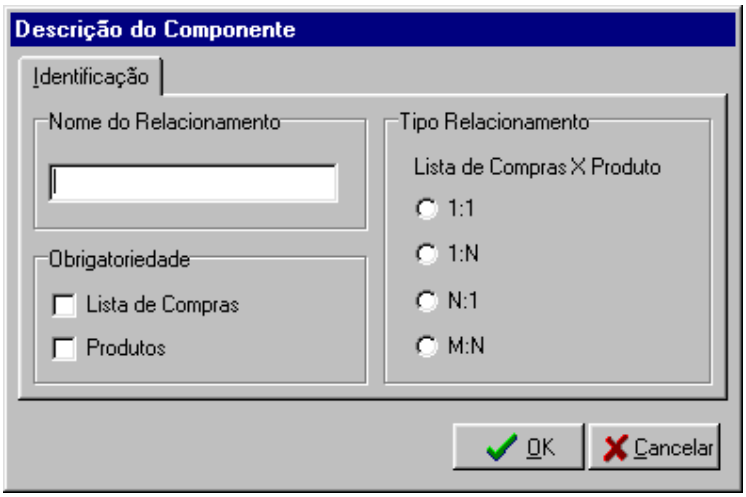


figura 48 - Relacionamento entre as Entidade Lista de Compras e Produtos

Após incluir todos as entidades e definir o relacionamento entre elas, o diagrama ficou concluído (Figura 49).

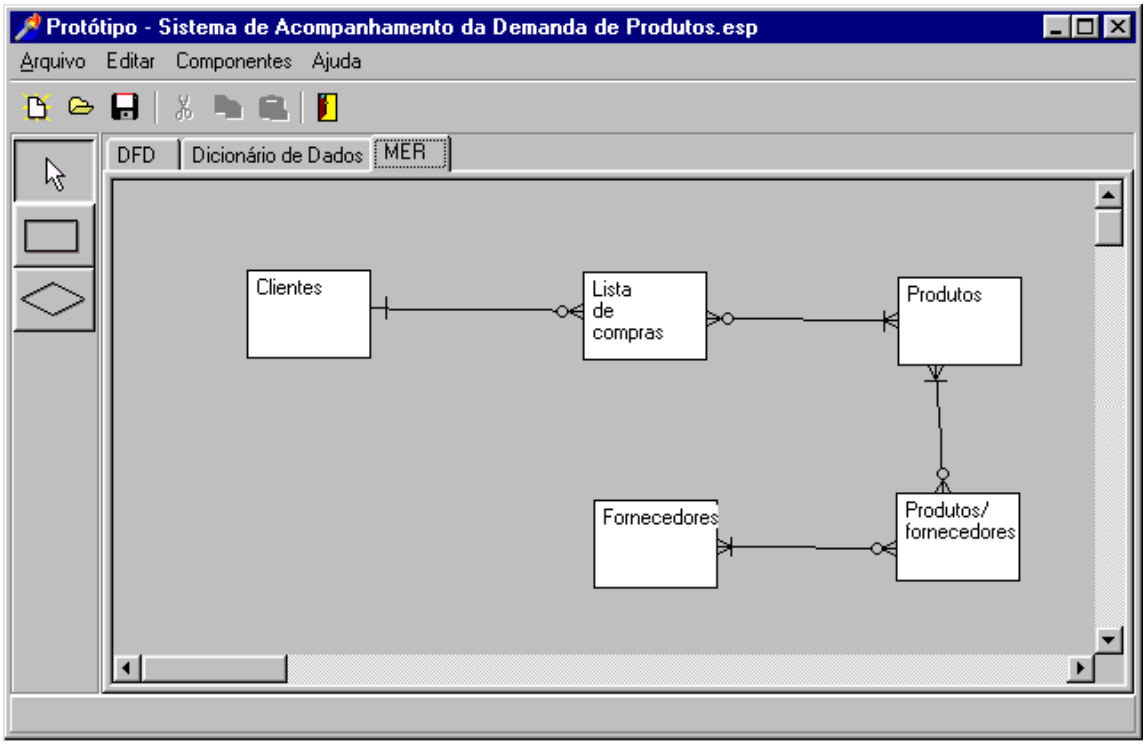


Figura 49 - MER do Sistema Exemplo

7 CONCLUSÃO

Após a finalização do trabalho e após a realização dos testes do protótipo pôde-se chegar às seguintes conclusões:

- a ferramenta possibilita ao usuário reutilizar com facilidade especificações de outros sistemas. Normalmente analogias entre especificações concretas permitem realizar um aproveitamento mais efetivo pois os componentes envolvidos são geralmente os mesmos. Isto quer dizer que com o reuso de especificações concretas tem-se poucas customizações a fazer;
- reuso de componentes abstratos também mostrou-se proveitoso. Porém o reuso de especificações abstratas demonstrou ser mais limitado que o reuso de especificações concretas. No protótipo foram inseridos apenas dois modelos de abstrações e pode-se observar que o reuso destas especificações geralmente exige a inclusão de novos componentes dos que aqueles que pertencem ao modelo;
- reuso do dicionário de dados está intrinsecamente ligado ao reuso de diagrama de fluxo de dados. Isto é, o reuso de dicionário de dados vai ocorrer quando ocorrer o reuso de componentes do DFD de especificações concretas;
- procedimento de identificação de componentes foi eficiente ao identificar analogias entre especificações de diagramas de fluxos de dados. O reuso de componentes do MER também ocorreu e facilitou a especificação do modelo;
- desenvolvimento de especificações de sistemas mais complexos apresentou um baixo nível de reusabilidade, pois as customizações aumentam gradativamente. Por isso é necessário que a ferramenta esteja bem "alimentada" de especificações de diversos sistemas. Por este motivo também que a ferramenta pode auxiliar principalmente os principiantes na construção de especificações de sistemas simples.

Uma limitação da ferramenta é que ela não oferece suporte ao desenvolvimento *top-down* do diagrama de fluxo de dados;

Algumas considerações devem ser feitas:

- durante o levantamento bibliográfico pode-se observar a quantidade limitada de textos apresentando o assunto onde inclusive os próprios autores afirmaram que o reuso de especificações é uma abordagem nova e que ainda precisa ser melhorada;
- reuso de especificações é visto pelos autores como algo muito bom de ser feito. Inclusive [HAL92] descreve que ferramentas com suporte ao reuso de especificações representam uma nova geração de ferramentas CASE. No trabalho de [YOU95] é expressa sua excitação ao falar sobre o assunto e que ferramentas de apoio ao reuso de especificação e geração de código poderão eliminar os esforços envolvidos em projetar, codificar e testar a implementação de uma especificação reusada.

As principais sugestões para trabalhos futuros inclui:

- melhorar o processo de reuso de MER, estudando os componentes relacionados passíveis de reuso. Por exemplo, se uma entidade "Pedido" fosse reutilizada, a entidade "Itens de Pedido" também poderia ser, inclusive o relacionamento entre elas;
- melhorar a interface da ferramenta a fim de tornar o desenvolvimento dos diagramas mais flexível. Agregar à ferramenta um módulo para fazer consistência do DFD e MER criados;
- reuso poderia ser ampliado para outras especificações, tais como o Diagrama Hieráquico Funcional, lista de eventos, especificações orientadas a objeto, etc;
- permitir o reuso de especificações criadas em outras ferramentas CASE, através da importação de dados do repositório destas ferramentas.

REFERÊNCIAS BIBLIOGRÁFICAS

[BAR94] BARBIERI, Carlos. **Modelagem e administração de dados**. IBPI Press, 1994.

[FIS90] FISHER, Alan S. **CASE – Utilização de ferramentas para desenvolvimento de Software**. São Paulo: Campus, 1990.

[FRE86] FREEMAN, Peter. **Tutorial: Software Reusability**. Washington : Computer Society Press of the IEEE, 1986.

[GAM95] GAMMA, Erich. **Design Patterns: Elements of Reusable Object-Oriented Software**. Massachusetts: Addison-Wesley Publishing Company, 1995.

[GAN90] GANE, Chris. **Case – O Relatório Gane**. Rio de Janeiro : Livros Técnicos e Científicos, 1990.

[GRA97] GRAHL, Everaldo Artur. **Apostila sobre Reutilização de Software**. 1997.

[HAL92] HALL, P. A. V. e Chapman. **Software Reuse and Reverse Engineering in Practive**. London, 1992.

[HUG98] HUGO, Marcel e Grahl, Everaldo Artur. **Apostila sobre metodologias de desenvolvimento de sistemas**. Blumenau, 1998.

[JUN97] JUNIOR, Osmar de Oliveira Braz. **Técnicas de Análise de Sistemas** 1997. Endereço eletrônico: http://tec1.unisul.rct-sc.br/osmarjr/.

[KUT99] KUTOVA, Marcos André Silveira. **Reuso de Software** 1999. Endereço eletrônico: http://www.intermidia.icmsc.sc.usp.br/~kutova/monografias/reuso.html.

[MAI92] MAIDEN, Neil A. e SUTCLIFFE, Alistair G. **Exploiting Reusable Specifications through Analogy**. Communications of the ACM. Abril 1992, Vol. 35. No. 4.

[MCC92] MCCLURE, Carma. **The Three Rs of Software Automation**. New Jersey : Prentice-Hall, 1992.

[MEN89] MENDES, Sueli. **Métodos para Especificação de Sistemas**. São Paulo : Edgard Blücher, 1989.

[MIT90] MITTERMEIR, Roland T. & Rossak, Wilhelm. **Reusability**. Austria, Institut F. Informatik Universitaet Klagenfurt, 1990.

[PIM99] PIMENTA, Alexandre. **Uma Ferramenta de Apoio à Reutilização para o PROSOFT** 1999. Endereço eletrônico: http://www.inf.ufrgs.br/~pimenta/artigo.html.

[POM94] POMPILHO, S. **Análise Essencial – Guia Prático de Análise de Sistemas**. Rio de Janeiro : IBPI Press, 1994.

[PRE95] PRESSMAN, Roger S. **Engenharia de Software**. São Paulo : Makron Books do Brasil Editora Ltda., 1995.

[RAM98] RAMOS, Débora Cristina Leira. **Ferramenta para Gerenciamento de Componentes Reutilizáveis em Access**. Blumenau, 1998. Trabalho de Conclusão de Curso – Centro de Ciências Exatas e Naturais. Universidade Regional de Blumenau.

[ROC87] ROCHA, Cavalcanti da e Regina, Ana. **Análise e Projeto Estruturado de Sistemas**. Rio de Janeiro: Campus, 1987.

[ROS94] ROSCA, Daniela. **Evolution and Reuse of Formal Specifications using Decision Structures and Analogy**. Norfolk, 1994.

[SPA99] SPANOUDAKIS, George e Constantopoulos, Panos. **Analogical Reuse of Requirements Specifications: A Computational Model**. 1999. Endereço eletrônico: http://www.cs.city.ac.uk/homes/gespan/publications.html

[YOU90] YOURDON, E. **Análise Estruturada Moderna**. Rio de Janeiro: Campus, 1990.

[YOU95] YOURDON, E. **Declínio e Queda dos Analistas e dos Programadores**. São Paulo: Makron Books, 1995.

[VER98] VERDIERI, Edna. **Reutilização de Software** 1998. Endereço eletrônico: http://www.unisul.rct-sc.br/unisul/cursos/ccp/engsoft/trabalhos/98b/ trab7/ objetivo.htm

[ZIR97] ZIRBES, Sérgio Felipe. **Reutilização de Modelos de Requisitos por Analogia: Experimentação e Conclusões**. Anais do IX Simpósio Brasileiro de Engenharia de Software. Recife, 1997. p. 415-429.