

UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIAS DA COMPUTAÇÃO
(Bacharelado)

**SOFTWARE PARA IDENTIFICAÇÃO DE COMPONENTES
REUSÁVEIS EM ORACLE**

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

SIMONE ÂNGELA SCHURT

BLUMENAU, JULHO/1999.

1999/1-54

SOFTWARE PARA IDENTIFICAÇÃO DE COMPONENTES REUSÁVEIS EM ORACLE

SIMONE ÂNGELA SCHURT

ESTE TRABALHO DE CONCLUSÃO DE CURSO FOI JULGADO ADEQUADO
PARA OBTENÇÃO DOS CRÉDITOS DA DISCIPLINA DE TRABALHO DE
CONCLUSÃO DE CURSO OBRIGATÓRIO PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

Prof. Everaldo Artur Grahl - ORIENTADOR

Prof. José Roque Voltolini da Silva - COORDENADOR DO TCC

BANCA EXAMINADORA

Prof. Everaldo Artur Grahl

Prof. Dalton Solano dos Reis

Prof. Wilson Pedro Carli

DEDICATÓRIA

Dedico este trabalho a minha família e à minha tia,
por tudo que fizeram por min.

AGRADECIMENTOS

Agradeço aos professores de Bacharelado em Ciências da Computação da Universidade Regional de Blumenau, por todo conhecimento que adquiri ao longo do curso. Em especial ao professor Everaldo Artur Grahl pela dedicação, orientação e incentivo na condução deste trabalho.

Aos meus pais, irmãs e tia Hedwig Weiss, sem os quais nada disso teria se realizado. Agradeço especialmente ao meu namorado Cleones Sebastião Alves pelo apoio, incentivo e compreensão que recebi, o qual foi de fundamental importância para conclusão deste trabalho.

A todos que de alguma forma contribuíram para a realização deste trabalho.

SUMÁRIO

LISTA DE FIGURAS.....	VIII
LISTA DE TABELAS.....	IX
RESUMO	X
ABSTRACT.....	XI
1 INTRODUÇÃO	1
1.1 ORIGEM.....	1
1.2 OBJETIVOS.....	2
1.3 ORGANIZAÇÃO DO TEXTO.....	2
2 REUTILIZAÇÃO.....	3
2.1 CONCEITOS.....	3
2.2 PROBLEMAS NA REUTILIZAÇÃO DE SOFTWARE	3
2.3 BENEFÍCIOS	6
2.4 COMPONENTES REUSÁVEIS.....	7
2.5 REQUISITOS PARA UM COMPONENTE SER REUSÁVEL.....	8
2.6 PROCESSO DE REUTILIZAÇÃO.....	9
2.7 IDENTIFICAÇÃO DE COMPONENTES.....	11
2.7.1 FASES PARA IDENTIFICAÇÃO DE COMPONENTES	12
2.7.2 MÉTRICAS PARA IDENTIFICAÇÃO DE COMPONENTES.....	13
2.7.2.1 VOLUME	14
2.7.2.2 COMPLEXIDADE CICLOMÁTICA	15
2.7.2.3 REGULARIDADE.....	16
2.7.2.4 FREQUÊNCIA DE REUTILIZAÇÃO	16
2.7.3 CRITÉRIOS PARA APLICAÇÃO DE MÉTRICAS.....	17
2.7.4 ESTUDOS DE CASOS.....	17
3 AMBIENTE ORACLE	20
3.1 BANCO DE DADOS RELACIONAL.....	20

3.2 FERRAMENTAS ORACLE	22
3.2.1 ORACLE FORMS	22
3.2.2 ORACLE REPORTS	24
3.2.3 SQL*PLUS.....	25
3.2.4 PL/SQL	26
3.2.5 DESIGNER/2000.....	26
4 FERRAMENTA PARA REUTILIZAÇÃO DE SOFTWARE.....	28
4.1 DEFINIÇÃO DOS RECURSOS	28
4.2 DEFINIÇÃO DO SISTEMA	28
4.2.1 DIAGRAMA DE CONTEXTO	29
4.2.2 MODELO ENTIDADE RELACIONAMENTO	29
4.2.3 DIAGRAMA DE FLUXO DE DADOS	32
4.2.4 DIAGRAMA HIERÁRQUICO FUNCIONAL.....	33
4.3 IMPLEMENTAÇÃO.....	34
4.3.1 DESCRIÇÃO DO SISTEMA	34
4.3.2 MENU	36
4.3.3 CADASTRO	36
4.3.3.1 IDENTIFICAR COMPONENTE	36
4.3.3.2 SISTEMAS.....	38
4.3.3.3 TIPO DE COMPONENTES.....	39
4.3.3.4 COMPONENTES.....	39
4.3.3.5 MÉTRICAS.....	40
4.3.4 CONSULTA	41
4.3.5 RELATÓRIO	42
4.4 TESTES	43
5 CONCLUSÃO.....	46
5.1 CONSIDERAÇÕES FINAIS	46
5.2 SUGESTÕES.....	46
ANEXO 1 – LISTAGEM DO COMPONENTE.....	47
ANEXO 2 – RELATÓRIO DE COMPONENTES REUSÁVEIS.....	52

REFERÊNCIAS BIBLIOGRÁFICAS..... 53

LISTA DE FIGURAS

1	PROPOSTA DE UM AMBIENTE DE REUTILIZAÇÃO	10
2	FASES DE IDENTIFICAÇÃO DE COMPONENTES	12
3	MÉTRICAS PARA IDENTIFICAÇÃO DE COMPONENTES REUSÁVEIS	15
4	AMBIENTE DE DESENVOLVIMENTO DO ORACLE FORMS	23
5	AMBIENTE DE DESENVOLVIMENTO DO ORACLE REPORTS	25
6	FERRAMENTAS DO DESIGNER/2000	27
7	DIAGRAMA DE CONTEXTO.....	29
8	MER (MODELO ENTIDADE RELACIONAMENTO)	30
9	DIAGRAMA DE FLUXO DE DADOS	33
10	DIAGRAMA HIERÁRQUICO FUNCIONAL.....	34
11	PROJETO FÍSICO DA FERRAMENTA.....	35
12	MENU DA FERRAMENTA	36
13	TELA PARA IDENTIFICAR DADOS DO COMPONENTE	37
14	TELA DE CADASTRO DOS SISTEMAS	38
15	TELA DE CADASTRO DOS TIPOS DE COMPONENTES.....	39
16	TELA DE CADASTRO DOS COMPONENTES.....	40
17	TELA DE CADASTRO DAS MÉTRICAS.....	41
18	TELA DE CONSULTA DOS VALORES DAS MÉTRICAS DOS COMPONENTES	42
19	TELA DE PARÂMETROS PARA RELATÓRIO DE COMPONENTES REUSÁVEIS	43

LISTA DE TABELAS

1	SOLUÇÕES PARA OS PROBLEMAS DA REUSABILIDADE.....	5
2	SOLUÇÕES PARA OS PROBLEMAS DA REUSABILIDADE.....	5
3	VALORES MÉDIOS DE MEDIDAS OBTIDAS.....	18
4	COMPONENTES COM FREQUÊNCIA DE REUTILIZAÇÃO MAIOR QUE 0.5.....	18
5	EXTREMOS DE VALORES ACEITÁVEIS NO MODELO DE REUTILIZAÇÃO.....	19
6	NOTAÇÃO UTILIZADA NO MER.....	29
7	ENTIDADE SISTEMA.....	30
8	ENTIDADE TIPO_COMPONENTE.....	31
9	ENTIDADE METRICA.....	31
10	ENTIDADE COMPONENTE.....	31
11	ENTIDADE COMPONENTE_METRICA.....	32
12	ENTIDADE PROCEDIMENTO.....	32
13	TABELA DE COMPONENTES ANALISADOS PELA FERRAMENTA.....	44
14	LIMITES DAS MÉTRICAS UTILIZADOS PELA FERRAMENTA.....	44
15	TABELA DE COMPONENTES REUSÁVEIS.....	45

RESUMO

Uma das atividades típicas no reuso de software é a identificação de componentes reusáveis. Este trabalho tem como objetivo aplicar técnicas estudadas na identificação de componentes no ambiente ORACLE. Estas técnicas de identificação auxiliam a encontrar, compreender e combinar os componentes. A ferramenta desenvolvida se utiliza de métricas para identificar os componentes e abrange o reaproveitamento de instruções desenvolvidas em *Forms* e *Reports* do ORACLE.

ABSTRACT

One of the typical features in software reuse is the identification of reusable components. The purpose of this issue is to apply studied techniques in component identification within ORACLE environment. These identification techniques help to find, understand and combine components. The developed tool makes use of metrics to identify components and includes the reutilization of instructions developed in ORACLE Forms and Reports.

1 INTRODUÇÃO

1.1 ORIGEM

A reutilização de software é um tema que atualmente está em evidência devido a preocupação existente com os altos custos com a manutenção de software e a baixa produtividade e qualidade dos sistemas.

O desenvolvimento de sistemas tradicional é lento, rígido e de um modo geral seus procedimentos não funcionam de maneira adequada. As soluções existentes requerem novos softwares, novos procedimentos e novas estruturas gerenciais, muitas vezes, entretanto, em choque com a própria cultura da organização [ARA95].

Segundo [MCC93], reutilização de software refere-se ao reaproveitamento de certos componentes que já foram construídos e estão disponíveis para novas aplicações. Trata-se do reaproveitamento de código do programa, especificações do projeto, planos, documentação, conhecimento e qualquer outra informação utilizada para criar um software.

Existem diversas maneiras de se contextualizar a reutilização de componentes, porém todas elas tem como objetivo ressaltar os benefícios trazidos pela reutilização. Dentre estes benefícios conforme [MCC93], [CHE94] e [FUR95], pode-se enfatizar a redução dos riscos a falhas, redução dos custos de desenvolvimento, melhoria da qualidade do software, melhoria da produtividade, melhoria da manutenção, aumento de confiabilidade no sistema, redução do tempo de desenvolvimento, compartilhamento do conhecimento adquirido no planejamento do software, aceleração e simplificação do processo de desenvolvimento.

Para se identificar um componente reutilizável em sistemas faz-se necessário a utilização de técnicas específicas para encontrar, compreender e combinar os componentes reutilizáveis. Neste trabalho estas técnicas serão aplicadas no ambiente ORACLE.

O ORACLE é um conjunto de ferramentas voltadas para o desenvolvimento de sistemas abrangendo todo o ciclo de vida dos mesmos. Durante o desenvolvimento de um sistema tem-se as fases de planejamento, análise, projeto e implementação. Isto pode variar conforme a metodologia empregada e no caso do ORACLE ele incorpora todas estas fases.

Diante deste contexto mostra-se oportuno a criação de um software para identificação dos componentes reutilizáveis. Desta forma pode-se obter um componente a ser reutilizado em um novo desenvolvimento, ou para manutenção de um sistema já existente de acordo com as necessidades do projeto que está sendo realizado.

1.2 OBJETIVOS

Este trabalho tem por objetivo principal especificar e implementar um software em Access que facilite a identificação de componentes reusáveis em ORACLE. Os componentes vão abranger instruções codificadas em ferramentas *Forms* (versão 4.5) e *Reports* (versão 2.5).

Os objetivos específicos abrangem o estudo da reusabilidade e o estudo de técnicas de identificação de componentes.

1.3 ORGANIZAÇÃO DO TEXTO

A seguir serão descritos brevemente cada capítulo do trabalho.

Este capítulo de introdução apresenta uma visão geral da origem deste trabalho, sua relevância e objetivos.

O segundo capítulo apresenta conceitos de reutilização, problemas e benefícios da reutilização, componentes reutilizáveis, processo de reutilização, técnicas e métricas para identificação de componentes e estudos de casos.

O terceiro capítulo apresenta as características do ambiente ORACLE, suas ferramentas e características.

O quarto capítulo apresenta a especificação e a descrição da ferramenta desenvolvida.

No último capítulo são apresentadas as conclusões e sugestões do trabalho.

2 REUTILIZAÇÃO

2.1 CONCEITOS

Produtividade e qualidade são aspectos críticos no desenvolvimento de software. Desenvolvedores de software são cada vez mais solicitados a fazer mais, com menos recursos, entregar os sistemas solicitados em prazos menores, reduzir custos e tempo de manutenção, aumentar os níveis de desempenho, confiabilidade e aumentar a segurança dos sistemas.

Isto vem a demonstrar o crescente consenso no uso da metodologia da reutilização que conforme [CHE94] define como sendo a utilização de uma técnica que pode ser empregada para reduzir os esforços despendidos no desenvolvimento e manutenção de programas. Segundo [MCC93], reutilização de software nada mais é do que o reaproveitamento de partes de um sistema que já foi construído e estão disponíveis para novas aplicações.

Com base nas definições citadas pode-se concluir que a reusabilidade visa o aproveitamento de qualquer produto de trabalho, anteriormente desenvolvido na construção de software, em qualquer fase do ciclo de vida, para a elaboração de novos sistemas.

2.2 PROBLEMAS NA REUTILIZAÇÃO DE SOFTWARE

Segundo [KUT97] um dos dilemas que vem interferindo na utilização integral da reutilização é a falta de componentes de software reusáveis ou a existência de componentes de software que são difíceis de encontrar, entender, adaptar e integrar.

Estes problemas são destacados também por [GRA96] e [SIL94], entretanto [TRA93] classifica estas dificuldades em três problemas gerais, que são:

- 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 ?

Além dos três grandes problemas [KUT97] lembra que o custo para construir componentes reusáveis, a falta de qualidade dos componentes encontrados em uma biblioteca, a dificuldade de modificar componentes para adequá-los a um novo software, ineficiência em

geral dos componentes existentes e a possibilidade de violação de direitos autorais, também exercem funções problemáticas na implantação de um programa de reusabilidade.

Ainda sob um aspecto global de problemas enfrentados ao se utilizar um programa de reutilização [MCC92] relaciona alguns outros problemas, citados por [SIL94], que são comumente encontrados na prática da reusabilidade. São eles:

- a) dificuldade de encontrar componentes comuns em softwares já prontos;
- b) falta de padronização nos programas;
- c) dependências 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) o aumento (em média de 25%) de custo para desenvolver um componente de software projetando-o para a reusabilidade;
- j) o desestímulo causado pelos benefícios da reusabilidade serem de longo prazo;
- k) a falta de prática de readaptação de componentes já existentes à reutilização.

No trabalho de [KUT97] são destacadas tecnologias para solucionar os problemas por ele citados, conforme mostra a tabela 1. Conforme apontado pelos autores, os problemas para a implantação da reusabilidade são muitos por isso um programa de reusabilidade deve ser estruturado de maneira que todos os problemas técnicos possam ser vencidos. Isto será um grande passo para se obter o sucesso esperado da reusabilidade.

No Workshop Internacional de Reusabilidade de Software ocorrido em julho de 1993 na Califórnia, onde os vários problemas foram apontados e discutidos, várias soluções para eles também foram apresentadas. A tabela 2 demonstra as soluções apresentadas por [TRA93].

Tabela 1 - Soluções para os problemas da reusabilidade

Problema	Solução
falta de componentes reusáveis (problema de população)	análise de domínio; aquisição/ representação de conhecimento; padronização de construtores.
aumento do custo de desenvolvimento para fazer componentes voltados a reutilização	ambientes de reengenharia; linguagens de programação que utilizem prototipação; geradores de aplicação.
falta de qualidade dos componentes	interfaces com especificações formais; verificações constantes nas implementações; terminologia padrão.
dificuldade de entender como utilizar componentes reusáveis	animação de algoritmos; visualização de dados;
dificuldade de modificar componentes	parametrização; polimorfismo;
ineficiência dos componentes	linguagens extensíveis; compiladores;
violação de direitos autorais	criptografia; autenticação; segurança geral do software.
dificuldade para integrar/compor componentes	linguagens de prototipação; captura/manipulação de conhecimento; ferramentas de integração e verificação de componentes.

Fonte: [KUT97]

Tabela 2 - Soluções para os problemas da reusabilidade

Problema	Solução
de população	análise de domínio; bibliotecas de construção; geração de componentes pelo domínio de linguagens.
de carga	esquemas de classificação, tais como o de facetas.
de construção	linguagens com propriedades genéricas, porém não há escopo de semântica da aplicação.

Fonte: [TRA93]

2.3 BENEFÍCIOS

A reutilização de software significa que as idéias e o código são desenvolvidos uma vez, e então utilizados para resolver muitos problemas de aplicação, deste modo aumentando a produtividade e qualidade. [MCD93] estima aproximadamente que 60% dos códigos no processo de dados são desenvolvidos desnecessariamente, e poderiam ser padronizados e reutilizados.

O aumento da qualidade é devido à diminuição do esforço necessário para a produção de código novo (programa novo). O aumento da qualidade advém do fato do código reutilizado já ter sido amplamente usado, modificado e testado em outros sistemas eliminando boa parte dos erros que poderiam ocorrer. Por exemplo, os japoneses relataram em 1980 que, através da promoção da reutilização, eles alcançaram um aumento de produtividade de 14% por ano durante vários anos (isto dobra a produtividade em cinco anos) [MCC93].

A razão pela qual atualmente está havendo um aumento no uso da reutilização, são os benefícios oferecidos que incluem conforme [ARA95]:

- a) redução do risco a falhas;
- b) melhoria da qualidade do software;
- c) melhoria da produtividade;
- d) melhoria da manutenção;
- e) os custos no desenvolvimento de software são reduzidos. Com a reusabilidade, menos componentes de software precisam ser especificados, projetados, implementados e testados;
- f) aumento da confiabilidade no sistema. Componentes reusáveis, que foram testados em sistemas que estão funcionando, são mais confiáveis que novos componentes;
- g) o tempo de desenvolvimento do software é reduzido. O reuso do software pode ser compartilhado. Reutilizar componentes de software necessita de um conhecimento dos planejadores. Pelo reuso de componentes, os módulos reutilizados compartilham entre os planejadores os conhecimentos deste componente;
- h) aceleração e simplificação do processo de desenvolvimento.

2.4 COMPONENTES REUSÁVEIS

Segundo [CAT96], componentes reusáveis representam um produto de trabalho do desenvolvimento de software que pode ser reaproveitado para a construção de um novo software. Apresenta-se, a seguir, uma especificação detalhada do que podem ser componentes reusáveis, na visão de vários autores.

Conforme [MCC93], podem ser componentes reusáveis:

- a) código de programa (programas inteiros, fragmentos de código);
- b) especificações de projeto (modelo de dados lógico, estruturas de processo, modelo de aplicação);
- c) planos (planos de controle de projeto, planos de teste);
- d) perícia e experiência (reuso do ciclo de vida, garantia da qualidade, conhecimento da área de aplicação);
- e) qualquer informação usada para criar software e documentação de software.

Na visão de [CAT96], componentes reusáveis são:

- a) requisitos;
- b) especificações de software e interface;
- c) arquitetura;
- d) projetos;
- e) códigos;
- f) testes;
- g) resultados de testes;
- h) documentos formais e informais usados pelos desenvolvedores em qualquer estágio do ciclo de vida.

Ainda, segundo [HAL94], dados também podem ser componentes reusáveis, no que se refere a uma padronização de formato de dados. Finalmente, de acordo com [KUT97], componentes reusáveis também podem ser objetos.

Em seu trabalho, [MCC93] afirma que existem muitas oportunidades para a prática de reuso porque existe muita redundância no desenvolvimento de sistemas de software:

- a) 40 a 60 % de todo código é reusável de um aplicação para outra;
- b) 60 % dos projetos de todas as aplicações do negócio são reusáveis;

- c) 75 % das funções de programas são comuns para múltiplos programas;
- d) 15 % do código de programas é único para uma aplicação específica.

2.5 REQUISITOS PARA UM COMPONENTE SER REUSÁVEL

Segundo [HAL94], nem todos os componentes de um sistema de software podem ser reutilizados. Um componente para ser reutilizado deve satisfazer critérios específicos como:

- a) funcionalidade:
 - aplicabilidade: é a probabilidade de, dado uma escolha aleatória de uma aplicação, um projeto existente que reusa um componente;
 - generalidade: quanto mais geral é um componente, mais reusado será;
 - coesão e acoplamento: a interface que um componente apresenta externamente será uma posição de uma unidade altamente conceitual e a dependência com outros componentes será pequena;
- b) ambiente:
 - portabilidade: a habilidade para usar parte de um software inalterado em outro ambiente diferente do que foi originalmente projetado;
- c) qualidade: claramente um componente não será utilizado se o mesmo não for considerado de alta qualidade.

Os requisitos de componentes reusáveis para [MCC92] são:

- a) independência: os componentes devem ser independentes e ter um comportamento previsível;
- b) isolamento de causa e efeito: os componentes devem executar seu comportamento independentemente de causas precedentes e efeitos subsequentes;
- c) auto-organização: os componentes devem saber de quais outros componentes eles precisam, de forma que o programador não tenha de se lembrar deles ou procurá-los.

Cita-se, a seguir, os critérios formulados por [MCC93]:

- a) aditividade: apto para combinar componentes com um mínimo de efeitos negativos;

- b) base matemática formal: admitir condições corretas a serem declaradas e combinação de componentes para preservar propriedades chaves dos mesmos;
- c) reservado: cada componente está embutido numa idéia;
- d) facilmente descrito: fácil de compreender;
- e) linguagem de programação independente;
- f) verificável: fácil de testar;
- g) interface simples: um número mínimo de parâmetros passados explicitamente;
- h) facilmente alterado: fácil para modificar com um mínimo de efeitos;
- i) reusável: tem alto potencial de reuso, grande probabilidade para ser usado em muitos sistemas.

Segundo [ARA95], existem ainda outras características que ajudam a identificar componentes altamente reusáveis como:

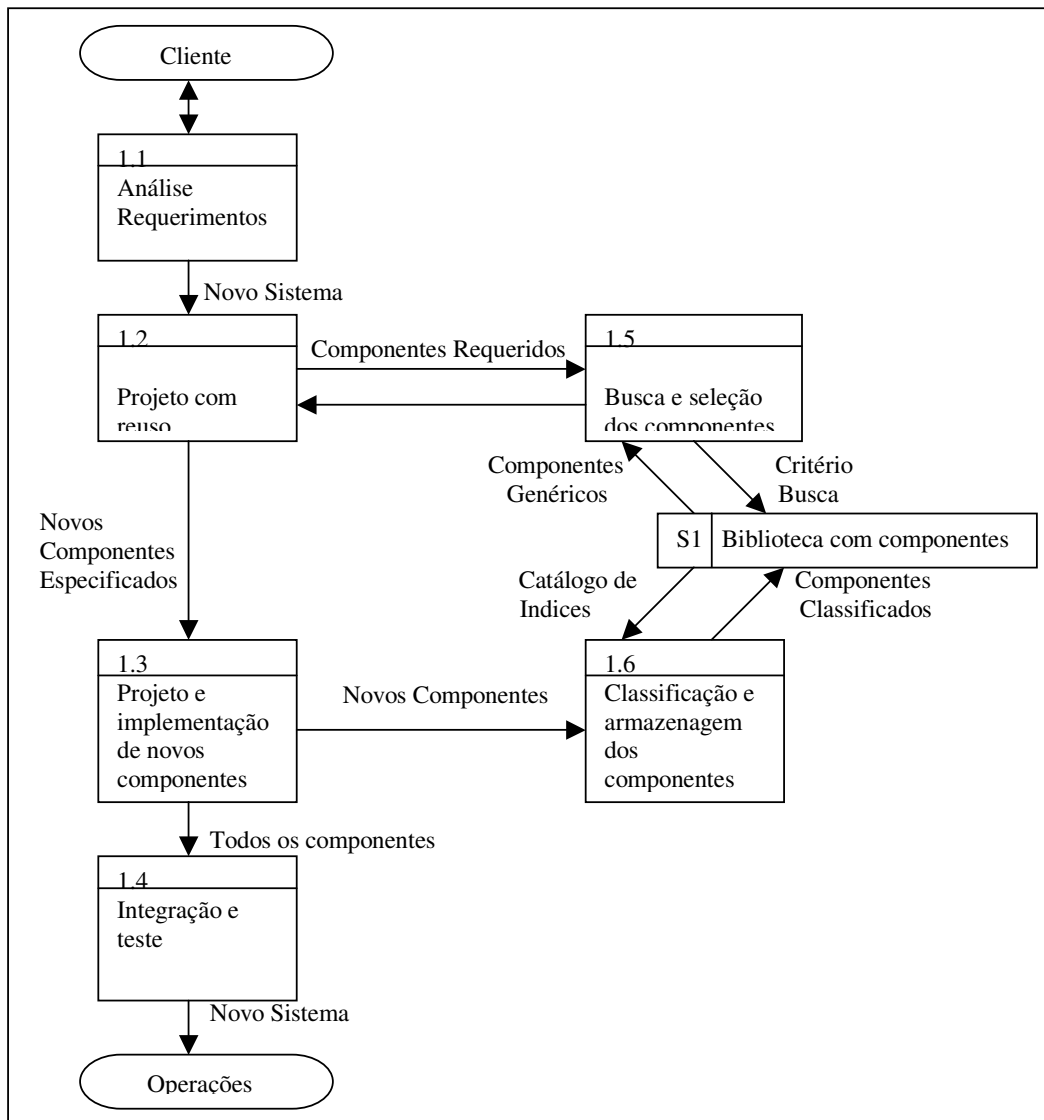
- a) componentes pequenos;
- b) bem documentados;
- c) interface simples – independente da implementação do componente;
- d) pouca entrada/saída;
- e) lógica simples e sequencial.

2.6 PROCESSO DE REUTILIZAÇÃO

Segundo [MCD93], o tradicional modelo de ciclo de vida de software requer uma dimensão acrescida para acomodar a possibilidade da reutilização em várias fases. Um modelo de reutilização deve estar capacitado a reutilizar componentes, sejam estes rotinas, objetos, idéias, especificações, *designs* ou algoritmos. O número de passos deve ser seguido no procedimento desde o reconhecimento da oportunidade de reutilização até a real utilização do componente em novas aplicações.

Um modelo de reutilização é apresentado na figura 1, segundo [MCD93]. Nas fases 1.1 à 1.4 representam um modelo tradicional de desenvolvimento de software e as fases 1.4 e 1.6, juntamente com uma biblioteca para armazenar os componentes, foram incorporados para permitir a reutilização. Na fase 1.5 é feita uma busca na biblioteca de componentes e seleção do componente necessário para o sistema. No caso de componentes novos, a possibilidade de reutilização deve ser considerada. Novos elementos do software que são necessários devem

ser considerados candidatos e adicionados. Após identificar o componente que é potencialmente reutilizável (fase 1.3), deve-se classificá-lo (fase 1.6) e armazená-lo na biblioteca de componentes reutilizáveis. Estes componentes devem ser o mais geral possível evitando-se super-generalização, o que poderia dificultar usos específicos ineficientes na operação. Para permitir a reutilização, a especificação deve fornecer informações claras da teoria e conceitos fundamentais do software, independente de qualquer implementação particular. Tendo selecionado um componente para reutilização, resta adaptar o componente para o seu uso e compô-lo com outros componentes e para alcançar os resultados desejados.



Fonte: [MCD93]

Figura 1 – Proposta de um ambiente de reutilização

Como o objetivo do trabalho está voltado para a identificação de componentes reusáveis, apresenta-se no item seguinte o funcionamento deste processo.

2.7 IDENTIFICAÇÃO DE COMPONENTES

Segundo [CAT96], a identificação de componentes é a análise de software já existente para extração de candidatos a componentes reusáveis. A identificação de software é importante porque permite reaver trabalho, conhecimento e experiência embutidos em aplicações já desenvolvidas.

A identificação de código reusável possui duas abordagens distintas. Existem técnicas baseadas na transformação de componentes existentes em componentes reusáveis e técnicas baseadas na identificação de componentes através do uso de métricas.

A identificação de componentes reusáveis através da transformação de componentes está baseada na sua generalização. O objetivo é eliminar as dependências externas de um componente. Pode-se citar a análise de domínio e a reengenharia.

Segundo [ARA95], a análise de domínio é o processo de identificação, coleção, organização, análise e representação de um modelo de domínio e arquitetura do software mediante um estudo dos sistemas existentes. Esta análise é importante na classificação do software, porque permite uma descrição da arquitetura e os diferentes relacionamentos entre os componentes.

Segundo [FUR95], reengenharia é o conjunto de técnicas orientadas à avaliação, reposicionamento e transformação de sistemas de informação existentes, com o objetivo de estender-lhes a vida útil e ao mesmo tempo, proporcionar-lhes uma melhor qualidade técnica e funcional.

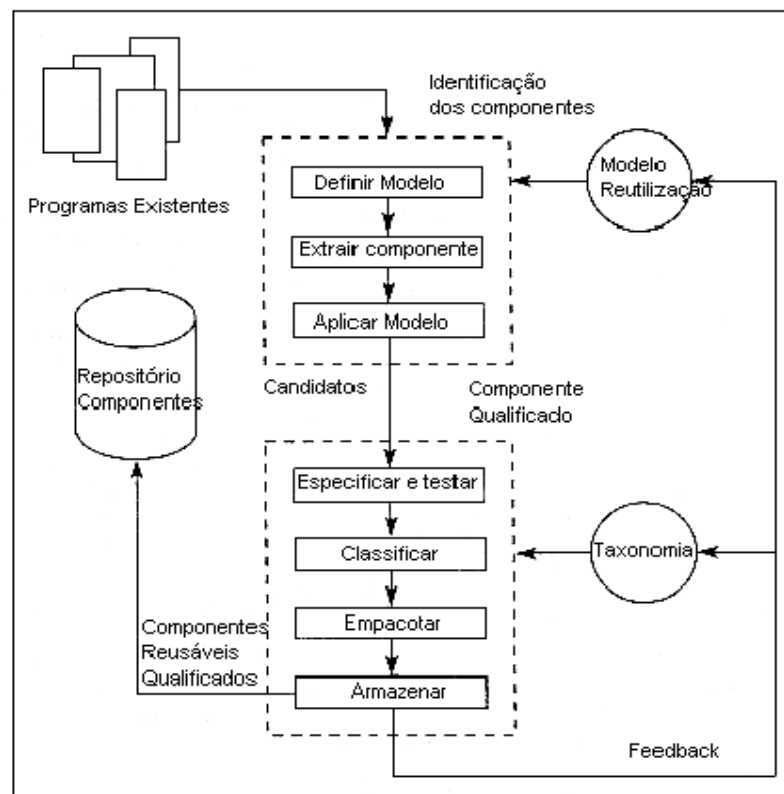
A identificação de componentes reusáveis através de métricas consiste em levantar todos os candidatos a componentes de sistemas existentes e aplicar um conjunto de métricas para verificar seu potencial de reuso. A utilização de métricas leva em consideração o grande volume de código a ser analisado, fornecendo um modo de automatização da fase de análise. Além disso, métricas permitem o *feedback* e melhorias, para facilitar o processo de extração em vários ambientes.

A técnica utilizada para a identificação de componentes neste trabalho é a utilização de métricas. No ítem 2.7.2 é apresentado este processo de forma mais detalhada.

2.7.1 FASES PARA IDENTIFICAÇÃO DE COMPONENTES

Conforme [ARN94], o processo de identificação de componentes se divide em duas fases como mostra a figura 2. Primeiro, escolhe-se os candidatos para possível reuso. Depois um analista com conhecimento do domínio da aplicação analisa o componente para determinar sua utilidade. Então, os componentes são armazenados com todas as informações que foram obtidas sobre eles.

A primeira fase pode ser totalmente automatizada. A intervenção humana necessária na segunda fase é a razão para divisão em duas fases, ao invés de procurar diretamente em programas existentes os componentes que podem ser reutilizáveis. A primeira fase reduz a quantidade de análises humanas necessárias na segunda fase através da limitação de componentes que sejam realmente válidos de serem analisados.



Fonte: [ARN94]

Figura 2 – Fases de identificação de componentes

Na fase de identificação dos componentes, unidades de programas são automaticamente extraídas, feitas independentemente e medidas de acordo com propriedades observadas relacionadas ao seu potencial de reutilização. Sendo assim, a fase de identificação consiste em três passos:

- a) **definição dos modelos de reusabilidade:** usando características de um componente potencialmente reutilizável, deve ser definido um conjunto de medidas automatizadas que capturem estas métricas e seus valores usando as orientações nos próximos passos e modificando-as até obter modelos de reusabilidade que maximizem as chances de selecionar candidatos para reutilização;
- b) **extração dos componentes:** extração de unidades de sistemas existentes e complementação para que possam ser utilizadas independentemente;
- c) **aplicação do modelo:** o modelo atual é aplicado a componentes completos extraídos. Componentes que possuam medidas que estão dentro dos limites de reusabilidade tornam-se candidatos a componentes a serem analisados pelos especialistas em domínio na fase de qualificação.

2.7.2 MÉTRICAS PARA IDENTIFICAÇÃO DE COMPONENTES

De acordo com [ARN94], um componente é simplesmente um reservatório para conter abstrações de expressões de estruturas de dados e algoritmos. Os atributos que tornam um componente reutilizável como bloco de construção para outros, são:

- a) **utilidade funcional no contexto da aplicação do domínio:** a utilidade funcional é afetada tanto pela familiaridade quanto pela variedade das funções realizadas pelo componente. A familiaridade de um componente pode ser dividida em três partes: a familiaridade dentro do sistema ou aplicação, a familiaridade comparando-se a sistemas diferentes do mesmo domínio ou familiaridade geral. Uma medida automatizada da utilidade funcional pode ser o número de vezes que a função ocorre dentro de um sistema analisado. O modelo de reutilização mede a utilidade funcional do componente, derivada da familiaridade de funções realizadas pelo mesmo, comparando o número de vezes que o componentes é invocado no sistema com o número de vezes que um componente útil é invocado. Componentes úteis podem ser encontrados em bibliotecas padrões dos ambientes de programação.

Quando estes ambientes estão próximos, a implementação do componente é regular. Alta regularidade sugere que a complexidade do componente indica a quantidade de funções que ele executa;

- b) **redução de custos:** custos de reutilização incluem custos de extração do sistema antigo, empacotamento, modificação e integração com o novo sistema. Pode-se medir os custos durante o processo ou utilizar métricas para prevêê-los. Para minimizar os custos de procura e extração, é necessário fragmentos de código pequenos e simples. Os custos podem ser influenciados pela capacidade de leitura do fragmento, volume, complexidade, redundância e estruturação da implementação dos componentes;
- c) **qualidade:** qualidades importantes na reutilização são: correção, capacidade de leitura, capacidade de efetuar testes, facilidade de modificação e performance.

Sintetizando estas considerações, a identificação de um componente reusável pode ser efetuada através de quatro métricas: volume, complexidade ciclomática, regularidade e frequência de reutilização, mostradas na figura 3.

2.7.2.1 VOLUME

Segundo [PRE95], o volume pode ser medido baseando-se no modo em que o programa utiliza a linguagem de programação. Primeiro define-se operadores e operandos.

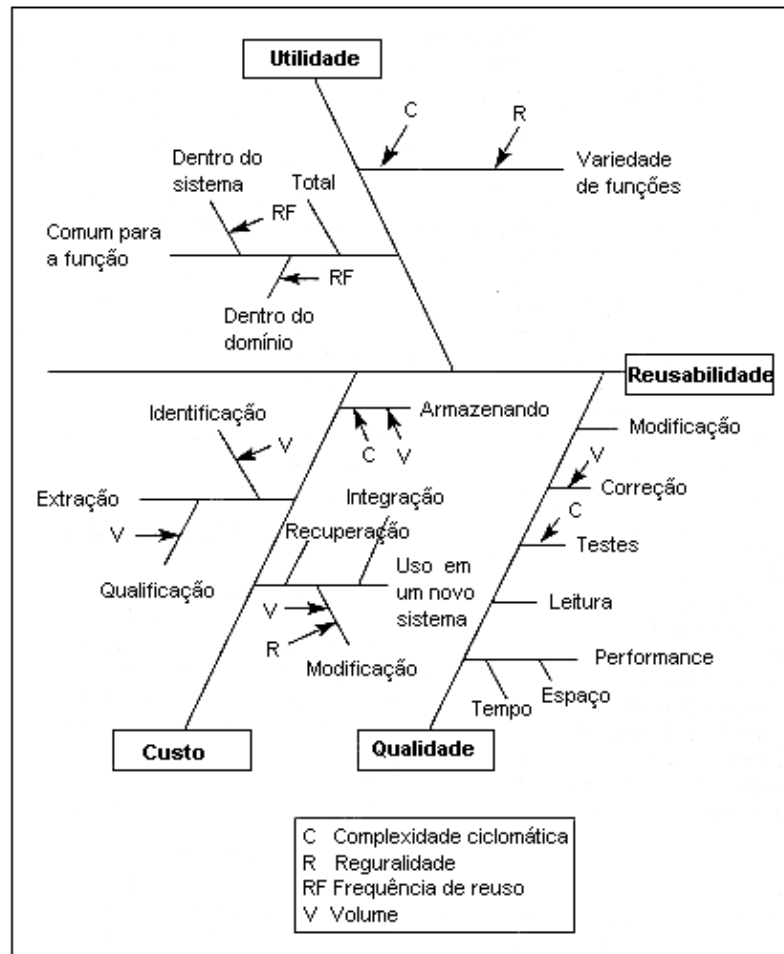
Os operadores representam elementos ativos do programa: operadores aritméticos, operadores de atribuição, funções, etc. Alguns operadores são fornecidos pela linguagem e alguns são definidos pelo usuário de acordo com as regras da mesma. O número de operadores distintos usados no programa é denotado por $n1$ e a contagem total de utilização destes operadores por $N1$.

Os operandos representam os elementos passivos do programa: constantes, variáveis, etc. O número de operandos distintos é denotado por $n2$ e a contagem total de utilização por $N2$.

Usando operadores e operandos define-se a seguinte fórmula:

$$V = (N1 + N2) \log_2 (n1 + n2)$$

O volume do componentes afeta tanto o custo de reutilização quanto a qualidade. Se um componente é muito pequeno e os custos de extração, recuperação e integração excederem seu valor intrínseco, a reutilização é impraticável. Se ele é muito grande, é mais passível de erro e tem baixa qualidade.



Fonte: [ARN94]

Figura 3 – Métricas para identificação de componentes reusáveis

2.7.2.2 COMPLEXIDADE CICLOMÁTICA

Pode-se medir a complexidade ciclomática, $V(G)$, de um componente através do número de ciclos no fluxograma do programa. Pode-se obter o número de ciclos do programa aplicando-se a fórmula:

$$V(G) = e - n + 2$$

Onde e é o nº de ramos do grafo de fluxo, e n é o nº de ramificações (nós).

Assim como o volume, reutilizar um componente pouco complexo poderá não cobrir o custo, enquanto componentes muito complexos podem indicar baixa qualidade, baixa capacidade de leitura, pouca capacidade de teste e alta possibilidade de erro. Por outro lado, alta complexidade com alta regularidade de implementação sugere alta utilidade funcional.

2.7.2.3 REGULARIDADE

Pode-se medir a economia de um componente ou o uso de práticas corretas de programação através da previsão de seu tamanho baseados em adoção de medidas de regularidade, com os seguintes indicadores:

$$N = N1 + N2$$

e o tamanho estimado:

$$N3 = n1 \log_2 n1 + n2 \log_2 n2$$

A proximidade da estimativa é uma medida da regularidade da codificação do componente:

$$R = 1 - \frac{N - N3}{N} = \frac{N3}{N}$$

A regularidade do componente mede a capacidade de leitura e não-redundância da implementação, deste modo, seleciona-se componentes cuja regularidade é próxima de 1.

2.7.2.4 FREQUÊNCIA DE REUTILIZAÇÃO

A frequência de reutilização pode ser estimada comparando-se o número de chamadas de um componente ao número de chamadas de uma classe de componentes reutilizáveis. Suponha-se que um determinado sistema seja composto por componentes definidos pelo usuário $X1, \dots, Xn$ e de componentes $S1, \dots, Sn$ definidos no ambiente de programação. Para um dado componente X , seja $n(X)$ o número de chamadas endereçadas a X no sistema. Associando a cada componente definido pelo usuário uma medida estática da sua reutilização

através do sistema, a razão entre o número de chamadas endereçadas ao componente C e a média do número de chamadas endereçadas ao componente padrão.

$$V\sigma(C) = \frac{n(C)}{\frac{1}{M} \sum_{i=0}^M n(S_i)}$$

A frequência de reutilização é a medida funcional do componente, assumindo-se que o domínio da aplicação usa convenções de nomenclatura, de modo que componentes com diferentes nomes não são funcionalmente os mesmos e vice-versa. Deste modo, só é necessário um limite mínimo para esta métrica.

2.7.3 CRITÉRIOS PARA APLICAÇÃO DE MÉTRICAS

Para completar o modelo básico de reutilização são necessários alguns critérios para selecionar os candidatos a reutilização com base nos valores das métricas apresentadas. O extremo de cada medida depende da aplicação, do ambiente, dos métodos de programação, da linguagem e muitos outros fatores não facilmente quantificáveis. Os limites de aceitabilidade são determinados através de experimentação podendo variar se acordo com o ambiente em que os componentes foram desenvolvidos. No próximo ítem são apresentados alguns estudos de casos onde pode-se ter uma noção destes limites.

2.7.4 ESTUDOS DE CASOS

Para identificar componentes reutilizáveis com aplicação de métricas foi desenvolvido um sistema chamado de CARE (*Computer Aided Reuse Engineering*), o qual identifica componentes desenvolvidos em linguagem C e ADA. Neste sistema o usuário deve selecionar as métricas de uma biblioteca e fornecer a cada métrica os limites de valores aceitáveis [ARN94].

Na tabela 3 são apresentadas valores médios de cada métrica obtidos com Sistema CARE.

Tabela 3 – Valores médios de medidas obtidas

Caso	Volume	Complexidade	Regularidade	Frequência de Reutilização
A	8.967	21,1	0,76	0,05
B	7.856	23,6	0,74	0,08
C	45.707	153,7	0,66	0,10
D	11.877	32,1	0,64	0,11
E	4.054	16,8	0,76	0,18
F	82.671	198,7	0,33	0,13
G	7.277	25,5	0,65	0,24
H	12.444	40,7	0,77	0,23

Fonte: [ARN94]

Os dados da última coluna da tabela 3 estão abaixo de 0.5. Deste modo, pode-se assumir que a frequência de reutilização de um componente acima de 0.5 é de componentes altamente reutilizáveis. A tabela 4 apresenta os dados de medida de componentes de alta reutilização, os quais apresentam frequência maior que 0.5.

Tabela 4 – Componentes com frequência de reutilização maior que 0.5

Caso	Volume	Complexidade	Regularidade	Frequência de Reutilização
A	2.249	7,0	0,89	> 0,50
B	2.831	4,8	0,77	> 0,50
C	13.476	43,8	0,68	> 0,50
D	4.444	8,5	0,80	> 0,50
E	1.980	10,7	0,87	> 0,50
F	156.199	384,3	0,40	> 0,50
G	1.904	5,4	0,70	> 0,50
H	8.884	31,1	0,75	> 0,50

Fonte: [ARN94]

Comparando as tabelas 3 e 4 vê-se um padrão regular. Os componentes altamente reutilizáveis tem volume e complexidade mais baixos do que a média, em torno de $\frac{1}{4}$ da média. Sua regularidade é levemente maior do que a média, geralmente acima de 0,70.

Segundo [ARN94], os estudos de casos mostram que, na maioria dos casos, pode-se obter resultados satisfatórios usando os valores da tabela 5 como extremos para valores aceitáveis.

Tabela 5 – Extremos de valores aceitáveis no modelo de reutilização.

Métricas	Valor Mínimo	Valor Máximo
Volume	2.000	10.000
Complexidade	5,00	15,00
Regularidade	0,70	1,30
Frequência de Reutilização	0,30	

Fonte: [ARN94]

3 AMBIENTE ORACLE

Segundo [CER95], ORACLE é um Sistema de Gerenciamento de Banco de Dados Relacional (SGBDR) produzido pela Oracle Corporation em 1977. A Oracle se tornou a líder mundial em ambientes de banco de dados relacionais, contando hoje com cerca de 57% das instalações mundiais em várias plataformas. No Brasil a Oracle possui a subsidiária Oracle do Brasil Sistemas, que iniciou suas atividades em outubro de 1988, constituindo-se também em empresa líder brasileira. Já instalou 4.000 bancos de dados Oracle em mais de 1.000 empresas, variando desde plataformas de grande, médio e pequeno porte a redes cliente/servidor com estações de trabalho.

O ORACLE possibilita o armazenamento de dados em tabelas (relações). Estas relações são representações bidimensionais (linhas X colunas) dos dados, onde as linhas representam os registros e as colunas (atributos) são as partes de informação contidas no registro.

Segundo [ROS96], o ORACLE pode ser comparado a um sistema operacional sobreposto ao sistema operacional de computador onde reside. Possui sua próprias estruturas de arquivo, de buffer, áreas globais e uma capacidade de se ajustar muito além dos capacidades fornecidas no sistema operacional. O ORACLE controla seus próprios processos, monitora seus registros, consistências e limpa a memória ao sair.

3.1 BANCO DE DADOS RELACIONAL

Um banco de dados é uma coleção de dados organizados e integrados, armazenados em forma de tabelas interligadas através de chaves primárias e estrangeiras, que constituem uma representação natural dos dados, sem imposição de restrições ou modificações. Sendo assim, adequado a qualquer computador, podendo ser utilizado por todas as aplicações relevantes sem duplicação de dados, e sem a necessidade de serem definidos em programas, pois utiliza as definições existentes nas bases de dados, através do Dicionário de Dados ativo e dinâmico, relata [CER95].

Segundo [ARA95], um banco de dados relacional pode ser definido como um banco de dados que aparece ao usuário como uma coleção de tabelas relacionadas e nada além de tabelas.

O ORACLE se baseia em três principais aspectos do modelo relacional: a estrutura, as operações e as regras de integridade.

As estruturas são os objetos que guardam os dados de um banco de dados. Essas estruturas e os dados podem ser manipulados através de operações.

As operações são as ações que permitem aos usuários manipular os dados e as estruturas de um banco de dados. Essas operações devem aderir a um conjunto de regras de integridade predefinidas.

As regras de integridade são as leis que governam quais operações são permitidas nos dados e nas estruturas de um banco de dados, com o propósito de protegê-los de operações indevidas.

O objetivo principal do sistema relacional é oferecer um banco de dados de fácil compreensão e bem-definido, o que facilita sua manipulação. Quando se deseja efetuar mudanças nas definições dos bancos de dados, podem-se inserir novas colunas às tabelas, sem que se tenha de fazer qualquer alteração nos programas de aplicação.

Dentre os benefícios de um banco de dados relacional, pode-se citar:

- a) simplicidade e uniformidade (o modelo relacional é compacto);
- b) independência total dos dados;
- c) interfaces de alto nível para usuários finais;
- d) visões múltiplas de dados;
- e) melhoria na segurança dos dados;
- f) redução significativa do atravancamento de aplicações e do tempo;
- g) gasto na manutenção;
- h) alívio da carga de trabalho do CPD (Centro Processamento de Dados);
- i) possibilidade de crescimento futuro e inclusão de novos dados devido à flexibilidade do sistema e independência de dados físicos.

Todas as operações que envolvem o acesso aos dados de um banco de dados são efetuadas através de comandos SQL (*Structured Query Language* ou Linguagem de Pesquisa Estruturada). A linguagem SQL é uma simples e poderosa linguagem de acesso a um banco

de dados que é usada como padrão para os sistemas de gerenciamento de banco de dados relacional, afirma [MOR95].

3.2 FERRAMENTAS ORACLE

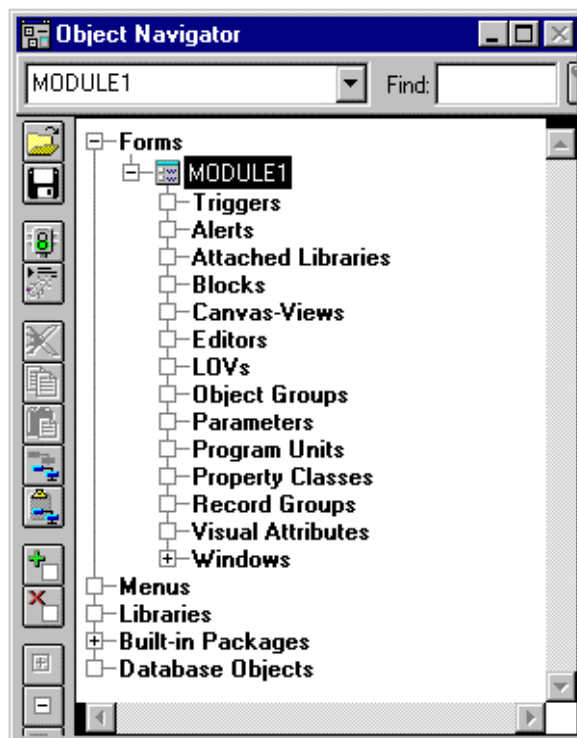
O ORACLE dispõe de uma gama variada de ferramentas que vão desde ferramentas de manipulação dos dados, de desenvolvimento e até para viabilizar comunicação com outras bases de dados, como segue.

3.2.1 ORACLE FORMS

O ORACLE Forms é uma ferramenta que permite desenvolver rápida e facilmente aplicações baseadas em telas e com controle feito através do tratamento de eventos (*triggers*). Através das aplicações geradas pelo ORACLE Forms, que utiliza a linguagem SQL, o usuário final pode realizar operações de inserção, consulta, alteração e deleção. Combina informações feitas por quem desenvolve, com informações do dicionário de dados para gerar aplicações completas. Segundo [ARA95], o ORACLE Forms implementa regras de integridade e gera mecanismos de consistência dos dados manipulados pela aplicação. O ORACLE Forms possui alguns recursos como:

- a) **Generate** - utilizado após a criação do Forms para gerar o arquivo fonte e atualizar as alterações;
- b) **Procedure** - local onde são declaradas pelo desenvolvedor as *procedures* que serão utilizadas naquele Forms não sendo visíveis para outros Forms. Procedures são um grupo de comandos SQL e PL/SQL agrupados como uma unidade lógica usada para resolver um problema ou tarefas específicas e relacionadas, em um banco de dados. Para ficar pública, uma *procedure* deve ser criada no servidor (equipamento que centraliza os dados do banco) e declarada pública.
- c) **Triggers** - tratamentos feitos para serem realizados a partir do disparo de um evento. Exemplo: quando um usuário vai mudar de um campo na tela para outro, este ato gerará em evento que pode ser tratado ou não ser tratado pelo usuário através de uma *trigger*.

Na figura 4, pode-se visualizar o ambiente de desenvolvimento do ORACLE Forms, com os blocos onde são declaradas as *triggers* e *procedures* (*Program Units*).



Fonte: [CER95]

Figura 4 – Ambiente de desenvolvimento do ORACLE Forms

A seguir serão citadas algumas vantagens do ORACLE Forms, segundo [CER95]:

- a) objetivos de interface gráfica para usuários;
- b) editor de textos integrado;
- c) acesso sem limite a dados;
- d) arquivos com formato de imagens e gráficos;
- e) capacidade de gerar aplicativos padrões;
- f) capacidade para gerar consultas padrões;
- g) protótipo de telas;
- h) capacidade procedural;
- i) suporte a idiomas pátrios;
- j) portabilidade de interface a usuário;
- k) integração total com outros produtos.

3.2.2 ORACLE REPORTS

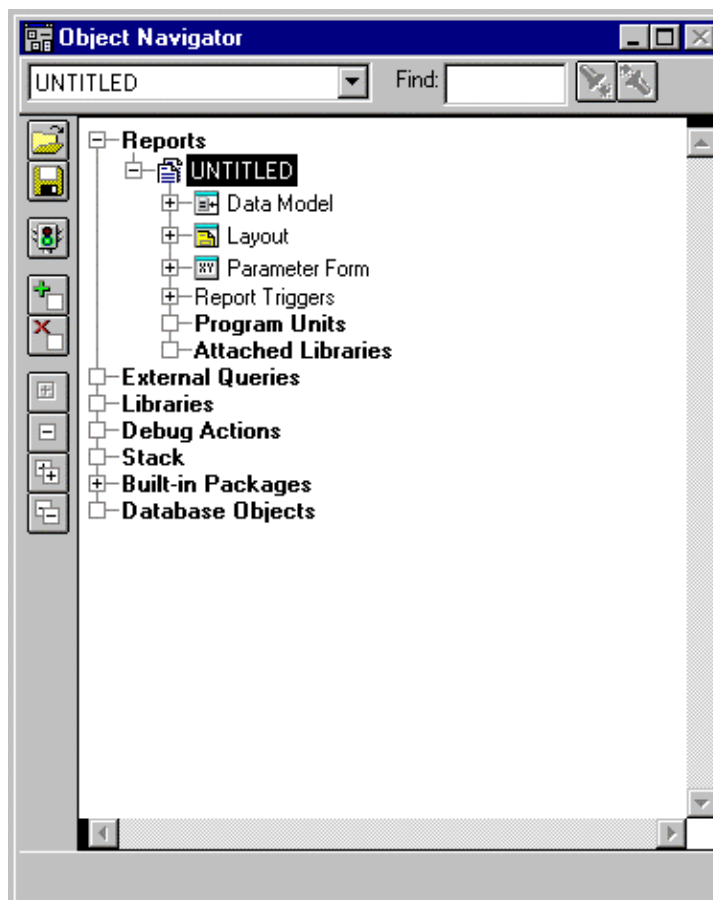
O ORACLE Reports é uma ferramenta que gera relatórios combinando gráficos e textos, formata os resultados das pesquisas para um relatório personalizado do usuário com a aplicação de alguns comandos. Podem-se especificar numeração de páginas, identificação de relatórios, cabeçalhos, rodapés, títulos, tamanhos de campos, formatos e adicionar lógica para controlar a execução do relatório. Assim como o ORACLE Forms, o ORACLE Reports também possui os recursos: *Generate*, *Procedure* e *Triggers*. O ORACLE Reports possui estilos padrões do relatório como:

- a) tabular;
- b) formulário;
- c) formato de carta;
- d) mestre / detalhe;
- e) etiquetas para endereçamento;
- f) matriz.

A seguir serão citadas algumas vantagens do ORACLE Reports, segundo [CER95]:

- a) controle de dados;
- b) capacidade extensiva de agregação;
- c) acesso sem limite a dados;
- d) formato de arquivo imagem ou gráfico;
- e) pintor gráfico comum;
- f) desenhador padrão de relatórios;
- g) capacidade procedural;
- h) suporte a idiomas pátrios;
- i) portabilidade de interface a usuário;
- j) integração a produtos;
- k) métodos de armazenamento de relatórios;
- l) segurança de relatórios.

A seguir, na figura 5 apresenta-se o ambiente de desenvolvimento do ORACLE Reports.



Fonte: [CER95]

Figura 5 – Ambiente de desenvolvimento do ORACLE Reports

3.2.3 SQL*PLUS

O SQL*PLUS é uma ferramenta de quarta geração com interface interativa com o banco de dados relacional ORACLE que tem a função de intermediar a comunicação do usuário com o banco.

[CER95] cita as seguintes características do SQL*PLUS:

- a) executar comandos SQL interativamente;
- b) manter a integridade dos dados;
- c) acesso a banco de dados ORACLE;
- d) fazer consistência;
- e) integração com o PL/SQL.

3.2.4 PL/SQL

PL/SQL (*Procedural Language/ Structured Query Language*) funciona integralmente com o servidor ORACLE, é uma extensão de SQL combinada com estruturas procedurais. Existem duas versões de PL/SQL: uma faz parte do mecanismo de banco de dados, a outra é um mecanismo incorporado a várias ferramentas ORACLE como, por exemplo, ORACLE Forms e ORACLE Reports. A PL/SQL de ferramenta tem sintaxe adicional projetada para suportar os requisitos das ferramentas. Por exemplo, para se colocar um botão de pressionar em um formulário a fim de navegar até a parte inferior da tela, o movimento seria codificado usando-se a PL/SQL em um sistema ORACLE Forms.

[CER95] cita as seguintes características do PL/SQL:

- a) suporte a SQL;
- b) programação estruturada em blocos;
- c) fluxo de controles de comandos;
- d) integrado com servidor ORACLE;
- e) integrado com o ORACLE Forms;
- f) integrado com o ORACLE Reports;
- g) integrado com SQL*PLUS.

3.2.5 DESIGNER / 2000

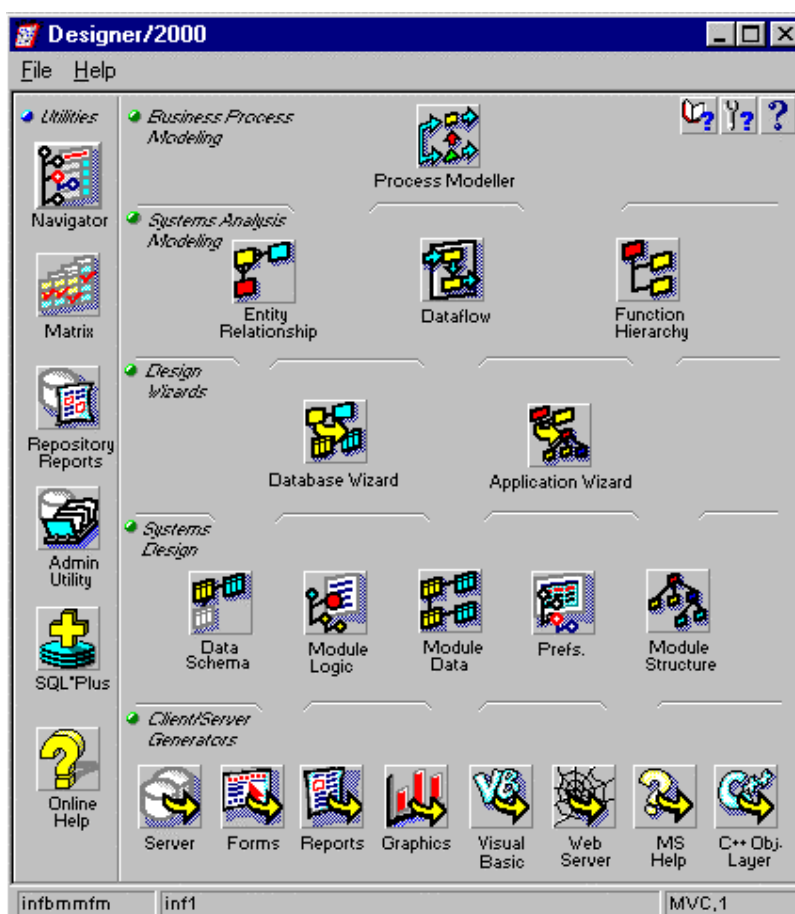
Segundo [CER95], Designer/2000 é um conjunto de ferramentas para modelar e projetar bancos de dados e suas aplicações. É um produto de grande auxílio para o entendimento global de informatização de uma empresa e também uma solução gráfica para a construção de diagramas. As principais ferramentas do Designer/2000 são:

- a) *Repository*;
- b) *Dataflow*;
- c) *Entity Relationship*;
- d) *Function Hierarchy*;
- e) *Matriz*;
- f) *Database Wizard*;
- g) *Application Wizard*.

Algumas das vantagens do Designer/2000:

- a) acesso multi-usuário ao repositório de dados para criação e manutenção das definições entidades e relacionamentos;
- b) acesso à assistentes de banco de dados (*Database Wizard*), que automatizam a construção do sistema, definindo um banco de dados padrão para as aplicações a partir de diagramas de entidade x relacionamento pré-definidos;
- c) geração de *layout* automático.

Na figura 6 pode-se visualizar as opções disponíveis no Designer/2000.



Fonte: [CER95]

Figura 6 – Ferramentas do Designer/2000

4 FERRAMENTA PARA REUTILIZAÇÃO DE SOFTWARE

Desenvolveu-se um software que analisa código fonte de componentes Forms e Reports desenvolvidos no ambiente ORACLE, capaz de identificar componentes reusáveis através da aplicação de métricas.

O desenvolvimento do trabalho foi realizado através das seguintes fases:

- a) definição dos recursos;
- b) definição do sistema;
- c) implementação;
- d) testes.

4.1 DEFINIÇÃO DOS RECURSOS

Para o desenvolvimento deste trabalho procurou-se utilizar uma base de dados relacional pelas vantagens oferecidas em relação ao armazenamento e acesso de dados.

Optou-se então pelo sistema de gerenciamento de banco de dados Microsoft Access 7.0 para a criação de aplicações para Windows. A estrutura de banco de dados Access combina as tabelas de dados que são inter-relacionadas e seus índices, formulários, relatórios, macros e códigos em Access Basic em um único arquivo do banco de dados com a extensão .MDB. O Access Basic é uma linguagem de programação que o Access incorpora, muito parecida com Visual Basic. A especificação do sistema foi feita com a ferramenta CASE do ORACLE, o Designer/2000.

4.2 DEFINIÇÃO DO SISTEMA

Nesta fase especificou-se o funcionamento do sistema, analisando-se o objetivo do sistema, bem como os dados gerados e requeridos para o alcance deste objetivo. O levantamento de dados foi feito de acordo com os dados que seriam necessários para se aplicar as métricas e poder identificar o componente.

A seguir será apresentado o Diagrama de Contexto, Modelo Entidade Relacionamento (MER), Dicionário de Dados, Diagrama de Fluxo de Dados e o Diagrama Hierárquico Funcional.

4.2.1 DIAGRAMA DE CONTEXTO

O Diagrama de Contexto tem por finalidade definir as entidades externas e a interação destas entidades com o sistema. Para visualização do Diagrama de Contexto da ferramenta vide a figura 7.

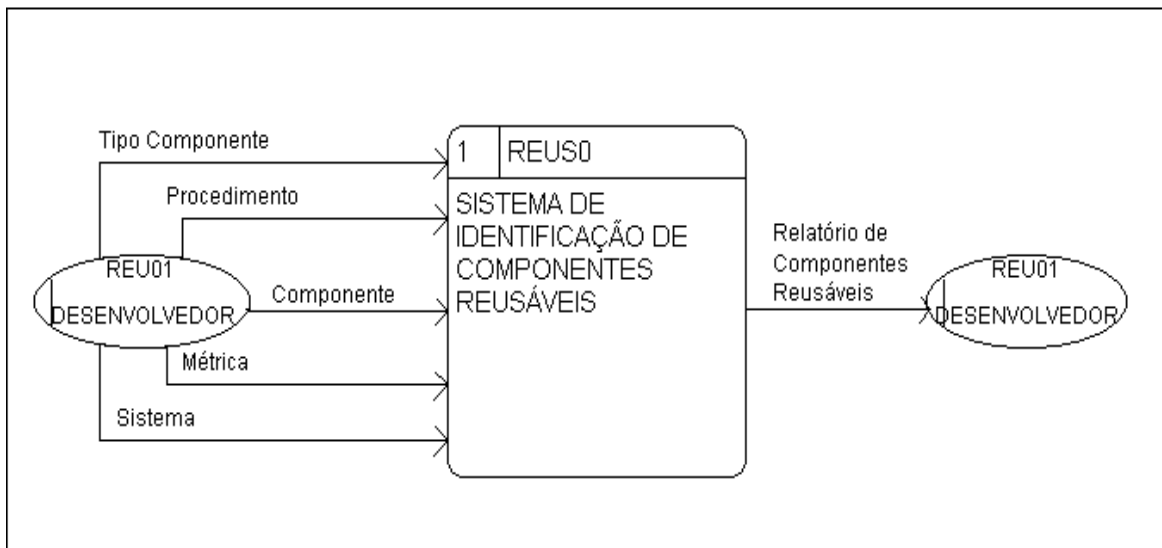


Figura 7 – Diagrama de Contexto

4.2.2 MODELO ENTIDADE RELACIONAMENTO

Nesta fase utilizou-se a técnica de modelagem de dados através do MER (Modelo de Entidade Relacionamento). A figura 8 mostra o Modelo Entidade Relacionamento da ferramenta.

A notação utilizada no MER é apresentada na tabela 6.

Tabela 6 – Notação utilizada no MER

Símbolo	Descrição
#	Chave primária
*	Chave estrangeira
0	Atributo
.....	Relacionamento opcional
_____	Relacionamento obrigatório

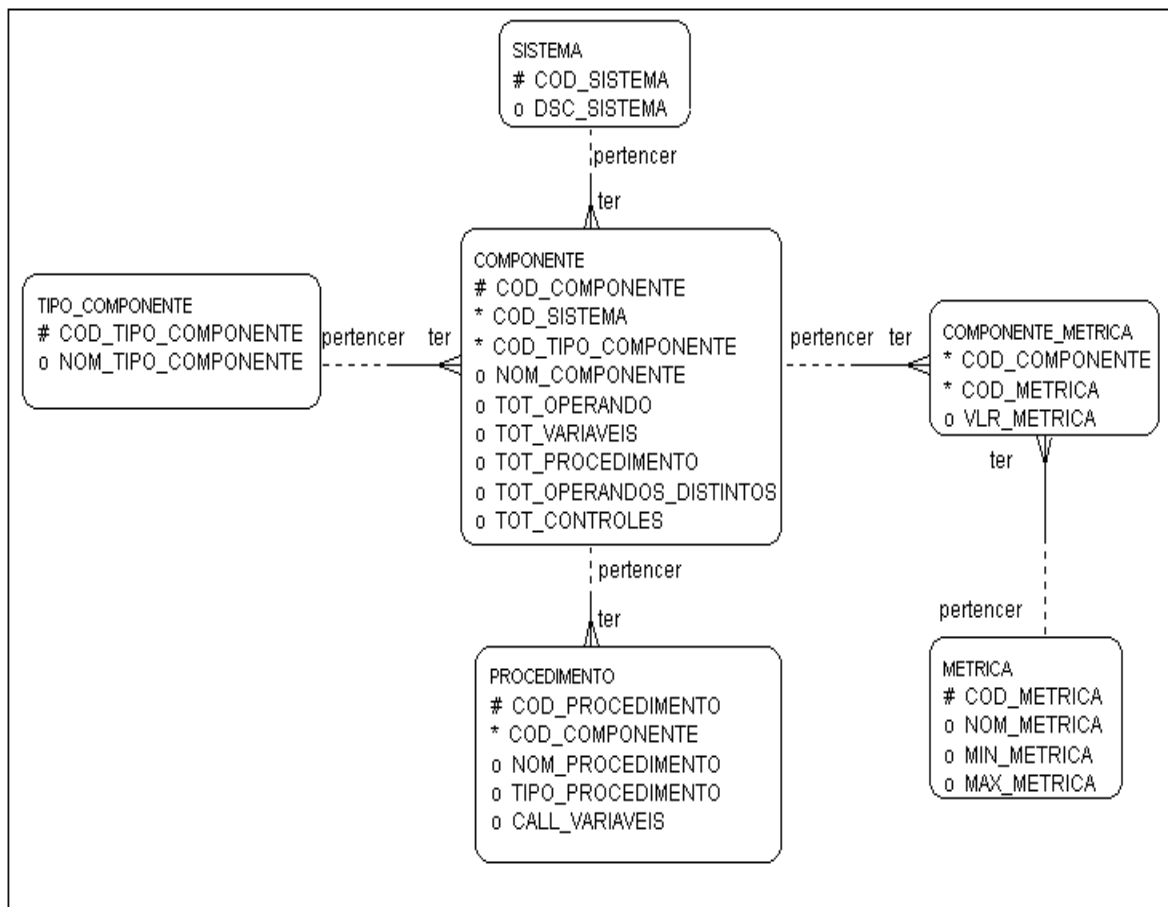


Figura 8 – MER (Modelo Entidade Relacionamento)

A partir do modelo entidade relacionamento da ferramenta pode-se construir o dicionário de dados do mesmo. Este dicionário tem por função fornecer alguns detalhes sobre as entidades do sistema, conforme mostram as tabelas 6, 7, 8, 9, 10 e 11.

Tabela 7 – Entidade SISTEMA

Nome	Tipo	Tamanho	Descrição
Cod_Sistema	AutoNumeração	4	Código do sistema
Dsc_Sistema	Texto	30	Descrição do sistema

Tabela 8 – Entidade TIPO_COMPONENTE

Nome	Tipo	Tamanho	Descrição
Cod_Tipo_Componente	AutoNumeração	4	Código do tipo de componente
Nom_Tipo_Componente	Texto	30	Descrição do tipo de componente

Tabela 9 – Entidade METRICA

Nome	Tipo	Tamanho	Descrição
Cod_Metrica	AutoNumeração	4	Código da métrica
Nom_Metrica	Texto	30	Nome da métrica
Min_Metrica	Número	10,4	Valor do limite mínimo
Max_Metrica	Número	10,4	Valor do limite máximo

Tabela 10 – Entidade COMPONENTE

Nome	Tipo	Tamanho	Descrição
Cod_Componente	AutoNumeração	4	Código do componente
Cod_Sistema	Número	4	Código do sistema
Cod_Tipo_Componente	Número	4	Código do tipo de componente
Nom_Componente	Texto	30	Nome do componente
Tot_Operandos_distintos	Número	10	Quantidade distinta de operandos
Tot_Operando	Número	10	Total de utilização dos operandos
Tot_Variaveis	Número	10	Total de variáveis
Tot_Controles	Número	10	Quantidade de ciclos / ramos
Tot_Procedimento	Número	10	Total de chamadas das procedures

Tabela 11 – Entidade COMPONENTE_METRICA

Nome	Tipo	Tamanho	Descrição
Cod_Componente	Número	4	Código do componente
Cod_Metrica	Número	4	Código da métrica
Vlr_Metrica	Número	10,4	Valor da métrica do componente

Tabela 12 – Entidade PROCEDIMENTO

Nome	Tipo	Tamanho	Descrição
Cod_Procedimento	AutoNumeração	4	Código da procedure ou função
Cod_Componente	Número	4	Código do Componente
Nom_Procedimento	Texto	30	Nome da procedure ou função
Tipo_Procedimento	Texto	30	Tipo procedimento (função / procedure)
Call_Variaveis	Número	10	Total de chamadas às variáveis dentro do procedimento

4.2.3 DIAGRAMA DE FLUXO DE DADOS

Através do Diagrama de Fluxo de Dados, conforme ilustra a figura 9, identificaram-se os processos básicos de operações executadas pela ferramenta.

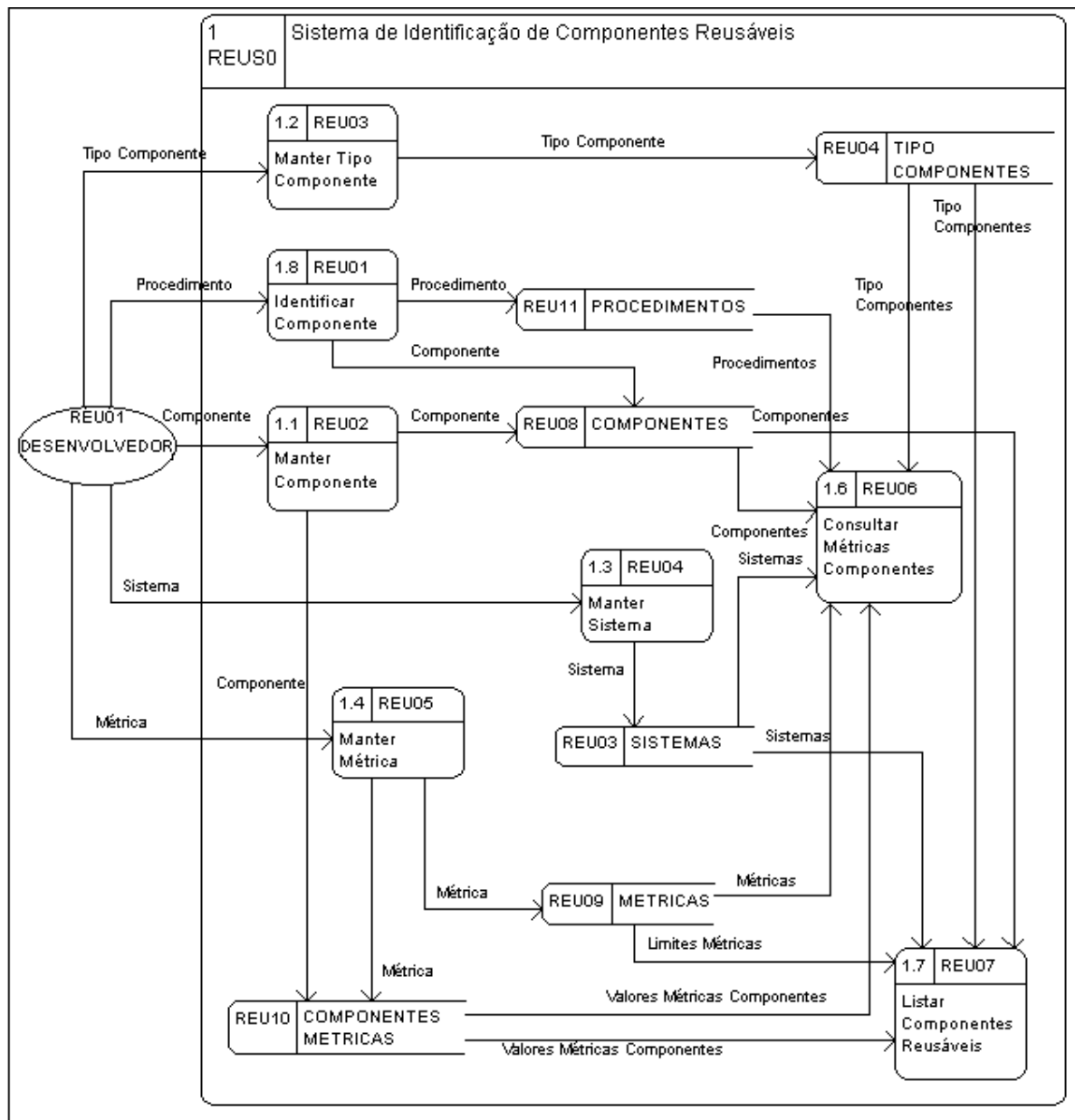


Figura 9 – Diagrama de Fluxo de Dados

4.2.4 DIAGRAMA HIERÁRQUICO FUNCIONAL

O Diagrama Hierárquico, segundo [MEL90], objetiva demonstrar uma visualização geral do sistema. O Diagrama Hierárquico Funcional da ferramenta está demonstrado na figura 10.

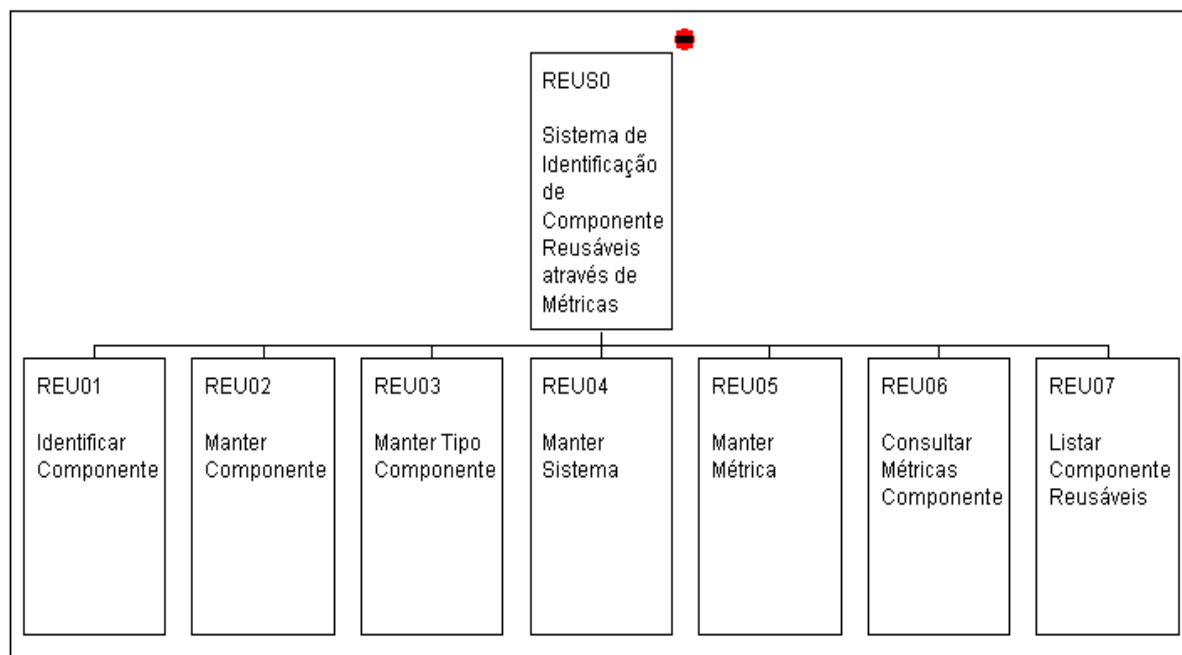


Figura 10 – Diagrama Hierárquico Funcional

4.3 IMPLEMENTAÇÃO

Nesta etapa, foi estabelecido o funcionamento do protótipo através da definição das telas de entrada de parâmetros e relatório de saída. Nos itens seguintes, apresenta-se o funcionamento do protótipo através do detalhamento destas telas e relatório.

4.3.1 DESCRIÇÃO DO SISTEMA

Inicialmente o desenvolvedor deve converter os arquivos fontes dos *Forms* e *Reports*, .FMB e .RDF, que são arquivos binários para arquivos textos, .FMT e .REX. Esta conversão é feita pela própria ferramenta onde os componentes foram desenvolvidos.

No Sistema de Reutilização o desenvolvedor deve informar o diretório onde se encontram os componentes já convertidos para que a ferramenta possa fazer o levantamento de informações necessário para aplicar as métricas. Neste levantamento são identificados os seguintes itens:

- quantidade de variáveis auxiliares;
- total de utilização das variáveis;
- quantidade de operadores e comandos da linguagem de programação SQL;

- d) total de utilização dos operadores e comandos;
- e) número de ciclos do componente;
- f) quantidade de procedures e funções;
- g) total de utilização das procedures e funções.

Todas as informações levantadas do componente referem-se as procedures e funções. No anexo 1 apresenta-se a listagem do componente FDV10YPP desenvolvido no ORACLE Forms onde pode-se visualizar suas procedures e funções.

Em seguida, o sistema calcula os valores das métricas e armazena as informações nas tabelas de dados. Métricas calculadas:

- a) volume;
- b) complexidade ciclomática;
- c) regularidade;
- d) frequência de reutilização.

O projeto físico da ferramenta pode ser visualizado na figura 11.

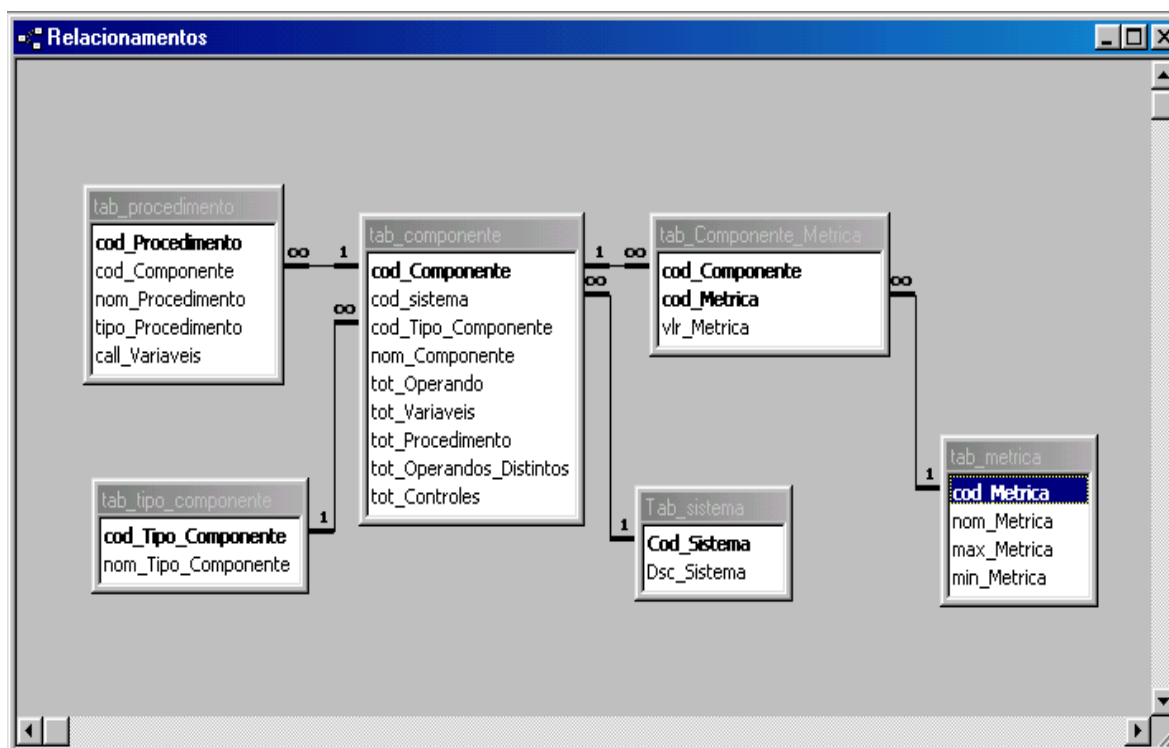


Figura 11 – Projeto Físico da Ferramenta

4.3.2 MENU

Ao iniciar sua execução, o software apresenta uma tela de menu onde o usuário pode optar pelas opções de arquivo, cadastro, consulta ou relatório. Na opção Arquivo pode-se optar pela saída do sistema, visualizar impressão, excluir registro, adicionar registro, primeiro, anterior, próximo ou último. A figura 12 mostra a tela de menu do software.

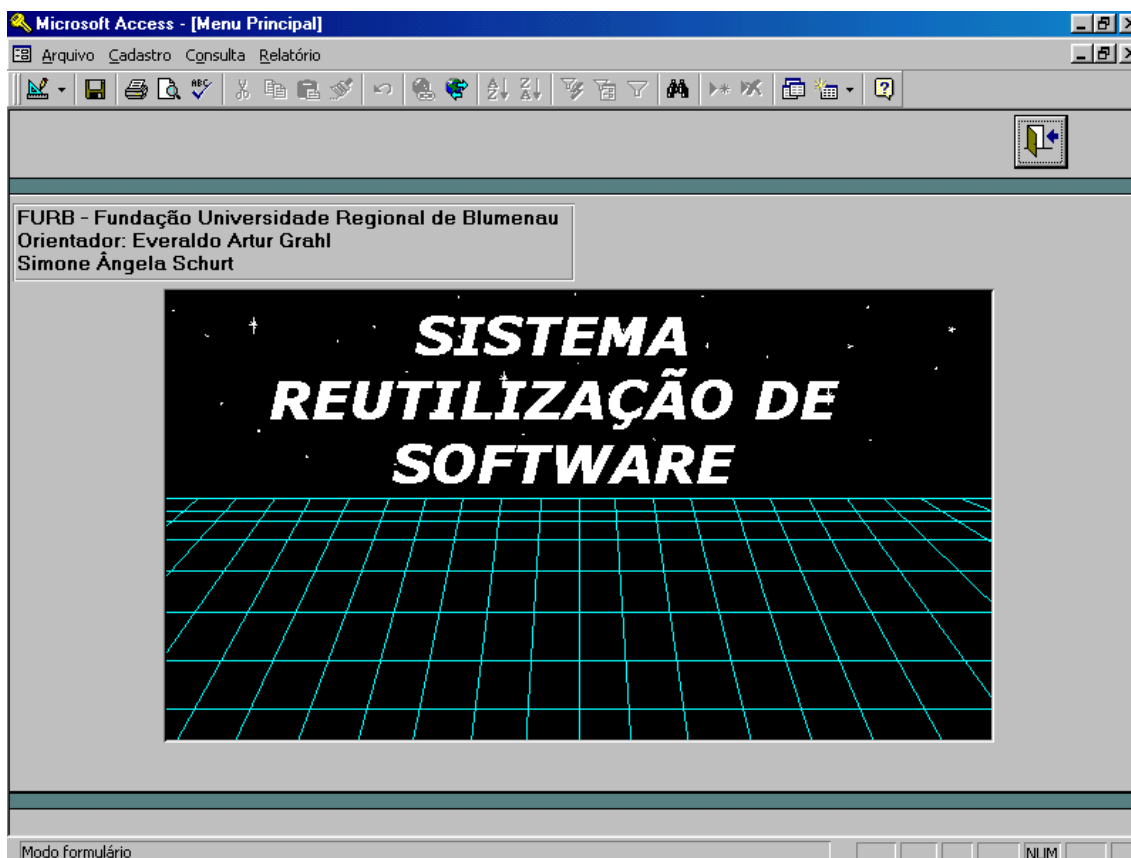


Figura 12 – Menu da Ferramenta

4.3.3 CADASTRO

Nesta opção, o usuário pode selecionar entre Identificar Componente, Sistemas, Tipos de Componentes, Componentes e Métricas. Nos próximos itens são demonstrados estas opções.

4.3.3.1 IDENTIFICAR COMPONENTE

Ao selecionar esta opção, é mostrada a tela que dispara o processo de levantamento de dados do componente e cálculo das métricas. O desenvolvedor deve informar o diretório onde

se encontra o componente e clicar no botão iniciar. Em seguida o software começa a ler o arquivo texto, calcular as métricas e atualiza os dados nas tabelas. A figura 13 exibe a tela para identificar componentes.

Identificar Componente

Componente: 23
 Tipo: 1 Forms
 Sistema: 1 Contas a Pagar
 Nome: c:\simone\componentes\tdv101pm.fmt
 Operandos: 115 Distintos: 21
 Variáveis: 13 Chamadas: 33
 Procedimentos: 6 Chamadas: 25
 Controles: 26

Métricas:

	Nome	Valor
1	Volume	752,9445
2	Complexidade Ciclomática	26
3	Regularidade	0,9482728
4	Frequência de Reutilização	0,24

Procedimentos:

Procedimento	Tipo	call(Variáveis)
BUTTON_HELP	Procedure	1
CGRI\$CHK_TIPO_VERBA_ESPEC	Procedure	2
CGUV\$CHK_TIPVERB_PK	Procedure	1
DEL_TIMER	Procedure	3
msg_alert	Procedure	12
TOOLBAR_ACTIONS	Procedure	14

Registro: 1 de 6

Registro: 1 de 18

Código da métrica utilizada para o componente

Figura 13 - Tela para identificar dados do componente

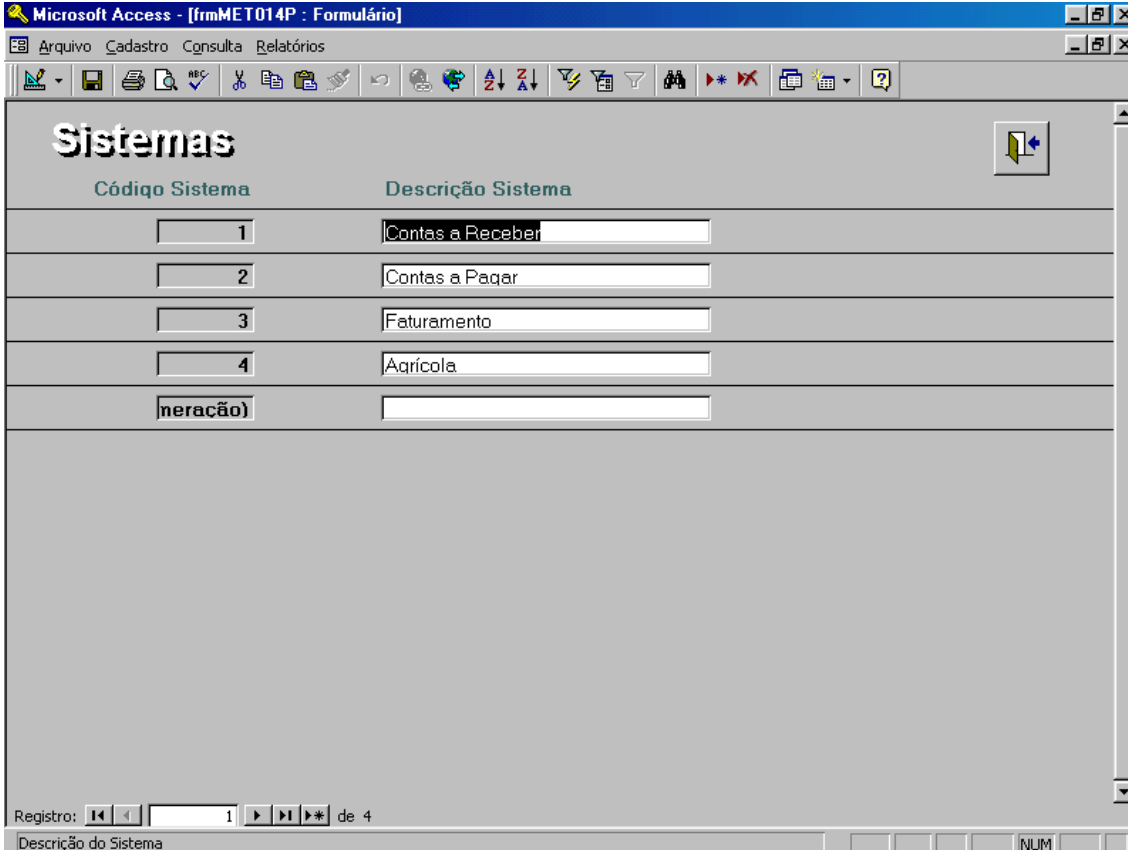
A tela Identificar Componente possui os seguintes campos:

- Componente: código do componente;
- Tipo: tipo do componente;
- Nome: diretório e nome do componente;
- Operandos: quantidade de operandos e comandos do componente;
- Total Oper.: quantidade de vezes que os operandos e comandos são utilizados no componente;
- Variáveis: quantidade de variáveis que o componente possui;
- Chamadas: quantidade de vezes que as variáveis são utilizadas;
- Procedimentos: quantidade de procedures e funções do componente;

- i) Chamadas: é a quantidade de vezes que as procedures e funções do componente são chamados;
- j) Controles: quantidade de ciclos / ramos do componente;
- k) Inicia: inicia a leitura do componente;
- l) Procedimentos: mostra as procedures e funções do componente, tipo e quantidade de vezes que são chamadas;
- m) Cálculo das Métricas: mostra as métricas e os valores calculados.

4.3.3.2 SISTEMAS

Ao selecionar esta opção, uma tela para cadastrar os sistemas é apresentada ao desenvolvedor. Esta tela apresenta o código e a descrição do sistema. A figura 14 exibe a tela Sistemas.



The screenshot shows a Microsoft Access form window titled "Microsoft Access - [frmMET014P : Formulário]". The form has a menu bar with "Arquivo", "Cadastro", "Consulta", and "Relatórios". Below the menu is a toolbar with various icons. The main area of the form is titled "Sistemas" and contains a table with two columns: "Código Sistema" and "Descrição Sistema". The table has five rows of data. The first row has "1" in the code column and "Contas a Receber" in the description column. The second row has "2" and "Contas a Pagar". The third row has "3" and "Faturamento". The fourth row has "4" and "Agrícola". The fifth row has "neracão" and an empty description field. At the bottom of the form, there is a status bar with "Registro: 1 de 4" and a "Descrição do Sistema" label.

Código Sistema	Descrição Sistema
1	Contas a Receber
2	Contas a Pagar
3	Faturamento
4	Agrícola
neracão	

Figura 14 – Tela de cadastro dos sistemas

4.3.3.3 TIPO DE COMPONENTES

Nesta opção é exibida uma tela para cadastrar os tipos de componentes que o software analisa. O campo Código Tipo Componente é AutoNumeração e Descrição Tipo Componente deve ser informada. Para este trabalho foram cadastrados os tipo *Forms* e *Reports* que são os tipo analisados pela ferramenta. A figura 15 ilustra a tela Tipo de Componentes.

Código Tipo Componente	Descrição Tipo Componente
1	Forms
2	Reports
neração	

Registro: 3 de 3

Descrição do Tipo de Componente

Figura 15 – Tela de cadastro dos tipos de componentes

4.3.3.4 COMPONENTES

Nesta tela são cadastrados os componentes. Possui os seguintes campos:

- Código do Componente: é do tipo AutoNumeração;
- Descrição do Componente: o desenvolvedor deve informar o nome do componente;
- Sistema: é exibida uma caixa de combinação com os sistemas cadastrados, onde o desenvolvedor deve selecionar o sistema a qual o componente pertence;

- d) Tipo Componente: é exibida uma caixa de combinação com os tipos de componentes cadastrados, onde o desenvolvedor deve selecionar o tipo do componente que esta sendo cadastrado.

A figura 16 apresenta a tela Componentes.

Código do componente	Descrição do Componente	Sistema	Tipo componente
3	FDV175PP	Contas a Receber	Forms
4	FDV11EPR	Contas a Receber	Reports
5	FDV10YPP	Contas a Pagar	Forms
neração)			

Figura 16 – Tela de cadastro dos componentes

4.3.3.5 MÉTRICAS

Ao selecionar esta opção, é apresentada a tela para cadastro de Métricas. Esta tela possui os seguintes campos:

- Código da Métrica: é do tipo AutoNumeração;
- Descrição da Métrica: nome da métrica;
- Valor Limite Mínimo: deve ser informado o valor mínimo para que um componente seja considerado reutilizável;

- d) Valor Limite Máximo: deve ser informado o valor máximo para que um componente seja considerado reutilizável.

A figura 17 apresenta a tela Métricas.

Código da Métrica	Descrição da Métrica	Valor Limite Mínimo	Valor Limite Máximo
1	Volume	0	0
2	Complexidade Ciclométrica	0	0
3	Regularidade	0	0
4	Frequência de Reutilização	0	0

Registro: 5 de 5
Descrição da Métrica

Figura 17 – Tela de cadastro das métricas

4.3.4 CONSULTA

Esta opção permite consultar os componentes cadastrados no sistema, informando os dados e valores das métricas dos componentes. Possui os seguintes campos:

- Código do Componente: Código do componente que esta sendo consultado;
- Descrição do Componente: nome do componente;
- Sistema: apresenta o código e nome do sistema;
- Tipo Componente: apresenta o código e descrição do tipo do componente;
- Métricas: nome das métricas;
- Valor da Métrica: Valor de cada métrica do componente.

A figura 18 ilustra a tela de consulta dos valores das métricas dos componentes.

The screenshot shows a Microsoft Access window titled "Microsoft Access - [frmMET011P : Formulário]". The menu bar includes "Arquivo", "Cadastro", "Consulta", and "Relatórios". The title bar of the form is "Valores das Métricas dos Componentes".

The form contains the following fields:

- Código do Componente:** 42
- Descrição do Componente:** c:\simone\componentes\fdv194pm.fmt
- Sistema:** 2 Contas a Receber
- Tipo Componente:** 1 Forms

Below these fields is a table with the following data:

Métrica	Valor da Métrica
Volume	1822,775
Complexidade Ciclomática	48
Regularidade	0,6305571
Frequência de Reutilização	0,2111111

At the bottom of the form, there is a record navigation bar showing "Registro: 27 de 27" and a "Modo formulário" label.

Figura 18 – Tela de consulta dos valores das métricas dos componentes

4.3.5 RELATÓRIO

Ao selecionar esta opção, é apresentada uma tela para seleção dos parâmetros para o relatório de componentes reusáveis. Neste relatório são listados todos os componentes que correspondem ao sistema e ao tipo de componente selecionado, e que possuem valor das métricas selecionadas entre o limite mínimo e máximo. A figura 19 ilustra a tela de parâmetros do relatório. No anexo 2 pode-se visualizar o *layout* do relatório gerado.

Microsoft Access - [frmMET016P : Formulário]

Arquivo Cadastro Consulta Relatórios

Relatório de Componentes Reusáveis

Sistema:

Tipo de Componentes:

OBS: Para trazer todos, deixe o campo em branco.

Métricas:

- Volume
- Complexidade Ciclomática
- Regularidade
- Frequência de Reutilização

Modo formulário NUM

Figura 19 – Tela de parâmetros para relatório de componentes reusáveis

4.4 TESTES

Nesta fase foram submetidos à ferramenta alguns *Forms e Reports*, obtidos na empresa Informática Blumenau Ltda, para verificar seu potencial de reuso. Estes componentes fazem parte de um Sistema de Dívidas e executam funções diversas como cadastramentos, consultas, controles, etc. A tabela 12 apresenta os componentes analisados e os valores obtidos das métricas.

Tabela 13 – Tabela de componentes analisados pela ferramenta

Componentes	Tipo	Métricas			
		Volume	Complexidade Ciclomática	Regularidade	Frequência Reutilização
FDV10APC	Forms	416,1677	15	0,8812	0,4286
FDV173PC	Forms	21.727	186	0,4451	0,2673
FDV15EPM	Forms	383,6331	22	0,6202	0,2778
FDV158PM	Forms	3.497,9640	76	0,3641	0,2592
FDV11EPP	Forms	1.542,9000	23	0,8761	0,2647
FDV10YPP	Forms	2.200,1510	60	0,4883	0,1222
FDV10EPM	Forms	1.245,6250	32	0,7986	0,2500
FDV194PM	Forms	1.822,7750	48	0,6306	0,2111
FDV101PM	Forms	752,9445	26	0,9483	0,2400
FDV106PS	Forms	1.261,297	49	0,7306	0,2340
FDV10HPC	Forms	416,9925	24	0,5706	0,3636
FDV114PM	Forms	655,7712	25	1,0404	0,2608
FDV11EPR	Reports	5.775,05	158	0,4139	0,6786
FDV10YPR	Reports	852,803	17	0,4732	1
FDV10NPR	Reports	33	2	1,7865	1
FDV10OPR	Reports	115,1016	10	0,3783	1
FDV10QPR	Reports	242,2272	9	0,7103	1
FDV13FPR	Reports	53,7744	0	2,5369	1

Os limites utilizados pela ferramenta para identificar os componentes reusáveis apresentam-se na tabela 13.

Tabela 14 – Limites das métricas utilizados pela ferramenta

Métricas	Valor Mínimo	Valor Máximo
Volume	500	2.000
Complexidade Ciclomática	10,00	30,00
Regularidade	0,3000	1,0000
Frequência de Reutilização	0,25	0,90

Após definir-se os limites de cada métrica pode-se visualizar na tabela 14 os componentes reusáveis e o percentual de reuso de cada componente. Os componentes com percentual 75 e 100 são considerados reusáveis. Isto pode ser comprovado pelo fato de alguns destes componentes já terem sido reaproveitados pelos desenvolvedores para desenvolver outros componentes, sendo efetuado algumas alterações ou ajustes.

Tabela 15 – Tabela de componentes reusáveis

Componentes	Volume	Complexidade Ciclomática	Regularidade	Frequência de Reutilização	Percentual Reuso (%)
FDV10APC		X	X	X	75
FDV173PC			X	X	50
FDV15EPM		X	X	X	75
FDV158PM			X	X	50
FDV11EPP	X	X	X	X	100
FDV10YPP			X		25
FDV10EPM	X		X	X	75
FDV194PM	X		X		50
FDV101PM	X	X	X		75
FDV106PS	X		X		50
FDV10HPC		X	X	X	75
FDV114PM	X	X	X	X	100
FDV11EPR			X	X	50
FDV10YPR	X	X	X		75
FDV10NPR					0
FDV10OPR		X	X		50
FDV10QPR			X		25
FDV13FPR			X		25

5 CONCLUSÃO

5.1 CONSIDERAÇÕES FINAIS

Os conceitos da reusabilidade, apesar de serem antigos, enfrentam ainda nos dias de hoje uma grande resistência a sua efetiva aplicação. Para que a reutilização possa evoluir e ser adotada pelas empresas deve-se solucionar alguns problemas, como:

- a) mudança de comportamento para que o desenvolvedor adquira o hábito de projetar componentes imaginando aplicações futuras;
- b) desprezo dos desenvolvedores para exercer efetivamente esta mudança;
- c) falta de um maior número de ferramentas voltadas a reutilização;
- d) redução dos custos para aquisição de uma ferramenta direcionada a reutilização.

A ferramenta desenvolvida conseguiu avaliar a potencialidade de reuso dos componentes *Forms* e *Reports* através das métricas. Os testes realizados demonstram que a ferramenta auxilia no processo de identificação de componentes reusáveis.

Foram encontradas limitações durante o desenvolvimento do trabalho em relação a ferramenta que seria utilizada para fazer a implementação. A proposta inicial do trabalho foi de implementar a ferramenta no próprio ORACLE, no caso Forms e Reports, porém estes aplicativos não possuem estrutura para efetuar leitura de arquivos textos, optando-se, então, pelo Access.

5.2 SUGESTÕES

As principais sugestões para futuros trabalhos nesta área são:

- a) um estudo mais aprofundado sobre métricas para reutilização;
- b) analisar outros componentes desenvolvidos no ambiente ORACLE;
- c) analisar além de procedures e funções também as triggers das ferramentas Forms e Reports;
- d) aplicar as métricas de identificação em componentes desenvolvidos na última versão do ORACLE : Forms 5.0 e Reports 3.0

ANEXO 1 – LISTAGEM DO COMPONENTE

```

/*

#ROS Script Version 2.2.2.2.0 - Production

Copyright (c) Oracle Corporation 1992, 1993. All rights
reserved.
*/

DESCRIBE  ROSSTRINGS
BEGIN
    UB4      groupid
    UB4      stringid
    UB4      lfid
    UB2      cs
    UB2      len
    BINARY   len:str
END

DESCRIBE  tool_plsql
BEGIN
    UB4      itemid
    TEXT     name
    UB2      type
    TLONG    plslfid_st
    BLONG    plslfid_ep
END

DESCRIBE  FRM45_PLSQLINFO
BEGIN
    TEXT     name
    UB4      itemid
END

DEFINE  tool_plsql
BEGIN
    itemid          = 1
    name            = <<"MSG_ALERT">>
    type            = 128
    plslfid_st      = (TLONG)
    <<"procedure msg_alert(
errm in char,      /* message */
errt in char,      /* message type */
rftf in boolean    /* raise form_trigger_failure ? */
) is              /* message parameters */
/*

```

```

* -----
----
* CHANGE HISTORY:
* DATE      PERSON      CHANGE
* -----
----
*
*/

alert_is alert;
alert_button number;

BEGIN

    IF (errt = 'F')
    THEN alert_is := FIND_ALERT('CFG_SYSTEM_ERROR');
    ELSIF (errt = 'E')
        THEN alert_is := FIND_ALERT('CFG_ERROR');
        ELSIF (errt = 'W')
            THEN alert_is := FIND_ALERT('CFG_WARNING_A');
            ELSIF (errt = 'I')
                THEN alert_is :=
FIND_ALERT('CFG_INFORMATION');
                ELSE MESSAGE(errm);
            end if;

    IF (errt IN ('F','E','W','I'))
    THEN
SET_ALERT_PROPERTY(alert_is,ALERT_MESSAGE_TEXT,errm);
        alert_button := SHOW_ALERT(alert_is);
    END IF;

    IF (rftf)
    THEN
        RAISE FORM_TRIGGER_FAILURE;
    END IF;

END;
">>
    plslfid_ep          = (BLONG) NULLP
END

DEFINE tool_plsql
BEGIN
    itemid              = 6
    name                = <<"PU_070">>
    type                = 2
    plslfid_st          = (TLONG)

```

```

<<"/* Executa procedure que faz definicoes iniciais do form */
begin
    set_window_property(FORMS_MDI_WINDOW,WINDOW_STATE,MAXIMIZE);
    set_window_property('CG$WINDOW_1',WINDOW_STATE,MAXIMIZE);
    exec_def_ini;
exception
    when others
    then
        raise form_trigger_failure;
end;

/* CCGN$CALL_GENERATOR_CODE */
/* Execute the WHEN-NEW-FORM-INSTANCE code that was created by
Forms */
/* Generator
*/
BEGIN
    BEGIN
        CG$WHEN_NEW_FORM_INSTANCE;
    EXCEPTION
        WHEN OTHERS THEN
            CGTE$OTHER_EXCEPTIONS;
    END;
END;
DEFINE tool_plsql
BEGIN
    itemid          = 17
    name            = <<"CG$WHEN_NEW_FORM_INSTANCE">>
    type            = 128
    plslfid_st      = (TLONG)
<<"PROCEDURE CG$WHEN_NEW_FORM_INSTANCE IS
BEGIN
/* CGLY$INIT_CANVASES */
/* Call procedure to ensure correct canvases are visible */
BEGIN
    CGLY$CANVAS_MANAGEMENT;
END;
END;
">>
    plslfid_ep      = (BLONG) NULLP
END

DEFINE tool_plsql
BEGIN
    itemid          = 18
    name            = <<"CGLY$CANVAS_MANAGEMENT">>
    type            = 128
    plslfid_st      = (TLONG)
<<"PROCEDURE CGLY$CANVAS_MANAGEMENT IS

```

```

/* Top level canvas management procedure */
current_canvas VARCHAR2(61) :=
get_item_property(:SYSTEM.CURSOR_ITEM,
ITEM_CANVAS);
base_canvas VARCHAR2(61);
canvas_list VARCHAR2(255);
BEGIN
IF ( (:CG$CTRL.CG$LAST_CANVAS IS NULL) OR
(:CG$CTRL.CG$LAST_CANVAS !=
current_canvas) ) THEN
:CG$CTRL.CG$LAST_CANVAS := current_canvas;
set_window_property( get_view_property( current_canvas,
WINDOW_NAME
), VISIBLE, PROPERTY_ON);
CGLY$GET_RELATED_CANVASES(current_canvas, base_canvas);
IF ( base_canvas = 'CG$PAGE_1') THEN
canvas_list := :CG$CTRL.CG$PAGE_1_LIST;
END IF;
CGLY$DISPLAY_CANVASES(canvas_list, current_canvas,
base_canvas);
IF ( base_canvas = 'CG$PAGE_1') THEN
:CG$CTRL.CG$PAGE_1_LIST := canvas_list;
END IF;
END IF;
END;
">>
plslfid_ep = (BLONG) NULLP
END
DEFINE tool_plsql
BEGIN
itemid = 22
name = <<"CONSISTE_ENTRADA">>
type = 128
plslfid_st = (TLONG)
<<"PROCEDURE consiste_entrada IS
BEGIN
if :blk_itens.grp_lc = 2 then
go_item('blk_lc.cod_lc');
first_record;
if :blk_lc.cod_lc is null then
msg_alert('Informe ao menos um(1) código de Linha de
Crédito!', 'I', true);
end if;
end if;
if :blk_itens.grp_bco = 2 then
go_item('blk_bco.cod_bco');
first_record;
if :blk_bco.cod_bco is null then

```

```

        msg_alert('Informe ao menos um(1) código de
Banco!','I',true);
    end if;
end if;
if :blk_itens.grp_ocr = 2 then
    go_item('blk_ocr.cod_ocr');
    first_record;
    if :blk_ocr.cod_ocr is null then
        msg_alert('Informe ao menos um(1) código de
NCR!','I',true);
    end if;
end if;
if :blk_itens.grp_geral <> 1 then
    go_item('blk_avulso.cod_prodor');
    first_record;
    if :blk_avulso.cod_prodor is null then
        msg_alert('Informe ao menos um(1) código
Avulso!','I',true);
    end if;
end if;
if :blk_itens.safra is null then
    msg_alert('Ano Safra deve ser informado !','I',true);
end if;
if :blk_itens.dat_liq is null then
    msg_alert('Data Liquidação deve ser informado
!','I',true);
end if;
if :blk_itens.grp_liq = 1 then
    if :blk_itens.grp_geral = 1 and :blk_itens.dat_venc to is
null then
        msg_alert('Para opção GERAL deve-se informar a Data de
Vencimento !','I',true);
    end if;
end if;
END;">>

```

ANEXO 2 – RELATÓRIO DE COMPONENTES REUSÁVEIS

Relatório de Componentes Reusáveis

Sistema: Contas a Pagar

Tipo Componente: Forms

Código Componente: 23

Nome Componente: c:\sisonelcomponentes\sfdv101pm.fmt

<i>Métrica</i>	<i>Valor Métrica</i>	<i>Limite Mínimo</i>	<i>Limite Máximo</i>
Complexidade Cidomática	26	10	30
Volume	752,9445	500	2000
Regularidade	0,9482728	0,3	1

Código Componente: 24

Nome Componente: c:\sisonelcomponentes\sfdv106ps.fmt

<i>Métrica</i>	<i>Valor Métrica</i>	<i>Limite Mínimo</i>	<i>Limite Máximo</i>
Volume	1261,297	500	2000
Regularidade	0,7306589	0,3	1

Código Componente: 25

Nome Componente: c:\sisonelcomponentes\sfdv10apc.fmt

<i>Métrica</i>	<i>Valor Métrica</i>	<i>Limite Mínimo</i>	<i>Limite Máximo</i>
Complexidade Cidomática	15	10	30
Regularidade	0,8812315	0,3	1
Frequência de Reutilização	0,4285714	0,25	0,9

Código Componente: 26

Nome Componente: c:\sisonelcomponentes\sfdv10epm.fmt

<i>Métrica</i>	<i>Valor Métrica</i>	<i>Limite Mínimo</i>	<i>Limite Máximo</i>
Volume	1246,625	500	2000
Regularidade	0,7986031	0,3	1
Frequência de Reutilização	0,25	0,25	0,9

REFERÊNCIAS BIBLIOGRÁFICAS

- [ARA95] ARAUJO, Sérgio Paulo Camacho. **Uma abordagem de apoio à reutilização de componentes em ferramentas ORACLE**. Blumenau, 1995. Trabalho de Conclusão de Curso – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau.
- [ARN94] ARNOLD, Robert S.; FRAKES, William B.. **Software Reuse and Reengineering**. Califórnia : Institute of Eletrical and Eletronics engenieurs, 1994.
- [CAT96] CATTONI, Simone Máldaliz. **Protótipo para a identificação de código reusável em Dataflex**. Blumenau, 1996. Trabalho de Conclusão de Curso – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau.
- [CER95] CERÍCOLA, Vicent Oswald. **ORACLE – Banco de Dados Relacional e Distribuído. Ferramentas para Desenvolvimento**. São Paulo : Makron – McGraw-Hill, 1995.
- [CHE94] CHENG, Jingewn. **A Reusability-Based Software Development Enviroment**. Austrália : University Monash, 1994.
- [FUR95] FURLAN, José Davi. **Reengenharia da Informação: Do Mito a Realidade**. Editora MAKRON Books do Brasil - São Paulo, 1995.
- [GRA96] GRAHL, Everaldo Artur. **Reutilização de Software**. Blumenau, 1996. Apostila - Pós Graduação em Informática. Santa Catarina, Universidade Regional de Blumenau.

- [HAL94] HALL, P. A .V.. **Software Reuse and Reverse Engineering in Practice**. London : Chapman & Hall, 1994.
- [KUT97] KUTOVA, Marcos André Silveira; MACEDO, Alessandra Alaniz. **Reuso de Software**. São Carlos, 1997. Trabalho de Conclusão de Curso – Ciências Exatas, Universidade de São Paulo.
- [MCC92] MCCLURE, Carma. **The Three Rs Of The software automation : Re-engineering, Reusability, Repository**. New Jersey : Prentice Hall, 1992.
- [MCC93] MCCLURE, Carma. **The Three Rs Of The software automation : Re-engineering, Reusability, repository**. New Jersey : Prentice Hall, 1993.
- [MCD93] MCDERMID, John. **Software Engineer's Reference Book**. Califórnia : CRC Press, 1993.
- [MEL90] MELENDEZ, Ruben Filho. **Prototipação de Sistemas de Informações : Fundamentos, Técnicas e Metodologias**. Rio de Janeiro : LCT, 1990.
- [MOR95] MORAIS, Rinaldo Oliveira. **ORACLE7 – Conceitos Básicos**. São Paulo : Érica, 1995.
- [PRE95] PRESSMAN, Roger S. **Engenharia de Software**. São Paulo : Makron Books, 1995.
- [ROS96] ROSÁRIO, Vilson Camargo. **Ferramenta de Auxílio à Administração do Banco de Dados ORACLE 7**. Trabalho de Conclusão de Curso. Universidade Regional de Blumenau, 1996.
- [SIL94] SILVEIRA, Janaina. **Reusabilidade de Software : Um Estudo Voltado á Análise de Domínio**. Trabalho de Conclusão de Curso. Universidade Regional de Blumenau, 1994.
- [TRA93] TRACZ, Will Editor. Second International Workshop on Software Reusability - IWSR2. **Software Engineering Notes**. Vol 18 Nro. 3. Califórnia, p. A73-A77, Julho de 1993.