AVALIAÇÃO DA REUTILIZABILIDADE DE COMPONENTES DE SOFTWARE

Cesar A. Comerlato, Geraldo B. Xexeo, Ana Regina C. da Rocha
Universidade Federal do Rio de Janeiro / COPPE - Sistemas
Caixa Postal 68511 - CEP 21945-970, Rio de Janeiro RJ - Brasil
email: cesar@lmp.coppe.ufrj.br

Resumo - Uma questão fundamental, quando se pensa em reutilização é a decisão do que reutilizar. Este trabalho apresenta atributos de qualidade relacionados à reutilizabilidade de código, avaliados através de variáveis linguísticas e lógica nebulosa (fuzzy), apresentando uma ferramenta para identificação de componentes de código FORTRAN reutilizáveis, a partir de um acervo de programas já existentes em um determinado ambiente. Isto permite a seleção de candidatos para inclusão em uma biblioteca de componentes reutilizáveis.

Palavras chave - lógica fuzzy, fortran, qualidade, reutilizabilidade, reutilização

1. Introdução

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 e confiabilidade, aumentar a segurança dos sistemas, etc. Neste contexto são imprescindíveis mudanças significativas na forma como o software é produzido atualmente.

A abordagem do problema de aumento da qualidade e produtividade pode ser resumida em três pontos principais: (1) otimizar a eficiência do processo; (2) reduzir a quantidade de trabalho refeito; (3) reutilizar produtos do ciclo de vida [BASI92].

A reutilização de componentes de software existentes em novos sistemas implica numa menor produção de software novo, causando um aumento da produtividade bem como da qualidade e confiabilidade. O aumento da produtividade é devido à diminuição do esforço necessário para a produção de código novo. O aumento da qualidade e confiabilidade advém do fato do código reutilizado já ter sido amplamente usado, modificado e testado em outros sistemas.

Este trabalho apresenta atributos de qualidade relacionados à reutilizabilidade de código, apresentando uma ferramenta para identificação de componentes reutilizáveis de código FORTRAN, a partir de um acervo de programas já existentes em um determinado ambiente. O modelo adotado partiu dos trabalhos de Prieto-Diaz e Freeman [PRIE87], Caldiera e Basili [CALD91] e Dunn e Knigth [DUNN93], para a identificação de componentes de código candidatos a reutilização.

A escolha da linguagem FORTRAN deve-se à sua preferência pela comunidade científica nacional e internacional para o desenvolvimento de software nas áreas científica e de engenharia. Consideramos, portanto, que a linguagem FORTRAN ainda continuará sendo usada, por um bom tempo, nas áreas científica e de engenharia. Assim sendo, através da identificação e seleção de candidatos à reutilização, a partir de sistemas já existentes, para formar parte de uma biblioteca de componentes reutilizáveis, pretendemos contribuir para o

aumento da produtividade e da qualidade no desenvolvimento de novos produtos nesta linguagem.

2. Reutilizabilidade de Componentes

Em trabalhos anteriores Rocha [ROCH83] propôs um modelo para avaliação da qualidade de software baseado nos seguintes conceitos:

- **Objetivos de qualidade:** são as propriedades gerais, que o produto deve possuir;
- **Fatores de qualidade:** determinam a qualidade na visão dos diferentes usuários do produto - usuário final e outros;
- **Critérios:**são atributos primitivos, possíveis de serem avaliados;
- **Processo de avaliação:** determinam as métricas a serem utilizadas, de forma a se medir o grau de presença, no produto, de um determinado critério;
- **Medidas:** são o resultado da avaliação do produto, segundo os critérios;
- **Medidas agregadas:** são o resultado da agregação das medidas obtidas ao se avaliar de acordo com os critérios, e quantificam os fatores.

Este modelo pode ser utilizado para avaliação da qualidade de produtos ao longo de todo o processo de desenvolvimento [CLUN87], [ANDR91], [BELC92], [BAHI92]. Neste trabalho, tratamos exclusivamente da avaliação a nível de código, especificamente de atributos de qualidade relacionados à reutilizabilidade em componentes de código.

Considerando a estrutura do modelo para avaliação da qualidade, identificamos os seguintes subfatores e critérios de qualidade, como fundamentais para a avaliação da reutilizabilidade de código (Tabelas 1 e 2).

Fator	Subfator	Critérios	Descrições
Reutilizabilidade	Estilo	Documentação Interna	Refere-se a característica do código fonte do componente apresentar informações significativas através de comentários.
		Organização Visual	É a característica de um componente ter uma boa apresentação quanto ao posicionamento de nomes, comandos, comentários, linhas em branco e na constatação de que foram utilizadas boas práticas de programação na sua implementação.
		Padronização	O componente obedece às normas e padrões estabelecidos pelo ambiente de programação da organização.
		Programação Estruturada	O componente obedece às normas da técnica de programação estruturada.

Tabela 1 - Subfator Estilo.

Atributos de qualidade são, na sua maioria, conceitos subjetivos e de difícil avaliação. Portanto, o uso de uma teoria que trate dessa subjetividade de forma adequada adapta-se bem às necessidades de medição desses atributos. Assim sendo, a teoria de conjuntos nebulosos (fuzzy) aliada ao uso de variáveis e termos linguísticos foi utilizada neste trabalho. Para melhor entendimento descrevemos, na próxima seção, estes conceitos.

Fator	Subfator	Critérios	Descrições
Reutilizabilidade	Generalidade	Independência do Tipo de Dados	A aptidão de um programa operar com vários tipos de dados da linguagem utilizada.
		Independência da Quantidade de Dados	Característica de um programa não possuir restrições, para utilização de qualquer volume de dados.
		Independência de Compilador	A codificação de um componente não inclui particularidades de um determinado compilador.
		Independência de Hardware	É o grau de independência do componente de software em relação ao hardware para o qual foi, originalmente, desenvolvido.
	Maturidade	Confiabilidade	A probabilidade de um componente executar satisfatoriamente sua função (sem falhas) durante um período de tempo.
		Número de Utilizações	Medida do número de vezes que o componente foi (re)utilizado.
		Vida Útil	É o período de tempo entre cada (re)utilização do componente.
	Simplicidade	Complexidade	Número de caminhos lógicos em um programa ou componente.
		Regularidade	É a razão entre a extensão estimada e a extensão real de um componente.
		Tamanho	Os componentes tem sua extensão compreendida entre valores mínimo e máximo estabelecidos como padrão.
	Modularidade	Fan-in	Número de módulos superiores (número de módulos chamantes).
		Fan-out	Número de módulos inferiores (número de módulos chamados).
		Acoplamento	É a relação de interdependência entre componentes.
		Não Memorização	O componente não possui memória de existência prévia, executando, a cada ativação, como se fosse a primeira vez, ou seja, sem memória de estados anteriores.

Tabela 2 - Subfatores Generalidade, Maturidade, Simplicidade e Modularidade.

3. Lógica Fuzzy

A lógica fuzzy é uma lógica de valores múltiplos que define níveis ou graus de pertinência de um elemento em um conjunto - uma forma prática de lidar com questões do mundo real. Segundo Zadeh [ZADE73], [ZADE88], o criador da lógica fuzzy, ela é *um tipo de lógica que utiliza graduações ou declarações qualificadas ao invés daquelas que são estritamente verdadeiro ou falso.*

Um conjunto fuzzy é *um conjunto que não tem um grau de pertinência rígido, ou seja, permite que os objetos tenham graus ou níveis de pertinência no intervalo unitário [0,1]* [ZADE84]. Bezdek [BEZD93] observa que os conjuntos fuzzy são uma generalização da

teoria convencional de conjuntos que permitem representar a imprecisão do cotidiano. Outro conceito importante é o de variável linguística, que segundo Zadeh [ZADE84] *são termos ordinários de linguagem que são usados para representar um conjunto fuzzy particular em um dado problema, tais como "grande", "pequeno", "médio" ou "OK".*

Conjuntos convencionais contém objetos que satisfazem propriedades específicas necessárias para serem membros do conjunto. O conjunto de números *H*, Figura 1 (a), de 6 a 8 é convencional e descrito por

$$H = \{ r \in \Re \mid 6 \leq r \leq 8 \}$$

De forma equivalente, *H* é descrito pela sua *função de pertinência* (FP),

$$p_H : \Re \rightarrow \{0,1\}$$

definida como

$$p_H(r) = \begin{cases} 1; & 6 \leq r \leq 8 \\ 0; & r < 6 \text{ ou } r > 8 \end{cases}$$

O conjunto convencional *H* e o gráfico de p_H são mostrados na Figura 1 (a) e (c) respectivamente. Enquanto que p_H mapeia todos os números reais *r* no intervalo fechado [0,1], os conjuntos convencionais correspondem a uma lógica binária: é ou não é, 0 ou 1, pertence ou não pertence [BEZD93].

Consideremos agora o conjunto *F* dos números reais *próximos a 7*. Uma vez que a propriedade "próximos a 7" é imprecisa (fuzzy), não existe somente uma função de pertinência para *F*. Pelo contrário, o modelador deve decidir, baseado na aplicação e nas propriedades desejadas para *F*, como ela deve ser.

As funções apresentadas na Figura 1 (b) e (d) podem ser representações úteis de *F*. p_{F1} é discreta (o gráfico escada), enquanto que p_{F2} é contínua (o gráfico triangular). Pode-se facilmente construir uma FP para *F* de forma que todo número tenha um grau de pertinência positivo em *F*, por exemplo.

Uma das grandes diferenças entre os conjuntos convencional e fuzzy, é que o primeiro tem sempre uma única função de pertinência, enquanto que todo conjunto fuzzy tem um número infinito de funções de pertinência que podem representá-lo. Isto representa, ao mesmo tempo, fragilidade e força; a unicidade é sacrificada, mas oferece um ganho paralelo em termos de flexibilidade, permitindo que os modelos fuzzy possam ser "ajustados" para se obter um aproveitamento máximo em uma dada situação [BEZD93].

Toda função $p : X \rightarrow [0,1]$ é um conjunto fuzzy. Apesar disto ser verdade do ponto de vista da matemática formal, muitas funções que se encaixam nesta definição não podem ser interpretadas convenientemente como uma realização conceitual de um conjunto fuzzy. Em outras palavras, funções que mapeiam *X* no intervalo unitário *podem* ser conjuntos fuzzy, mas *tornam-se* conjuntos fuzzy quando, e somente quando, elas combinam alguma descrição semântica plausível de propriedades imprecisas dos objetos em *X* [BEZD93].

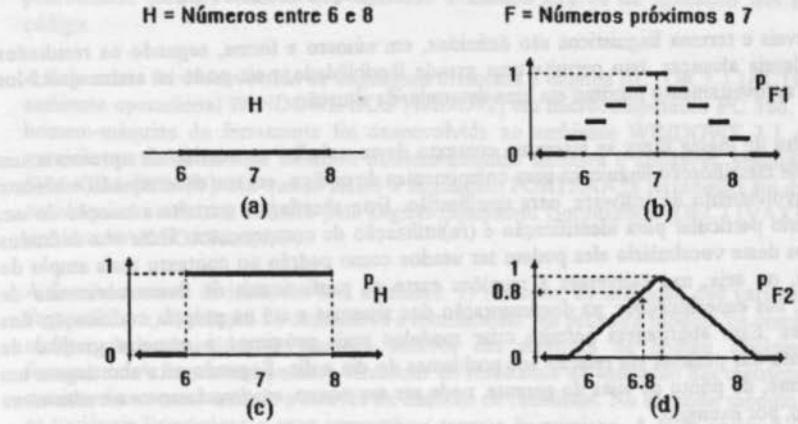

Figura 1 - O subconjunto H (a) e as funções de pertinência para os subconjuntos de $\Re$, convencional (c) e fuzzy (b), (d) [BEZD93].

Exemplificando, vamos considerar uma classificação de programas por tamanho, medido em linhas de código fonte (LCF). A Figura 2 apresenta um gráfico com quatro conjuntos fuzzy representados pelas quatro curvas, da esquerda para a direita, como "Mínimo", "Pequeno", "Médio" e "Grande", respectivamente. Portanto, a variável linguística **Tamanho** pode assumir quatro valores distintos: **"Mínimo"**, **"Pequeno"**, **"Médio"** e **"Grande"**, que são aqui chamados de termos linguísticos. Temos, então, *termo linguístico*: linhas → [0,1]. O mapeamento deste conceito pode ser, por exemplo, o caso no qual um programa com 30 linhas tem um grau de pertinência de 1.0 em **Pequeno**, enquanto que um componente de 55 linhas tem um grau de pertinência de somente 0.7 na mesma classe.

A lógica fuzzy viola o "princípio da não-contradição" no sentido de que, por exemplo, um programa pode ser pequeno e médio (pertencer a dois conjuntos) ao mesmo tempo. É o caso do programa de 55 linhas que tem um grau de pertinência de 0.7 em **Pequeno** e aproximadamente 0.2 em **Médio**. Temos, portanto, classificações diferentes para um programa ao mesmo tempo.

A escolha de um componente pode ser feita através do grau máximo de pertinência de um componente em uma variável linguística. Por exemplo, caso se desejasse o maior grau de pertinência do componente de 55 LCF, do exemplo anterior na variável linguística **Tamanho**, ele seria classificado como **Pequeno**, uma vez que o seu grau de pertinência no termo linguístico **Pequeno** é 0.7.

Um critério de seleção poderia ser selecionar componentes segundo um grau de pertinência diferente de zero em um termo linguístico específico. Por exemplo, vamos considerar que o universo de componentes é composto pelos componentes de 30 e 55 LCF dos exemplos anteriores. Se fossemos escolher componentes com grau de pertinência diferente de zero para o termo linguístico **Pequeno** teríamos como resultado os dois componentes do universo (30 e

55 LCF). Caso estivéssemos selecionando componentes classificados segundo o termo linguístico **Médio** teríamos como resposta somente o componente de 55 LCF.

As variáveis e termos linguísticos são definidos, em número e forma, segundo os resultados que se deseja alcançar. Isto permite uma grande flexibilidade, pois pode-se assim ajustá-los para um aproveitamento máximo em uma determinada situação.

O conceito de lógica fuzzy se insere no contexto deste trabalho no sentido de apresentar um modelo de classificação linguística para componentes de código, em um determinado ambiente de desenvolvimento de software, para reutilização. Esta abordagem permite a criação de um vocabulário particular para identificação e (re)utilização de componentes. Uma vez definidos os termos deste vocabulário eles podem ser usados como padrão no contexto mais amplo do ambiente, ou seja, nas conversas e reuniões entre os profissionais de desenvolvimento de software, nas especificações, na documentação dos sistemas e até na própria codificação dos programas. Esta abordagem permite criar modelos mais próximos à maneira gradual de pensamento das pessoas em relação aos problemas do dia a dia. Segundo esta abordagem um componente, do ponto de vista do gerente, pode ser *um pouco*, *moderadamente* ou *altamente* complexo, por exemplo.

Assim, os valores numéricos podem ser interpretados através de variáveis e termos linguísticos que oferecem uma interpretação mais lógica desses valores numéricos, ainda que relativamente subjetiva. Deve-se levar em consideração o fato de que a interpretação das variáveis e termos linguísticos pode ter significados diferentes para usuários diferentes, devendo portanto ser seguido um padrão estabelecido para o ambiente ou, pelo menos, diretivas básicas para a definição das variáveis e termos linguísticos.

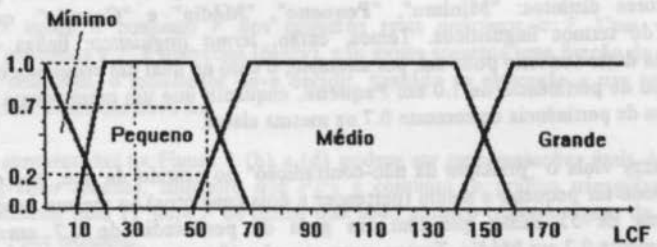

Figura 2 - Exemplo de uma variável linguística, **Tamanho**, para classificação do tamanho de um componente em número de linhas de código fonte (LCF). A variável **Tamanho** é definida por quatro termos linguísticos: **"Mínimo"**, **"Pequeno"**, **"Médio"** e **"Grande"**.

4. ReFOR: Uma Ferramenta para Seleção de Módulos Reutilizáveis

Para que o processo de avaliação da qualidade seja, de fato, vantajoso, viável e utilizável, é preciso que se usem ferramentas automatizadas. A coleta manual de dados, durante a avaliação, é um processo propenso a erros, além de demandar muito tempo. Ferramentas automatizadas implicam na melhoria da coleta e da análise dos dados, oriundos da aplicação das métricas, na redução dos custos desse processo e na disponibilidade mais rápida dos resultados sempre que for necessário.

Assim sendo, foi construída a ferramenta ReFOR cujo objetivo é selecionar módulos FORTRAN candidatos a fazer parte de uma bilbioteca de componentes reutilizáveis, a partir

da aplicação de métricas e utilização de variáveis e termos linguísticos. A utilização de variáveis e termos linguísticos tem por objetivo permitir o estabelecimento de uma linguagem padronizada para os valores dos atributos avaliados através da aplicação das métricas de código.

A ferramenta foi desenvolvida na linguagem orientada a objetos ACTOR 3.1 [ACTO91], sob o ambiente operacional WINDOWS/DOS [WIND92] em microcomputador PC 386. A interface homem-máquina da ferramenta foi desenvolvida no ambiente WINDOWS 3.1 de forma a garantir aos usuários uma interface homem-máquina intuitiva e amigável, segundo o padrão SAA-CUA [IBM89]. No que se refere à linguagem FORTRAN, a ferramenta foi desenvolvida considerando a sintaxe definida pela Digital Equipment Corporation (DEC) [VAXF84] para o produto VAX FORTRAN 4.0.

A ferramenta está dividida em três módulos: 1) medição, 2) definição das variáveis e termos linguísticos, e, 3) seleção de candidatos à reutilização. No primeiro módulo os componentes de código são submetidos à medição através das métricas de código implementadas na ferramenta. A ferramenta permite visualizar os resultados da aplicação das métricas através da sua interface com o usuário e através da emissão de relatórios. No segundo módulo são criadas as variáveis linguísticas e seus respectivos termos linguísticos. A toda variável linguística é associada uma métrica escolhida pelo usuário. Pode-se aí, também, modificar e excluir variáveis linguísticas e seus respectivos termos linguísticos. Por fim, no último módulo, são estabelecidos os critérios para seleção de candidatos à reutilização segundo as escolhas feitas pelo usuário. O processo de seleção é ativado pelo usuário. O resultado da seleção pode ser visualizado através de funções da interface gráfica e através de relatórios. A Figura 3 apresenta a estrutura da ferramenta ReFOR.

4.1 Descrição do Funcionamento de ReFOR

A ferramenta apresenta, na sua janela principal, uma lista com todos os componentes de código que podem ser submetidos à medição. Esses componentes de código fonte são programas, subrotinas e funções, que devem, obrigatoriamente, ter sido previamente compilados. A ferramenta apresenta a relação de todos os componentes que encontrar no diretório alvo. Estes estão classificados em ordem alfabética pelo nome do componente, tipo (FUNCTION, SUBROUTINE ou PROGRAM). A medição pode ser efetuada individualmente (por componente) ou todos de uma só vez.

O usuário pode, então, executar a medição de um componente (o selecionado na lista de componentes) ou pode escolher executar a medição de todos os componentes da lista. O sistema sinaliza quando acabou de medir um componente (ou todos) e marca, na lista de componentes, aqueles que foram processados. Pode-se, então, visualizar (ou imprimir) os resultados da avaliação segundo a opção escolhida no menu. Nesta primeira etapa os componentes (ou componente) são submetidos às métricas e critérios estabelecidos e os resultados são os valores numéricos das medidas. O usuário pode, ainda nesta etapa, solicitar um relatório geral, ou individual, sobre os componentes avaliados. Estes relatórios tem por objetivo apresentar ao usuário um plano geral das medidas efetuadas, fornecendo subsídios para a criação das variáveis linguísticas.

Para criação das variáveis linguísticas, o usuário utiliza o editor de variáveis linguísticas. Para criar uma variável linguística nova, o usuário deve fornecer o nome da variável, a escala do eixo das abcissas (o intervalo de variação desejado), a unidade de medida da variável e associar

uma métrica à variável, dentre as disponíveis na lista apresentada. O eixo das ordenadas (y) tem escala fixa de zero a um (0,1), pois é nele que será identificado o grau de pertinência. A partir deste ponto podem ser criados os termos linguísticos da variável linguística. Para criar um termo linguístico é necessário fornecer o nome do termo linguístico e os valores numéricos da abcissa em função dos vértices da curva escolhida. Cada termo linguístico criado é apresentado em uma cor diferente e, é inserido na lista de termos linguísticos.

Após criar as variáveis linguísticas, o usuário pode configurar a seleção de candidatos à reutilização segundo suas necessidades. Ele pode, então, selecionar as variáveis linguísticas desejadas e o(s) respectivo(s) termo(s) linguístico(s) escolhido(s). O usuário pode pedir uma classificação de todos os componentes, conforme uma configuração efetuada, para ter uma visão geral dos componentes do ponto de vista das variáveis linguísticas. Nesta última etapa, pode-se avaliar os resultados através de relatórios gerais. A análise destes relatórios permite efetuar ajustes nas variáveis linguísticas, de forma a otimizar o processo segundo a intenção do usuário, realimentando o processo segundo as suas necessidades.

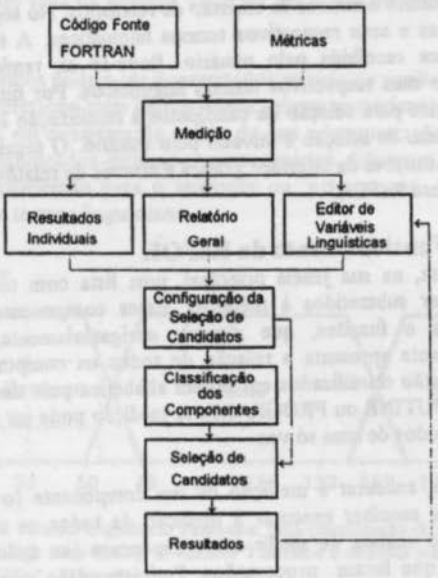

Figura 3 - Estrutura da ferramenta ReFOR.

4.2 Métricas Utilizadas pela Ferramenta

A partir dos subfatores e critérios identificados como relacionados à reutilizabilidade, selecionamos alguns para implementação na primeira versão da ferramenta ReFOR. São avaliados os seguintes critérios:

a) **Tamanho** é avaliado através das seguintes métricas:

- Métrica de tamanho (N) de Halstead [HALS77]: $N = N_1 + N_2$
- Métrica de volume (V) de Halstead: $V = N \times \log_2 n$

- Número de linhas de código: contagem do número de linhas de código, excluídas as linhas em branco e as linhas de comentário.

b) **Regularidade** é avaliada através da seguinte métrica:
- Métricas de tamanho estimado ($N^$) e de tamanho real (N) de Halstead: $r = N^ / N$

c) **Complexidade** é avaliada através das seguintes métricas:
- Número ciclomático de McCabe[McCA76]: número de IF's
- Número ciclomático de McCabe modificado: número de IF's + AND's + OR's + XOR's + NOT's + EQV's + NEQV's

d) **Padronização** é avaliada através da seguinte métrica:
- Bloco de comentários inicial do componente: o bloco de comentários inicial do componente é avaliado segundo o padrão exigido (nome do autor, data de criação, data e motivo das modificações, descrição do objetivo, descrição dos parâmetros de E/S e descrição das variáveis usadas).

e) **Organização Visual** é avaliada através da seguinte métrica:
- Escore de zero a dez onde é exigido que:
  1) se tenha uma linha em branco antes e depois de uma chamada de rotina;
  2) uma linha em branco antes e depois de um bloco de comentários;
  3) a numeração dos labels deve estar em ordem crescente;
  4) o deslocamento de pelo menos um branco, para esquerda (indentação), nos comandos seguintes a um IF, a um DO e nas linhas de continuação.

  Cada um dos quatro itens é calculado através da percentagem de atendimento à exigência e convertido para um valor na escala de zero a dez. O escore é calculado através da média aritmética dos quatro itens relacionados.

f) **Documentação Interna** é avaliada através da seguinte métrica:
- Percentagem de comentários do componente.

g) **Programação Estruturada** é avaliada através da seguinte métrica:
- Escore de zero a dez onde é exigido:
  1) nenhuma ocorrência do comando ENTRY (cada ocorrência subtrai três do escore inicial dez);
  2) somente um comando RETURN (no caso de subrotinas; cada ocorrência adicional subtrai um do escore inicial);
  3) número mínimo de comandos GOTO (cada ocorrência subtrai dois do escore inicial).

  A partir do escore inicial dez são aplicadas às subtrações relacionadas acima.

h) **Fan-out** é avaliado através da seguinte métrica:
- Número de componentes chamados: contagem do número de componentes chamados.

i) **Fan-in** é avaliado através da seguinte métrica:
- Número de vezes que é chamado: contagem do número de vezes que o componente é chamado (a partir do fan-out).

j) **Não Memorização** é avaliado através da seguinte métrica:

- Escore de zero a dez onde:
  1) cada ocorrência do comandos COMMON (local e bloco) subtrai três do valor inicial dez;
  2) cada ocorrência do comando EQUIVALENCE subtrai dois do valor inicial dez.

5 Exemplo de Utilização da Ferramenta

Durante o processo de validação da ferramenta ReFOR foram avaliados quarenta e nove componentes (subrotinas e programas) de um subsistema do Sistema Integrado de Computadores de Angra (SICA), desenvolvido pelo Programa de Engenharia Nuclear da COPPE. Este subsistema foi escolhido por apresentar o código FORTRAN mais "puro", isto é, com o mínimo de chamadas de funções do sistema operacional, gerenciadores de tela e banco de dados. A Figura 4 apresenta um relatório individual, de uma subrotina FORTRAN, com o resultado das medidas efetuadas pela ferramenta ReFOR, na primeira etapa.

```
Relatório Individual: SUBROUTINE GRP_TO_GRC
Linhas de Código
   19  linhas de código fonte
    4  linhas de comentário não-alfanumérico
    2  linhas de comentário com palavras
   60  linhas de comentário no bloco inicial
   66  linhas de comentário (total)
   19  linhas em branco
  104  linhas (total do componente)
 18.3  % de linhas em branco
 63.5  % de linhas de comentário
 18.3  % de linhas de código fonte
Complexidade
  2 - Número ciclomático de McCabe
  2 - Número ciclomático de McCabe modificado
Métricas de Halstead
 20 - Número de operadores              (n1)
 45 - Número de ocorrência operadores   (N1)
 14 - Número de operandos               (n2)
 25 - Número de ocorrência operandos    (N2)
 34 - Vocabulário                       (n = n1+n2)
 70 - Tamanho                           (N = N1+N2)
139 - Tamanho estimado                  (N^)
  2. - Regularidade                     (r = N^/N)
356 - Volume                            (V)
Fan-out
   0.
Fan-in
   3.
Organização Visual
   8.6 - Organização Visual (0 a 10)
Não-Memorização
  10. - Não-Memorização (0 a 10)
Programação Estruturada
  10. - Programação Estruturada (0 a 10)
```

Figura 4 - Relatório individual das medidas de uma subrotina, efetuado pela ferramenta ReFOR.

Operadores e ocorrências * 1

PDF compression, OCR, web optimization

&	5
,	5
/	1
-	2
=	4
.EQ.	1
BYTE	1
ELSE	1
END IF	1
END	1
IF	1
IMPLICIT	1
INCLUDE	1
INTEGER*4	1
REAL*4	1
RETURN	1
SUBROUTINE	1
THEN	1
fim-de-linha	14
n1= 20	N1= 45

Operandos e ocorrências	
A	1
Z	1
'P2500$PARAMETROS:SISTEMA.INC'	1
9.	1
32.	1
VALIDO	1
5.	1
GRF_TO_GRC	1
0.0	1
INVALIDO	2
GR_FARENHEIT	3
STS_GR_FARENHEIT	3
GR_CELSIUS	4
STS_GR_CELSIUS	4
n2= 14	N2= 25

Figura 4 (cont.) -Relatório individual das medidas de uma subrotina, efetuado pela ferramenta ReFOR.

A Figura 6 apresenta uma classificação de dez componentes segundo as variáveis linguísticas apresentadas nas Figuras 5(a), 5(b), 5(c) e 5(d). O critério de classificação utilizado foi o de maior grau de pertinência em um termo linguístico de uma variável linguística.

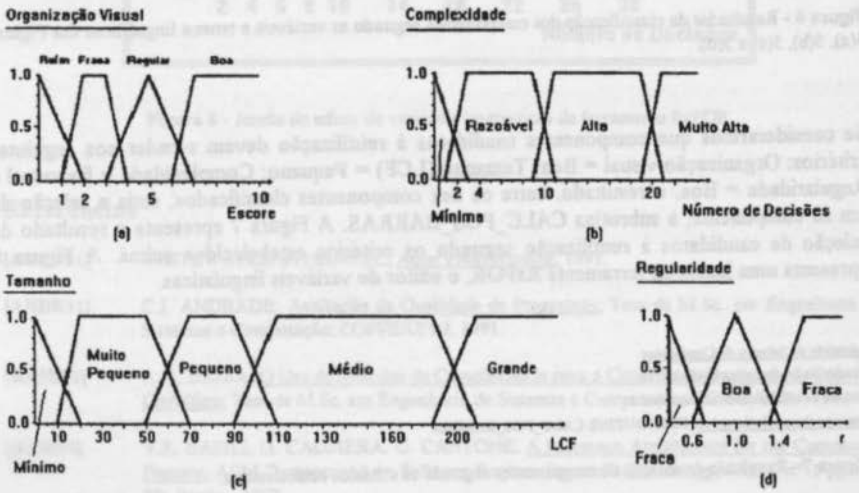

Figura 5 - Variáveis e termos linguísticos usados na classificação dos componentes: (a) organização visual; (b) complexidade; (c) tamanho; (d) regularidade.

Relatório de Classificação: Número de componentes classificados = 10

Componente SUBROUTINE AVGINT
Organização visual = Boa (9.2)/ Tamanho (LCF) = Médio (157 LCF)/ Complexidade = Muito Alta (26)/ Regularidade = Boa (0.98)

Componente SUBROUTINE AVGINT_LEITURA_HISTÓRICO
Organização visual = Boa (8.8)/ Tamanho (LCF) = Grande (264 LCF)/ Complexidade = Muito Alta (55)/ Regularidade = Boa (0.80)

Componente SUBROUTINE CALC_POS_BARRAS
Organização visual = Boa (8.9)/ Tamanho (LCF) = Pequeno (74 LCF)/ Complexidade = Razoável (9)/ Regularidade = Boa (0.74)

Componente SUBROUTINE CONC_BORO_COMP
Organização visual = Boa (8.6)/ Tamanho (LCF) = Muito_Pequeno (34 LCF)/ Complexidade = Razoável (3)/ Regularidade = Fraca (1.49)

Componente SUBROUTINE CONC_IODO_POT_CTE
Organização visual = Boa (7.3)/ Tamanho (LCF) = Muito_Pequeno(35 LCF)/ Complexidade = Razoável (4)/ Regularidade = Fraca (1.51)

Componente SUBROUTINE CONC_PM_POT_CTE
Organização visual = Boa (7.3)/ Tamanho (LCF) = Muito_Pequeno(35 LCF)/ Complexidade = Razoável (4)/ Regularidade = Fraca (1.51)

Componente SUBROUTINE CONC_SM_POT_CTE
Organização visual = Boa (7.9)/ Tamanho (LCF) = Muito_Pequeno(52 LCF)/ Complexidade = Razoável (5)/ Regularidade = Boa (1.18)

Componente SUBROUTINE CONC_XE_POT_CTE
Organização visual = Boa (7.6)/ Tamanho (LCF) = Muito_Pequeno(56 LCF)/ Complexidade = Razoável (5)/ Regularidade = Boa (1.19)

Componente SUBROUTINE CONC_XENONIO_EQUILIBRIO
Organização visual = Boa (6.9)/ Tamanho (LCF) = Muito_Pequeno(23 LCF)/ Complexidade = Razoável (2)/ Regularidade = Fraca (1.97)

Componente SUBROUTINE GRF_TO_GRC
Organização visual = Boa (8.6)/ Tamanho (LCF) = Muito_Pequeno(19 LCF)/ Complexidade = Razoável (2)/ Regularidade = Fraca (2.)

Figura 6 - Resultados da classificação dos componentes segundo as variáveis e termos linguísticos das Figuras 5(a), 5(b), 5(c) e 5(d).

Se considerarmos que componentes candidatos à reutilização devem atender aos seguintes critérios: Organização visual = Boa; Tamanho (LCF) = Pequeno; Complexidade = Razoável e Regularidade = Boa, o resultado, entre os dez componentes classificados, seria a seleção de um só componente, a subrotina CALC_POS_BARRAS. A Figura 7 apresenta o resultado da seleção de candidatos à reutilização segundo os critérios estabelecidos acima. A Figura 8 apresenta uma janela da ferramenta ReFOR, o editor de variáveis linguísticas.

Relatório de Seleção de Candidatos
Número de componentes avaliados: 10
Número de componentes selecionados: 1
Componentes selecionados: SUBROUTINE CALC_POS_BARRAS

Figura 7 - Resultado da seleção de componentes segundo os critérios estabelecidos.

PDF compression, OCR, web optimization using

6 Conclusão

Este trabalho apresentou um conjunto de atributos que devem ser avaliados ao se considerar candidatos a inclusão em uma biblioteca de componentes reutilizáveis. Foi também desenvolvida uma ferramenta (ReFOR) que apóia a avaliação de código FORTRAN segundo estes atributos. Uma descrição mais detalhada dos atributos e da ferramenta pode ser encontrada em [COME94].

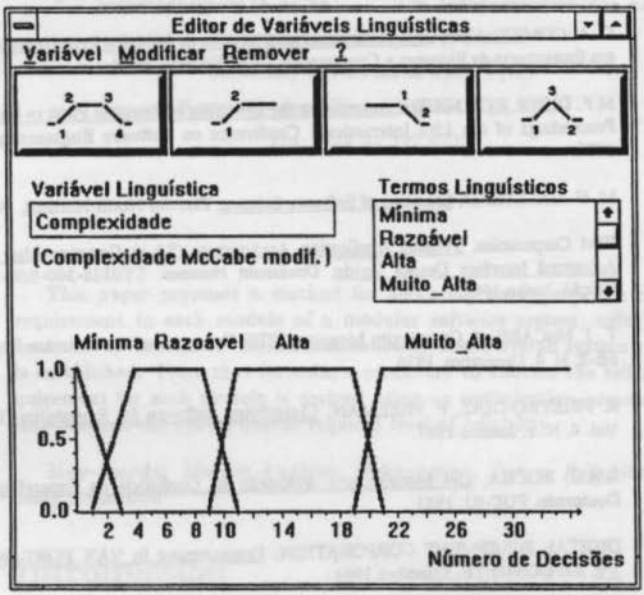

Figura 8 - Janela do editor de variáveis linguísticas da ferramenta ReFOR.

Referências

[ACTO91] WHITEWATER/SYMANTEC; Actor Programming; 1991.

[ANDR91] C.J. ANDRADE; Avaliação da Qualidade de Programas; Tese de M.Sc. em Engenharia de Sistemas e Computação; COPPE/UFRJ, 1991.

[BAHI92] A. S. BAHIA; O Uso de Métricas de Complexidade para o Controle da Qualidade de Software Científico; Tese de M.Sc. em Engenharia de Sistemas e Computação; COPPE/UFRJ, 1992.

[BASI92] V.R. BASILI, G. CALDIERA, G. CANTONE; A Reference Architecture for the Component Factory; ACM Transactions on Software Engineering and Methodology, Vol. 1 nº 1 pp. 53-80; Janeiro, 1992.

[BELC92] A. D. BELCHIOR; Controle da Qualidade de Software Financeiro; Tese de M.Sc. em Engenharia de Sistemas e Computação; COPPE/UFRJ, 1992.

[BEZD93] J. BEZDEK; Fuzzy Models - What Are They, and Why? (Editorial); IEEE Transactions on Fuzzy Systems, Vol. 1, No. 1; Fevereiro, 1993.

[CALD91] G. CALDIERA, V. R. BASILI; Identifying and Qualifying Reusable Software Components; Computer pp. 61-70; Fevereiro 1991.

[CLUN87] C. CLUNIE; Verificação e Validação de Especificações; Tese de M.Sc. em Engenharia de Sistemas e Computação; COPPE/UFRJ, 1987.

[COME94] C. A. COMERLATO; Uma Ferramenta para Seleção de Módulos Reutilizáveis; Tese de M.Sc. em Engenharia de Sistemas e Computação; COPPE/UFRJ, 1994.

[DUNN93] M.F. DUNN, J.C KNIGTH; Automating the Detection of Reusable Parts in Existing Software, Proceedings of the 15th International Conference on Software Engineering; pp. 381-390; 1993.

[HALS77] M. H. HALSTEAD; Elements of Software Science; Elsevier North-Holland, 1977.

[IBM89] IBM Corporation; Systems Application Architecture (SAA) Common User Access (CUA) Advanced Interface Design Guide; Document Number: SY0328-300-R00-1089; Primeira Edição, Junho 1989.

[MCCA76] T. J. McCABE; A Complexity Measure; IEEE Transactions on Software Engineering, Vol. SE-2, N. 4, Dezembro, 1976.

[PRIE87] R. PRIETRO-DIAZ, P. FREEMAN; Classifiying Software for Reusability; IEEE Software, Vol. 4, N. 1, Janeiro 1987.

[ROCH83] A.R.C. ROCHA; Um Modelo para Avaliação da Qualidade de Especificações; Tese de Doutorado, PUC-RJ; 1983.

[VAXF84] DIGITAL EQUIPMENT CORPORATION; Programming in VAX FORTAN - VAX/VMS V4; AA-DO34D-TE, Setembro 1984.

[WIND92] MICROSOFT; Microsoft Windows - Guia do Usuário; Microsoft Corporation 1992.

[ZADE73] L.A.ZADEH; Outline of a New Approach to the Analysis of Complex Systems and Decision Processes; IEEE Transactions on Systems, Man, and Cybernetics; Vol. SMC-3, N. 1, Janeiro 1973.

[ZADE84] L.A.ZADEH; Making Computers Think Like People; IEEE Spectrum, Agosto 1984.

[ZADE88] L.A.ZADEH; Fuzzy Logic; IEEE COMPUTER, Abril 1988.

PDF compression, OCR, web optimization