POKE-TOOL - UMA FERRAMENTA PARA SUPORTE À APLICAÇÃO DOS CRITÉRIOS POTENCIAIS USOS PARA TESTE DE PROGRAMAS

POKE-TOOL - A TOOL TO SUPPORT POTENTIAL USES CRITERIA APPLICATION TO PROGRAM TESTING

Prof. Dr. José Carlos MALDONADO*
Marcos Lordello CHAIM**
Prof. Dr. Mario JINO***

ABSTRACT

This work describes implementation and operation of a multilanguage tool to support Potential Uses Criteria application, named POKE-TOOL (**PO**tencial Uses **CRI**teria **Tool** for Program Testing). Presently, this tool assists application of structural testing criteria to programs written in C, automating unit testing and permittinf more complex programs to be tested. POKE-TOOL also organizes generated data through a data base of testinf data.

KEY WORDS: Structural Software Testing, Automated Tool, Data Flow Analysis.

SUMÁRIO

Este trabalho descreve a implementação e a operação de uma ferramenta multilinguagem de suporte à aplicação dos critérios Potenciais Usos, denominada POKE-TOOL (**PO**tencial Uses **CRI**teria **Tool** for Program Testing). Atualmente, esta ferramenta auxilia a aplicação de critérios de teste estruturais em programas escritos na linguagem C, automatizando o teste de unidades e permitindo que programas mais complexos possam ser testados. A POKE-TOOL também organiza os dados gerados com a criação de uma base de dados da atividade de teste.

PALAVRAS-CHAVE: Teste Estrutural de Software, Ferramenta Automatizada, Análise de Fluxo de Dados.

1. INTRODUÇÃO

1.1 Motivação

À medida que os sistemas de software cresceram em tamanho e complexidade, o esforço requerido para testar esses sistemas tem crescido muito além das expectativas, implicando altos custos e produtos com baixa confiabilidade. Tem-se verificado que embora se gaste, em geral, até 50% do orçamento para desenvolvimento do software em atividades de teste, um número significativo de defeitos permanece sem ser detectado nos softwares liberados; esses defeitos normalmente têm um impacto grande na operação normal do sistema. Este alto custo das atividades de teste ensejou o desenvolvimento de métodos mais eficientes na detecção de defeitos no software e de ferramentas automáticas para auxiliar na produção de testes efetivos e na análise dos resultados dos testes.

Basicamente, os métodos utilizados para testar as unidades que compõem um software são baseados em duas técnicas: funcional e estrutural. Com respeito ao teste estrutural vários critérios para a seleção de casos de teste foram estabelecidos. Os primeiros critérios a surgirem foram baseados unicamente no fluxo de controle dos programas. Assim, têm-se critérios que requerem que todo os blocos de comandos seqüênciais de um programa sejam executados pelos menos uma vez (chamado critério todos-nós), ou todos os comandos de

(*) Professor do Departamento de Ciências da Computação e Estística Instituto de Ciências Matemáticas de São Carlos - ICMSC-USP.
(**) Pesquisador do Centro Nacional de Pesquisa Tecnológica em Informática para Agricultura Empresa Brasileira de Pesquisa Agropecuária - EMBRAPA.
(***) Professor de Engenharia de Computação e Automação Industrial Faculdade de Engenharia Elétrica - FEE - UNICAMP

transferência (critério todos-ramos), ou mesmo todos os caminhos do programa (critério todos-caminhos). Note-se, entretanto, que este último critério pode levar a um número infinito de casos de teste.

Com o intuito de tornar o teste de unidades mais eficiente, novos critérios de teste foram introduzidos (HER76, LAS83, NTA84, RAP85, MAL88a, MAL88a, MAL88b, URA88). Estes novos critérios de teste baseiam seus requisitos de teste na análise de fluxo de dados da unidade. A análise de fluxo de dados foi inicialmente utilizada para otimização de código por compiladores e, em geral, estabelece que a ocorrência de uma variável pode ser de dois tipos: definição e uso. Uma ocorrência é entendida como definição quando a variável recebe um valor através, por exemplo, de um comando de atribuição ou um comando de entrada de dados. Um uso ocorre quando a variável não está sendo definida. No contexto de teste de software, critérios de teste baseados em análise de fluxo de dados requerem que as interações que envolvem definições de variáveis de programa e subseqüentes referências a essas definições sejam testadas. Portanto, esses critérios baseiam-se nas associações entre uma definição de uma variável e os seus possíveis subseqüentes usos para a derivação de casos de teste.

Especificamente, Maldonado, Chaim e Jino (MAL88a, MAL88b), introduziram os Critérios Potenciais Usos baseados no conceito potencial uso; os Critérios Potenciais Usos são critérios de teste estrutural, baseados na análise de fluxo de dados e consistem, fundamentalmente, em variações da família de critérios apresentada por Rapps e Weyuker (RAP82, RAP85); são denominados: critérios todos-potenciais-du-caminhos, todos-potenciais-usos e todos-potenciais-usos/du. As associações são requeridas independentemente da ocorrência explícita de uma referência a uma determinada definição; se um uso pode existir - um potencial uso - a potencial associação é requerida. Uma associação é representada por (i, J, x) onde i é um nó onde existe uma definição de variável x e j é um nó onde existe um potencial uso de x. Um uso pode existir em j se for possível alcançar este nó através de pelo menos um caminho onde não ocorra redefinição de x.

O uso de uma ferramenta de software para auxílio ao teste de programas pode ser vinculado a um critério de teste de duas maneiras. A ferramenta de software pode utilizar o critério de teste como um guia para a geração de casos de teste que satisfaçam o critério; outra possibilidade é a utilização critério para a análise de cobertura de um conjunto de casos de teste, isto é, verificar se os casos de teste aplicados preencheram os requisitos de teste do critério.

Dessa maneira, pode-se concluir que as ferramentas automatizadas de suporte à aplicação dos diversos critérios de teste de programas são a chave para o aumento na produtividade e da eficiência da atividade de teste, pois permitem que programas maiores e mais complexos sejam testados e também automatizam a aplicação dos mais variados critérios, em especial, daqueles baseados em análise de fluxo de dados.

1.2 Organização do Trabalho

Este trabalho descreve a implementação e a operação de uma ferramenta de suporte à aplicação dos critérios Potenciais Usos, denominada POKE-TOOL (**PO**tential Uses **CRI**teria **Tool** for Program Testing) (MAL89)1. Essa ferramenta auxilia a aplicação de critérios de teste estruturais em programas escritos na linguagem C, automatizando o teste de unidades e permitindo que programas mais complexos possam ser testados. A POKE-TOOL também organiza os dados gerados com a criação de uma base de dados da atividade de teste. Dessa maneira, este tipo de ferramenta procura aumentar a produtividade na fase de teste e aumentar a qualidade dos softwares liberados.

Ela fornece ao usuário os caminhos necessários para satisfazer os Critérios Potenciais Usos e é capaz de verificar se o conjunto de casos de teste fornecido pelo usuário executou todos os caminhos requeridos. A POKE-TOOL é uma ferramenta flexível, isto é, não atrelada a nenhuma linguagem de programação específica, e permite que o usuário a configure para a linguagem de programação de seu interesse.

Na Seção seguinte são apresentadas a arquitetura estabelecida para a ferramenta e a implementação deste trabalho; na Seção 3 os aspectos de configuração da POKE-TOOL; na Seção 4, os aspectos operacionais da ferramenta são apresentados através de um exemplo de utilização; e, na Seção 5, as conclusões são apresentadas.

2. ARQUITETURA E IMPLEMENTAÇÃO DA POKE-TOOL

Nesta seção são apresentadas a arquitetura e a implementação da ferramenta POKE-TOOL.

Com a implementação da ferramenta POKE-TOOL pretende-se, além de auxiliar o uso prático dos critérios Potenciais Usos, viabilizar a realização de comparações entre estes critérios e os demais critérios de teste estrutural, bem como a avaliação da adequação destes critérios a classes de erros.

2.1 Arquitetura Funcional da Ferramenta POKE-TOOL

A arquitetura funcional da ferramenta POKE-TOOL, ilustrada na Figura 1, foi inicialmente proposta por Maldonado, Chaim e Jino (MAL89).

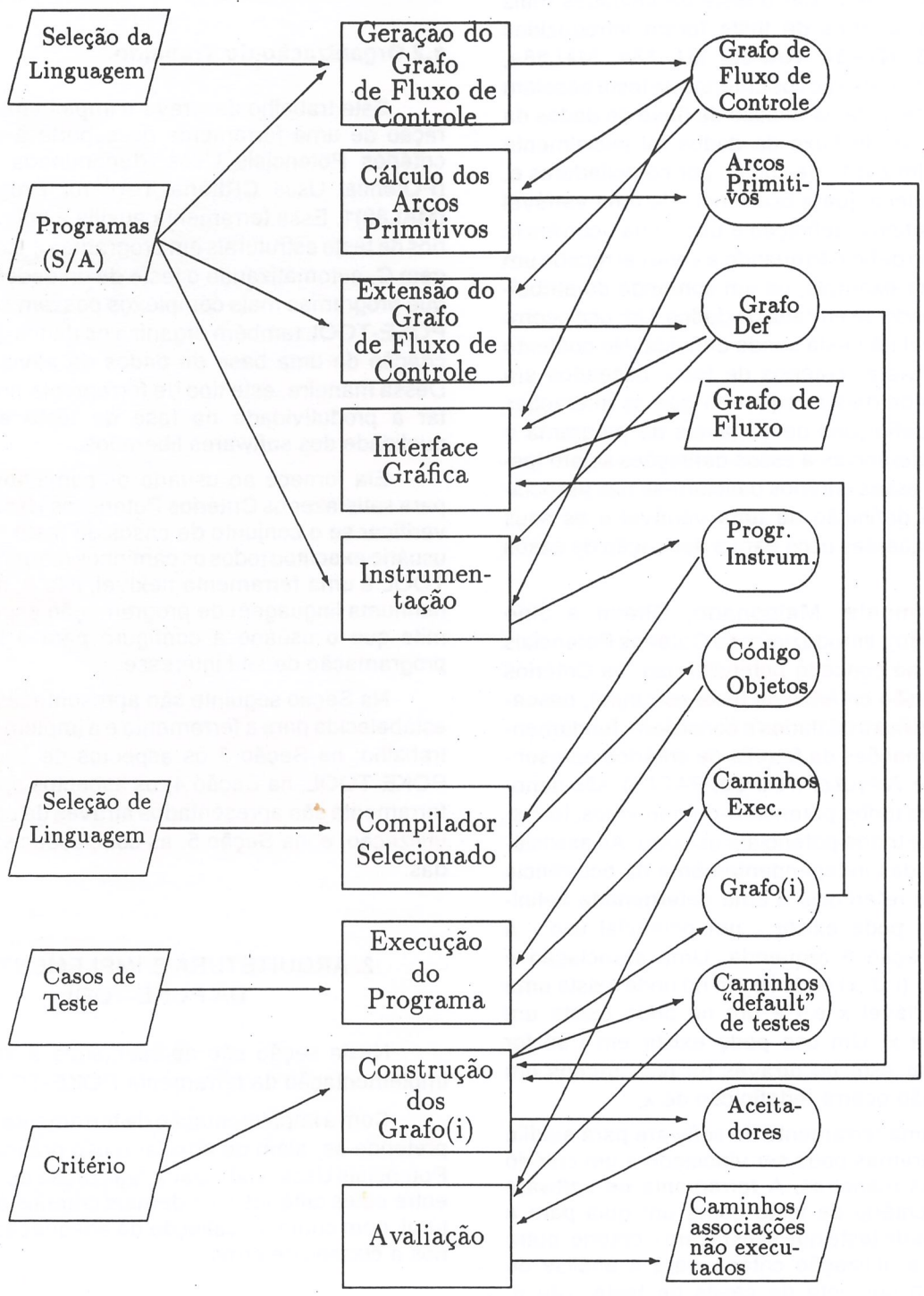

Figura 1 - Arquitetura funcional da Ferramenta de Teste Estrutural POKE-TOOL.

Inicialmente, foi implementada uma versão que apóia o teste de programas escritos na linguagem C e implementa todas as funções básicas que devem ser realizadas por uma ferramenta de teste estrutural de programas (DEU82).

Semelhantemente à ferramenta de teste implementada por Frankl e Weyuker - ASSET (FRA85) -, a POKE-TOOL tem como entrada o programa a ser testado, a seleção da critério a ser utilizado e um conjunto de casos de teste. Na hipótese do conjunto de casos de teste fornecido ser vazio, a POKE-TOOL fornecerá um conjunto de caminhos necessários para satisfazer o critério selecionado, orientando dessa forma a própria seleção de casos de teste. Se o conjunto de casos de teste fornecido não for vazio, será produzida uma relação de caminhos requeridos pelo critério mas ainda não executados.

De forma resumida, pode-se dizer que a partir do programa fonte, a POKE-TOOL determina o grafo de fluxo de controle (Geração de Grafo de Fluxo de Controle). A seguir, este grafo é estendido incorporando-se informações de fluxo de dados obtendo-se o grafo def (Extensão do Grafo de Fluxo de Controle); o conjunto de arcos primitivos é calculado, utilizando-se o algoritmo de redução de herdeiros de fluxo de dados (REHFLUXDA), obtendo-se o grafo com redução de herdeiros para fluxo de dados (Cálculo dos Arcos Primitivos) (CHU87). Esses arcos primitivos são utilizados para construir descritores (expressões regulares) dos caminhos (associações) requeridos. A Instrumentação auxilia na determinação dos caminhos efetivamente executados pelo conjunto de casos de teste fornecido. Os descritores são utilizados para verificar se o critério selecionado foi satisfeito; esta verificação dá-se pela implementação de aceitadores de expressões regulares. A partir do grafo def do conjunto de arcos primitivos constroem-se os grafo(i), caracterizando-se os arcos primitivos em cada um desses grafo(i) (Construção dos Grafo(i)). Em seguida, os descritores dos caminhos (associações) requeridos são elaborados. Na hipótese da POKE-TOOL ser utilizada na avaliação da adequação de um conjunto de casos de teste, são determinados os caminhos (associações) efetivamente executados e verifica-se se os aceitadores correspondentes aos descritores dos caminhos (associações) requeridos estão no estado final (o critério foi satisfeito); caso contrário, é fornecida ao usuário uma lista de caminhos requeridos pelo critério e não executados pelo conjunto de casos de teste (Avaliação).

Entre os caminhos requeridos pode existir um caminho não executável; a ferramenta POKE-TOOL, na sua primeira versão, não dá nenhum suporte para determinação da executabilidade de um determinado caminho - o que é uma questão indecidível; métodos heurísticos foram desenvolvidos e estão sendo incorporados à POKE-TOOL para tratar esse problema (FRA87) através de facilidades para a manipulação de caminhos não executáveis (VER92).

Observe-se que a ferramenta POKE-TOOL pode constituir-se também num suporte extremamente útil tanto às atividades de depuração como às de manutenção de programas.

2.2 Aspectos de Implementação da POKE-TOOL

A POKE-TOOL apóia a aplicação dos critérios Potenciais Usos para o teste de unidades. As unidades são entendidas aqui como procedimentos de uma linguagem procedimental.

A ferramenta pode ser configurada para testar programas escritos em linguagens procedimentais com gramáticas livres de contexto que satisfaçam algumas limitações. Basicamente, para configurar a ferramenta, é necessário que o configurador especifique o analisador léxico e o analisador sintático para a linguagem alvo. Chaim (CHA91) descreve como realizar estas tarefas; uma síntese desse processo é fornecida na Seção 3.

A POKE-TOOL é uma ferramenta interativa cuja operação é orientada para sessão de trabalho. Na sessão de trabalho, o usuário realiza todas as tarefas de teste, a saber: análise estática da unidade; preparação para o teste; submissão de casos de teste; avaliação de casos de teste; e gerenciamento dos resultados de teste. Na POKE-TOOL, a sessão de trabalho é dividida em duas fases: uma estática e outra dinâmica. Na fase estática a ferramenta analisa o código fonte, obtém as informações necessárias para a aplicação dos critérios e instrumenta o código fonte, com inserção de pontas de prova (instruções de escrita), que produzem um "trace" do caminho executado, gerando uma nova versão da unidade em teste - versão instrumentada -, que viabiliza a posterior avaliação da adequação de um dado conjunto de casos de teste. Terminada essa fase, a POKE-TOOL apresenta, sob solicitação do usuário, entre outras coisas, o conjunto de associações requeridas pelos critérios todos-nós e todos-arcos, o conjunto de caminhos requeridos pelo critério todos potenciais-du-caminhos e o conjunto de associações requeridas pelo critério todos potenciais-usos e todos potenciais-usos/du. Com estas informações o usuário pode projetar seus casos de teste a fim de que eles executem os caminhos ou associações exigidos.

A fase dinâmica consiste no processo de execução e avaliação de casos de teste. Porém, antes de executar os casos de teste, é necessário que se gere o programa executável a partir da versão instrumentada da unidade a ser testada. A POKE-TOOL, depois de execução dos casos de teste, realiza a avaliação destes de acordo com

um dos três Critérios Potenciais Usos. O resultado da avaliação é um conjunto de caminhos ou associações que restam ser executados para satisfazer os critérios e o percentual da cobertura do conjunto de casos de teste. Ainda nessa fase, a ferramenta fornece o conjunto de caminhos ou associações que foram executados, as entradas e saídas, bem como os caminhos percorridos na execução de cada um dos casos de teste. O processo de execução/avaliação deve continuar até que todos os caminhos ou associações restantes tenham sido satisfeitos (executados ou detectada a sua não executabilidade). Eventualmente, o usuário pode querer interromper a sessão de trabalho sem ter atingido uma das duas situações anteriores. Neste caso, a POKE-TOOL provê meios para a interrupção da sessão, armazenamento dos dados gerados e do estado da ferramenta até o momento. Posteriormente, pode-se recuperar a sessão de trabalho e recomeçar os testes.

A POKE-TOOL é composta de vários módulos que se comunicam através de arquivos. Esses módulos implementam funções ou parte de funções descritas anteriormente na arquitetura funcional da ferramenta. Na Figura 2 é apresentado um diagrama contendo os módulos e os diversos produtos gerados. Nessa figura, os retângulos representam os módulos, os losangos representam as entradas fornecidas pelos usuários à ferramenta e os círculos os produtos gerados; as linhas tracejadas representam o fluxo de controle e as linhas cheias o fluxo de informação.

O módulo **poketool** é o responsável pela comunicação entre a ferramenta e o usuário e pela seqüenciação das atividades de teste através da ativação dos demais módulos.

O módulo **li** é sensitivo à linguagem na qual está escrita a unidade em teste, pois realiza a tradução dessa unidade para uma versão escrita na linguagem intermediária (LI). A LI é uma linguagem cuja função é identificar os comandos que provocam desvio no fluxo de execução da unidade, ela possui dois tipos de comandos: comandos seqüênciais e comandos de desvio de fluxo. A cada comando da LI estão associados ponteiros que identificam o código fonte da unidade em teste relativo ao comando Li.

O módulo **chanomat** gera o garfo de fluxo de controle (GFC) da unidade - grafo de programa - e uma nova versão da unidade em LI onde cada comando está associado ao nó correspondente. Estes dois módulos implementam a função Grafo de Fluxo de Controle da POKE-TOOL.

O módulo **pokernel** é o responsável pelo restante da análise estática da unidade em teste, gerando as informações estáticas adicionais, necessárias ao teste dinâmico da unidade. O **pokernel** implementa as seguintes funções da POKE-TOOL: Cálculo dos Arcos Primitivos; Extensão do Grafo de Fluxo de Controle; Instrumentação, Construção dos Grafo(i) e Geração dos Descritores.

O módulo **gera executável** fornece as condições para a geração do programa executável da versão instrumentada e engloba a função Compilador Selecionado.

O módulo **executa caso de teste**, como o próprio nome diz, controla a execução dos casos de teste salvando as entradas, a saída e o caminho executado para cada caso de teste; implementa a função Execução do Programa da POKE-TOOL.

Finalmente, o módulo **avaliador** verifica quais os caminhos ou associações executados pelos casos de teste e fornece uma análise da cobertura do conjunto de casos de teste fornecido; implementa a função Avaliação da POKE-TOOL.

A atual implementação da POKE-TOOL foi desenvolvida para o sistema operacional MS-DOS e foi escrita na linguagem C, sendo que a configuração presente suporta o teste também de unidades (funções) escritas em C. Está em andamento a validação da configuração da POKE-TOOL para as linguagens COBOL e FORTRAN.

3. ASPECTOS DE CONFIGURAÇÃO DA POKE-TOOL

3.1. Introdução

A arquitetura da POKE-TOOL estabeleceu uma distinção entre as funções dependentes do código fonte das demais funções, permitindo que essas funções sejam isoladas em módulos distintos dos módulos que realizam funções independentes da linguagem. Isto permite que os módulos responsáveis por essas atividades possam ser reutilizados nas configurações da ferramenta para as diversas linguagens. Notadamente, o Cálculo dos Arcos Primitivos, a Geração dos Grafo(i) e Descritores e o mecanismo de Avaliação utilizado são completamente independentes da linguagem fonte.

De qualquer maneira, as informações dependentes da linguagem fonte são necessárias. Para obtê-las é necessário produzir um programa semelhante a um "front end" de compilador; este "front end" pode ser genérico ou específico para a linguagem fonte a ser tratada. O "front end" genérico é obtido através de algoritmos que são configurados para tratar os aspectos léxicos, sintáticos e semânticos específicos da linguagem em questão.

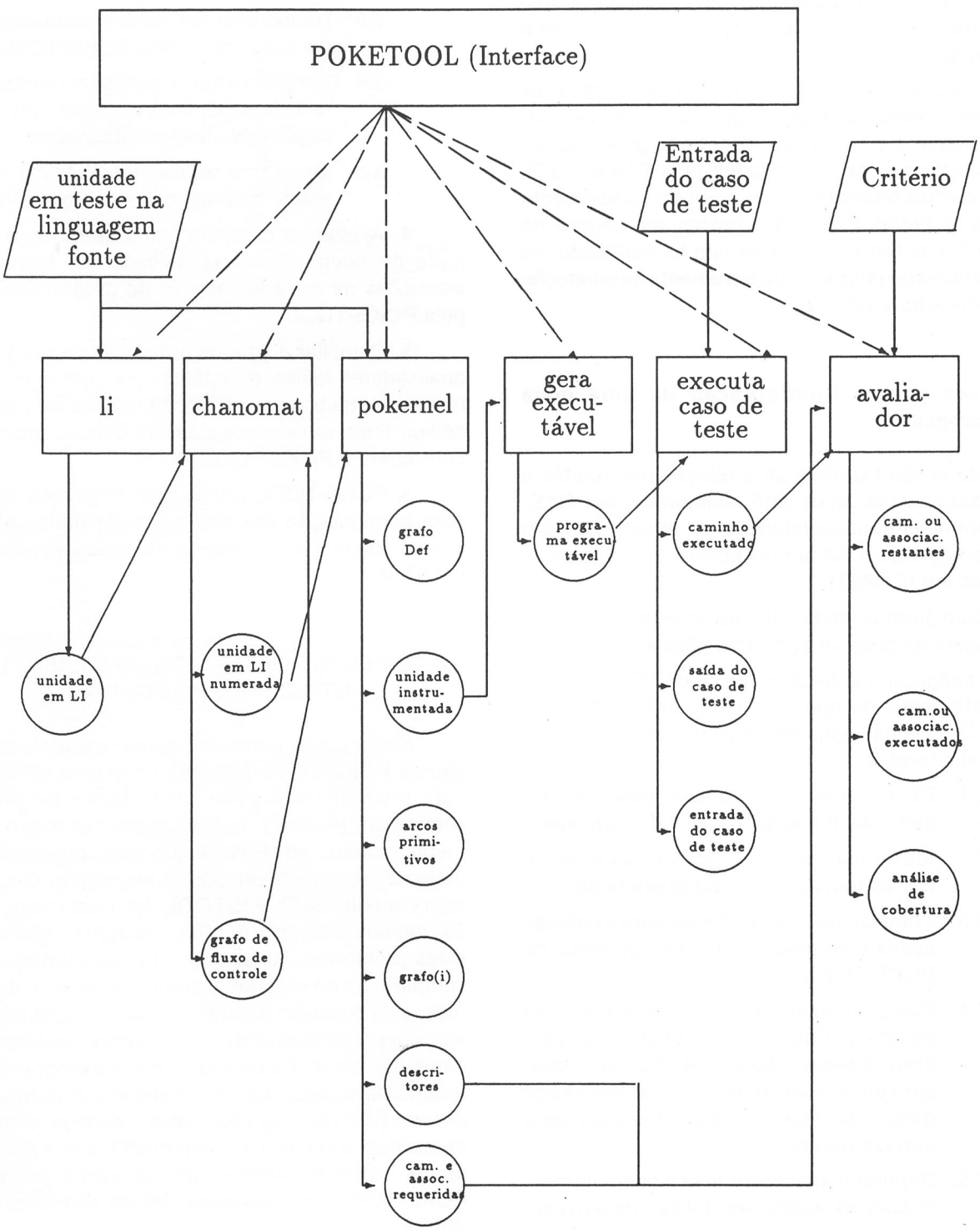

Figura 2 - Módulos da POKE-TOOL

A POKE-TOOL utiliza o enfoque genérico. A razão para essa opção foi privilegiar a reutilização de código (e esforço) em detrimento da otimização. Com o enfoque genérico, estamos reutilizando também parte do código usado nos módulos dependentes da linguagem e utilizando outras ferramentas, como descrito nos módulos **li** e **chanomat**.

Não obstante a existência de soluções otimizadas e, portanto, melhores do ponto de vista de eficiência, para o enfoque específico, os algoritmos genéricos de análise sintática e léxica utilizados pela POKE-TOOL têm uma complexidade (n log n, onde n é o comprimento da cadeia de entrada) que não constitui um gargalo no desempenho da ferramenta; este fato foi verificado, na prática, através da utilização da ferramenta na condução de um "benchmark"(MAL91).

3.2 Passos para a Configuração de uma nova Linguagem

Nesta seção fazemos uma relação das tarefas a serem realizadas por um usuário configurador da POKE-TOOL para obter uma configuração da ferramenta para uma nova linguagem. As tarefas abaixo descritas estão detalhadas em (CHA91).

1. Configurar o módulo **li** para realizar a tradução da linguagem de programação alvo para a LI.

2. Configurar o analisador léxico da POKE-TOOL. Essa tarefa demanda que o usuário conheça bem os aspectos léxicos da linguagem alvo e está dividida em algumas sub-tarefas.

   2.1. Desenvolver um autômato finito (SET81) que realize a análise léxica da linguagem.

   2.2. Identificar as ações semânticas associadas às transições do autômato finito.

   2.3. Transcrever esse autômato para a notação aceita pelo analisador léxico genérico da POKE-TOOL.

   2.4. Complementar o analisador léxico através de rotinas ("ações semânticas") que realizem a separação dos átomos da linguagem fonte, colocando-os nas estruturas de dados da POKE-TOOL utilizadas para armaze-ná-los.

   2.5. Depurar o analisador léxico conjuntamente com as ações semânticas desenvolvidas.

3. Configurar o analisador sintático da POKE-TOOL. Esta tarefa demanda que o usuário conheça bem a sintaxe e a semântica da linguagem alvo e está dividida em algumas sub-tarefas.

   3.1. Descrever a gramática da linguagem fonte na forma de grafo sintáticos (WIR76).

   3.2. Identificar os pontos onde ativar rotinas semânticas nos grafos sintáticos.

   3.3. Transcrever os grafos sintáticos para a notação aceita pela POKE-TOOL.

   3.4. Complementar o analisador sintático com rotinas semânticas que realizem a identificação das variáveis definidas.

   3.6. Depurar o analisador sintático conjuntamente com as rotinas desenvolvidas.

4. Ajustar os procedimentos que inserem código fonte na nova versão da unidade em teste com as instruções da nova linguagem de programação aceita pela POKE-TOOL.

5. Compilar e ligar os arquivos que constituem os analisadores léxico e sintático genéricos, as ações e rotinas semânticas, e os procedimentos de inserção de código fonte atualizados com os demais arquivos que constituem a POKE-TOOL.

A POKE-TOOL provê uma biblioteca de rotinas para manipulação das estruturas de dados utilizadas. Essas rotinas e as estruturas de dados são detalhadas em (CHA91).

4. ASPECTOS OPERACIONAIS E EXEMPLO DE UTILIZAÇÃO DA POKE-TOOL

Nesta seção apresenta-se um exemplo de utilização da POKE-TOOL para um programa escrito em C. Este programa faz parte do conjunto de programas contido em (KER81). Esses programas foram originalmente escritos em Pascal; um sub-conjunto deles foi traduzido manualmente para a linguagem C e testados com o auxílio da POKE-TOOL. No total foram testados 29 programas segundo os três critérios PU implementados; estes programas constituem um "benchmark" para a comparação de critérios estruturais de teste de programas, pois possuem algumas características interessantes: são programas escritos por programadores profissionais, são exemplos de boa técnica de programação e já foram utilizados para avaliar a eficácia de outros critérios de teste (WEY88, WEY90). Dessa maneira, com o intuito de verificar a eficácia dos critérios PU e sua relação com outros critérios, submeteram-se esses programas à POKE-TOOL; os resultados obtidos são discutidos em (MAL91).

O programa apresentado aqui é chamado **entab** (pag. 32, (KER81) e sua função é copiar a entrada via teclado para a saída, substituindo cadeias de espaços em branco por caracteres de tabulação de tal maneira que

visualmente a saída é igual à entrada, porém, com menor número de caracteres. Este programa encontra-se no arquivo ENTAB. C; por isso, a partir de agora, utilizaremos o nome de seu arquivo para referenciarmos este programa.

Para distinguir entre telas e mensagens do sistema, entradas do usuário e comentários, foi adotada a seguinte convenção: o que for relativo à ferramenta será descrito em "font" de máquina de escrever, o texto entrado pelo usuário é impresso em negrito e os comentários a respeito do funcionamento da ferramenta são escritos em itálico.

POKETOOL - Ferramenta de Apoio aos Critérios Potenciais Usos - Vers. 0.2

Bem-vindo!!

INICIAÇÃO

Entre com o nome do arquivo que contem a unidade a ser testada (digite "fim"para terminar a sessão):

⇒**entab.c**

Mensagens .....

** Determinando o Grafo de Fluxo de Controle**

** Ferramenta Geradora de GFC's foi bem sucedida**

** Calculando os arcos primitivos...**

** Carregando Tabela de Transição léxica...**

** Fazendo a análise sintática do código fonte... **

** Gerando descritores... **

** O Núcleo da POKE-TOOL foi bem sucedido**

O usuário deve entrar com o nome do arquivo que contém a unidade em teste. Quando o usuário digita a tecla "ENTER", a POKE-TOOL dá início à fase estática da sessão de trabalho que consiste na análise do código fonte através dos módulos **li, chanomat e pokernel**. No decorrer desta fase, a ferramenta envia uma série de mensagens que indicam o andamento do processo de análise do código fonte.

POKETOOL - Ferramenta de Apoio aos Critérios Potenciais Usos - Ver. 0.2

MENU PRINCIPAL

a. Resultados da Análise Estática
b. Resultados da Análise Dinâmica
c. Gera Programa Executável
d. Executa Caso de Teste
e. Avaliação Caso de Teste
f. Termina Sessão

Entre com a opção desejada:

⇒**a**

Mensagens .....

Terminando o processo de preparação da unidade em teste é apresentada a tela MENU PRINCIPAL, indicando que a sessão de trabalho pode prosseguir. Neste ponto a POKE-TOOL já determinou o grafo de fluxo de controle (GFC), o conjunto de arcos primitivos, o GFC estendido com as informações de fluxo de dados (grafo def), a versão instrumentada da unidade em teste, as associações e caminhos requeridos por cada critério Potencial Uso e os descritores utilizados na avaliação da cobertura desse mesmos critérios.

Selecionando a opção a no MENU PRINCIPAL, o usuário poderá visualizar os resultados obtidos da análise estática através da tela RESULTADOS DA ANÁLISE ESTÁTICA.

POKETOOL - Ferramenta de Apoio aos Critérios Potenciais Usos - Ver. 0.2

RESULTADOS DA ANÁLISE ESTÁTICA

a. Arquivo Modificado
b. Grafo Def
c. N'os Requeridos pelo Critério Todos-N'os
d. Arcos Primitivos Requeridos pelo Critério Todos-Arcos
e. Potenciais Du-caminhos Requeridos
f. Associações Requeridas
g. Retorna para o Menu Principal

Entre com a opção desejada:

⇒**d**

Mensagens .....

O usuário, ao selecionar alguma das opções de menu acima, provoca uma chamada de sistema para o utilitário "more"; é apresentado na tela o arquivo associado à opção selecionada. Em seguida a tela RESULTADOS DA ANÁLISE ESTÁTICA é apresentada novamente.

As opções a, b, c, d, e e f do menu apresentam os resultados da análise estática da unidade. Na tela acima é selecionada a opção d; é então mostrado o arquivo ARCPRIM.TES que contém os arcos primitivos requeridos pelo critério todos-arcos.

ARCOS PRIMITIVOS DO MODULO ESTAB. C

arco (4, 5) e' primitivo
arco (4, 6) e' primitivo
arco (7, 8) e' primitivo
arco (9, 14) e' primitivo
arco (10, 11) e' primitivo
arco (10, 12) e' primitivo
arco (14, 2) e' primitivo
arco (14, 15) e' primitivo

Antes de executar um caso de teste é necessário gerar o programa executável que contém a unidade em teste instrumentada. Este programa será executado quando forem submetidos os casos de teste. Para gerar este programa deve-se selecionar a opção c do MENU PRINCIPAL.

POKETOOL - Ferramenta de Apoio aos Critérios Potenciais Usos - Ver. 0.2

GERA PROGRAMA EXECUTÁVEL

OBSERVAÇÃO: Para gerar o programa executável para teste, você recebe o "prompt" do sistema operacional. Você deverá compilar e "linkar" o arquivo TESTEPROG. C junto com os seus outros arquivos no lugar do módulo a ser testado. O novo programa executável deverá ter o nome TESTEPROG.

Digite qualquer tecla para entrar no "shell" do sistema operacional...

Para retornar 'a POKE-TOOL digite "exit"...

A tela GERA PROGRAMA EXECUTÁVEL apresenta uma mensagem indicando como o usuário deve proceder para gerar o programa executável. Em seguida, o usuário recebe o "prompt" do sistema operacional para compilar a unidade em teste instrumentada e ligá-la com as demais unidades.

C:\USR\CHAIM\POKETOOL>make testeprog

Neste caso foi editado um arquivo "makefile" que compila e liga as unidades que compõem o programa executável.

C:\USR\CHAIM\POKETOOL>exit

Ao retornar à POKE-TOOL retorna-se à tela MENU PRINCIPAL.

Para executar um caso de teste basta selecionar a opção d e a tela EXECUTA CASO DE TESTE será apresentada.

POKETOOL - Ferramenta de Apoio aos Critérios Potenciais Usos - Ver. 0.2

EXECUTA CASO DE TESTE

Seu programa necessita de parâmetros de entrada (S/N) (N)?

⇒**n**

Seu programa tem entrada pelo teclado (S/N) (N)?

⇒**s**

Mensagens .....

** A saída está sendo direcionada para o arquivo output.tes**

Na primeira vez que se executa um caso de teste a tela EXECUTA CASO DE TESTE pergunta se o programa aceita parâmetros de entrada na linha de comandos e se o programa aceita entrada via teclado. O usuário responde essas perguntas e o primeiro caso de teste é executado. Nas próximas execuções de casos de teste essas perguntas não serão feitas.

col 1 2 34 rest

col 1 2 34 rest

Tecle qualquer tecla para retornar à POKE-TOOL...

A unidade ENTAb.C não aceita parâmetros de entrada através da linha de comandos mas, se aceitasse, estes parâmetros de entrada seriam salvos. A entrada via teclado, a saída na tela do caso de teste e o caminho percorrido pelos casos de teste são também salvos. Depois de executado um caso de teste a tela MENU PRINCIPAL é reapresentada e o usuário pode decidir entre "rodar" mais um caso de teste ou avaliar os já executados com relação a algum critério PU.

Selecionando-se a opção e, obtém-se a tela AVALIAÇÃO.

POKETOOL - Ferramenta de Apoio aos Critérios Potenciais Usos - Ver. 0.2

AVALIAÇÃO

a. Critério Todos-No's
b. Critério Todos-Arcos
c. Critério Todos-Potenciais-Usos
d. Critério Todos-Potenciais-Usos/du
e. Critério Todos-Potenciais-Du-Caminhos
f. Retorna para Menu Principal

Entre com a opção desejada:

⇒**c**

Mensagens .....

** Realizando a avaliação do caso de teste **
** Avaliação do caso de teste foi bem sucedida **
** Terminada a avaliação de um caso de teste **

O usuário tem três critérios PU, mais os critérios todos nós e todos arcos, implementados na POKE-TOOL; ele pode selecionar através do menu da tela AVALIAÇÃO um deles. O usuário, ao selecionar um dos critérios, por exemplo todos-potenciais-usos, faz com que a POKE-TOOL dê início ao processo de avaliação; durante a avaliação a ferramenta envia mensagens que indicam o andamento do processo. Terminada a avaliação, a POKE-TOOL apresenta o arquivo que contém as associações não executadas pelo conjunto de casos de teste avaliado. Depois de apresentado o arquivo, a tela AVALIAÇÃO é representa. Abaixo, apresentamos o ar-

quivo resultado obtido para a avaliação de oito casos de teste do exemplo ENTAB.C segundo o critério todos-potenciais-usos.

ASSOCIAÇÕES DO CRITÉRIO TODOS POT-USOS não executadas:

<1, (14,2), { col, tabstops }>
<1, (4,5), { col, tabstops }>
<2, (8,7), { newcol }>
<2, (7,8), { newcol }>
<4, (9,14), { newcol, tabstops }>
<4, (14,15), { newcol, tabstops }>
<4, (10,11), { newcol, tabstops }>
<5, (14,15), { col }>
<5, (2,3), { col }>
<5, (14,2), { col }>
<5, (9,14), { col }>
<5, (10,11), { col }>
<8, (14,15), { col }>
<8, (3,7), { col }>
<8, (6,3), { col }>
<8, (4,6), { col }>
<8, (4,5), { col }>
<8, (14,2), { col }>
<8, (9,14), { col }>
<8, (10,11), { col }>
<11, (10,12), { col }>
<11, (10,11), { col }>
<12, (7,8), { col }>

Cobertura Total = 71.250000
Média da Cobertura dos Grafo(i) = 70.431721

Neste ponto já é possível observar os resultados obtidos da fase dinâmica da sessão de trabalho. Para tanto, é necessário selecionar a tela RESULTADOS DA ANÁLISE DINÂMICA do MENU PRINCIPAL.

POKETOOL - Ferramenta de Apoio aos Critérios Potenciais Usos - Ver. 0.2

RESULTADOS DA ANÁLISE DINÂMICA

a. Entrada dos Casos de teste
b. Entrada do teclado
c. Saída dos Casos de Teste
d. Caminhos percorridos pelos Casos de Teste
e. No's executados para o Critério Todos-No's
f. Arcos Primitivos executados para o Critérios Todos-Arcos
g. Associações executadas para o Critério Todos-Potenciais-Usos
h. Associações executadas para o Critério Todos-Potenciais-Usos/du
j. Retorna para o Menu Principal

Entre com a opção desejada:

⇒**b**

Entre com o número do caso de teste (último caso de teste número 8):

⇒**7**

Mensagens .....

As opções **a, b, c** e **d** apresentam, respectivamente, os pârametros de entrada, entradas via teclado, as saídas na tela e os caminhos percorridos pelos casos de teste. Porém, essas opções estão relacionadas com vários arquivos porque existe um arquivo para cada caso de teste. Quando o usuário seleciona, por exemplo, a entrada do teclado, a POKE-TOOL pergunta a qual caso de teste o usuário está se referindo; se ele selecionar o sétimo caso de teste, a POKE-TOOL irá apresentar o arquivo com a entrada do teclado relativo a este caso de teste. Fato análogo ocorre com as outras opções acima e os conjuntos de arquivos associados.

col 1 2 34 rest

As opções **e, f, g, h** e **i** apresentam os nós, arcos, associações e caminhos efetivamente exercitados pelos casos de teste avaliados.

Para terminar a sessão de trabalho deve-se retornar ao MENU PRINCIPAL. Selecionando-se a opção **f** é iniciado o processo de término da sessão. A POKE-TOOL pergunta ao usuário se ele deseja que os arquivos gerados pela ferramenta sejam salvos em um sub-diretório com o nome da unidade; em caso afirmativo, a POKE-TOOL cria, se necessário, este sub-diretório e salva os arquivos nele; em caso contrário, a POKE-TOOL termina a execução sem salvar as informações geradas.

5. CONCLUSÕES

A POKE-TOOL analisa o código fonte obtendo o grafo de fluxo de controle e determinando os caminhos e associações requeridos pelos critérios PU; esses caminhos e associações são listados em relatórios que podem ajudar o usuário a projetar seus casos de teste; a POKE-TOOL gera também uma nova versão - versão instrumentada - da unidade em teste contendo instruções de escrita que fornecem o caminho percorrido pelo caso de teste; os casos de teste são avaliados e a ferramenta fornece relatórios que indicam quais os caminhos (associações) que não foram ainda satisfeitos para o critério selecionado, bem como os caminhos (associações) que foram satisfeitos para esse critério; ainda, a POKE-TOOL fornece uma série de informações (entradas, saídas e caminhos executados pelos casos de teste) que retratam como foi realizada e organizada a atividade de

teste. Como resultado da análise de cobertura, são apresentadas duas métricas que indicam a cobertura dos requisitos de teste (exigidos pelo critério selcionado) obtida para o conjunto de casos de teste.

A POKE-TOOL tem uma característica que a distingue das demais ferramentas de teste estrutural de software: ela é uma ferramenta configurável, ou seja, é possível obter instanciações da ferramenta para que ela suporte outras linguagens de programação.

A atual configuração da POKE-TOOL suporta o teste de programas escritos na linguagem C. Esta configuração foi validada através através da realização de um "benchmark"dos critérios PU utilizando um conjunto de programas da literatura (KER81). Este "benchmark" constituiu-se no teste de 29 programas segundo os três critérios PU implementados na POKE-TOOL. Os resultados obtidos desse "benchmark" (MAL91) são promissores e indicam que os critérios são factíveis de serem utilizados em ambiente industrial. mais ainda, a utilização da POKE-TOOL nesse "benchmark" mostrou que a ferramenta implementada é uma versão inicial de um produto e não apenas um protótipo.

A atual interface implementada na POKE-TOOL é simples e necessita o acréscimo de recursos gráficos que permitam apresentar ao usuário o grafo de fluxo de controle da unidade em teste e tornem a ferramenta mais amigável. A simplicidade da interface se deve à ênfase dada nesse trabalho à implementação e validação do cerne operacional da ferramenta; esses dois trabalhos estão em andamento e essas facilidades serão incorporadas na próxima versão da POKE-TOOL.

A POKE-TOOL é uma ferramenta que foi projetada para ser configurável para várias linguagens. A versão apresentada aqui está configurada para a linguagem C. Atualmente já existem outras configurações da POKE-TOOL para as linguagens COBOL e FORTRAN.

REFERÊNCIAS

(CHA91) CHAIM, M. L., MALDONADO, J. C. & JINO, M. Manual de Configuração da POKE-TOOL Campinas (SP), DCA/RT/008/91 - FEE/UNICAMP, 1991 (Relatório Técnico Interno).

(CHU87) CHUSHO, T "Test data selction and quality estimation based on the concept of essential branches for path testing" IEEE Trans. Software Eng., 13(5):509-517, May, 1983.

(DEU82) DEUTSCH, M. S. Software verification and validations, Englewood Cliffs, Prentice-Hall, 1982.

(FRA85) FRANKL, P. G. & WEYUKER, E. J. "Data flow testing tool" In: Proc. IEEE Softfair, San Francisco, CA, Dec. 1985 p. 46-53.

(FRA87) FRANKL, P. G. The use of data flow information for the selection and evaluation of software test data. Ph D. Thesis, New York University, 1987.

(HER76) HERMAN, P. M. "Data Flow Approach to Program Testing" Australian Computer Journal, 8(3), Nov., 1976.

(KER81) KERNIGHAN, B. W. & PLAUGER, P. Software Tools in Pascal. Addison-Wesley Publishing Company, Reading, Massachusetts, 1981.

(LAS83) LASKI, J. W. & KOREL B. "A data flow oriented program testing strategy" IEEE Trans. Software Eng., 9(3):347-354, May, 1983.

(MAL88a) MALDONADO, J. C. CHAIM, M. L. JINO, M. "Seleção de casos de testes baseada em fluxo de dados através dos critérios potenciais uso" In: Proc. Simp. Eng. Software, Canela, R. S., Out. 1988.

(MAL88b) MALDONADO, J. C. CHAIM, M. L. JINO, M. "Resultados do estudo de uma família de critérios de teste de programas baseado em fluxo de dados," Campinas (SP), DCA/RT/001/88 - FEE/UNICAMP, 1988 (Relatório Técnico Interno).

(MAL89) MALDONADO, J. C. CHAIM, M. L. JINO, M. "Arquitetura de uma Ferramenta de Software de Apoio aos Critérios Potenciais Usos" In: Proc. Cong. Nac. de Informática, 22, São Paulo, S. P., Set. 1989.

(MAL91) MALDONADO, J. C. Critérios Potenciais Usos: Uma Contribuição ao Teste Estrutural de Software. Dissertação de Doutorado, Faculdade de Engenharia Elétrica UNICAMP, 1991.

(NTA84) NTAFOS, S. C. "On required element testing" IEEE Trans. Software Eng., 10(6):795-803, Nov., 1984.

(RAP82) RAPPS, S. & WEYUKER, E. J. "Data flow analysis techniques for test data selection" In: Proc. Int. Conf. Software Eng., Tokio, Japão, Sept., 1982 p.272-278.

(RAP85) RAPPS, S. & WEYUKER, E. J. "Selecting software test data using data flow information" IEEE Trans. Software Eng., 11(4):367-375, April, 1985.

(SET81) SETZER V. W. & de MELO I. S. H. A Construção de um Compilador terceira edição, Editora Campus, Rio de Janeiro, 1981.

(VER92) VERGILIO, R. S. Determinação de Caminhos Não-Executáveis na Ferramenta POKE-TOOL. Dissertação de Mestrado, Faculdade de Engenharia Elétrica UNICAMP, 1992.

(WEY88) WEYUKER E. J. "An Empirical Study of the Complexity of Data Flow Testing" In: Proc. Second Workshop on Software Testing, Verification, and Analysis, Banff, Canada, Jul. 1988, p.188-195

(WEY90) WEYUKER E. J. "The Cost of Data Flow Testing: An Empirical Study' IEEE Trans. Software Eng., 11(4):121-128, Fev., 1990.

(WIR76) WIRTH N. Algorithms + Data Structures = Programs. Englewood Cliffs, NJ, Prentice-Hall, 1976.

(URA88) URAL, H. & YANG, B. "A structural testing criterion" Information Processing Letters, 28(3):157-163, Jul., 1988.