**UNIVERSIDADE METODISTA DE PIRACICABA**

**FACULDADE DE CIÊNCIAS EXATAS E DA NATUREZA**

**MESTRADO EM CIÊNCIA DA COMPUTAÇÃO**

# DISPONIBILIZAÇÃO DA FERRAMENTA DE TESTE POKE-TOOL COMO APLICAÇÃO WEB

**LUIZ EDUARDO GUALTI**

**ORIENTADOR:** PROF. DR. Plínio Roberto Souza Vilela

**PIRACICABA, SP**
**2006**

**UNIVERSIDADE METODISTA DE PIRACICABA**

**FACULDADE DE CIÊNCIAS EXATAS E DA NATUREZA**

**MESTRADO EM CIÊNCIA DA COMPUTAÇÃO**

# DISPONIBILIZAÇÃO DA FERRAMENTA DE TESTE POKE-TOOL COMO APLICAÇÃO WEB

**LUIZ EDUARDO GUALTI**

**ORIENTADOR:** PROF. DR. Plínio Roberto Souza Vilela

Dissertação apresentada ao Mestrado em Ciência da Computação, da Faculdade de Ciências Exatas e da Natureza, da Universidade Metodista de Piracicaba – UNIMEP, como requisito para obtenção do Título de Mestre em Ciência da Computação.

**PIRACICABA, SP**
**2006**

Gualti, Luiz Eduardo
G945d Disponibilização da ferramenta de teste POKE-TOOL como aplicação WEB / Luiz Eduardo Gualti. – Piracicaba: Universidade Metodista de Piracicaba, 2006.
82 p.

Dissertação (Mestrado em Ciência da Computação). UNIMEP-SP.
Orientador: Prof. Dr. Plínio Roberto Souza Vilela.
Inclui bibliografia.

1. Engenharia de software. 2. Teste de software.
3. Aplicações Web. I. Título.

CDD – 001.6425

# DISPONIBILIZAÇÃO DA FERRAMENTA DE TESTE POKE-TOOL COMO APLICAÇÃO WEB

AUTOR: LUIZ EDUARDO GUALTI

Orientador: Prof. Dr. Plínio Roberto Souza Vilela

Dissertação de Mestrado defendida e aprovada em 28 de abril de 2006 pela Banca Examinadora constituída dos Professores:

_____

Prof. Dr. Plínio Roberto Souza Vilela

UNIMEP

_____

Prof. Dr. Marcos Lordello Chaim

Universidade de São Paulo

_____

Prof. Dr. Luís Augusto Consularo

UNIMEP

*À*

*Minha esposa Mariana e nossas filhas Stella e Clara*

*Aos*

*Meus pais Luiz e Gracinda*

# AGRADECIMENTOS

Ao professor Dr. Plínio Roberto Souza Vilela pela instrução, orientação, compreensão e incentivo dispensados no desenvolvimento deste trabalho.

Aos Salesianos do Liceu Coração de Jesus, mantenedor do Centro Unisal, por terem custeado parte das despesas do curso.

A minha família, pelo carinho, apoio e sobretudo compreensão, durante meu envolvimento com o curso.

Aos colegas que de forma direta ou indireta, colaboraram no desenvolvimento deste trabalho.

“In God we Trust, Everything else we Test”

www.softrel.org

# RESUMO

As duas últimas décadas testemunharam um avanço tecnológico expressivo, principalmente no que diz respeito ao desenvolvimento de protocolos e aplicativos de software para Internet. É fato que uma série de softwares acadêmicos desenvolvidos nessas décadas utilizam-se de tecnologias específicas, normalmente planejados para uso *stand-alone*, muitas vezes necessitando de instalação e configuração do ambiente para funcionarem.

A configuração do ambiente necessário para a operação de um software específico pode trazer dificuldades para uma organização. Em geral, precisa-se re-estabelecer todo um ambiente de execução apropriado para a ferramenta em questão. No caso específico de ferramentas de teste essa situação é ainda mais limitante, a ferramenta a ser utilizada precisa muitas vezes operar no mesmo ambiente utilizado para o desenvolvimento.

Um exemplo de ferramenta de teste é a POKE-TOOL. Para ser utilizada necessita ser instalada e configurada no computador do usuário, além de requerer conhecimentos de administração do sistema operacional e da própria ferramenta.

Com a eficiente disponibilização da ferramenta POKE-TOOL como uma aplicação Web, o usuário necessitará apenas de acesso HTTP à Internet. Portanto não serão necessários: acesso FTP ou SSH, efetuar qualquer instalação, direitos de administração no sistema operacional, profundos conhecimentos do sistema operacional ou da própria ferramenta.

**PALAVRAS-CHAVE:** Engenharia de Software, Teste de Software e Aplicações Web.

## *ABSTRACT*

The last few decades have witnessed a huge technological advance, especially on the development of new protocols and software applications for the Internet. In the other hand a number of applications in the academic world are tools that require special software environment setup to work properly and were in most cases developed to work in stand-alone mode.

The configuration of the required environment for the operation of a specific software can bring difficulties for an organization. It is, in general, necessary to re-create an appropriate execution environment for the tool. In the specific case of testing tools, this situation is even more limited, the selected testing tool has to operate in the same software environment used for the development.

An example of testing tool is POKE-TOOL. To be used it needs to be installed and configured in the user's computer, besides requiring administrator privileges on the operational system. With the availability of POKE-TOOL as a Web application, the user will need only HTTP access to the Internet. Therefore they will not need: FTP or SSH access, to make any installation, administrator privileges, deep knowledge of the operational system or even a detailed knowledge of the testing tool itself.

**KEYWORDS:** Software Engineering, Software Testing and Web applications.

# SUMÁRIO

# LISTA DE FIGURAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| ANSI C | também chamada de “C ANSI”, refere-se à linguagem de programação estruturada “C”, padronizada pelo *American National Standard Institute* (Instituto Nacional Americano de Padronização). |
| CGI | *Common Gateway Interface* (Interface de Ligação comum), tecnologia que permite serem geradas páginas WEB dinâmicas, com passagem de parâmetros através de um navegador, para serem interpretados por um servidor. |
| FTP | *File Transfer Protocol* (Protocolo de Transferência de arquivos). |
| GCC | *The GNU Compiler Collection* (conjunto de compiladores GNU), chamado originalmente de GNU C Compiler (Compilador C GNU) pois só compilava programas em linguagem C, foi complementado para compilar C++, Fortran, Ada, Java, Objective-C, dentre outras. Apesar de ser o compilador padrão para sistemas operacionais Unix e Linux, possui versões para outros sistemas operacionais. |
| GNU | *GNU is Not Unix* (GNU não é Unix), surgiu como um projeto para criação de um sistema operacional compatível com Unix, mas totalmente livre (qualquer pessoa teria o direito de usá-lo e distribuí-lo sem ter que pagar licenças de uso). |
| HTML | *HyperText Markup Language* (linguagem de marcação de hipertexto). |
| HTTP | *HyperText Transmission Protocol* (protocolo de transmissão de hipertexto). |
| IEEE | *Institute of Electrical and Electronics Engineers* (Instituto de Engenheiros Eletricistas e Eletrônicos), uma organização profissional sem fins lucrativos, sendo seus sócios engenheiros |

| | |
|---|---|
| | eletricistas, engenheiros da computação, cientistas da computação, profissionais de telecomunicações, etc. Um de seus papéis mais importantes é o estabelecimento de padrões computacionais e de dispositivos. |
| LINUX | contração de Linus e Unix, em homenagem a Linus Torvalds, criador do núcleo do Sistema Operacional GNU. Também chamado de GNU/Linux, é um livre e popular sistema operacional da família Unix que implementa o padrão POSIX. |
| PHP | *Hypertext Preprocessor* (pré-processador de hipertexto), chamado originalmente de *Personal Webpage Tools* (ferramentas para páginas pessoais) é muito utilizada para gerar conteúdo dinâmico na WEB. |
| POKE-TOOL | *Potencial Uses Criteria Tool for program testing*, uma ferramenta de teste de software que suporta os critérios: todos os nos, todos os ramos, todos os usos, todos os potenciais usos e todos potenciais usos/DU. |
| POSIX | *Portable Operating System Interface* (Interface portátil de sistema Operacional e o X representa o nome Unix) é nome de uma família de normas relacionadas, definidas pelo IEEE. O objetivo é normalizar a interface de programação de aplicações para a família de sistemas operacionais UNIX. |
| PSP | *Personal Software Process* (Processo pessoal de desenvolvimento de software). |
| RAD | *Rapid Application Development* (Desenvolvimento Rápido de Aplicações). |
| SQL | *Structured Query Language* (Linguagem Estruturada de Consulta). |
| SSH | *Secure Shell*, é um programa e protocolo que permite a comunicação segura entre computadores. |

UNIX — é um sistema operacional portátil, multitarefa e multiusuário. Atualmente, UNIX é o nome dado a uma grande família de Sistemas Operacionais que partilham muitos dos conceitos dos Sistemas UNIX originais. Alguns exemplos destes Sistemas Operacionais são: FreeBSD, OpenBSD e NetBSD, Solaris, IRIX, AIX, HP-UX, Tru64, Linux e Mac OS X.

URI — *Universal Resource Identifier (Identificador Universal de Recursos*), cada recurso disponível na Web — documento HTML, imagem, video, programa, etc. — tem um endereço que pode ser codificado por um URI

URL — *Universal Resource Locator* (Localizador Uniforme de Recursos) é o endereço de um recurso (um arquivo, uma impressora etc.), disponível em uma rede (inclui a Internet e a intranet).

WEB — O mesmo que WWW.

WWW — *World Wide Web* (teia do tamanho do mundo) é uma rede de computadores na Internet que fornece informações em forma de hipertexto, acessíveis através de softwares navegadores, usando o protocolo de comunicação HTML.

# CAPÍTULO 1

## INTRODUÇÃO

### 1.1. CONTEXTO

A sociedade, a indústria e o comércio usam e dependem, claramente e cada vez mais, dos sistemas computacionais. Sob o ponto de vista técnico esses sistemas são compostos de hardware e software. Com os avanços tecnológicos na produção do hardware foi possível alcançar alta confiabilidade, desempenho e armazenamento a custos acessíveis. Nesse contexto, atualmente, o foco principal de preocupação com qualidade e eficiência de desenvolvimento passou naturalmente a ser o software.

A garantia da qualidade do software depende de uma série de procedimentos e técnicas propostas por vários autores. Um dos processos imprescindíveis na garantia de qualidade do software é o teste de software. O teste é definido como o processo de identificar defeitos no software, permitindo que esses defeitos sejam encontrados e eventualmente removidos.

A aplicação apropriada do processo de teste depende da definição de critérios de teste. Critérios de teste definem parâmetros para se decidir quando a atividade de teste deve ser encerrada, ou mesmo decidir se um conjunto de casos de teste foi ou não adequado ao teste de um determinado programa.

Os critérios de teste se valem de informações sobre o programa em teste para determinar os requisitos de teste. Essas informações podem vir diretamente da especificação do programa, o que caracteriza os critérios funcionais, ou da estrutura do programa – código fonte – caracterizando os critérios estruturais.

Existem vários critérios estruturais, alguns baseados na análise do fluxo de controle do programa, outros baseados na análise do fluxo de dados. Esse tipo de análise é, de modo geral, extremamente trabalhosa e demorada para se

fazer manualmente, dependendo de ferramentas apropriadas para a sua aplicação.

Uma das ferramentas de teste estrutural de software que suporta a aplicação de critérios baseados em fluxo de controle e fluxo de dados é a POKE-TOOL. Essa ferramenta é o foco principal deste trabalho de pesquisa.

## 1.2. MOTIVAÇÃO

Uma versão da POKE-TOOL, suportando a realização de teste de programas escritos em linguagem ANSI C em sistemas operacionais Linux encontra-se disponível para *download*, após, é preciso instalá-la, o que requer conhecimentos de administração do sistema operacional Linux.

Para utilizá-la é necessário conhecer seu funcionamento e dependendo do programa a ser testado, também podem ser necessários direitos de administração do sistema operacional.

Estas necessidades específicas e sobretudo o tempo envolvido com toda esta preparação para uso, dificultam - quando não impedem - sua utilização e, conseqüentemente, não estimulam seu aprimoramento e desenvolvimento de suporte a testes em outras linguagens procedimentais.

## 1.3. OBJETIVOS

Especificar e implementar uma aplicação Web cujo propósito é disponibilizar os recursos da ferramenta POKE-TOOL na Internet, ampliando e viabilizando sua utilização a qualquer pessoa. Como extensão desses objetivos, a aplicação Web também acompanhará e direcionará os usuários durante sua utilização, evitando as confusões e resultados errados que podem ocorrer em decorrência da má utilização da ferramenta POKE-TOOL, por exemplo, com a tentativa de avaliação dos resultados de testes sem que estes tenham sido feitos, a realização de testes em segmentos que não tenham sido instrumentados, confusões entre versões do programa e seus casos de testes, dentre outras.

## 1.4. ORGANIZAÇÃO DO TRABALHO

No Capítulo 2 são apresentados os conceitos utilizados nesse trabalho: uma introdução ao teste de software e os principais critérios de teste, a ferramenta POKE-TOOL, e uma introdução à Web, linguagens e protocolos utilizados.

O Capítulo 3 apresenta a disponibilização da ferramenta de teste POKE-TOOL como uma aplicação Web, cuja implementação originou a Webpoketool. Também são apresentadas equivalências entre elas e considerações sobre a compatibilidade delas com outras linguagens de programação, outras plataformas operacionais além das utilizadas como padrão pela ferramenta e considerações sobre o ambiente de teste.

No Capítulo 4 – são apresentados os resultados obtidos, são discutidos os problemas encontrados e superados. Também são apresentadas sugestões para novos trabalhos relacionados.

No Apêndice A encontra-se a o código fonte do programa Cal.c, usado na sessão de teste apresentada no Capítulo 3.

O Apêndice B corresponde ao complemento feito nos *scripts* utilizados pela POKE-TOOL.

# CAPÍTULO 2

## REVISÃO BIBLIOGRÁFICA

### 2.1. TESTE DE SOFTWARE

Para MYERS (1979), testar é um processo de execução de um programa com o intuito de encontrar erros e um bom caso de teste é aquele que tem alta probabilidade de encontrar um erro ainda não descoberto.

"Um produto de software é um programa de computador combinado com os itens que o tornam inteligível, utilizável e extensível". (PETERS, 2001, p.4).

> "No processo de desenvolvimento de software todos os erros são erros humanos e, apesar do uso dos melhores métodos de desenvolvimento, das melhores ferramentas de suporte e de pessoal treinado, os erros permanecem presentes nos diversos produtos de software produzidos e lançados no mercado, o que faz com que as atividades de revisão e teste tenham um papel fundamental no processo de desenvolvimento de software" (ROCHA, 2001, p.73)

Segundo PAULA FILHO (2003) o objetivo central de toda a metodologia de teste é maximizar sua cobertura, ou seja, a quantidade potencial de defeitos que podem ser detectados por meio do teste.

Para FELIZARDO (2006), a definição de VV&T - Validação, Verificação e Teste, abrange muitas das atividades relacionadas com a Garantia da Qualidade de Software (GQS). Para MALDONADO (1991), validação e verificação são duas atividades centrais de GQS. A validação é uma atividade

que tem como objetivo garantir que o produto correto está sendo desenvolvido e a maior parte de esforço de validação tem sido realizada no final do período de desenvolvimento, quando o produto é testado e confirmada sua validação com os requisitos pré-estabelecidos. Já a verificação é uma atividade que tem como objetivo assegurar, durante o ciclo de vida do software, que o produto esteja sendo construído corretamente.

Ainda segundo MYERS (1979), o teste é tido como bem sucedido se encontrar um erro anda não descoberto. O teste pode indicar a presença de erros mas não garante a inexistência deles.

MOLINARI (2003) complementa que o conceito de verificação inclui dois critérios fundamentais:

a) o software tem de executar todas as funções desejadas;

b) o software, durante sua execução, deve passar por caminhos que tenham sido testados em alguma combinação com outras funções.

O teste de software está presente em praticamente todos os modelos de desenvolvimento de software: queda d'água, dente de serra, dente de tubarão, espiral, v, w, e borboleta - segundo MOLINARI (2003), nos modelos: estrela, Shneiderman; segundo ROCHA (2003) e, segundo PRESSMAN (2002), também presente nos modelos: prototipagem, RAD, incremental e espiral ganha-ganha.

> O Teste de Software é executado em níveis diferentes durante todo o ciclo de vida do software. O teste começa com componentes individuais de software – cada componente deve ser verificado em termos funcionais e estruturais. O teste também é necessário durante a integração dos componentes de software para garantir que cada combinação deles é satisfatória e o teste do sistema e o teste de aceitação acompanham o teste de componente e o de integração (PETERS, 2001, p.383).

Para ROCHA (2001), quanto mais próximos de sua origem os erros forem detectados, menor será o custo e a dificuldade das correções.

> "Uma estratégia de teste de software deve acomodar testes de baixo nível, que são necessários para verificar se um pequeno segmento de código-fonte foi corretamente implementado, bem como testes de alto nível, que validam as principais funções do sistema, com base nos requisitos do cliente". (PRESSMAN, 2002, p. 470).

Embora durante todo o processo de desenvolvimento de software sejam utilizadas técnicas, métodos e ferramentas a fim de evitar que erros sejam introduzidos no produto, a atividade de teste continua sendo de fundamental importância para a eliminação dos erros que persistem, MALDONADO (1991).

IEEE (1990) inclui as seguintes definições: um erro (*error*) é a diferença entre uma condição ou valor computado, medido ou observado e a verdade, especificada ou o valor ou condição teoricamente correto; um defeito (*fault*) é um passo ou um processo ou a definição de dados incorreta em um programa computacional; uma falha (*failure*) é um resultado computado incorretamente, por exemplo, 12 quando o resultado correto é 10; um engano (*mistake*) é um ato humano que produz um resultado incorreto, como uma ação incorreta de um programador ou operador.

VERGILIO (1997) classifica os erros sob duas perspectivas: causa e efeito. As técnicas de teste têm o objetivo de revelar a existência de um defeito estudando a falha por ele produzido, já as técnicas de depuração têm o objetivo de estudar e eliminar a causa que o gerou.

Segundo PFLEEGER (2004), um defeito (por exemplo, no projeto do software) pode levar a outros defeitos (por exemplo, codificação incorreta, manual do usuário incorreto). Já uma falha é a divergência entre o comportamento real do sistema e o comportamento requerido. A falha pode ser descoberta durante os testes, a operação e a manutenção do software. Para PETERS (2001), o

software conterá uma falha sempre que for incapaz de desempenhar uma função específica.

Se um código com defeitos nunca for executado, ou se um estado específico nunca ocorrer, então o defeito nunca fará o código falhar. (PFLEEGER, 2004, p. 5).

No teste estrutural estamos interessados no desenvolvimento de casos de teste baseados na estrutura (lógica interna) do código em teste. Existem diversas classes de teste que dependem de quão completo e consumidor de tempo o processo de teste deva ser. São categorias básicas os testes de abrangência de instrução, abrangência de ramificação e abrangência de caminho (PETERS, 2001, p.396).

PFLEEGER (2004) define um caso de teste como sendo uma escolha específica de dados de entrada a serem utilizados para testar um programa. Define também o teste como sendo um conjunto limitado de casos de testes

Para ROCHA (2001), as técnicas de teste são classificadas de acordo com a origem das informações utilizadas para estabelecer os requisitos de teste, sendo:

a) técnica funcional ou caixa preta (segundo WHITTAKER (2003), também conhecida como *specification-based testing* e *behaviorial testing*) – estabelece os requisitos de teste com base na especificação. Para DeMILLO (1987), o comportamento e a estrutura interna do programa não são considerados;

b) técnica estrutural ou caixa branca (segundo WHITTAKER (2003), também conhecida como *code-based testing*) – estabelece os requisitos de teste com base em uma determinada implementação, verificando se atende aos detalhes do código e solicitando a execução de partes ou de componentes elementares do programa. Esta definição também é compartilhada por DeMILLO (1987), MYERS (1979) e PRESSMAN (2002);

c) técnica com base em erros – segundo DeMILLO (1980) estabelece requisitos de teste explorando os erros típicos e comuns cometidos durante o desenvolvimento de software;

d) técnica com base em máquinas de estado finito - Para MENEZES (1997), as máquinas de estado finito são definidas como modelos matemáticos que respondem a entradas e saídas discretas, assumindo um número finito de estados, onde são conservadas apenas as informações do passado relevantes a ações a serem tomadas na próxima entrada. Segundo ROCHA (2001), esta técnica é utilizada para modelar o aspecto comportamental de sistemas de software.

Segundo PFLEEGER (2004), os testes de caixa branca são também denominados testes de caixa aberta.

O grafo de fluxo de controle (ou grafo de programa) é uma notação simples para a representação do fluxo de controle e que auxilia seu entendimento. Cada círculo, chamado de nó do grafo de fluxo de controle, representa um ou mais comandos procedimentais. A Figura 2.1 apresenta a notação do grafo de fluxo de controle para construções estruturadas.

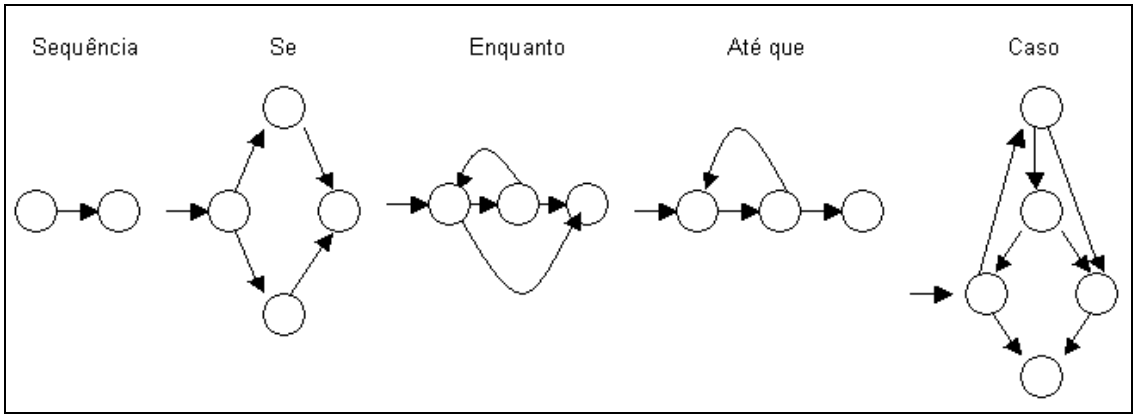



**FIGURA 2.1 – NOTAÇÃO DE GRAFO DE FLUXO DE CONTROLE (PRESSMAN, 2002, P. 439).**

Segundo CHAIM (2001), um programa pode ser mapeado para um grafo de fluxo de controle se a linguagem em que foi desenvolvido for do estilo Algol. Segundo a UNIVERSIDADE DE MICHIGAN (2006a), as linguagens deste estilo são baseadas no Algol 60: Algol 68, Pascal, Simula, PL/1, BCPC, Modula2, Euclid, Samaltalk, Concurrent Pascal, ADA, C, C++ e Java. Também segundo A UNIVERSIDADE DE MICHIGAN (2006b), esta relação pode ser complementada pelas linguagens: Modula-3, Oberon e Delphi. Como também já foram feitos estudos do uso da ferramenta POKE-TOOL, que se utiliza do grafo de fluxo de controle, com as linguagens Cobol, Clipper, Fortran e aplicações usando SQL (aplicações de banco de dados relacional), podemos garantir a possibilidade de utilização do grafo de fluxo de controle para quase todas as linguagens de programação.

Para facilitar o entendimento do grafo de fluxo de controle, vamos compará-lo a um fluxograma, que é utilizado para mostrar a estrutura de controle do programa, como pode ser visto na Figura 2.2, baseada em PRESSMAN (2002, p. 439). Nesta comparação observamos que:

a) uma seqüência de caixas de processamento e um losango de decisão podem ser mapeados em um único nó;

b) as setas do grafo de fluxo de controle, chamadas arestas ou ligações representam o fluxo de controle e são análogas às setas de fluxograma.

Uma aresta deve terminar em nó, mesmo que não represente nenhum comando procedimental. As áreas limitadas por arestas e nós são chamadas regiões. Ao contarmos as regiões, devemos incluir a área fora do grafo de fluxo de controle como uma região.

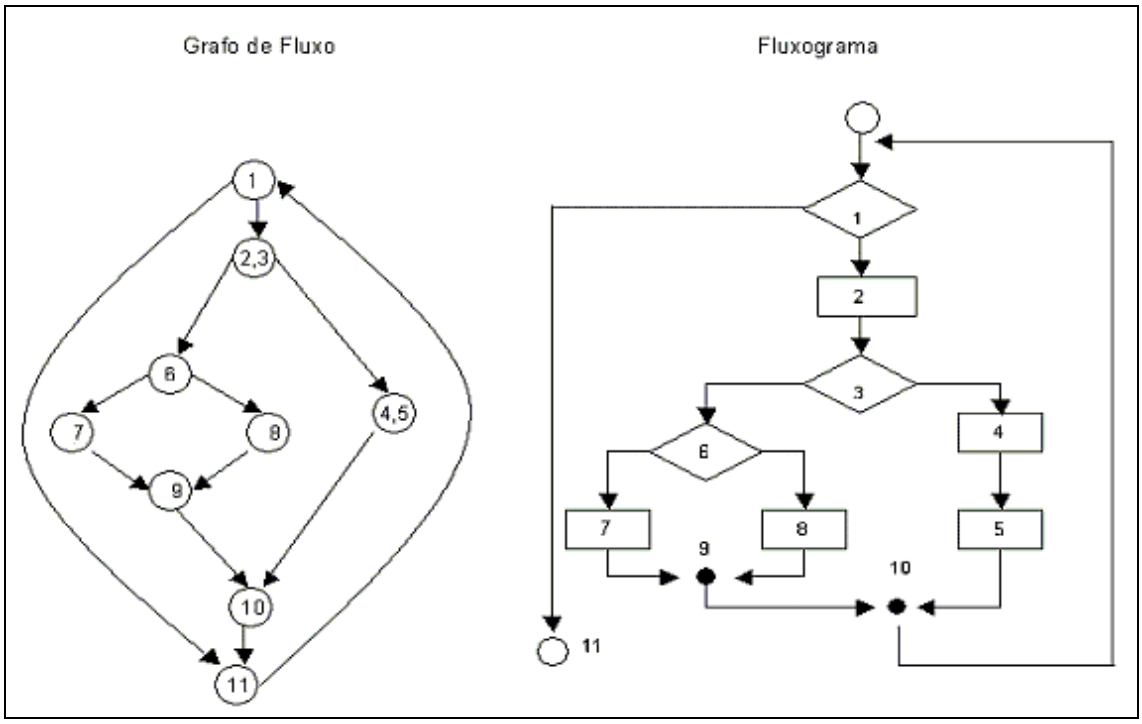



**FIGURA 2.2 – COMPARAÇÃO ENTRE O GRAFO DE FLUXO DE CONTROLE E O FLUXOGRAMA.**

Quando um ou mais operadores booleanos lógicos (ou, e, não-e, não-ou) está presente em um comando condicional, ou seja, quando existem condições compostas, a geração do grafo de fluxo de controle torna-se ligeiramente mais complicada. Neste caso, um nó separado deve ser criado para cada condição. Estes nós originados de condições compostas são chamados de nós predicados e segundo PRESSMAN (2002) se caracterizam por duas ou mais arestas saindo deles.

A técnica estrutural, em que se insere esta dissertação, é classificada por ROCHA (2001) de acordo com critérios baseados:

a) na Complexidade: os requisitos de testes são baseados na complexidade do programa. O critério mais conhecido é o da Complexidade Ciclomática, proposta por McCABE (1976).

b) no Fluxo de Controle: usam apenas características do controle da execução do programa, como comandos ou desvios para determinar

quais estruturas são necessárias para a realização dos testes. Os mais conhecidos são os critérios:

a. Todos-Nós (*all-nodes*, ou Todos Nós): exige que durante a execução do programa, seja executado, pelo menos uma vez, cada vértice do grafo de fluxo de controle (ou, em outras palavras, que cada comando do programa seja executado pelo menos uma vez).

b. Todos-Arcos (*all-edges*, ou Todos Ramos): exige que cada aresta do grafo de fluxo de controle do programa seja executada ao menos uma vez. Para FELIZARDO (2006) equivale a dizer que cada desvio do programa, seja exercitado pelo menos uma vez.

c. Todos-Caminhos (*all-paths*, ou Todos Caminhos): requer que todos os caminhos possíveis do programa sejam executados pelo menos uma vez, em geral é impraticável segundo PRESSMAN (2002).

c) no Fluxo de Dados: associa ao grafo de fluxo de controle informações sobre o fluxo de dados do programa, explorando associações entre pontos do programa em que é atribuído um valor a uma variável (chamado de definição de variável) e pontos em que esse valor é utilizado (chamado de referência ou uso da variável). Com base nessas associações são determinados os caminhos a serem testados. Neste critério incluem-se a família de Critérios Baseados em Fluxo de Dados de Rapps e Weyuker - RAPPS (1985) - e a família de Critérios Potenciais Usos de Maldonado, Jino e Chaim - MALDONADO (1988) e MALDONADO (1991).

O teste de caminho básico é uma das técnicas de estrutura de controle e foi proposta inicialmente por McCABE (1976). Para PRESSMAN (2002) esta técnica permite ao projetista de casos de testes originar uma medida da complexidade lógica de um projeto procedimental e usa esta medida como guia para definir um conjunto básico de caminhos de execução.

Para SOMMERVILLE (2003), o ponto de partida dos testes de caminho é um grafo de fluxo de controle de programa, segundo ele, um modelo em esqueleto de todos os caminhos de programa. O grafo de fluxo de controle é construído substituindo-se declarações de controle de programa por diagramas equivalentes, sendo, os nós, representantes das decisões e os ramos, o fluxo de controle.

Na Figura 2.3 observamos uma comparação entre o programa fonte "identifier.c" e seu grafo de fluxo de controle. Observe que, para referência, à esquerda das instruções do programa estão representados números que equivalem à seqüência dos nós no grafo de fluxo de controle: /*01*/, que representa o primeiro bloco no grafo de fluxo de controle, e assim subseqüentemente. Cada arco representa uma transferência de controle entre os blocos.

"Com base no grafo de fluxo de controle são identificados os componentes (nós, arcos ou caminhos) que devem ser executados para satisfazer determinado critério, caracterizando assim o Teste Estrutural". FELIZARDO (2006).

Para PETERS (2001), as principais categorias de abrangência do teste orientado por fluxo de dados são o bloco básico e as categorias definidas por RAPPS (1982) e RAPPS (1985): Todos-usos (*all-uses*, Todos usos); C-uso (*c-use*, uso em C); P-usos (*p-use*, uso em P); DU-caminho (*du-paths*, caminhos de definição e uso).

Para ilustrar essas formas de abrangência de teste, o seguinte fragmento de um código em Java, obtido de PETERS (2001) é apresentado:

---

```
sum = 0;
prod = 0;
i = 1;
while (i <= n)
        {
            sum+ = 1;
            prod* = i;
```

```
        i++
    }
```

---

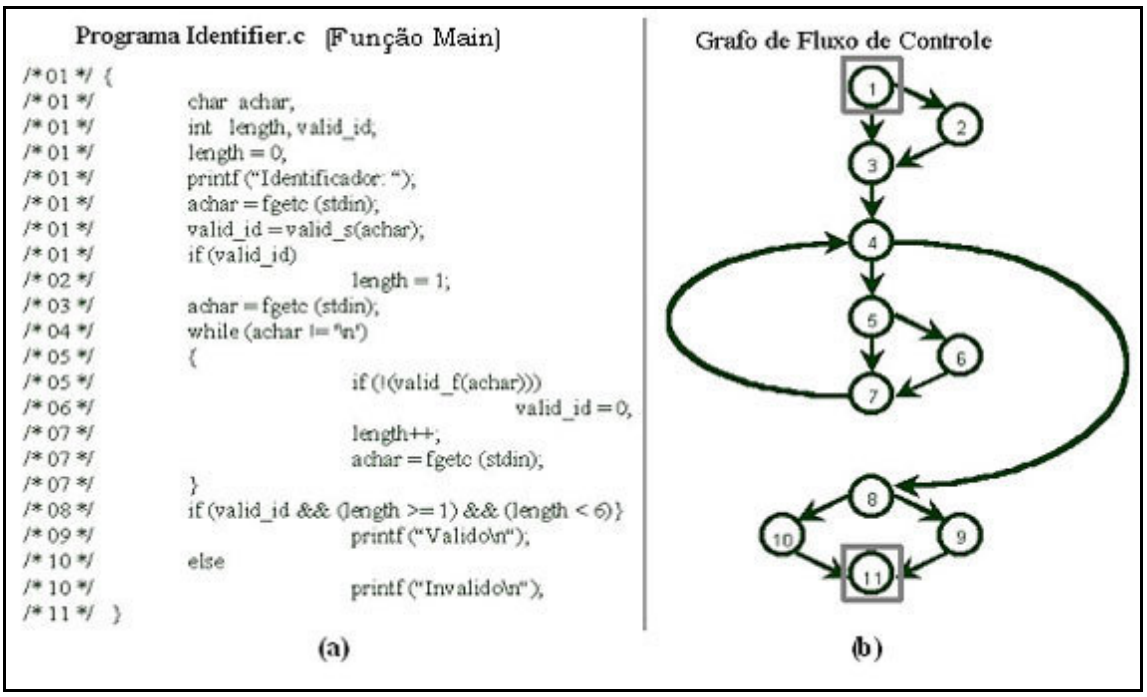



**FIGURA 2.3 - PROGRAMA IDENTIFIER.C E O GRAFO DE FLUXO DE CONTROLE (MALDONADO, 2001).**

Os *blocos* básicos constituem as partes consecutivas do código executadas ao mesmo tempo, sem qualquer ramificação. No código acima, existem dois blocos, a saber:

- primeiro bloco:

```
sum = 0;
prod = 0;
i = 1;
```

- segundo bloco:

```
sum+= 1;
prod*=1;
i++;
```

C-uso, P-uso e Todos-usos são categorias mais especializadas dos critérios de abrangência do fluxo de dados, com ênfase em um par *definição-uso* e uma distribuição desse par por todo o código:

a) uma *definição* refere-se a uma instrução em que o valor inicial é atribuído a uma variável. Por exemplo: “i = 1”; “sum = 0” e “prod = 1”;

b) o uso é a ocorrência desta variável. Por exemplo, os usos da primeira variável são: “i++” e “prod* = i”. o uso da variável soma ocorre na instrução: “sum+= 1;”.

Se o uso aparecer em uma expressão computacional, o par é um C-uso. Nele um caminho começa a partir da definição da variável e termina em uma instrução em que é envolvido no cálculo. Por exemplo: “sum = 0;” (sendo a definição) e “sum + = 1” (sendo o uso);

Se o uso ocorrer dentro de um predicado, o par resultante é um P-uso. Para WEYUKER (1990), ocorrem em comandos como *while...do*, *if...then..else* ou *repeat...until*. Nesta categoria um caminho começa a partir da definição da variável e termina em uma instrução em que aparece dentro de um determinado predicado. Por exemplo, P-uso seria “i = 1;”(sendo a definição) e “while (i <= n)” (sendo o uso).

“C-uso e P-uso são denominados Todos-usos”. PETERS (2001, p.409). Nele um caminho começa a partir da definição da variável e, através de um caminho livre de definição, ou seja, um caminho em que a variável não seja redefinida termina em uma instrução em que se torna usada. Dois exemplos desta abrangência são:

a) “sum = 0;” (definição) e “sum += 1;” (uso) e

b) “prod = 0;” (definição) e “prod* = i” (uso).

Um DU-caminho é um caminho de uma definição de variável em relação ao seu uso que não contém uma redefinição da variável. Um exemplo é uma execução da seqüência que começa em ”i=1;”, faz um loop uma vez no corpo

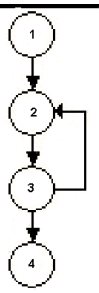
da instrução “while (i <=.n)” e, depois, continua. Os caminhos com dois ou mais *loops* para “sum+ = i” não são DU-caminhos, uma vez que a instrução “i++” redefine o valor da variável “i” .

A Figura 2.4 apresenta uma comparação entre eles em um caso prático, a função T_CalcFatorial.BotaoCalcular”.

```
(01) procedure T_CalcFatorial.BotaoCalcular(Sender: TObject);
(01) var vConta, vFat, vCalc: Integer;
(01) begin
(01)    vFat := StrToInt(txtFat.Text);
(01)    vCalc := 0;

(02)    for vConta := 1 to vFat - 1 do
(03)        vCalc := vCalc + (vFat * vConta);

(04)    showmessage('O fatorial de ' + IntToStr(vFat) + ' é ' + IntToStr(vCalc));
(04) end;
```




| | |
|---|---|
| Todos-Nós: {1,2,3,4}<br><br>Todos-Arcos: (1,2,3,2,3,4) | Todos-Caminhos: {(1,2,3,4)<br>(1,2,3,2,3,4)<br>. . .<br>(1,2,3,2,3,...,2,3,4)} |

**FIGURA 2.4 - DIFERENÇAS ENTRE OS CRITÉRIOS DE FLUXO DE CONTROLE (GONÇALVES, 2003, P. 15).**

> “O método de teste do fluxo de dados seleciona caminhos de testes de um programa de acordo com a localização das definições e do uso das variáveis do programa. Diversas estratégias de teste de fluxo de dados foram estudadas e comparadas”. (PRESSMAN, 2002, p. 447)

Os critérios baseados em fluxo de dados usam informações do fluxo de dados do programa para derivar os requisitos de teste. Adicionando-se ao grafo de fluxo de controle do programa informações sobre o fluxo de dados, obtém-se o chamado grafo Def-Uso (*def/use graph*), definido por RAPPS (1982) e RAPPS (1985). O grafo de fluxo de controle mostrado na Figura 2.5 apresenta os pontos que exibem definição e uso de variáveis. Com base nesse grafo de fluxo controle são determinados quais caminhos devem ou não ser exercitados. O uso de uma variável se dá por dois tipos de uso: computacional ou uso predicativo - o uso computacional determina o uso de uma variável de forma computacional e o uso predicativo de forma predicativa, ou seja, quando uma

variável é usada no teste do programa. O critério Todos-Usos requer que todas as associações entre uma definição de variável e seus subseqüentes usos sejam exercitadas pelos casos de teste, sendo que uma associação é estabelecida entre uma atribuição de valor a uma variável (ou seja, sua definição) e um subseqüente uso dessa variável através de um caminho livre de definição, ou seja, um caminho em que a variável não seja redefinida.

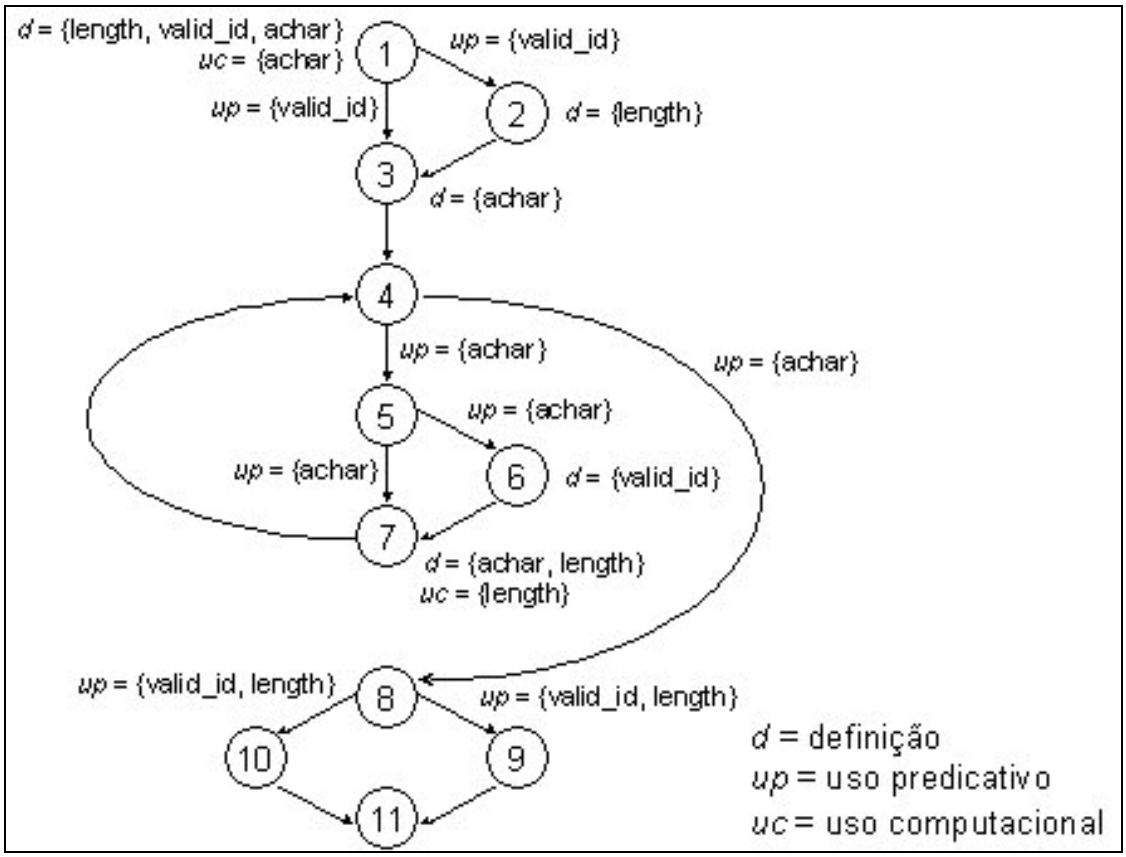



**FIGURA 2.5 - GRAFO DEF-USO DO PROGRAMA IDENTIFIER.C (MALDONADO, 2001).**

Para FELIZARDO (2006), o critério Todos-Potenciais-Usos requer a execução das Potenciais-Associações. Uma Potencial-Associação é uma associação na qual elementos são caracterizados independentemente da ocorrência explícita de uma referência (um uso) a uma determinada definição. Se um uso desta definição pode existir, ou seja, existir um caminho livre de definição até um certo nó ou arco (um potencial-uso) a potencial-associação entre a definição e o potencial-uso é caracterizada e, eventualmente requerida pelo critério Todos-Potenciais-Usos.

A ferramenta de teste POKE-TOOL apóia a utilização dos “Critérios Potenciais Usos” (PU), que 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; são denominados: critérios todos-potenciais-du-caminhos, todos-potenciais-usos e todos-potenciais-usos/du (CHAIM, 1991, p.12).

O desenho manual de grafos de fluxo de controle de programa é trabalhoso e erros normalmente ocorrem, o que leva esta atividade a ser dispendiosa, tanto em relação ao tempo quanto em relação ao seu custo. A importância da visualização automatizada dos grafos de fluxo de controle de programa, levou VILELA (1994) a desenvolver a Viewgraph, que além de gerar grafos de fluxo de controle baseada em arquivos gerados pela POKE-TOOL, pode atuar também sobre a depuração do programa.

Devido à sua arquitetura modular, que estabeleceu distinção entre as funções dependentes do código fonte das demais funções, a POKE-TOOL, além de suportar atualmente a linguagem ANSI C em Linux, provê facilidades para suporte a diversas outras linguagens de programação.

## 2.2. WEB

NIEDERST (2001) define a Web como somente uma das formas de compartilhamento das informações, um subtipo da informação na Internet.

Com a introdução da Web, principalmente por permitir que os usuários de computadores localizem e visualizem documentos baseados em multimídia sobre quase todos os assuntos, a Internet explodiu para se tornar um dos principais mecanismos de comunicação do mundo. Antes disso, a maioria dos softwares era projetada para funcionar em computadores não se comunicavam entre si - atualmente pode-se escrever aplicativos que se comunicam com centenas de milhares de computadores no mundo. DEITEL (2003) afirma que Internet funde tecnologias de computação e comunicações, torna informações acessíveis mundialmente de forma instantânea e conveniente, e facilita o contato entre comunidades de interesse comum.

O protocolo HTTP (que é um protocolo de camada de aplicação), define como deve ocorrer a comunicação na Web. Ele é implementado em dois tipos de softwares:

- no computador do usuário (também chamado cliente), onde recebe o nome de navegador (originário do inglês *browser*);
- no computador que fornece os dados requeridos pelo usuário, onde recebe o nome de servidor - são conhecidos como sevidores Web e sevidores HTTP, originados das traduções do inglês *server*, Web *server* e HTTP *server*.

Com o protocolo HTTP, a comunicação entre estes dois softwares ocorre independentemente dos sistemas operacionais que estiverem sendo utilizados em qualquer dos computadores.

O protocolo HTTP opera sobre o protocolo TCP/IP. Para permitir a localização, cada servidor Web possui um número único, seu endereço de rede IP (do inglês *IP address*). Normalmente, para facilitar sua localização do servidor, é atribuído a ele um nome, chamado de domínio (*hostname*). Na Web há um consenso de que os servidores possuam seu nome iniciado por www, mas isto não significa que todos os endereços Web iniciem com este prefixo.

A associação entre endereço IP e o nome do servidor é feito por um sistema de de gerenciamento de nomes, que traduz o nome do servidor em seu endereço de rede. Este procedimento ocorre transparentemente ao usuário.

Todas as páginas Web podem ser encontradas através de seus endereços, conhecidos como URLs, que definem suas localizações, que são exclusivas. Tomando-se como exemplo: http://www.unimep.br/lts/index.html, temos:

- “http://” que é a definição do protocolo a ser utilizado;
- “www.unimep.br” o domínio;
- “/lts/” o endereço; e

- “index.htm” a página desejada.

A HTML é uma linguagem de marcação utilizada para produzir páginas na Internet que são disponibilizadas em servidores. Esses códigos podem ser interpretados pelos navegadores para exibir as páginas da Web.

Resumidamente, quando um usuário solicita uma página Web, digitando seu endereço ou clicando sobre um hiperlink, o navegador envia para o servidor mensagens de requisição HTTP para os objetos da página. O servidor recebe as requisições e responde com uma mensagem de resposta HTTP que contém os objetos, que são apresentados no navegador.

Diferentemente das outras linguagens estruturadas, a HTML utiliza marcações que são conhecidas como etiquetas (do inglês tags), que consistem em breves instruções tendo uma marca de início e outra de final, mediante as quais se determinam a ligação (definição da localização) e formatação de recursos multimídia (textos, imagens, sons, animações, vídeos) que podem compor uma página e associações dela com outras páginas. Estes recursos multimídia são definidos como objetos e as páginas como documentos, por KUROSE (2003). Estas ligações, pontos passivos para outros documentos, são conhecidos como hiperlinks (âncoras). Portanto quando o usuário coloca o cursor sobre um hiperlink e clica sobre ele, o navegador solicitará ao serivdor novos dados que correspondem ao item selecionado.

Segundo SCAMBRAY (2003), devido à capacidade de extensão do HTML e suas variações, é possível embutir uma grande quantidade de funcionalidades em conteúdos Web aparentemente estáticos – HOAG (2002) comenta que este protocolo foi projetado inicialmente para apresentar o conteúdo estático de páginas. A necessidade de obtenção de conteúdo dinâmico (também definido como ativo), normalmente contido em bancos de dados, culminou com o desenvolvimento de linguagens de programação orientadas para a Internet, como o PHP, ASP, ActiveX, Java e outras.

Para Wikipedia (2006), o PHP é uma linguagem extremamente modularizada, o que a torna ideal para instalação e uso em servidores web. Muito parecida, em

tipos de dados, sintaxe e mesmo funções, das linguagens C e C++. Pode ser embutida no código HTML. Além disso, destaca a extrema facilidade com que o PHP lida com servidores de base de dados, como MySQL, PostgreSQL, Microsoft SQL Server e Oracle. A própria Wikipedia, uma enciclopédia livre do projeto Wikimedia Foundation (2006), que disponbilizada na internet, utiliza exclusivamente a linguagem PHP e bases de dados MySQL.

“O PHP é capaz de receber o envio de qualquer *browser* que siga a norma RFC-1867 (o que inclui Netscape Navigator 3 ou posterior, Microsoft Internet Explorer 3 com um patch da Microsoft, ou posterior sem patch*)”.* PHP (2006b).

# CAPÍTULO 3

## DISPONIBILIZAÇÃO DA FERRAMENTA DE TESTE POKE-TOOL COMO APLICAÇÃO WEB

### 3.1. A ESCOLHA DO AMBIENTE DE DESENVOLVIMENTO E FUNCIONAMENTO

A implementação da aplicação Webpoketool necessitou da instalação de alguns serviços: um servidor Web, uma linguagem de programação orientada para Web e um Sistema Gerenciador de Banco de Dados, pois o armazenamento de alguns dados permite o direcionamento correto do usuário na realização dos testes.

Por ser o sistema Operacional Linux imprescindível para a ferramenta POKE-TOOL - na qual a aplicação Webpoketool se baseia, a escolha dos softwares seria norteada também por este sistema operacional.

O software para servidor Web mais utilizado na Internet é o Apache HTTP Server, um dos projetos do The Apache Software Foundation (APACHE, 2006), chamado comumente de Apache.

Quando especificamos o sistema operacional Linux, a adoção do Apache é praticamente unânime.

"O Apache possui uma reputação merecida de segurança e desempenho". SCAMBRAY (2003, p. 46).

Segundo HOAG (2002), em 1996 o servidor Apache já havia se tornado o servidor mais popular, adquirindo respeito pela comunidade de usuários por ser sólido e robusto. Devido à sua arquitetura independente da plataforma operacional, pode ser executado na maioria das versões do Unix, Linux, Windows, no Netware 5.x e no Os/2.

Na Figura 3.1 encontramos uma estatística de utilização dos servidores Web mais conhecidos.

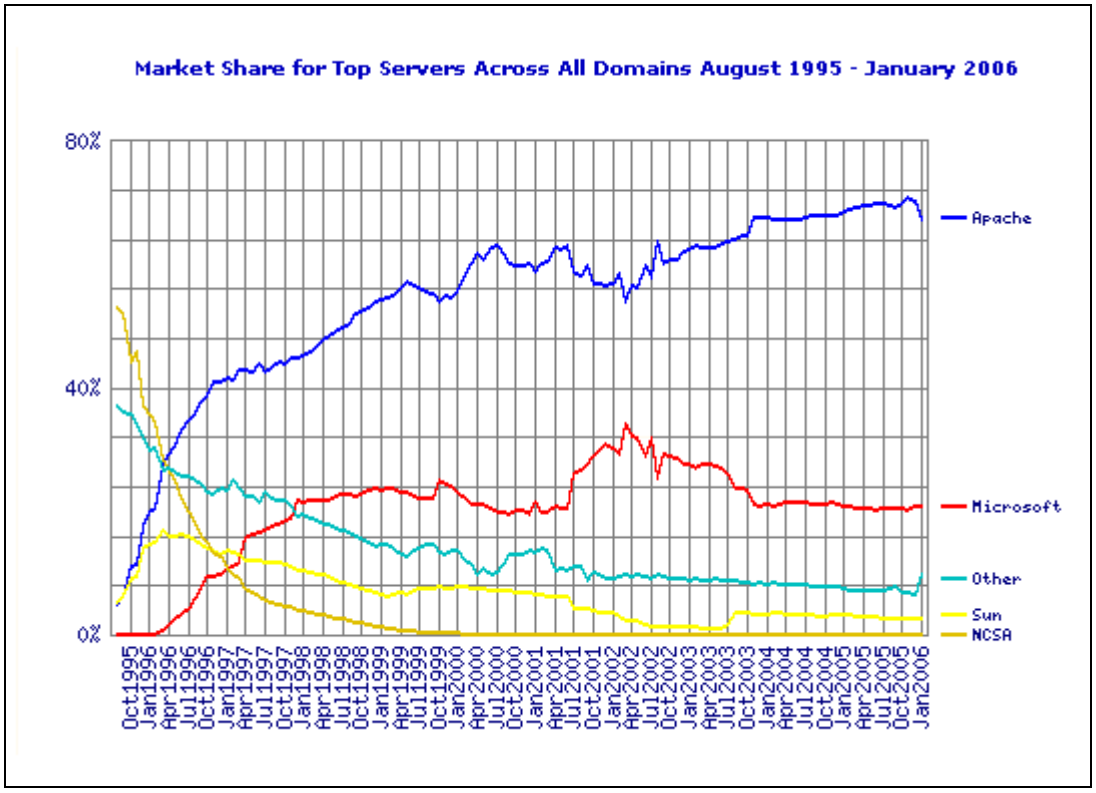



**FIGURA 3.1 – DISTRIBUIÇÃO DO MERCADO DE SERVIDORES WEB (NETCRAFT, 2006).**

Com o cuidado para que qualquer usuário pudesse utilizar a aplicação sem a necessidade de instalação de suporte a pacotes adicionais, como por exemplo flash, javascript, active-x, optou-se pela adoção de uma linguagem de programação que não necessitasse de *plugins* para funcionar. Outro requisito é que a linguagem pudesse ser utilizada de forma interpretada (em contrapartida à utilização de *scripts* CGI), permitindo que a manutenção e complementação das páginas possa ser feita por qualquer conhecedor de PHP. Dois outros grandes fatores levaram à escolha da linguagem de programação PHP:

a) forte semelhança com a linguagem C, também utilizada pela POKE-TOOL, facilitando a migração de novos recursos, a medida que sejam desenvolvidos;

b) grande quantidade de usuários, o que pode ser visto na Figura 3.2. Ainda segundo a Netcraft, em dezembro de 2005 o PHP estava presente em 22.172.983 domínios e 1.277.375 endreços IPs na Internet

(desconsiderando servidores localizados em redes internas), garantindo a presença de muitos programadores para esta linguagem.

Confirmando CONVERSE (2002), o resultado final de páginas escritas em PHP é o próprio HTML. A linguagem PHP é um módulo oficial do Apache, também gratuito, é incorporado em páginas HTML, é multiplataforma, é estável e rápido.

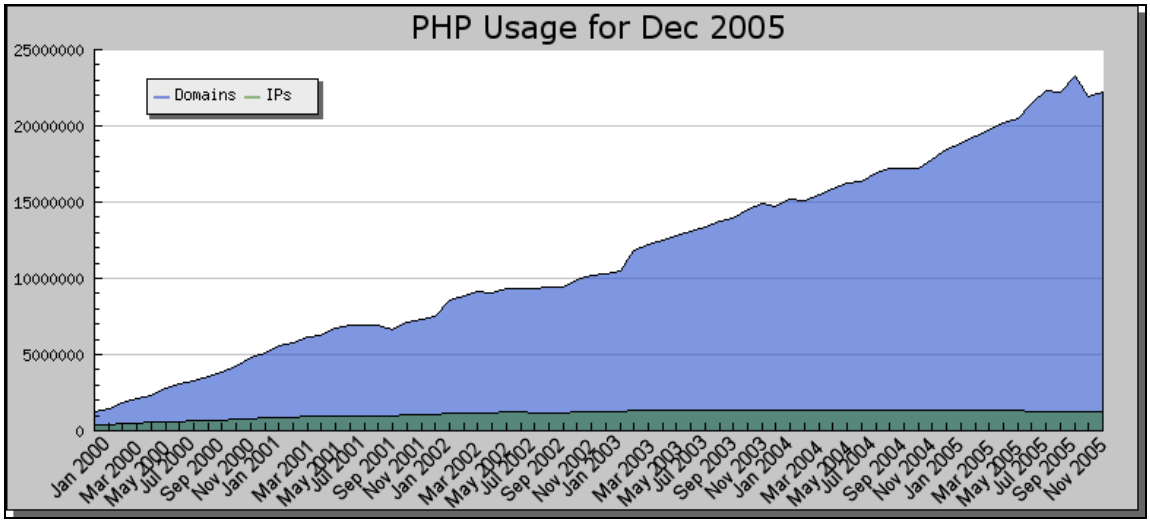



**FIGURA 3.2 – USO DO PHP EM 2005 (PHP, 2006A).**

“Os bancos de dados relacionais são, de longe, o tipo de banco de dados mais comumente utilizado”. Welling (2003, p.132).

O conteúdo de uma base de dados relacional pode ser complementando e usado por muitas pessoas ao mesmo tempo, uma das principais razões para a adoção do SQL. A flexibilidade da base de dados relacional e a acessibilidade do SQL possibilitam a criação de aplicações mais eficientes do que as que se utilizam de outros tipos de bancos de dados.

Para PATRICK (2002), o SQL (Structured Query Language, linguagem estruturada de consulta) é uma linguagem de programação declarativa utilizada para obter dados armazenados em uma base de dados relacional.

Para a seleção do serviço de banco de dados, suportável pelo Apache e pela linguagem PHP, levou-se em consideração que o Mysql é o terceiro banco de dados mais utilizado no momento - considerando-se todos os sitemas

operacionais - se considerarmos o sistema operacional Linux, ele é o mais comumente utilizado, como pode ser percebido na estatística de uso apresentada na Figura 3.3, baseada em pesquisa da empresa Software Development, realizada em junho de 2005.

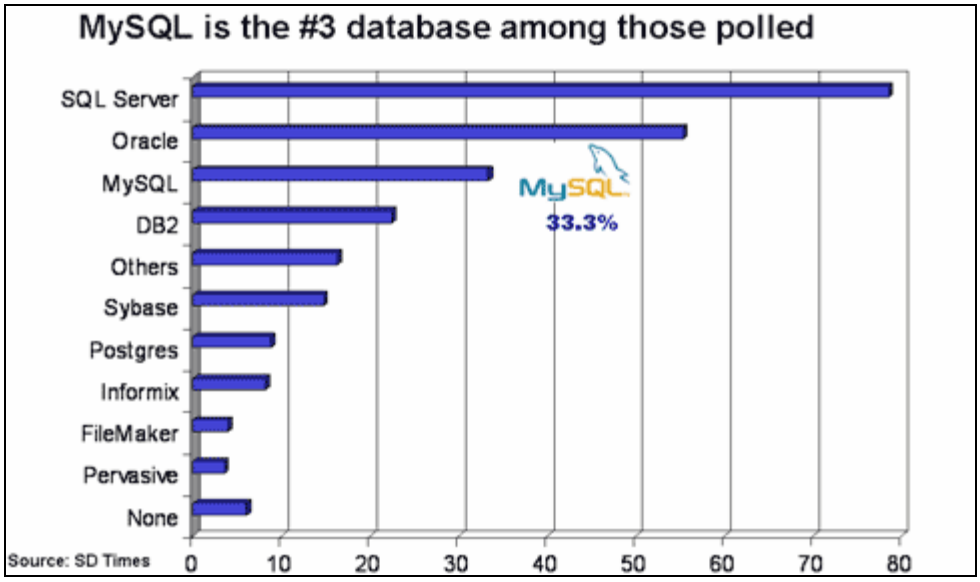



**FIGURA 3.3 – PARTICIPAÇÃO DO MYSQL NO MERCADO (MYSQL, 2006B).**

Tanto o Apache, como o PHP e o MySQL estão presentes em várias distribuições Linux e podem ser instalados no momento da instalação do Sistema Operacional ou posteriormente.

Uma característica fundamental para o desenvolvimento da Webpoketool foi pensá-la de forma a não exigir exclusividade de ambiente. Portanto, ela compartilha com outras aplicações, serviços e páginas, um servidor Linux que contenha a instalação do Apache, PHP e Mysql.

## 3.2. DESCRIÇÃO DA WEBPOKETOOL

É uma aplicação Web, já que pode ser utilizada remotamente via *browser.* Escrita em linguagem PHP e HTML, com suporte a banco de dados Mysql. A Webpoketool não é um Serviço Web (*Web Service*) pois não se utiliza dos protocolos: XML (*Extensible Markup Language*), SOAP (*Simple Object Access Protocol*) e as camadas: WSDL (*Web Services Definition Language*) e UDDI (*Universal Discovery Description*). Um grande ponto positivo desta tecnologia seria a utilização do serviço por diferentes sistemas operacionais e linguagens.

Para direcionar os usuários à execução correta das etapas de teste, permitindo continuidade, os dados de acesso são armazenados em um banco de dados. Por este motivo é necessário o cadastramento dos usuários e seus projetos de teste.

Quando um usuário se conecta à Webpoketool, após a identificação e da seleção do projeto, será sugerido, por padrão, que ele continue com o processo iniciado. Basicamente podemos considerar o processo de testes na Webpoketool sendo dividido em etapas que devem ocorrer subsequencialmente, a saber:

- envio do programa fonte e eventuais *include(s)* para instrumentação;
- instrumentação do programa fonte de acordo com os critérios desejados pelo usuário;
- *donwload* do pacote contendo, inclusive, o programa fonte instrumentado e compilado;
- realização dos testes;
- envio dos casos de testes;
- análise da precisão dos testes de acordo com critérios desejados.

## 3.3. CARACTERÍSTICAS ESPECIAIS DA WEBPOKETOOL

O desenvolvimento e a manutenção da Webpoketool seriam beneficiados com a diretiva de configuração “register_globals”. Com esta diretiva ativada o PHP permitiria o uso direto de variáveis, oriundas inclusive de formulários HTML, em qualquer das páginas da aplicação.

Como o PHP não requer a inicialização de variáveis para que elas sejam utilizadas, a ativação da diretiva “*register globals*” aumentaria a possibilidade de escrita de códigos inseguros, além de consumir mais recursos do servidor, uma vez que as variáveis permaneceriam em memória desde o momento da criação até o encerramento da utilização da aplicação. Porém, o fato mais agravante é que estas variáveis poderiam ser manipuladas pelos usuários, por exemplo, para dar acesso a informações privilegiadas ou prejudicar projetos de testes de outros usuários.

Com o intuito de garantir a segurança da aplicação, requerer o mínimo de recursos do servidor em que a Webpoketool está instalada e prevendo tanto evitar problemas com outras aplicações no mesmo servidor como evitar manutenções em decorrência da provável desativação posterior desta diretiva, optou-se por manter a diretiva “regiter_globals” desativada.

Para controlar a passagem de variáveis entre páginas há a possibilidade de utilização do controle de sessões, iniciando uma sessão após a autenticação do usuário e concluindo-a com a expiração ou com o término de uso da aplicação.

O controle de sessões poderia requerer a habilitação do uso de *cookies* nos navegadores dos usuários, os *cookies* são pequenos arquivos que armazenariam dados de interação com algumas páginas Web.

Uma alternativa à passagem de variáveis entre páginas com a opção “registrer_globals” desativada seria o uso da etiqueta HTML chamada *hidden*, como pode ser visto no segmento de código HTML e PHP abaixo:

<INPUT TYPE="hidden" NAME="midusuario" VALUE="<?php echo $midusuario;?>">

Este parâmetro passaria para a página subseqüente a variável midusuario com o valor que ela possui atualmente. No entanto, voltamos à possibilidade do usuário modificar os valores destas variáveis e causar danos irreparáveis ao ambiente.

Para evitar estes problemas, optou-se pelo desenvolvimento de um controle de sessões específico, que gera uma chave criptografada de 30 caracteres baseada horário e data (*timestamp*), que é passada entre as páginas e armazenada em uma tabela do banco de dados, que também armazena dados do projeto de testes e do usuário após a autenticação, agilizando a utilização da aplicação.

O controle da validade (expiração e revalidação) da sessão é feito pelas próprias páginas da aplicação, o que garante:

- atender maior quantidade de usuários simultaneamente, pois a conexão é considerada somente no momento que ocorre a transição entre uma página e outra;
- proibir que o usuário inicie mais de uma sessão de teste com o mesmo projeto, evitando sobreposições indesejadas;
- maior poder de controle sobre as seções, pois é feito exclusivamente pela aplicação.

Por utilizar um banco de dados integrado à aplicação Web, para o armazenamento do andamento dos projetos de testes, direcionamento dos usuários à execução de tarefas subseqüentes e evitar confusões (como, por exemplo, a tentativa de avaliação de um caso de teste cuja função não foi instrumentada), a aplicação estaria naturalmente suscetível à aplicação de um truque chamado injeção de comandos SQL (*SQL Injection*), normalmente utilizado por *hackers* para burlar autenticação, acesso e manipulação de dados privados.

Para SCAMBRAY (2003) a injeção SQL pode ser usada em Web *sites* que realizam autenticação baseada em formulários e que se utilizam de bancos de dados SQL, dentre outras aplicações, para contornar a autenticação. Para ele a melhor maneira de se evitar a injeção SQL é realizar a validação de entrada.

Exemplificando, tendo como base uma página de identificação do usuário, que contivesse um formulário de entrada de dados com dois campos a saber: usuário e senha; e considerando as variáveis $mapelido e $msenha como sendo as variáveis que armazenarão os dados digitados para serem confrontados com os valores contidos em uma tabela chamada usuarios contendo os campos: apelido e senha, poderíamos consultar a validade se a seguinte instrução retornasse pelo menos uma tupla:

```
$consulta = ("SELECT apelido, senha FROM webpoketool.usuarios WHERE apelido = '$mapelido'  and senha = '$msenha'");
```

Uma injeção SQL para acesso indevido à aplicação, contornado a autenticação do usuário, ocorreria se o usuário entrasse com o seguinte valor para o primeiro campo do formulário: ' OR 1=1 --' teríamos a seguinte situação, o caractere ' fecharia o campo de entrada do usuário, o comando OR continuaria com a consulta, o comando 1=1 retornaria verdadeiro e finalmente o comando -- que comentaria os comandos seguintes (senha do usuário), portanto, o resultado da consulta seria verdadeiro e garantiria acesso ao sistema sem que houvesse autenticação.

Para evitar esta vulnerabilidade foi desenvolvida uma função de validação dos dados digitados nos formulários que retira caracteres e palavras equivalentes a comandos SQL:

```
function validacampo($campo)
{
$campo
=addslashes(fgetss(trim(preg_replace(sql_regcase("/(from|select|insert|delete|update|where|drop all|drop
table|index|alter|create|references|reload|shutdown|process|file|describe|desc|grant|revoke|super|execute|lock tables|slave|replication|show tables|\*|#|--
|\\\\)/"),"",$campo))));
```

```
return $campo;
}
```

Esta função também evitaria o truque de injeção HTML e injeção PHP, vulnerabilidades similares à injeção SQL, mas utilizáveis em páginas HTML e PHP.

No entanto, durante os testes para impedir a injeção SQL observou-se que o problema estava relacionado exclusivamente à presença dos apóstrofos. Removê-los prejudicaria nomes de usuários e projetos que o contivessem, assim como seria possível remover alguma palavra importante caso coincidisse com algum dos comandos SQL citados na função.

As vulnerabilidades acima foram resolvidas com a utilização de duas funções do PHP:

- mysql_escape_string - que adiciona o caractere de escape antes de eventuais apóstrofos, sendo usada em consultas e novas inclusões de dados.

- fgetss – que remove etiquetas HTML e PHP.

Segundo WELLING (2003) o serviço Apache permite, em sua configuração padrão, 150 conexões simultâneas, esta quantidade tem se mostrado bastante suficiente, mesmo porque, conexões excedentes são armazenadas em uma fila de espera, conforme relata APACHE (2006b). Caso seja necessário aumentar este número, o administrador do sistema poderá editar o arquivo de configuração “httpd.conf”, item “MaxClients”, incluindo o valor necessário

Apesar da Webpoketool poder compartilhar o serviço de banco de dados Mysql com outras aplicações no servidor onde ela está hospedada, atualmente não há outra aplicação que o esteja utilizando. Para MYSQL (2006), a quantidade máxima padrão de conexões simultâneas aceitáveis é 100 processos, podendo ser modificada. Na aplicação Webpoketool os acessos ao banco de dados são ativados exclusivamente nos momentos em que são necessárias as operações (consultas, atualizações, exclusões e inclusões) e desativados em seguida.

Havendo a possibilidade de utilização deste servidor com outras aplicações que utilizem o Mysql, o administrador do servidor poderá aumentar a quantidade de conexões simultâneas aceitáveis através da edição do arquivo de configuração “mysqld“ , item “max_connections current value: <novovalor> “, sendo <novovalor> a quantidade desejada.

Para SCAMBRAY (2003), o os elementos de uma aplicação Web que os intrusos irão mais facilmente perceber e procurar explorar de imediato, são as vulnerabilidades do próprio software do servidor Web, recomendando fortemente a atualização do servidor, além de sugerir a desativação de dois módulos que formatam a listagem de diretórios, através do *script*: ./configure –disable-module=dir –disable-module=autoindex .

## 3.4. EXEMPLO DE UMA SESSÃO DE TRABALHO COM A WEBPOKETOOL COMPARADA COM A SESSÃO DE TRABALHO DA POKE-TOOL.

O acesso à Webpoketool ocorre através da Internet. Atualmente ela está hospedada em http://www.webpoketool.com.br, servidor pertencente ao Laboratório de Teste de Software da Unimep, Campus Taquaral. A estrutura básica de páginas da aplicação pode ser visualizada na Figura 3.4, detalhes são apresentados subseqüentemente.

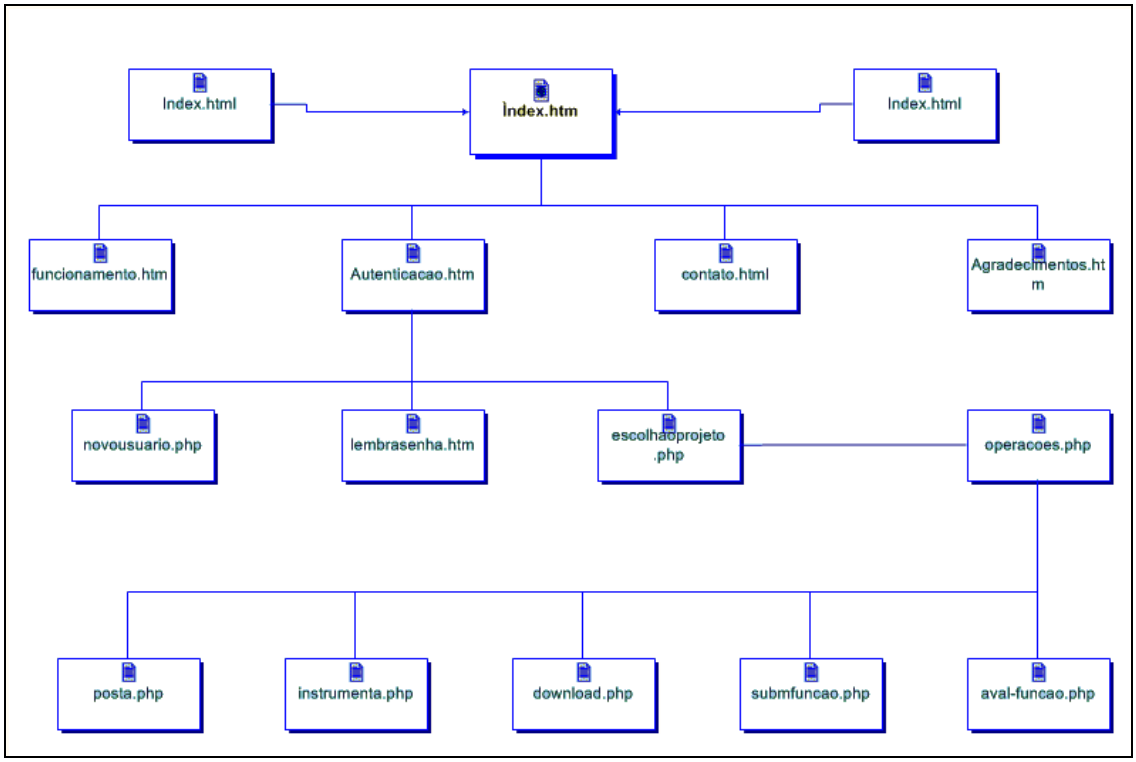



**FIGURA 3.4. ESTRUTURA BÁSICA DE PÁGINAS DA WEBPOKETOOL.**

NA página inicial da Webpoketool são apresentados os objetivos do projeto. Nesta página o usuário poderá acessar as páginas: descrição do funcionamento; agradecimentos, contato e a página de utilização. Ao clicar sobre o *hiperlink* de utilização, a página de identificação do usuário é apresentada, como pode ser visualizado na Figura 3.5.


**FIGURA 3.5 – IDENTIFICAÇÃO DO USUÁRIO.**

Caso o usuário não tenha sido cadastrado, ao clicar sobre o *hiperlink* de cadastramento, acessará uma página com a solicitação de alguns dados pessoais, que facilitarão a comunicação no decorrer da utilização da aplicação Web - esta página pode ser observada na Figura 3.6. Após o cadastramento será apresentada uma página de conclusão do cadastramento, que pode ser observada na Figura 3.7. Todas as páginas envolvidas com a utilização da aplicação Webpoketool possuem o rodapé que pode ser visto na Figura 3.5, facilitado o contato com o responsável pelo *site*, removido das figuras apresentadas subseqüentemente para evitarmos duplicidade.

O apelido e a senha serão a chave de acesso do usuário ao sistema, portanto, caso já exista um usuário com o mesmo apelido, será solicitado que o segundo usuário informe um apelido que ainda não tenha sido utilizado.

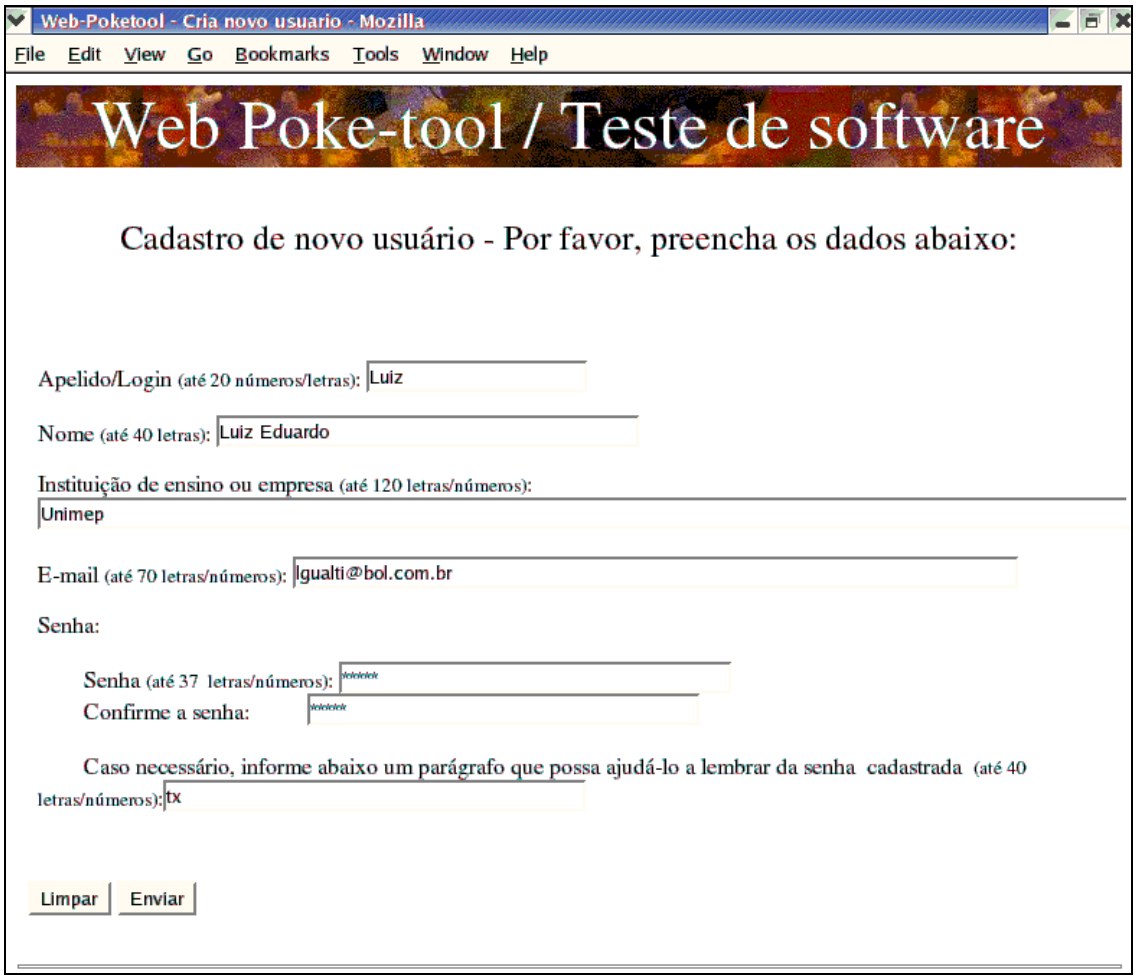



**FIGURA 3.6 – CADASTRAMENTO DE NOVO USUÁRIO DA WEBPOKETOOL.**

Caso o internauta já seja usuário do sistema e tenha se esquecido de sua senha, ao clicar sobre o *hiperlink* “Caso você tenha se esquecido da senha, clique aqui” da página de identificação (Figura 3.5), e após o usuário informar o apelido, será apresentada uma página com sua dica de senha - que pode ser vista na Figura 3.8.

Após o cadastramento do novo usuário, este deve clicar sobre o *hiperlink* “iniciar a utilização”, que permitirá o acesso direto à aplicação, que o levará à página de cadastramento de projetos.

O acesso à manutenção dos projetos também pode ser realizado após as páginas de identificação do usuário (Figuras: 3.9, 3.10 e 3.11).


**FIGURA 3.7 - PÁGINA DE CONFIRMAÇÃO DE SUCESSO NO CADASTRAMENTO.**

Caso o usuário não tenha nenhum projeto cadastrado, a aplicação o direcionará para o cadastramento do novo projeto – esta página de cadastramento pode se visualizada na Figura 3.12.


**FIGURA 3.8 – PÁGINA DE APRESENTAÇÃO DA DICA DE SENHA.**

Não tendo nenhum projeto cadastrado, a aplicação direcionará o usuário para o cadastramento de um projeto.

Caso o usuário já tenha cadastrado um ou mais projetos, poderá dar continuidade à utilização/realização dos testes, assim como poderá incluir novos projetos – vide Figura 3.16.

A aplicação Web permite o uso concomitante da quantidade de projetos que o usuário necessitar.

Para dificultar que alguns usuários se identifiquem como outros, por tentativa e erro, de forma manual ou com o uso de ferramentas automatizadas, optou-se pela solicitação do usuário e da senha em diferentes páginas da aplicação. Este funcionamento é representado pelas Figuras 3.9, 3.10 e 3.11.


**FIGURA 3.9 – INFORMANDO O USUÁRIO.**


**FIGURA 3.10 – ACESSO À DIGITAÇÃO DA SENHA PELO USUÁRIO VÁLIDO.**

Após o usuário ter sido validado, é iniciado um controle especial de sessão (atividade), com uma chave encriptada e única, que é encaminhada à página

de solicitação da senha. Este controle de sessão é armazenado em uma tabela da aplicação e possui um período de validade.

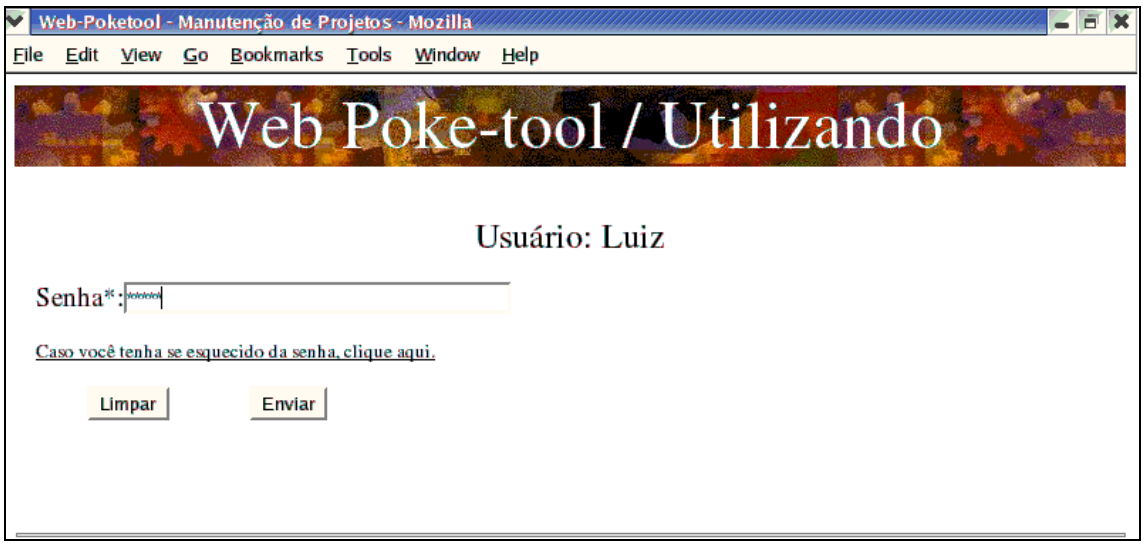



**FIGURA 3.11 – USUÁRIO VÁLIDO INFORMANDO A SENHA.**

Após o usuário ter informado sua senha, a aplicação confirma se a sessão ainda está válida, estando, detecta qual usuário está relacionado à sessão e confirma se a senha está correta. Estando correta, é permitida a continuidade de uso da aplicação.

Em comparação com a POKE-TOOL, supondo a existência de um computador especialmente preparado para a utilização desta ferramenta, também seria necessário informar usuário e a respectiva senha. Estando no modo gráfico, que é comumente utilizado, após a autenticação, será necessário que o usuário inicie o *shell*, que é imprescindível para o funcionamento da POKE-TOOL.

Em muitos casos é necessário que o usuário possua direitos de administração do sistema, necessários durante a instrumentação e testes, que normalmente serão realizados no mesmo computador.

Voltando à aplicação Wepoketool, após o usuário ter sido identificado, é apresentado a ele uma página de escolha de projeto, como pode ser visto na Figura 3.12, em que o usuário poderá dar continuidade a trabalhos realizados em projetos que já tenha cadastrado ou incluir um novo projeto de testes.

A aplicação Webpoketool disponibilizará uma área especial para cada usuário e projeto. Estas áreas não são visíveis aos usuários, porém utilizadas pela aplicação.

Em comparação ao uso da ferramenta POKE-TOOL, seria equivalente à criação de uma pasta de trabalho, que pode ser exemplificado pela Figura 3.13. Sendo possível que o usuário confunda-se com testes que já tenha realizado e no caso de computadores de uso comum, preparados especialmente para uso da ferramenta, ainda pode se possível que um usuário confunda-se ou interfira no trabalho de outro usuário.


**FIGURA 3.12 – INCLUSÃO DE NOVO PROJETO DE TESTES.**

Na Figura 3.14 – Cadastramento de novo projeto, o usuário poderá optar por controlar a submissão dos casos de testes.

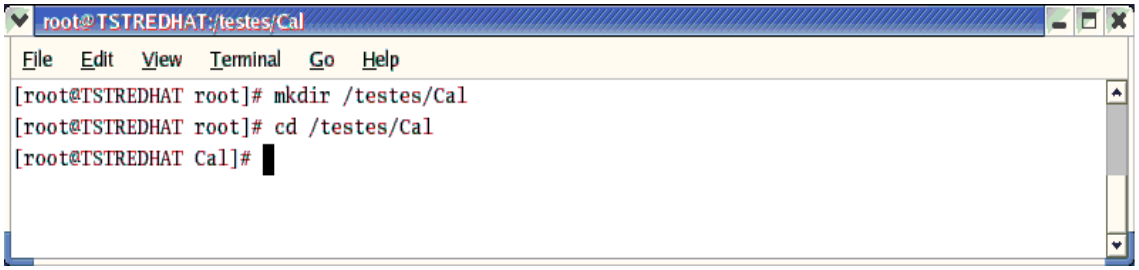



**FIGURA 3.13 – CRIAÇÃO DE UMA PASTA PARA ALOJAR OS TESTES NA POKE-TOOL.**

Após o término de uma sessão de testes, o usuário irá submeter o arquivo path.tes e a própria aplicação se encarregará de definir qual a numeração de seqüência do caso de teste que está sendo enviado, evitando sobreposições indesejadas.


**FIGURA 3.14 – CADASTRAMENTO DE NOVO PROJETO.**

Após o cadastramento do projeto, o usuário da aplicação recebe uma página de confirmação do sucesso – como pode ser visto na Figura 3.15 - ou aviso da existência de outro projeto com mesmo nome (do mesmo usuário), permitindo modificá-lo.


**FIGURA 3.15 – CONCLUSÃO DO CADASTRAMENTO DO PROJETO.**

Após o cadastramento do projeto, esse passa a estar disponível na página de seleção de projetos, como pode ser visto na Figura 3.16.


**FIGURA 3.16 – SELEÇÃO DO PROJETO OU CADASTRAMENTO DE NOVO PROJETO.**

O próximo passo será o envio do programa fonte, do computador do usuário para a instrumentação pela Webpoketool. Detalhes podem ser vistos na Figura 3.17 – Seleciona arquivo fonte para envio.

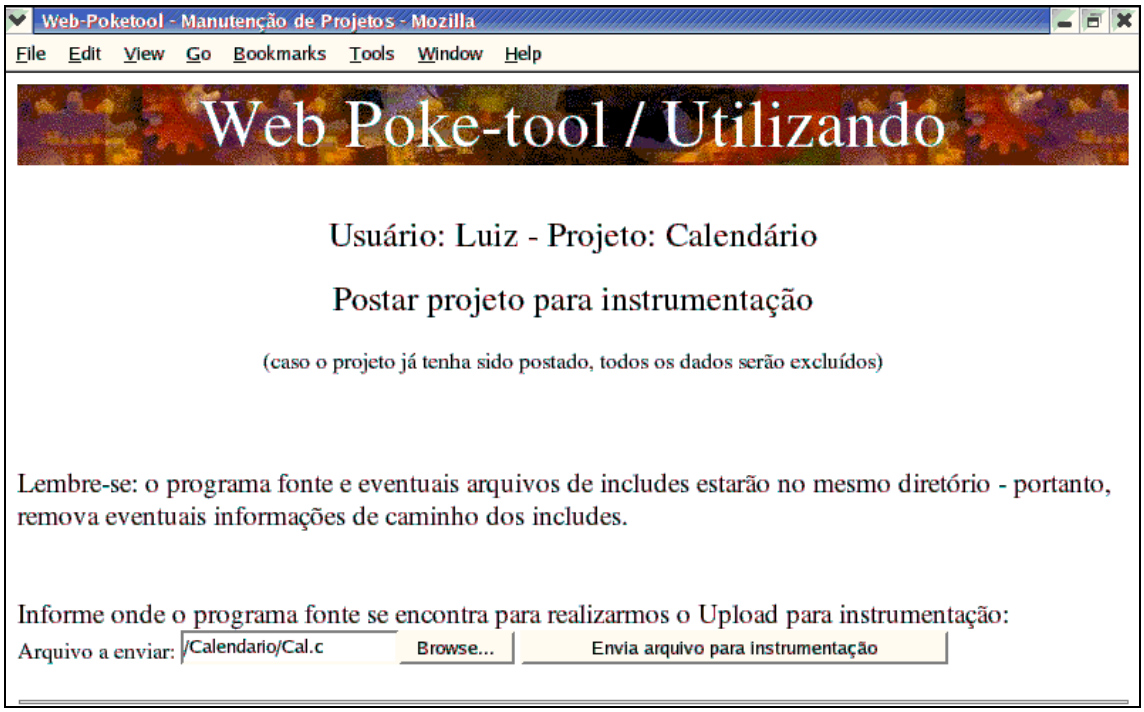



**FIGURA 3.17 - SELECIONA ARQUIVO FONTE PARA INSTRUMENTAÇÃO.**

O PHP é capaz de receber arquivos através de navegadores, portanto ao clicar sobre o botão *browse*, o usuário poderá localizar mais facilmente o arquivo a postar. Um exemplo da página de seleção de arquivos pode ser visto na Figura 3.18.

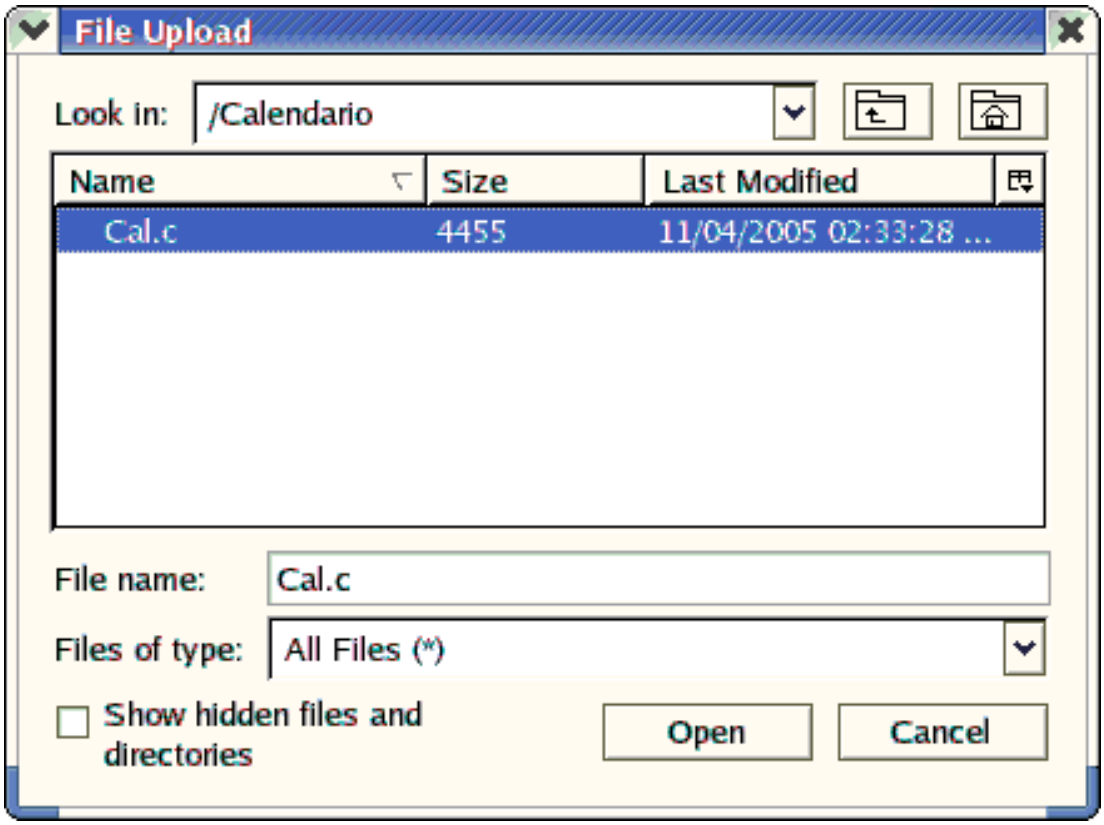



**FIGURA 3.18 – DETALHE DA CAIXA DE DIÁLOGO PARA SELEÇÃO DO PROGRAMA FONTE A SER ENVIADO**

Após o envio do programa fonte, será solicitado ao usuário se ele possui arquivos de *include* a serem enviados, como pode ser visto na Figura 3.19. Em caso positivo, a Figura 3.20 apresenta a página de envio de *include* e após a transferência retorna o controle à página anterior (Figura 3.19).


**FIGURA 3.19 – QUESTIONAMENTO SOBRE O USO DE *INCLUDES*.**


**FIGURA 3.20 – ENVIO DE *INCLUDES*.**

Em comparação com a POKE-TOOL, no caso de utilização de um computador de uso comum, especialmente preparado para uso desta ferramenta, será necessário que o usuário obtenha seu(s) programa(s) fonte(s), o que pode ocorrer das mais variadas formas, por exemplo, cópia em disquete, pasta compartilhada e transferência via FTP.

Após o envio do programa fonte, e de eventuais *includes* serão apresentadas opções para instrumentação (Figura 3.21 – Página de seleção de critérios a

serem utilizados na instrumentação do programa fonte).

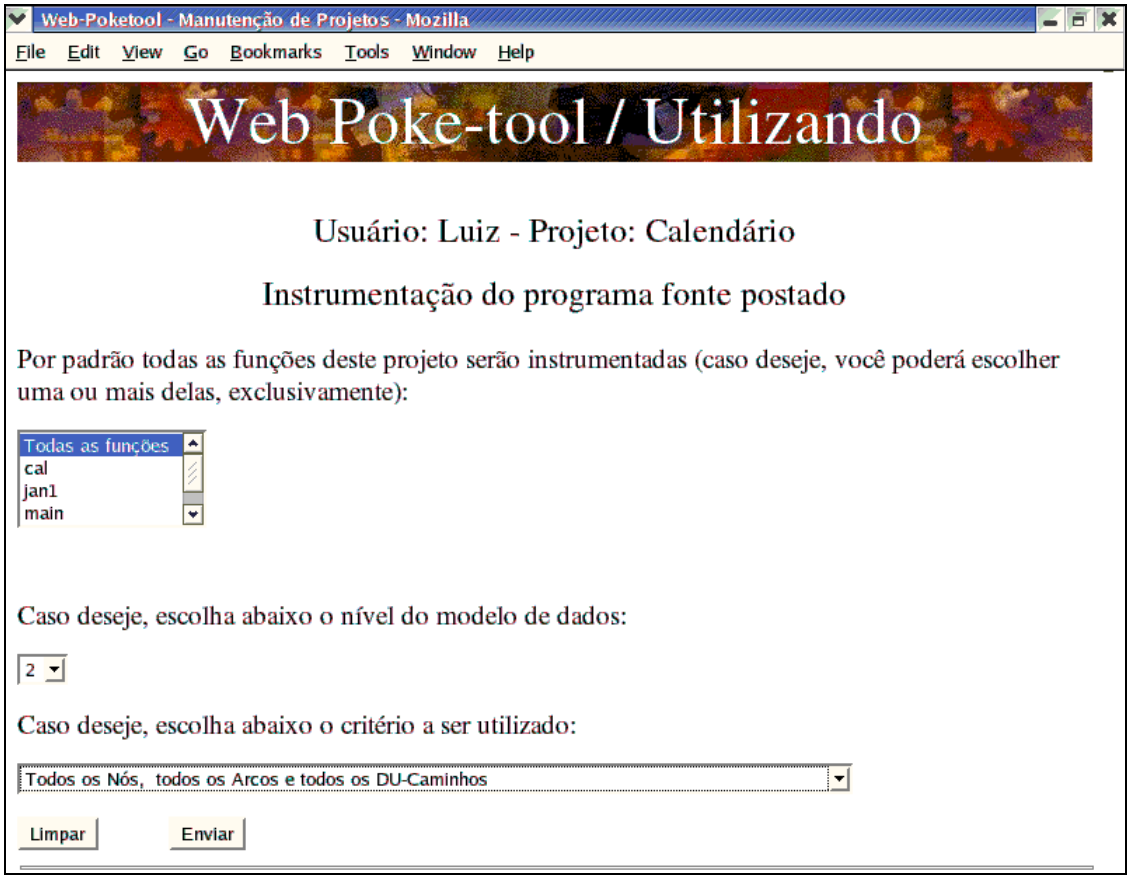



**FIGURA 3.21 – PÁGINA DE SELEÇÃO DOS CRITÉRIOS A SEREM UTILIZADOS NA INSTRUMENTAÇÃO DO PROGRAMA FONTE.**

Nesta tela é permitindo a seleção de quais funções devem ser instrumentadas, do nível do modelo de dados e do critério de instrumentação desejado: “Todos Nós e todos Arcos”, ou “Todos Nós, todos Arcos, todos P-usos e Todos-usos”, ou “Todos os Nós, todos os Arcos e todos os DU-Caminhos”, “Todos os Nós, todos os Arcos, todos Pot-Usos e Pot-Usos/DU”, ou “Todos os Nós, todos os Arcos, todos Pot-DU-Caminhos”, “Todos os Nós, todos os Arcos, todos P-Usos, Todos-usos, todos Pot-Usos e todos Pot-Usos/DU”.

A Figura 3.22 representa a equivalência da aplicação Webpoketool para a ferramenta POKE-TOOL. A codificação usada na POKE-TOOL pode confundir usuários, sobretudo usuários novos, também é necessário que o usuário lembre-se de qual função deseja instrumentar e se desejar instrumentar todas,

colocar uma a uma, após o respectivo parâmetro “–f”. É constante o uso do Help pelo usuário para relembrar a sintaxe dos comandos.

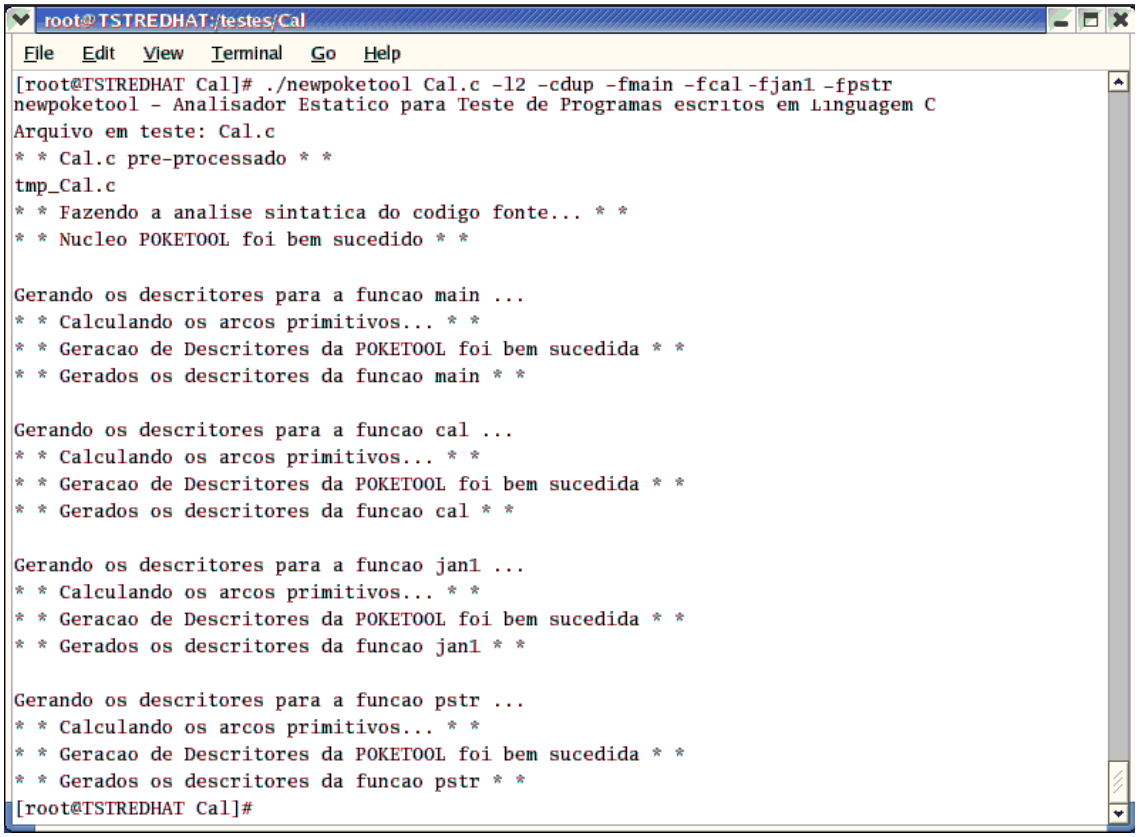



**FIGURA 3.22 – EXEMPLO DE EXECUÇÃO DA NEWPOKETOOL (POKE-TOOL).**

Concluída esta etapa, o programa fonte instrumentado, juntamente com os arquivos de controle dos testes e respectivas pastas, são compactados e disponibilizados pela aplicação Web para *download*, como pode se visto na Figura 3.23 – Preparação para *download* do arquivo compactado, e após o usuário clicar sobre ‘continua’ uma imagem semelhante à Figura 3.24 - Detalhe para o *download* do arquivo compactado contendo o programa instrumentado, arquivos e pastas necessários – aparecerá permitindo ao usuário informar onde deseja que este arquivo seja armazenado, como na Figura 3.25 - Detalhe da seleção do local para armazenamento do arquivo compactado.

Após a descompactação do arquivo o usuário contará com a estrutura de pastas e arquivos necessários a execução dos casos de testes. São apresentadas as Figuras 3.26, 3.27 e 3.28 como exemplos destas pastas e

arquivos, levando-se em consideração a instrumentação do programa Cal.c com modelo de dados padrão e critério “Todos os Nós, todos os Arcos, todos P-Usos, Todos-usos, todos Pot-Usos e todos Pot-Usos/DU”.

O código fonte do programa Cal.c é apresentado no Apêndice A.


**FIGURA 3.23 – PREPARAÇÃO PARA DOWNLOAD DO ARQUIVO COMPACTADO.**

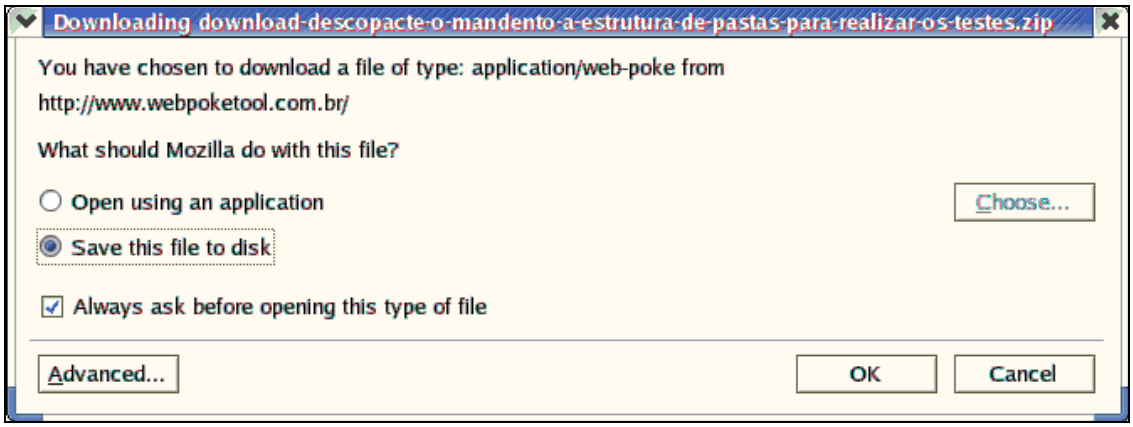



**FIGURA 3.24 – DETALHE PARA O DOWNLOAD DO ARQUIVO COMPACTADO CONTENDO O PROGRAMA INSTRUMENTADO, ARQUIVOS E PASTAS NECESSÁRIOS.**

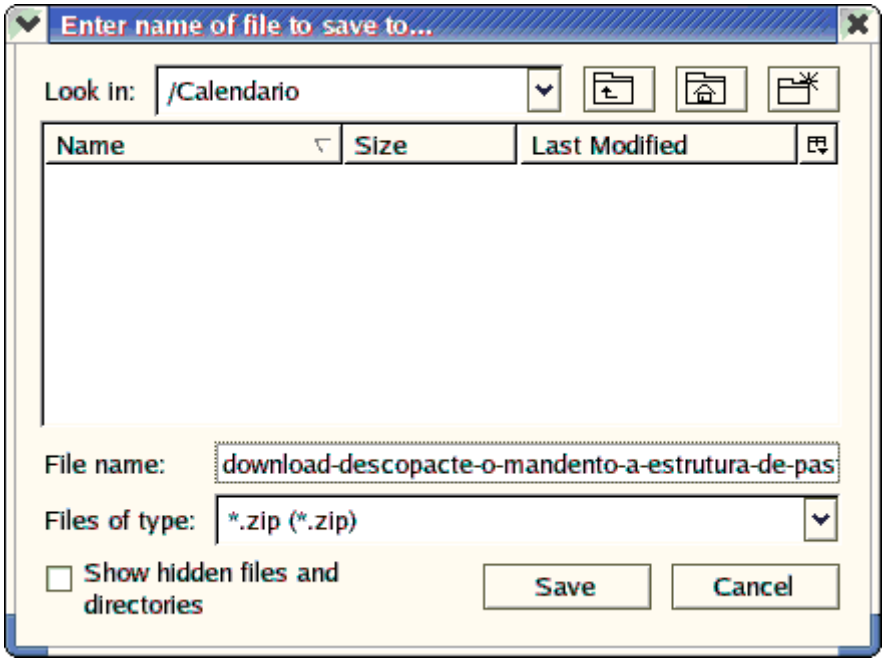



**FIGURA 3.25 – DETALHE DA SELEÇÃO DO LOCAL PARA ARMAZENAMENTO DO ARQUIVO COMPACTADO.**

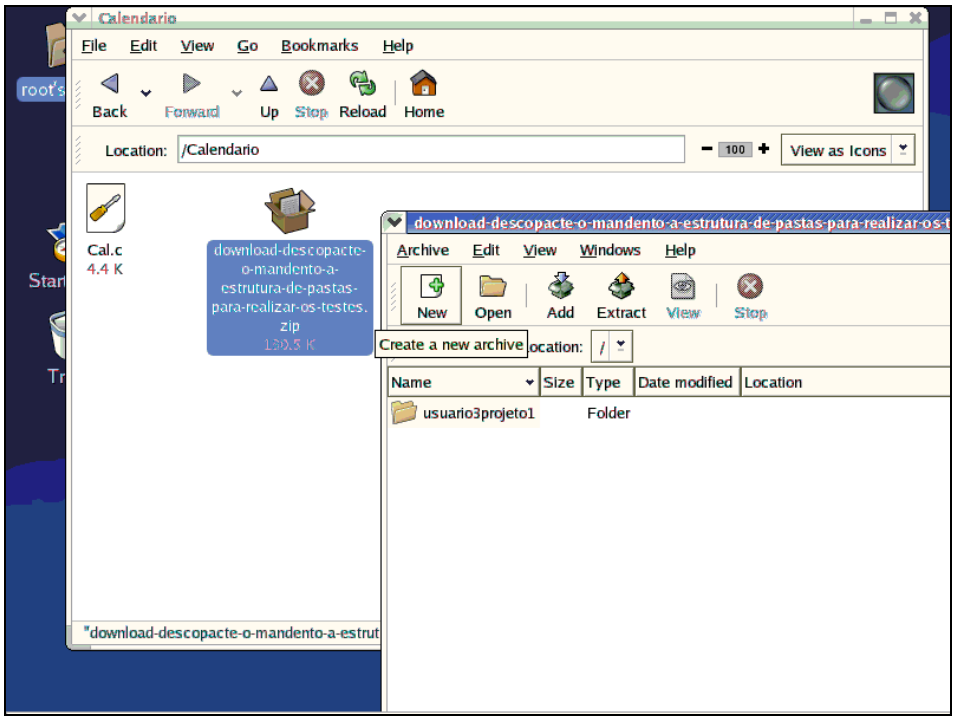



**FIGURA 3.26 – VISUALIZAÇÃO DO DESKTOP DO USUÁRIO APÓS A BAIXA DO PACOTE.**

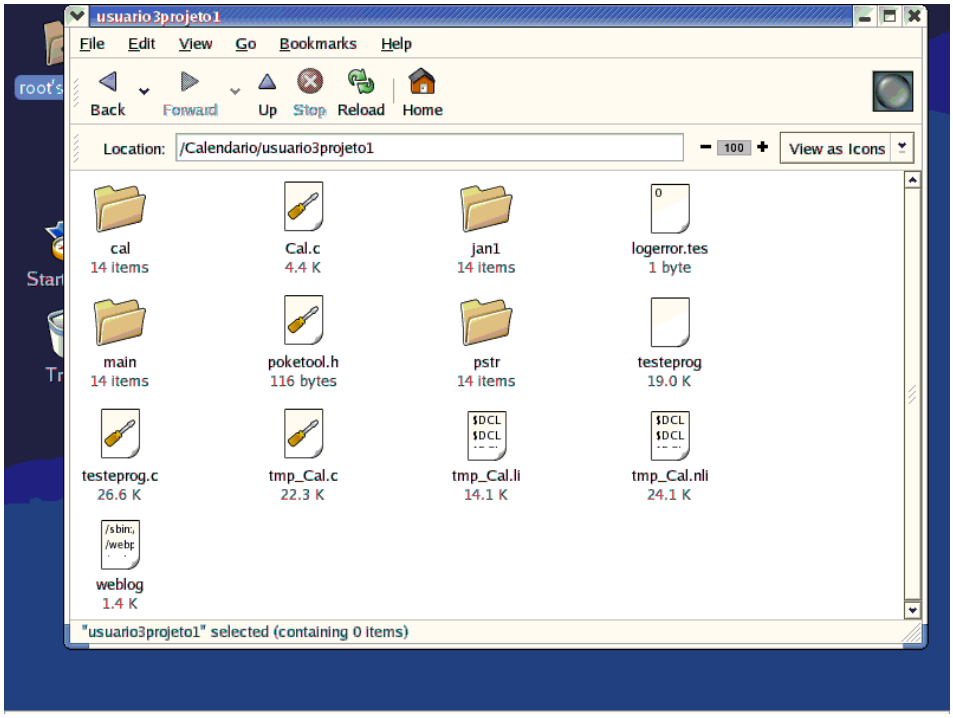



**FIGURA 3.27 - DETALHE PARA PASTAS E ARQUIVOS CRIADOS.**

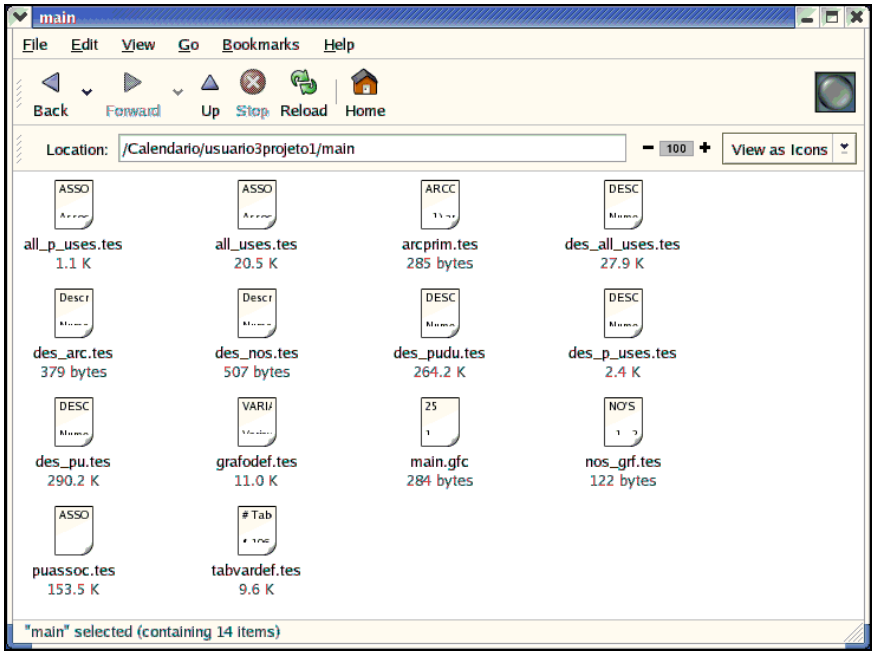



**FIGURA 3.28 – DETALHE DA PASTA 'MAIN' APÓS A INSTRUMENTAÇÃO DO PROGRAMA CAL.C**

Todas as vezes em que o usuário se conectar à Webpoketool, após a identificação e da seleção do projeto, será sugerido, por padrão, que ele

continue com o processo iniciado. O processo de testes na Webpoketool é dividido em etapas que devem ocorrer subseqüencialmente: envio do programa fonte e eventuais *include(s)* para instrumentação, instrumentação do programa fonte de acordo com os critérios desejados pelo usuário, *download* do pacote contendo, inclusive, o programa fonte instrumentado e compilado, realização dos testes, envio dos casos de testes e análise da precisão dos testes de acordo com critérios desejados.

Ainda tratando do direcionamento do usuário para a execução correta das etapas, a Webpoketool não permite que o usuário tente executar uma etapa que não seja subseqüente à última etapa realizada.

Norteada por este comportamento, a Webpoketool não permite:

- o pedido de instrumentação para funções que não existam;
- envio de casos de testes para funções que não tenham sido instrumentadas,
- a análise da precisão dos testes para funções que não tenham tido casos de testes enviados e, conseqüentemente, para funções que não tenham sido instrumentadas.

A Webpoketool, flexível, também permite que o usuário volte a realizar tarefas que já tenha realizado anteriormente, redefinindo automaticamente a próxima etapa.

Não há equivalência destes procedimentos na POKE-TOOL e durante a utilização desta ferramenta é comum ocorrerem confusões como as citadas acima.

No caso da ferramenta POKE-TOOL, a compilação do programa fonte instrumentado deve ocorrer manualmente, como pode ser visto na Figura 3.29, conseqüentemente, exigindo conhecimento do usuário.

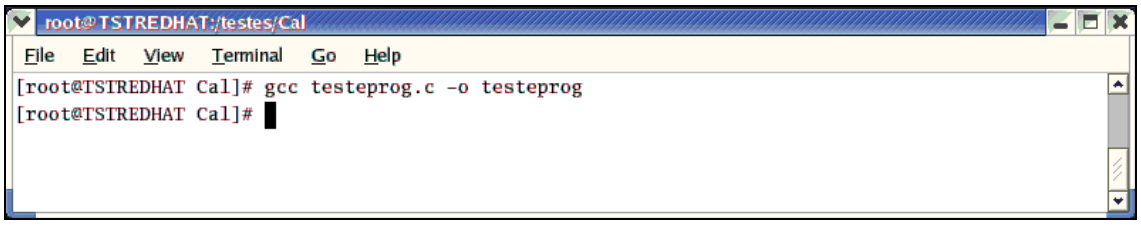



**FIGURA 3.29 – COMPILANDO O PROGRAMA FONTE INSTRUMENTADO (POKE-TOOL).**

A Figura 3.30 apresenta a página de seleção de atividade. É importante lembrar que na Webpoketool que a etapa sugerida para execução é mostrada subseqüentemente às etapas já realizadas, evitando que alguma etapa seja esquecida, prejudicando os casos de testes.


**FIGURA 3.30 – PÁGINA DE SUGESTÃO DE PRÓXIMA ATIVIDADE PARA CONTINUIDADE DOS CASOS DE TESTES.**

Exemplificando, se o usuário ainda não tiver enviado o arquivo, só será apresentada a opção de envio. Outro exemplo, se o usuário já tiver enviado o arquivo e também o tiver instrumentado, será apresentada a ele, por padrão, a opção de *download* e também permitindo que ele, caso deseje, refaça alguma etapa já concluída.

A execução dos casos de testes é similar tanto na Webpoketool como na POKE-TOOL. Exemplos seguem nas Figuras: 3.31, 3.32 e 3.33.

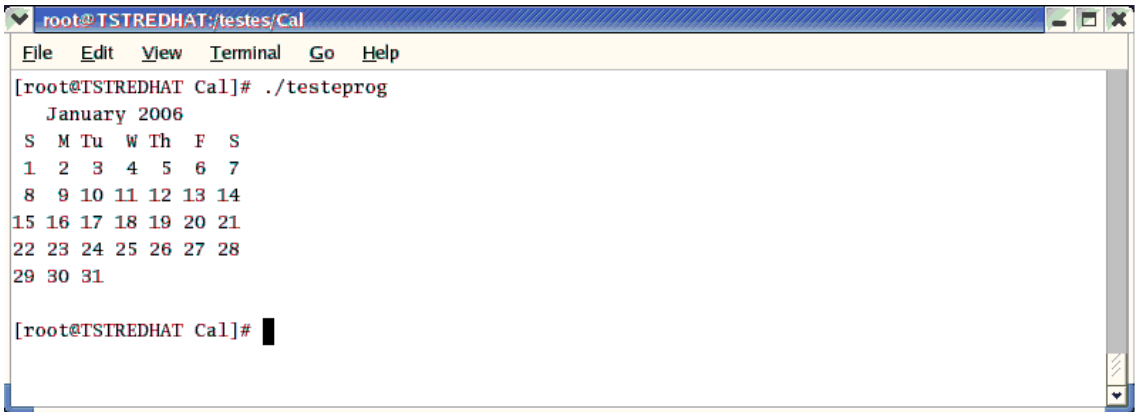



**FIGURA 3.31 – EXECUÇÃO DO PRIMEIRO TESTE**


**FIGURA 3.32 – EXECUÇÃO DO SEGUNDO TESTE.**

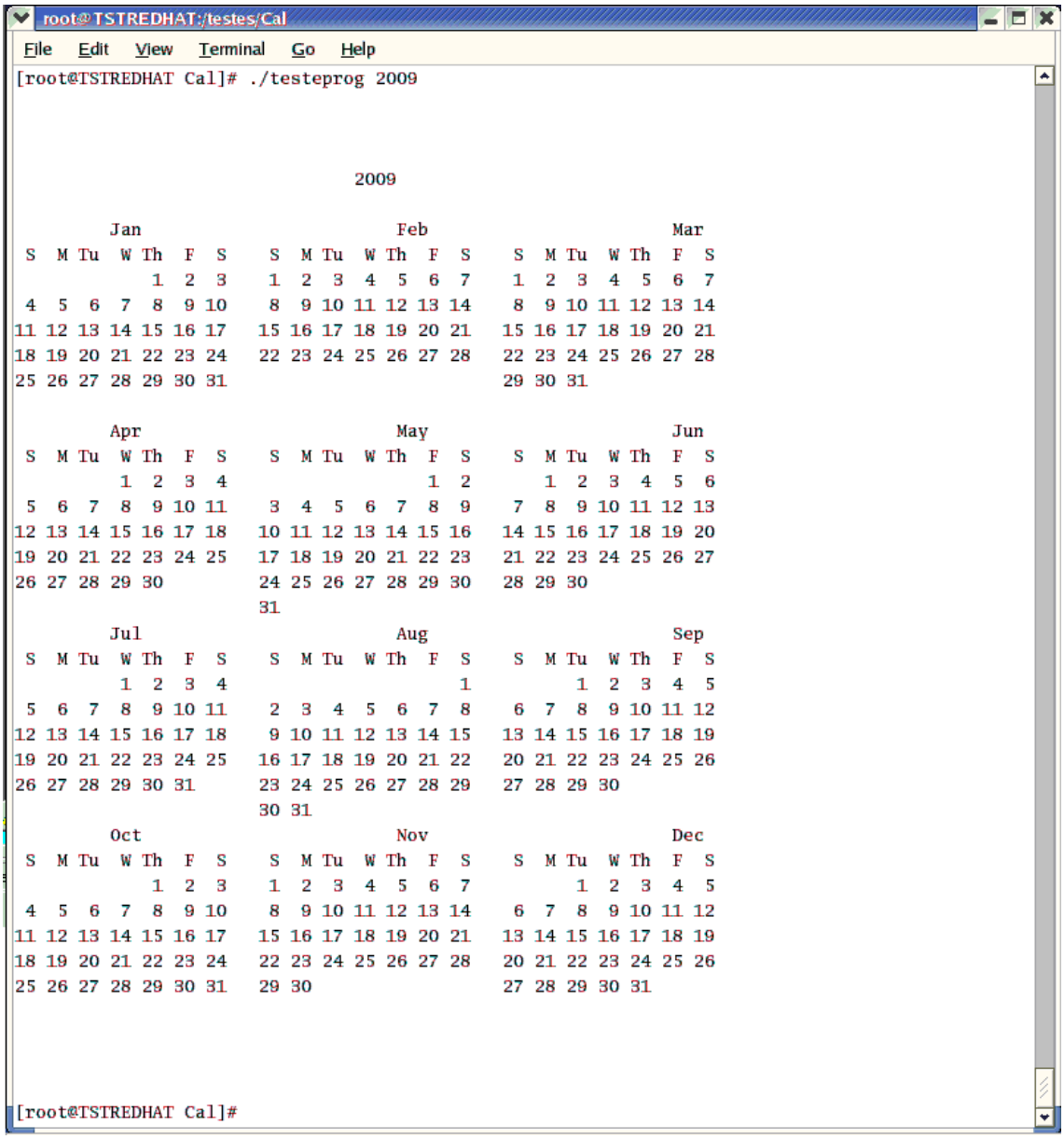



**FIGURA 3.33 – EXECUÇÃO DO TERCEIRO TESTE.**

Após o usuário realizar o caso de teste, bastará submetê-los à Webpoketool. Estes casos de testes estão em arquivos chamados ‘path.tes’, localizados nas pastas de mesmo nome das funções que tenham sido utilizadas durante os testes.

Para submetê-los à aplicação, após a identificação do usuário, a seleção do projeto e a solicitação de envio de caso de teste, será apresentada uma página semelhante à da Figura 3.34, permitindo ao usuário selecionar para

qual (das) função(ões) que tenha(m) sido instrumentada(s) o caso de teste está relacionado.


**FIGURA 3.34 – SELEÇÃO DA FUNÇÃO PARA ENVIO DO CASO DE TESTE.**

Após a submissão do(s) caso(s) de teste, correspondente ao o envio de cada arquivo path.tes, será apresentada a página de opção de novos envios para a mesma função - nos casos de controle de casos de teste habilitado, a própria Webpoketool pode definir a seqüência do caso de teste - ou para outra(s) função(ões), desde que tenha(m) sido instrumentada(s). A Figura 3.35 apresenta a possibilidade de seleção de outra função.

Para projetos em que o usuário tenha escolhido não controlar os casos de testes, fazendo com que todos os testes que realize e submeta, independente de quantos forem, sejam considerados somente como um caso de teste, para evitarmos confusões, as páginas representadas pelas Figuras: 3.36 e 3.37 substituem, respectivamente as páginas representadas pelas Figuras: 3.35 e 3.38.


**FIGURA 3.35 – DETALHE DE SELEÇÃO DA FUNÇÃO PARA ENVIO DO CASO DE TESTE.**


**FIGURA 3.36 – SELEÇÃO DA FUNÇÃO PARA ENVIO DO CASO DE TESTE EM PROJETOS SEM CONTROLE DE CASOS.**

A Figura 3.38 apresenta a possibilidade de atribuição de uma numeração especial para cada caso de teste a ser submetido. Caso esta numeração não seja informada, a aplicação incluirá a numeração subseqüente às já postadas para a função selecionada.


**FIGURA 3.37 – SELEÇÃO DO ARQUIVO PATH.TES PARA ENVIO EM PROJETOS SEM CONTROLE DE CASOS.**


**FIGURA 3.38 – SELEÇÃO DO ARQUIVO PATH.TES PARA ENVIO.**

Terminada a etapa de submissão dos casos de testes, será possível analisar a abrangência dos testes. A Figura 3.39 apresenta como sugestão a avaliação dos casos de testes - no entanto, o usuário pode escolher por continuar a submeter novos casos de testes


**FIGURA 3.39 – MENU DE SELEÇÕES PERMITINDO A AVALIAÇÃO DOS TESTES.**

Optando pela avaliação dos casos de testes submetidos, é apresentada uma página de definição de parâmetros, como pode ser visto na Figura 3.40.


**FIGURA 3.40 – ESCOLHA DA FUNÇÃO CUJOS TESTES SERÃO ANALISADOS**

Caso o usuário tenha optado por não controlar os casos de teste submetidos os itens de escolha de seqüência de caso de teste inicial e final não serão solicitados, ficando definido apenas o padrão.


**FIGURA 3.41 – SELEÇÃO DE CRITÉRIOS PARA A AVALIAÇÃO DA FUNÇÃO PREVIAMENTE SELECIONADA.**

Após a definição da função que será analisada, como pode ser visto na Figura 46, os resultados são processados e é apresentada uma página permitindo a visualização dos resultados (Figura 3.42).

Comparando-se a Figura 3.43 com as Figuras: 3.44 e 3.45, observa-se a equivalência entre o resultado, para as mesmas instrumentações, casos de testes e análise, entre a aplicação Webpoketool e a ferramenta POKE-TOOL – *script* newpokeaval. Observa-se que a sintaxe usada no programa newpokeaval pode confundir usuários, sobretudo usuários novos.


**FIGURA 3.42 – VISUALIZAÇÃO DOS RESULTADOS DE ABRANGÊNCIA DOS TESTES.**

No caso da POKE-TOOL também é necessário que o usuário se lembre de qual função ele teria instrumentado e gerado o caso de teste. Considerando-se a utilização de múltiplos casos de testes a situação é agravada, pois o usuário deve lembrar da quantidade de casos executados em cada função.

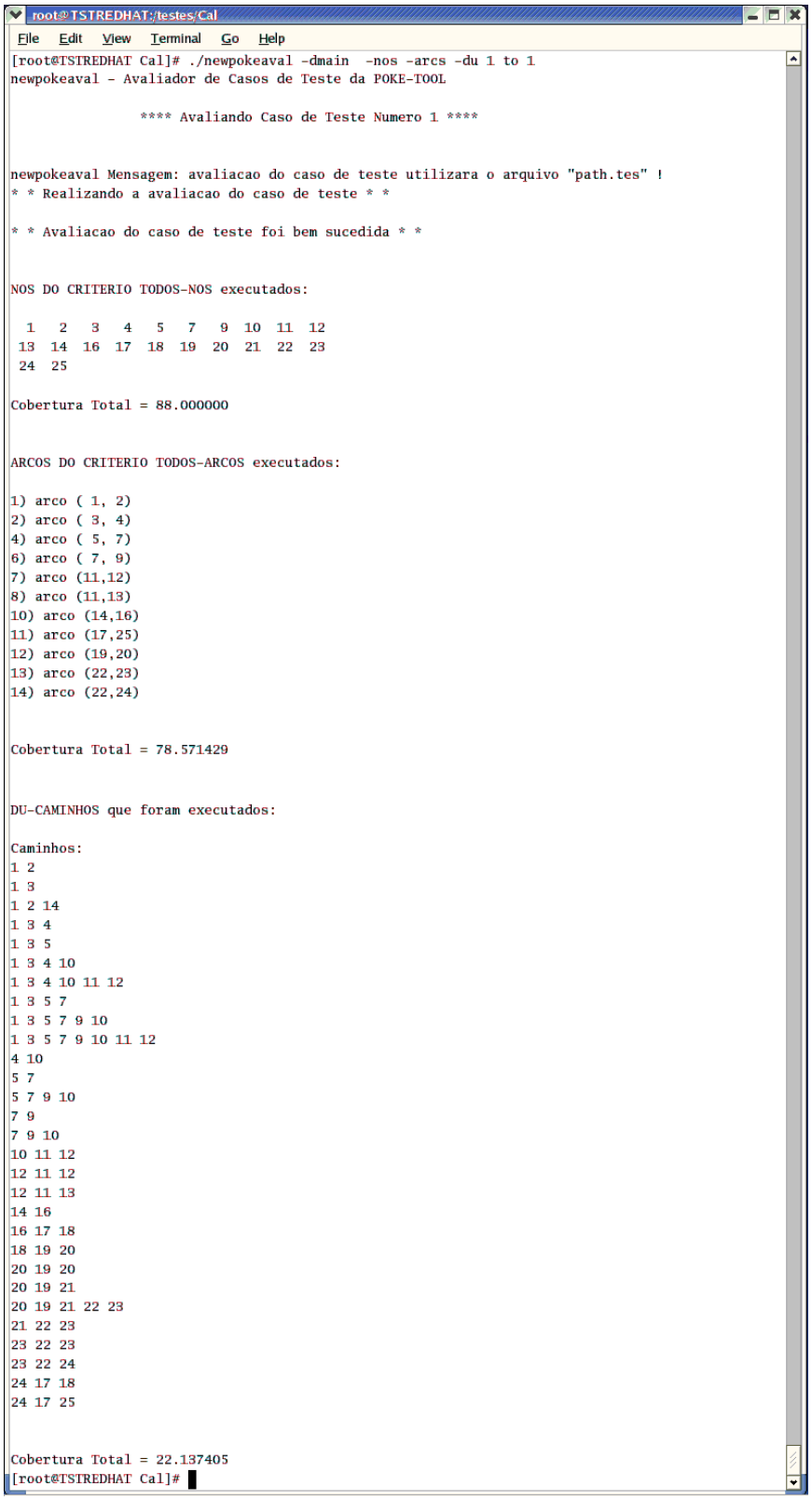



**FIGURA 3.43 – EXEMPLO DE EXECUÇÃO DA NEWPOKEAVAL (POKE-TOOL).**

Estes problemas não ocorrem na Webpoketool que direciona o usuário para a correta utilização.

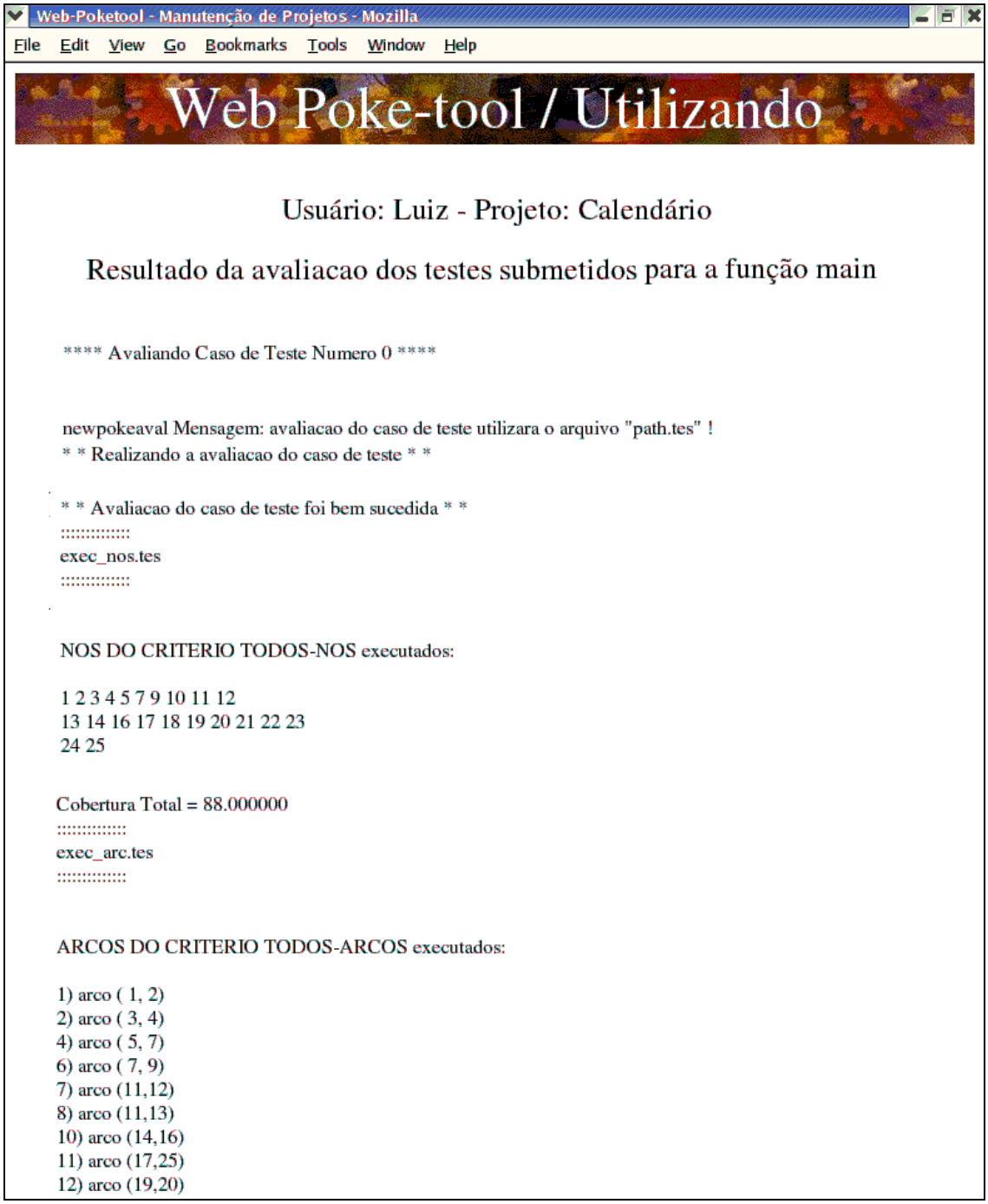



**FIGURA 3.44 – APRESENTAÇÃO DA PRECISÃO DOS TESTES.- PRIMEIRA PARTE.**

```
13) arco (22,23)
14) arco (22,24)


Cobertura Total = 78.571429
::::::::::::::
exec_dupaths.tes
::::::::::::::


DU-CAMINHOS que foram executado

Caminhos:
                                                    s:
1 2
1 3
1 2 14
1 3 4
1 3 5
1 3 4 10
1 3 4 10 11 12
1 3 5 7
1 3 5 7 9 10
1 3 5 7 9 10 11 12
4 10
5 7
5 7 9 10
7 9
7 9 10
10 11 12
12 11 12
12 11 13
14 16
16 17 18
18 19 20
20 19 20
20 19 21
20 19 21 22 23
21 22 23
23 22 23
23 22 24
24 17 18
24 17 25


Cobertura Total = 22.137405
```

**FIGURA 3.45 – APRESENTAÇÃO DA PRECISÃO DOS TESTES – ÚLTIMA PARTE.**

## 3.5 WEBPOKETOOL/POKE-TOOL, CONSIDERAÇÕES A RESPEITO DO AMBIENTE IDEAL

A POKE-TOOL é uma ferramenta orientada à sessão, ou seja, o usuário deve fornecer o programa para teste, compilá-lo, gerar um programa instrumentado, executar os casos de teste, avaliar os casos de teste e visualizar os resultados. Na versão utilizada (última) a execução se dá por meio de *scripts*, que chamam programas utilitários do sistema operacional que podem não estar disponíveis na instalação do usuário.

A POKE-TOOL está disponível para download, o usuário, com um pouco de conhecimento em administração do sistema operacional Linux poderá instalá-la e configurá-la em seu microcomputador. A POKE-TOOL analisa a adequação de um conjunto de casos de teste e fornece medidas de cobertura dos principais critérios de teste da técnica estrutural, de relevância indiscutível para disciplinas de teste de software. Disponibilizar os recursos da POKE-TOOL através da Web evitaria a necessidade de direitos e conhecimentos de administração do sistema operacional, permitindo a usuário direcionar seus esforços unicamente na aplicação dos critérios em seus programas.

Estando a Internet disponível em todas as instituições de ensino, disponibilizar este recurso nela tornaria a ferramenta facilmente alcançável. Idealizando, o ambiente Web deveria distinguir seus usuários e projetos e permitir que executassem seus programas/casos de testes diretamente no ambiente – este último recurso pode ser observado em ASKIGOR (2006). No entanto, foram levantadas preocupações reais, cujas principais seguem:

- os recursos (arquivos, pastas e serviços) usados pelos programas dos usuários durante os casos de testes poderiam não estar disponíveis nesse servidor, impossibilitando a execução;

- permitir a execução de programas no servidor Web poderia por em risco a própria estabilidade do ambiente, pois seria praticamente impossível evitar a execução de algum código que pudesse revelarem brechas de segurança, que por exemplo, permitissem o

acesso a programas e áreas de trabalho de outros usuários, causar lentidão e/ou instabilidade dos serviços disponíveis ou ainda, atacar terceiros;

- haveria a possibilidade de um dos programas travar processos no servidor, entrando em loop indesejadamente;

- como garantir a disponibilidade de memória incluindo espaço em disco, para armazenar todos casos de testes de todos os usuários da aplicação Web, assim como capacidade de processamento.

Ainda a cerca do ambiente ideal, se os problemas acima fossem problemas facilmente transponíveis, haveria de se pensar tornar a aplicação Web em um Serviço Web, disponibilizando framework completo para o desenvolvimento e testes, evitando a submissão dos programas. Por final, o ambiente também poderia se assemelhar ao de uma máquina virtual para cada projeto, com espaço e configurações de framework, compiladores... que pudessem ser customizadas pelos usuários e reaproveitados entre seus projetos.

Para manter a segurança da aplicação Web e evitar constantes investimentos em recursos computacionais no servidor, optamos por manter a execução dos casos de teste no microcomputador do usuário e as outras etapas na Webpoketool.

Centralizando o uso da POKE-TOOL na Webpoketool, sempre que ocorrerem alterações (correções ou melhorias) na POKE-TOOL bastará aplicá-las na Webpoketool para que todos os usuários sejam beneficiadas.

## 3.6 WEBPOKETOOL/POKE-TOOL, CONSIDERAÇÕES A RESPEITO DO USO DE PLATAFORMAS OPERACIONAIS NÃO LINUX E PROGRAMAS FEITOS EM OUTRAS LINGUAGENS DE PROGRAMAÇÃO.

A POKE-TOOL pode ser configurada para trabalhar com linguagens de programação procedurais. CHAIM(1991) descreve como esta tarefa deve ser realizada.

O compilador GCC, usado pela POKE-TOOL está disponível para várias plataformas computacionais e suportando hoje as linguagens C, C++, Objective-C, Fortran, Java, e Ada, além das bibliotecas destas linguagens, conforme GCC (2006).

Existem diferenças relevantes relacionadas com os formatos de arquivos nas variadas plataformas computacionais. Para usá-las na Webpoketool seriam adequações desses formatos. Esta diferença entre plataformas operacionais nos permite classificá-las em 3 grandes grupos: DOS, MAC e UNIX.

i) No grupo DOS, os términos de linha são identificados por dois caracteres representadores: CR (*carriage return*, respectivamente "\015") e LF (*line feed*, respectivamente "\012"). Neste grupo temos os sistemas operacionais: Dos, Windows, MsWindows32, Vms, Os2, Netware, Cygwin e Network);

ii) no grupo MAC, os términos de linha são identificados simplesmente pelo caractere representador CR (*carriage return*, equivalente a "\015"). Neste grupo temos os sistemas operacionais Mac e Mac-Os;

iii) finalmente, no grupo UNIX, os términos de linha são identificados simplesmente pelo caractere representador LF (*line feed*, equivalente a "\012"). Neste grupo temos os sistemas operacionais Unix, Linux, FreeBSD.

Outra diferença relevante está relacionada com as referências a subdiretórios, que nos sistemas do grupo Unix usam o caractere “/” e no sistemas do tipo DOS o caractere “\”.

Portanto, usuários da Webpoketool que não se utilizassem de sistemas Operacionais do grupo UNIX, teriam seus arquivos convertidos de forma transparentemente em três subseqüentes etapas:

a) a primeira etapa ocorreria após o envio do programa fonte, com a conversão do formato de arquivo utilizado pelo usuário para o padrão Unix;

b) a segunda etapa ocorreria após a instrumentação do programa fonte convertido e antes da disponibilização dele (e de arquivos e pastas de controle) para *download*, desta vez convertendo-os do formato Unix para o formato do sistema operacional do usuário. Nesta etapa, para usuários de sistemas operacionais tipo DOS, também ocorre uma modificação no programa fonte instrumentado, relacionada com a estrutura de pastas, que no UNIX referencia-se por “/” e no sistema Operacional do usuário por “\”;

c) a última etapa ocorre durante o envio dos casos de teste, em cada uma destas postagens, realizando a conversão do arquivo “path.tes”para o padrão Unix.

Apesar destes problemas serem transponíveis na Webpoketool, o fato da instrumentação e atividades subseqüentes e dela dependentes atuarem sobre o programa fonte pré-processado, limitam a atual versão da POKE-TOOL – e conseqüentemente a Webpoketool, à compatibilidade exclusiva com a linguagem ANSI C em sistemas operacionais Linux.

# CAPÍTULO 4

## CONCLUSÕES

### 4.1 RESULTADOS, DISCUSSÕES

A aplicação Webpoketool evita a necessidade de instalação da POKE-TOOL, permite a execução de múltiplos casos de testes, inclusive para o mesmo usuário, concomitantemente, armazenando dados de controle das atividades, e direcionando-o na execução dos passos necessários subseqüentemente, tornando seu uso praticamente intuitivo.

A Webpoketool mostrou-se funcional para diversas distribuições Linux, incluindo distribuições Live-CD (Knoppix e Kurumin), que podem ser utilizadas facilmente por discentes, inclusive a partir de suas casas e locais de trabalho, pois não há necessidade de instalação do sistema Operacional Linux.

Após a escolha do ambiente englobando servidor com Linux, HTTP Apache, PHP e Mysql, dificuldades no desenvolvimento foram superadas. Também tiveram que ser superadas algumas dificuldades de configuração para que a Webpoketool utilizasse um servidor localizado no Laboratório de Teste de Software da Unimep, juntamente com outras aplicações. Conseqüentemente ocorreram muitas revisões e complementações na Webpoketool desde o início do desenvolvimento.

Mesmo não tendo sido parte do escopo desta dissertação, tentou-se superar o problema de incompatibilidade com outras plataformas operacionais, ocasionado pelo pré-processamento usado pela POKE-TOOL. Após tentativas em conjunto com pessoas do desenvolvimento do GCC para as plataformas Unix e DOS, concluímos que a solução envolveria a construção de um novo compilador, único, que funcionasse para várias plataformas (*cross-platform*), após algumas pesquisas encontrei um projeto que estava em andamento no *site* http://www.sourceforge.net, iniciado por pessoas relacionadas com a

distribuição Linux Mandriva, após contato soube-se que o projeto não havia sido continuado.

Sem sucesso também foram feitos testes de compilação entre plataformas, usando a distribuição Linux Cygwin, que é executada sobre sistemas operacionais Windows.

## 4.2 CONSIDERAÇÕES FINAIS

Foi especificada uma aplicação Web denominada Webpoketool, com o propósito de disponibilizar pela Internet os recursos da ferramenta de teste POKE-TOOL, ampliando e viabilizando sua utilização dela a qualquer pessoa.

Os requisitos básicos desta aplicação foram definidos: a aplicação também acompanharia e direcionaria os usuários durante sua utilização, evitando confusões que levariam conclusões erradas e que a ferramenta POKE-TOOL, que vem sendo atualizada desde seu desenvolvimento, não sofresse alterações.

A aplicação Webpoketool especificada foi implementada com: a escolha do ambiente de desenvolvimento direcionado para Web, da adição de recursos de controle dos processos de teste dos usuários e de recursos de segurança. Esta implementação constitui o protótipo atual da Webpoketool, disponibilizada na Internet através do endereço http://www.webpoketool.com.br.

Durante os testes realizados com distribuições Linux e programas escritos em linguagem C, houve equivalência dos resultados obtidos com a Webpoketool e a POKE-TOOL.

## 4.3 SUGESTÕES PARA TRABALHOS FUTUROS

Os recursos da versão atual da Webpoketool baseiam-se nos recursos da versão atual da POKE-TOOL, portanto, à medida que esta última seja complementada (com, por exemplo, o desenvolvimento de novos módulos de suporte a outras linguagens de programação e a otimização do módulo de

instrumentação, substituindo os códigos pré-processados por códigos convencionais), a Webpoketool também deverá ser atualizada.

A execução remota dos casos de testes não torna a Webpoketool compatível com outras plataformas operacionais pois a pré-intrumentação usada pela POKE-TOOL adiciona códigos especificamente compatíveis com o sistema Operacional Linux, portanto o desenvolvimento de um único compilador com as características do GCC, mas que funcione para várias plataformas operacionais (*cross-platform*), com capacidade de traduzir instruções de pré-processamento entre estas plataformas, aliada à conversão dos tipos de arquivos e referências de pastas, permitirá a utilização da POKE-TOOL e conseqüentemente a Webpoketool em outras plataformas operacionais.

A Webpoketool poderá ser complementada com sua tradução para outras línguas - internacionalização, ROCHA (2003), o que beneficiará uma grande quantidade de estudantes que não conhecem o idioma português.

Também poderá ser complementada com o controle versões dos projetos submetidos e com o armazenamento de dados dos erros encontrados poderá ser complementada para suportar PSP, que segundo CÔRTES (2001), permitirá propor uma lista de verificação mais ajustada ao perfil de defeitos do programador e fazer avaliações da evolução do nível da qualidade dos projetos.

## REFERÊNCIAS BIBLIOGRÁFICAS

APACHE Software Foundation. Disponível em: <http://www.apache.org>. Acesso em 02 de jan. 2006.

ASKIGOR. Disponível em: <http://www.askigor.org/>. Acesso em 28 de abr. 2006.

CHAIM, Marcos Lordello. *POKE-TOOL – Uma Ferramenta para Suporte ao Teste Estrutural de Programas Baseado em Análise de Fluxo de Dados*. 1991. 176 f. Dissertação (Mestrado em Engenharia Elétrica) - Faculdade de Engenharia Elétrica da Unicamp. Campinas, 1991.

CHAIM, Marcos Lordello. *Depuração de Programas Baseada em informação de Teste Estrutural*. 2001. 222 f. Tese (Doutorado em Engenharia Elétrica) - Faculdade de Engenharia Elétrica e de Computação da Unicamp. Campinas, 2001.

CONVERSE, Tim; Joyce Park. *PHP4: A Bíblia*. Tradução de Edson Fumankiewicz. Rio de Janeiro: Elsevier, 2003. 868 p.

CÔRTES, Mário L.; Thelma C. dos S. Chiossi. *Modelos de Qualidade*. Campinas: Ed. da Unicamp, Instituto da Computação, 2001. 148 p.

DeMILLO, Richard A. Mutation Analysis as a Tool for Software Quality Assurance. In: Proceedings of the COMPSAC'80. *IEEE Computer Society Press,* p. 390-393, Los Alamitos, CA, oct. 1980.

DEITEL, H.M; P.J. Deitel. *Java, como programar*. 4. ed. Tradução de Carlos Arthur Lang Lisbôa. Porto Alegre: Bookman, 2003. 1386 p.

FELIZARDO, Kátia R. *Uma introdução geral ao teste de software*. Disponível em: http://linhadecodigo.com.br>. Acesso em 02 de jan. 2006.

GCC. Disponível em: <http://www.gnu.org/software/gcc/>. Acesso em 02 de jan. 2006.

HOAG, Melanie. *Servidor Web usando Apache*. Tradução Toni Ricardo Cavalheiro. São Paulo: Berkeley Brasil, 2002. 472 p.

IEEE, *Std 610.12-1990*: IEEE Standard Glossary of Software Engineering Terminology. New York, 1990. 83 p.

KUROSE, James F.; ROSS, Keith W. *Rede de Computadores e a Internet: uma nova abordagem*. Tradução de Arlete Símille Marques. São Paulo: Addison Wesley, 2003. 548 p.

MALDONADO, José C.; CHAIM, Marcos L.; JINO, Mário. Seleção de casos de testes Baseados em Fluxo de Dados através dos Critérios Potenciais Usos. In: *Proc.* II Simpósio Brasileiro de Engenharia de Software, Canela, RS, Out, 1988, pp. 24-35.

MALDONADO, José Carlos. *Critérios potenciais-usos: uma contribuição ao teste estrutural de software*. 1991. 247 f. Tese (Doutorado em Engenharia Elétrica) - Faculdade de Engenharia Elétrica da Unicamp. Campinas, 1991.

MALDONADO, J.C.; Martiniano, L. A. F.; Dória, E.S.; Fabbri, S.C.C.P.F.; Mendonça, M. Readers project: replication of experiments - a case study Using Requeriments Documents. *In*: *ProTeM-CC-Project Evaluation Workshop - International Cooperation***,** CNPq. Rio de Janeiro: CNPQ, 2001. p. 85-117.

McCABE, Thomas J. A Complexity Measure, *IEEE Transactions on Software Engineering,* vol. SE-2, n. 4, p. 308-320, dec. 1976.

MENEZES, Paulo Fernando Blauth*. Linguagens formais e autômatos.* Porto Alegre: Sagra Luzzatto, 1997. 168p.

MOLINARI, Leonardo. *Testes de Software: Produzindo Sistemas Melhores e Mais Confiáveis*. São Paulo: Érica, 2003. 228 p.

MYERS,Glenford J. *The art of software testing*. New York: John Wiley & Sons, 1979. 177 p.

MYSQL, *MySQL 3.20, 4.0, 4.1 Reference Manual*. Disponível em: <http://downloads.mysql.com/docs/refman-4.1-en.a4.pdf>. Acesso em 02 de jan. 2006a.

_____, *Marketshare*. Disponível em: <http://www.mysql.com/why-mysql/marketshare/>. Acesso em 02 de jan.2006b.

NETCRAFT. *Web Server Survey*. <http://news.netcraft.com/archives/web_server_survey.HTML>. Acesso em 02 de jan. 2006.

NIEDERST, Jennifer. *Learning Webdesign - a begginer's gide to hml, gaphics and beyond.* Sebastianpool - CA: O'Reilly, 2001. 388 p.

PATRICK, John J. *SQL Fundamentos*. 2. ed. Tradução de Texto Digital. São Paulo: Berkeley, 2002. 710 p.

PAULA FILHO, Wilson de P. *Engenharia de software: fundamentos, métodos e padrões*. 2. ed.. Rio de Janeiro: LTC, 2003. 602 p.

PETERS, James F; PEDRYCZ, Witold. *Engenharia de Software: Teoria e Prática*. Rio de Janeiro: Campus, 2001. 624 p.

PFLEEGER, Shari Lawrence. Witold. *Engenharia de Software: Teoria e Prática*. 2. ed. Tradução Dino Franklin. São Paulo: Prentice Hall, 2004. 535 p.

PHP. Disponível em: <http://www.php.net>. Acesso em 02 de jan. 2006a.

_____. *Php_manual_pt_BR.chm*. Disponível em: <http://br2.php.net/get/php_manual_pt_BR.chm/from/a/mirror>. Acesso em 02 de jan. 2006b.

PRESSMAN, Roger S. *Engenharia de Software*. 5. ed. Tradução Mônica Maria G. Travieso. Rio de Janeiro: McGraw-Hill, 2002. 843 p.

RAPPS, Sandra; WEYUKER, Elaine J. Data Flow Analysis Techniques for Test Data Selection. In: *Proceedings of the 6$^{t}$ International Conference on Software Engineering,* p. 272-278, Toquio, Japan, sep. 1982.

_____. Selecting Software Test Data Using Data Flow Information, *IEEE Transactions on Software Engineering,* vol. SE-11, n. 4, p. 367-375, apr. 1985.

ROCHA, Ana Regina Cavalcante da; MALDONADO, José Carlos; WEBER, Kival Chaves. *Qualidade de Software*. São Paulo: Prentice Hall, 2001. 303 p.

ROCHA, Heloisa Vieira; BARANAUSKAS, Maria Cecília Calani. *Design e Avaliação de Interfaces Humano-Computador*. Campinas: NIED/UNICAMP, 2003. 244 p.

SCAMBRAY, Joel; SHEMA, Mike. *Segurança contra hackers: aplicações Web.* Tradução de Bazán Tecnologia e Lingüística. São Paulo: Futura, 2003. 394 p.

SOMMERVILLE, Ian. *Engenharia de Software*. Tradução André Maurício de Andrade Ribeiro. São Paulo: Addison Wesley, 2003. 592 p.

University of Michigan. *Lecture #1 CIS 400 9/8/99*. Disponível em <http://www.engin.umd.umich.edu/CIS/course.des/cis400/maxim/lectures/lecture1.htm>. Acesso em 02 de jan. 2006a.

_____. *Evolution of the Major Programming Languages*. Disponível em < http://www.engin.umd.umich.edu/CIS/course.des/cis400/maxim/lectures/chp2.htm>. Acesso em 02 de jan. 2006b.

VILELA, Plínio Roberto Souza. *Uma Ferramenta para Auxílio Visual ao Teste e Depuração de Programas*. 1994. 86 f. Dissertação (Mestrado em Engenharia Elétrica) - Faculdade de Engenharia Elétrica da Unicamp. Campinas, 1994.

VERGILIO, Silvia Regina. *Critérios restridos de teste de software: uma contribuição para gerar dados de teste mais eficazes*. 1997. 133 f. Tese (Doutorado em Engenharia Elétrica) - Faculdade de Engenharia Elétrica e de Computação da Unicamp. Campinas, 1997.

WELLING, Luke; THOMSON, Laura. *PHP e MySQL: Desenvolvimento Web*. 2. ed. Tradução de Altair Dias Caldas de Moraes e Cláudio Belleza Dias. Rio de Janeiro: Campus, 2003. 676 p.

WEYUKER, Elaine J. The Cost of Data Flow Testing: An Empirical Study. *IEEE Transactions on Software Engineering,* vol. 16, n. 2, p. 121-128, feb. 1990.

WIKIPEDIA. *PHP*. Disponível em <http://pt.wikipedia.org/wiki/php>. Acesso em 02 de jan. 2006.

WIKIMEDIA. *Wikimedia Foundation*. Disponível em <http://wikimediafoundation.org/wiki/home>. Acesso em 02 de jan. 2006.

WHITTAKER, James A. *How to Break Software: a practical guide to testing: an example-rich explanation of how to effectively test software that anyone can undestand and use immediately*. USA: Pearson EducationAddison Wesley, 2003. 178 p.

## REFERÊNCIAS CONSULTADAS

APACHE Software Foundation. *Apache HTTP Server Version 1.3 - Documentaiton*. Disponível em: <http://httpd.apache.org/docs/1.3/>. Acesso em 02 de jan. 2006.

COMER, Douglas E. *Redes de Computadores e Internet*. 2. ed. Tradução de Marinho Barcellos. Porto Alegre: Bookman, 2001. 552 p.

DeMILLO, Richard A.; McCRACKEN, W. Michael; MARTIN, R.J.; PASSAFIUME, John F. *Software Test and Evaluation*. Menlo Park, California: The Benjamin/Cummings Publishing Company, 1987. 537 p.

DEMARCO, Tom. *Structured analysis and system specification*. New York: Yourdon Inc, 1978. 352 p.

GONÇALVES, Klausner Vieira. *Testes de Software em aplicação de Banco de Dados Relacional*. 2003. 59 f. Dissertação (Mestrado em Engenharia Elétrica e Computação) - Instituto de Computação da Unicamp. Campinas, 2003.

IEEE. *Guide to the software engineering body of knowledge: 2004 version*. Disponível em: <http://www.swebok.org/ironman/pdf/SWEBOK_Guide_2004.pdf >. Acesso em 02 de jan. 2006.

MENASCÉ, Daniel A. ; ALMEIDA, Virgilio A.F. *Capacity planning for Web performance: metrics, models, and methods.* Upper Saddle River: Pearson Education, 1998. 321 p.

McCOMB, Gordon. *Javascript Sourcebook*. Tradução Altair Dias Caldas de Moraes. São Paulo: Makron Books, 1997, 736 p.

HETZEL, William. *Guia Completo ao teste de software*. Rio de Janeiro: Campus, 1987. 206 p.

MYSQL. Disponível em: <http://www.mysql.com>. Acesso em 02 de jan. 2006.

NIEDERAUER, Juliano. *PHP para quem conhece PHP*. São Paulo: Novatec, 2004. 480 p.

O'REILLY. *Learning the Unix operating System*. 5 ed. Disponível em: <http://iq140.formymind.com/HTML/study/ebook2/lunix/index.htm>. Acesso em 02 de jan. 2006.

_____. *Sed and Awk*. 2. ed. Disponível em: <http://iq140.formymind.com/HTML/study/ebook2/sedawk/index.htm>. Acesso em 02 de jan. 2006.

_____. *Unix in a Nutshell*. 3. ed. Disponível em: <http://iq140.formymind.com/HTML/study/ebook2/unixnut/index.htm> . Acesso em 02 de jan. 2006.

_____. *Unix Power Tools*. 3. ed. Disponível em: <http://iq140.formymind.com/HTML/study/ebook2/upt/index.htm>. Acesso em 02 de jan. 2006.

RAMALHO, José Antonio Alves. *HTML avançado*. São Paulo: Makron Books, 1997. 659 p.

RED HAT. *Red Hat Linux Manuals* .Disponível em: <http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/pdf/rhl-ig-x86-en-9.pdf>. Acesso em 02 de jan. 2006.

SHAY, William A. *Sistemas Operacionais*. Tradução Mário Moro Fechio. São Paulo: Makron Books, 1996. 758 p.

SILBERSCHATZ, Avi; GALVIN, Peter Baer; GAGNE, Greg. *Operating System Concepts.* 6. ed. New York: John Wiley & Sons, 2002. 887 p.

SOARES, Wallace. *PHP5: Conceitos, Programação e Integração com Banco de Dados*. São Paulo: Éricka, 2004. 523 p.

TANENBAUM, Andrew S. *Sistemas Operacionais Modernos***.** 2. ed. Tradução Ronaldo A.L. Gonçalves e Luis A. Consularo. São Paulo: Pearson Prentice Hall, 2003. 695 p.

TANENBAUM, Andrew S.; WOODHULL, Albert S., *Operating Systems: design and implementation*. 2. ed. New Jersey: Prentice-Hall, 1997. 939 p.

TOSCANI, Simão S; OLIVEIRA, Rômulo S. de; CARISSIMI, Alexandre da S. *Sistemas Operacionais e Programação Concorrente.* Porto Alegre: Sagra Luzzatto, 2003. 247 p.

VINCENZI, Auri Marcelo Rizzo. *Subsídios para o estabelecimento de Estratégias de Testes baseados na técnica de Mutação*. 1998. 149 f. Dissertação (Mestrado em Ciência da Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação ICMC/USP. São Carlos, 1998.

VERGILIO, Silvia Regina. *Caminhos não executáveis: Caracterização, Previsão e Determinação para Suporte ao Teste de Programas*. 1992. 172 f. Tese (Mestrado em Engenharia Elétrica) - Faculdade de Engenharia Elétrica e de Computação da Unicamp. Campinas, 1992.

VMWARE. Disponível em: <http://www.vmware.com>. Acesso em 02 de jan. 2006.

WORLD WIDE WEB CONSORTIUM. Disponível em: <http://www.w3c.org>. Acesso em 02 de jan. 2006.

_____. *HTML 4.01 Specification*. Disponível em: <http://www.w3.org/TR/HTML401/>. Acesso em 02 de jan. 2006

# APÊNDICE A

Código fonte do programa Cal.c:

```
#ifndef lint
static char sccsid[] = "@(#)cal.c       4.4 (Berkeley) 87/05/28";
#endif

#include <sys/types.h>
#include <time.h>
#include <stdio.h>

char    dayw[] = {
        " S  M Tu  W Th  F  S"
};
char    *smon[]= {
        "January", "February", "March", "April",
        "May", "June", "July", "August",
        "September", "October", "November", "December",
};
char    string[432];
main(argc, argv)
char *argv[];
{
        register y, i, j;
        int m;

        if(argc == 2)
                goto xlong;
        /*
         * print out just month
         */
        if(argc < 2) {                  /* current month */
                time_t t;
                struct tm *tm;

                t = time(0);
                tm = localtime(&t);
                m = tm->tm_mon + 1;
                y = tm->tm_year + 1900;
        } else {
                m = atoi(argv[1]);
                if(m<1 || m>12) {
                        fprintf(stderr, "cal: %s: Bad month.\n", argv[1]);
                        exit(1);
                }
                y = atoi(argv[2]);
                if(y<1 || y>9999) {
```

```
                fprintf(stderr, "cal: %s: Bad year.\n", argv[2]);
                exit(2);
            }
        }
        printf("   %s %u\n", smon[m-1], y);
        printf("%s\n", dayw);
        cal(m, y, string, 24);
        for(i=0; i<6*24; i+=24)
            pstr(string+i, 24);
        exit(0);

xlong:
        /*
         * print out complete year
         */
        y = atoi(argv[1]);
        if(y<1 || y>9999) {
            fprintf(stderr, "cal: %s: Bad year.\n", argv[1]);
            exit(2);
        }
        printf("\n\n\n");
        printf("                              %u\n", y);
        printf("\n");
        for(i=0; i<12; i+=3) {
            for(j=0; j<6*72; j++)
                string[j] = '\0';
            printf("         %.3s", smon[i]);
            printf("                       %.3s", smon[i+1]);
            printf("                      %.3s\n", smon[i+2]);
            printf("%s   %s   %s\n", dayw, dayw, dayw);
            cal(i+1, y, string, 72);
            cal(i+2, y, string+23, 72);
            cal(i+3, y, string+46, 72);
            for(j=0; j<6*72; j+=72)
                pstr(string+j, 72);
        }
        printf("\n\n\n");
        exit(0);
}

pstr(str, n)
char *str;
{
        register i;
        register char *s;

        s = str;
        i = n;
        while(i--)
```

```
                if(*s++ == '\0')
                        s[-1] = ' ';
        i = n+1;
        while(i--)
                if(*--s != ' ')
                        break;
        s[1] = '\0';
        printf("%s\n", str);
}

char    mon[] = {
        0,
        31, 29, 31, 30,
        31, 30, 31, 31,
        30, 31, 30, 31,
};

cal(m, y, p, w)
char *p;
{
        register d, i;
        register char *s;

        s = p;
        d = jan1(y);
        mon[2] = 29;
        mon[9] = 30;

        switch((jan1(y+1)+7-d)%7) {

        /*
         *      non-leap year
         */
        case 1:
                mon[2] = 28;
                break;

        /*
         *      1752
         */
        default:
                mon[9] = 19;
                break;

        /*
         *      leap year
         */
        case 2:
                ;
```

```
	}
	for(i=1; i<m; i++)
		d += mon[i];
	d %= 7;
	s += 3*d;
	for(i=1; i<=mon[m]; i++) {
		if(i==3 && mon[m]==19) {
			i += 11;
			mon[m] += 11;
		}
		if(i > 9)
			*s = i/10+'0';
		s++;
		*s++ = i%10+'0';
		s++;
		if(++d == 7) {
			d = 0;
			s = p+w;
			p = s;
		}
	}
}

/*
 *	return day of the week
 *	of jan 1 of given year
 */

jan1(yr)
{
	register y, d;

/*
 *	normal gregorian calendar
 *	one extra day per four years
 */

	y = yr;
	d = 4+y+(y+3)/4;

/*
 *	julian calendar
 *	regular gregorian
 *	less three days per 400
 */

	if(y > 1800) {
		d -= (y-1701)/100;
		d += (y-1601)/400;
```

```
        }

/*
 *      great calendar changeover instant
 */

        if(y > 1752)
                d += 3;

        return(d%7);
}
```

## APÊNDICE B

Modificações que antecedem os scripts newpoketool e newpokeaval, que foram renomeados para webpoketool e webpokeaval, respectivamente:

```
#
pathmunge () {
        if ! echo $PATH | /bin/egrep -q "(^|:)$1($|:)" ; then
          if [ "$2" = "after" ] ; then
            PATH=$PATH:$1
          else
            PATH=$1:$PATH
          fi
        fi
}
# Path manipulation
if [ `id -u` = 0 ]; then
        pathmunge /sbin
        pathmunge /usr/sbin
        pathmunge /usr/local/sbin
fi
pathmunge /usr/X11R6/bin after
unset pathmunge
# No core files by default
ulimit -S -c 0 > /dev/null 2>&1
USER="`id -un`"
LOGNAME=$USER
MAIL="/var/spool/mail/$USER"
HOSTNAME=`/bin/hostname`
HISTSIZE=1000
if [ -z "$INPUTRC" -a ! -f "$HOME/.inputrc" ]; then
    INPUTRC=/etc/inputrc
fi
```

```
#POKETOOL:
NEWPOKE=/webpoketool/bin
NEWLITABS=/webpoketool/bin/litabs
NEWPOKETABS=/webpoketool/bin/poketabs
PATH=$PATH:/webpoketool/bin:/usr/bin
export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUTRC
NEWPOKE NEWLITABS NEWPOKETABS
echo $PATH
echo $NEWPOKE
echo $NEWLITABS
echo $NEWPOKETABS
for i in /etc/profile.d/*.sh ; do
    if [ -r "$i" ]; then
        . $i
    fi
done
unset i
#################################################################
```