Instituto de Ciências Matemáticas de São Carlos

ISSN-0109-2569

Proteum
Manual do Usuário
versão 1.1 - C

MÁRCIO EDUARDO DELAMARO
JOSÉ CARLOS MALDONADO

Nº 23

RELATÓRIOS TÉCNICOS DO ICMSC

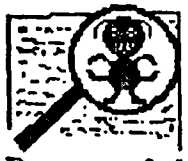
SÃO CARLOS
Out. / 1994

SYSNO 872259
DATA / /
ICMC - SBAB

Proteum - Manual do Usuário

Versão 1.1 - C
09/09/94

Márcio Eduardo Delamaro[*]
José Carlos Maldonado[#]



[*] Apoio financeiro da CAPES.
[#] Apoio financeiro do CNPq.

Conteúdo.

1. - Introdução. 1
2. - Ambiente 1
3. - Sessões de Teste 2
    3.1. - Criando um Novo Teste 3
    3.2. - Retomando um Teste 6
    3.3. - Salvando uma Sessão 7
4. - Abandonar o Sistema 8
5. - O Conjunto de Casos de Teste 9
    5.1. - Incluindo Casos de Teste 10
    5.2. - Consultando o Conjunto de Casos de Teste 12
    5.3. - Eliminando Casos de Teste 14
    5.4. - Importando Casos de Teste 15
        5.4.1. - Casos de Teste do Proteum 15
        5.4.2. - Casos de Teste da POKE-TOOL 16
        5.4.3. - Casos de Teste de Arquivos ASCII 16
6. - Trabalhando com os Mutantes 17
    6.1. - Gerando os Mutantes 17
    6.2. - Executando os Mutantes 19
    6.3. - Visualizando os Mutantes 21
    6.4. - Selecionando Sub-conjuntos de Mutantes 23
        6.4.1. - Seleção por Operador de Mutação 23
        6.4.2. - Seleção por Blocos 24
7. - Verificando o Andamento do Teste 24
8. - Criando Relatórios 26
    8.1. - Criando Relatórios sobre Casos de Teste 26
9. - Configurando o Proteum 29
10. - Diferenças em Relação à Versão 1.0 - C 30

Lista de Figuras.

3.1. - Menu Principal 2
3.2. - Iniciando uma Sessão de Teste 2
3.3. - Criando um Novo Teste 3
3.4. - Programa myprog.c 4
3.5. - Confirmando a Criação de Novo Teste 6
3.6. - Retomando um Teste 7
3.7. - Escolhendo entre Salvar ou Abortar Sessão 8
4.1. - Abandonar o Proteum 9
5.1. - Manipulando Casos de Teste 10
5.2. - Fornecendo os Parâmetros Iniciais do Caso de Teste 11
5.3. - Incluindo um Caso de Teste 11
5.4. - Confirmando Caso de Teste 12
5.5. - Visualizando um Caso de Teste 13
5.6. - Habilitando ou Desabilitando um Caso de Teste 14
5.7. - Eliminando Casos de Teste 14
5.8. - Importando Casos de Teste 15
5.9. - Importando Casos de Teste do Proteum 16
5.10. - Importando Casos de Teste da POKE-TOOL 16
5.11. - Importando Casos de Teste de Arquivos ASCII 17
6.1. - Trabalhando com os Mutantes 17
6.2. - Selecionando Classes de Operadores 18
6.3. - Selecionando Operadores Individualmente 18
6.4. - Gerando Mais Mutantes 19
6.5. - Executando os Mutantes 20
6.6. - Analisando os Mutantes 22
6.7. - Selecionando Mutantes por Bloco de Programa 24
7.1. - Verificando o Andamento do Teste 25
8.1. - Selecionando Relatório Simples 26
8.2. - Relatório Simples 27
8.3. - Relatório Completo 28
9.1. - Configurando o Ambiente 29

1. - Introdução.

Este manual apresenta as principais características operacionais do Proteum (**Program Testing Using Mutants**), ferramenta de teste baseada no critério **Análise de Mutantes** [DEM78].

O proteum é uma ferramenta multilinguagem, ou seja, pode ser configurada para o teste de programas escritos em diferentes linguagens. Este manual refere-se especificamente à versão **1.1 - C**, ou seja, à versão configurada para a linguagem C, implementada para rodar em estações SUN, sob o ambiente OPENWINDOWS.

2. - Ambiente

No diretório em que o Proteum estiver instalado, por exemplo *usr bin proteum*, devem estar presentes os seguintes arquivos:

Executáveis:

proteum
tcase_ex (monitora a execução de casos de teste)

Tabelas:

chave c (tabela de palavras chaves do C)
tabsin.c (tabela do grafo sintático do C)

O Proteum deve ser chamado a partir de uma janela de comandos (console ou cmdtool) dentro do Openwindows. Antes de executar-se o Proteum é necessário indicar em que diretório estão esses arquivos. Isto é feito atribuindo-se à variável de ambiente *PROTEUMHOME* o nome do diretório onde o Proteum está instalado. Para isso deve-se executar o seguinte comando do UNIX, supondo-se que o diretório seja */usr/bin/proteum*.

```
setenv PROTEUMHOME /usr/bin/proteum
```

Esse comando pode ser incluído no arquivo *.cshrc* no diretório *HOME* do usuário. Assim, ao entrar no ambiente Openwindows, o usuário já terá essa variável de sistema com o valor assinalado em todas as janelas de comandos..

Depois disso, basta executar o arquivo */usr/bin/proteum/proteu*.

3. - Sessões de Teste

Iniciada a execução do Proteum, é apresentada ao usuário a tela da Figura 3.1.

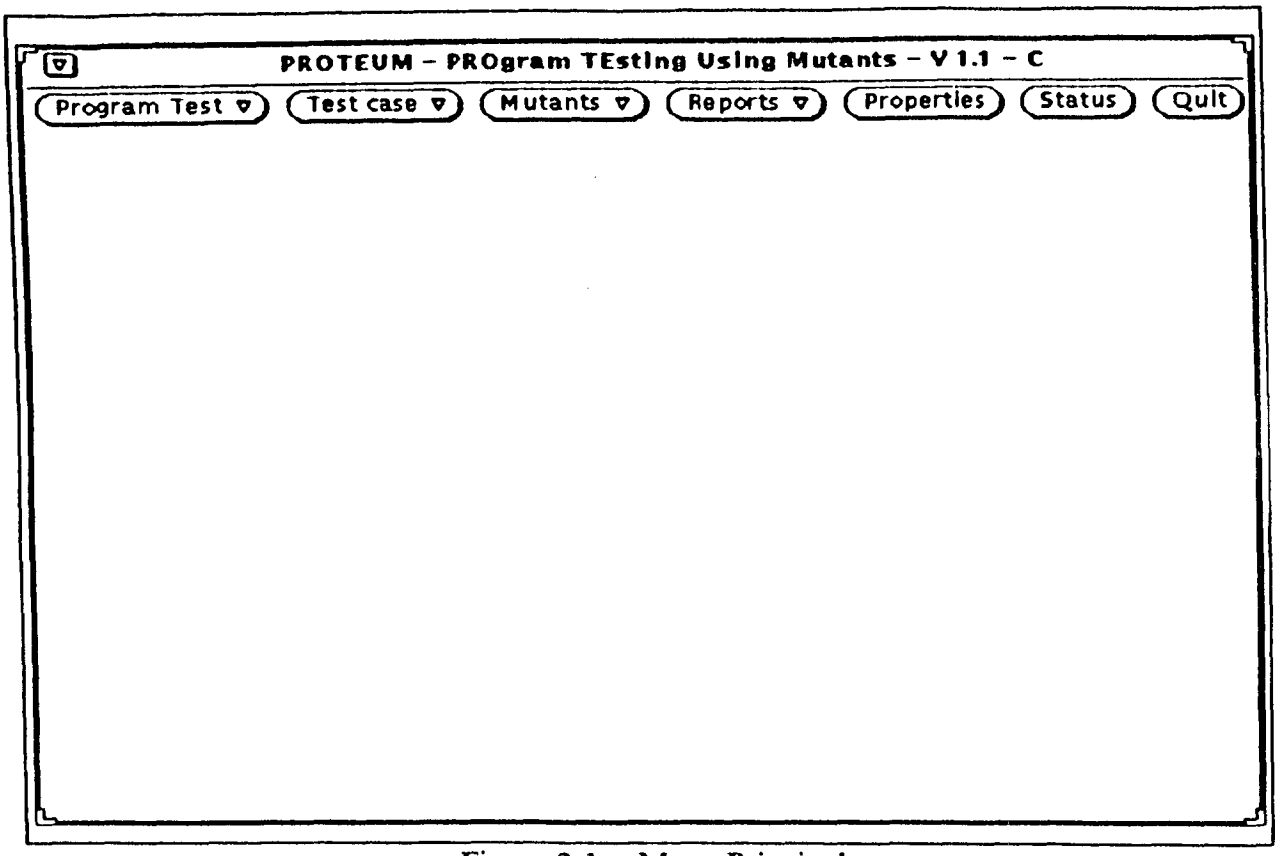


Figura 3.1. - Menu Principal

A operação do Proteum é baseada em **sessões de teste**, ou seja, o usuário pode iniciar o teste de um programa, interrompê-lo a qualquer momento e posteriormente retomá-lo do ponto em que foi abandonado. Então, o primeiro passo é informar ao Proteum qual o teste que se deseja conduzir, quer seja um novo teste ou um teste já iniciado anteriormente e interrompido.

Isto é feito escolhendo-se o botão *Program Test* do menu principal. Assim o usuário pode escolher iniciar um novo teste, através da opção *New* mostrada na Figura 3.2., ou retomar um teste já existente, através da opção *Load* (Seção 3.2).

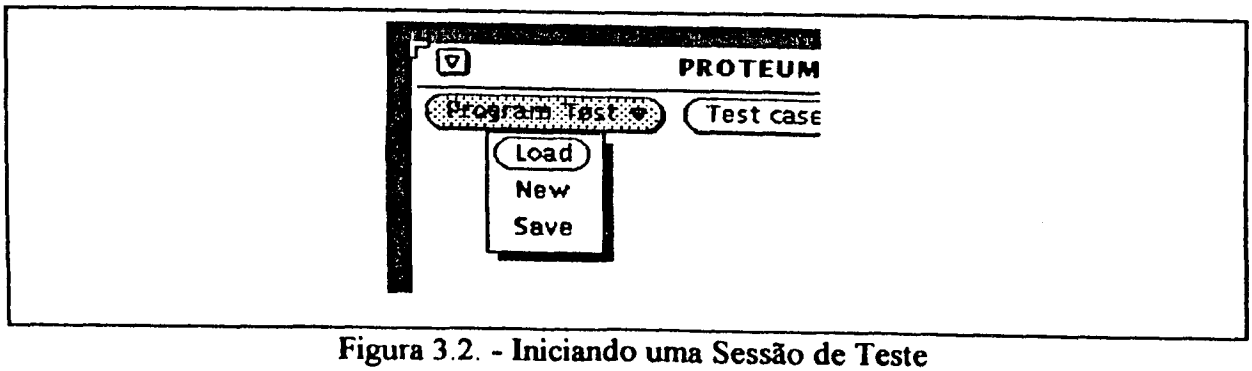


Figura 3.2. - Iniciando uma Sessão de Teste

3.1. - Criando um Novo Teste

Ao selecionar a opção *New*, o usuário recebe um painel onde deve fornecer as características do teste que deseja criar. São informações gerais e que definem, entre outras coisas, qual programa deseja-se testar. A Figura 3.3. mostra como é esse painel. A seguir são explicados os campos desse painel que devem ser preenchidos pelo usuário.

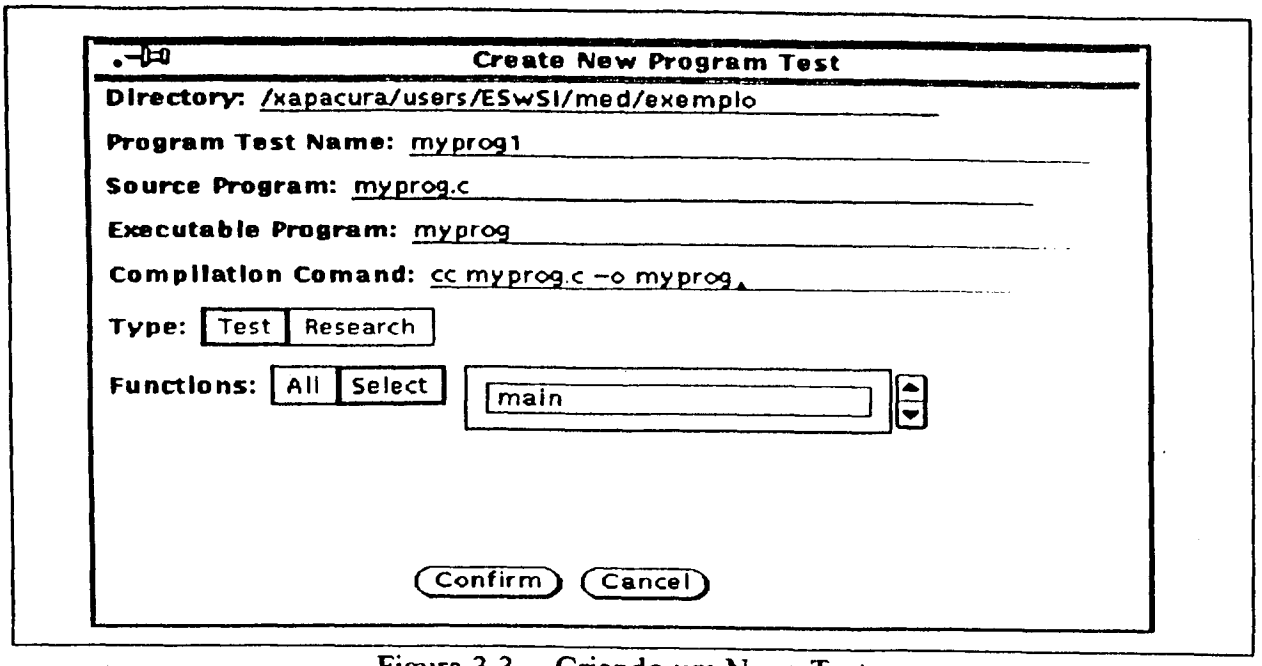


Figura 3.3. - Criando um Novo Teste

Directory:

Neste campo o usuário deve fornecer o diretório onde será montado o teste. Neste diretório deve estar o programa fonte - inclusive arquivos de "include" - e o executável que se deseja testar. É nesse diretório, que pode-se chamar de "diretório de trabalho", que o Proteum cria os arquivos de trabalho, descritos a seguir.

Caso o diretório não exista, o Proteum indica o erro e não aceita a escolha do usuário.

Program Test Name:

Deve ser fornecido um nome pelo qual o teste será identificado. Pode-se imaginar que o próprio nome do programa fonte ou do executável seja suficiente para identificar um teste. Isso nem sempre é verdade pois algumas vezes deseja-se testar um mesmo programa de maneiras diferentes, sendo assim necessário batizar com nomes diferentes cada um desses diferentes testes.

Por exemplo, se deseja-se testar o programa *myprog*, criado a partir do fonte *myprog.c* (mostrado na Figura 3.4.), pode-se dar um nome ao teste como *myprog1*. Neste

caso, os arquivos de trabalho criados pelo Proteum, no diretório que o usuário escolheu, são:

myprog1.PTM

Esse arquivo contém informações gerais sobre o teste. Basicamente, são os dados que o usuário fornece na criação do teste.

myprog1.TCS
myprog1.IOL

Esses dois arquivos guardam, em conjunto, os casos de teste que o usuário fornece para testar o programa.

myprog1.IND
myprog1.MUT

Respectivamente, arquivo de índices dos descritores de mutantes e arquivo com os descritores dos mutantes gerados.

myprog1.LI

Arquivo com uma representação simplificada do arquivo fonte - chamada de *Linguagem Intermediária (LI)*, e que auxilia na geração dos mutantes. Esse arquivo é gerado a cada sessão de trabalho. Portanto, ao contrário do que acontece com os outros arquivos de trabalho, se esse arquivo sofrer qualquer dano isto não representa nenhum problema pois ele será reconstituído ao iniciar-se uma nova sessão de teste.

myprog1.c

Arquivo gerado pelo pré-processador C do UNIX, chamado *cpp*. É sobre esse arquivo que o Proteum trabalha, e não sobre o arquivo fonte fornecido pelo usuário. Esse arquivo também é gerado no início de cada sessão de teste.

```
#include        <stdio.h>

main()
{
int     a, b;

  scanf("%d %d", &a, &b);
  if (a > b)
        printf("\n%d eh maior que %d", a, b);
  esle
  if ( a < b)
        printf("\n%d eh menor que %d", a, b);
  else
        printf("\n%d eh igual a %d", a, b);
  return 0;
}
```

Figura 3.4. - Programa *myprog.c*

Source Program:

Nome do arquivo do programa fonte. Deve sempre ter o sufixo *.c*. Como já foi dito, deve estar no diretório escolhido pelo usuário. Caso não exista um arquivo com esse nome, o Proteum avisa ao usuário e não aceita a escolha. No caso, deve-se fornecer o nome *myprog.c* .

Executable Program:

Nome do programa executável (*myprog*), que deve estar no mesmo diretório. Caso não exista, o Proteum apresenta mensagem de erro.

Compilation Command:

Nesse campo o usuário deve fornecer o comando do sistema usado para criar o programa executável a partir do programa fonte. Isso permite que o usuário utilize qualquer compilador e diversas opções de compilação e linkedição. A restrição que se faz é que tanto o nome do arquivo fonte quanto do executável devem aparecer no comando; caso isso não ocorra, o Proteum não aceita e sinaliza um erro.

O Proteum utiliza esse campo do painel para poder criar programas executáveis a partir dos mutantes que cria. Para isso toda ocorrência do nome do arquivo do programa fonte em teste é substituído pelo nome do fonte do mutante e analogamente para o nome do executável. Assim se o usuário escolheu o comando *cc myprog.c -o myprog* como comando de compilação, o Proteum interpreta-o como *cc <arquivo fonte> -o <arquivo executável>*.

Podem ser incluídas opções de linkedição e outros arquivos, necessários para a construção do programa executável. Por exemplo: *cc myprog.c yourprog.o -lmath -o myprog* cria o executável *myprog* compilando o *myprog.c* e linkeditando-o com o programa objeto *yourprog.o* e com a biblioteca *libmath.a*

Type:

O usuário pode selecionar para esse campo uma das opções que determinam o tipo de comportamento que o Proteum terá para o teste: *Test* ou *Research*. A primeira faz com que a execução e eliminação dos mutantes seja feita da maneira convencional, ou seja: executa-se cada mutante com os casos de teste até que um deles o mate ou até que todos os casos de teste tenham sido utilizados. Escolhendo o tipo *Research*, cada mutante é executado com todos os casos de teste, independentemente de já estar morto ou não. Essa abordagem permite o levantamento de dados sobre o conjunto de casos de teste e mesmo sobre a utilização do Proteum e do critério Análise de Mutantes.

Functions:

Pode-se selecionar quais funcões ou sub-rotinas devem ser testadas ou, em outras palavras, quais trechos do arquivo fonte devem sofrer mutações e quais não devem. O usuário pode selecionar a opção *All*, fazendo com que todo o programa seja testado ou escolher quais funções devem ser testadas selecionando a opção *Select*. Neste caso, o Proteum consulta o arquivo fonte e apresenta uma lista das funções que compõem o programa e permite ao usuário escolher quais deseja testar.

Depois de preenchidos todos os campos, basta selecionar o botão *Confirm* para que a operação seja executada. Durante a leitura do painel o usuário pode desistir da operação a qualquer momento, selecionando o botão *Cancel* ou acionando duas vezes o pino no canto superior esquerdo do painel.

Caso já exista no diretório escolhido um teste com o mesmo nome, o Proteum avisa o usuário que pode então decidir se deseja sobrepor o teste, ou seja, continuar com a criação, perdendo o teste que já existia ou se deseja desistir da operação. Tal situação é mostrada na Figura 3.5.

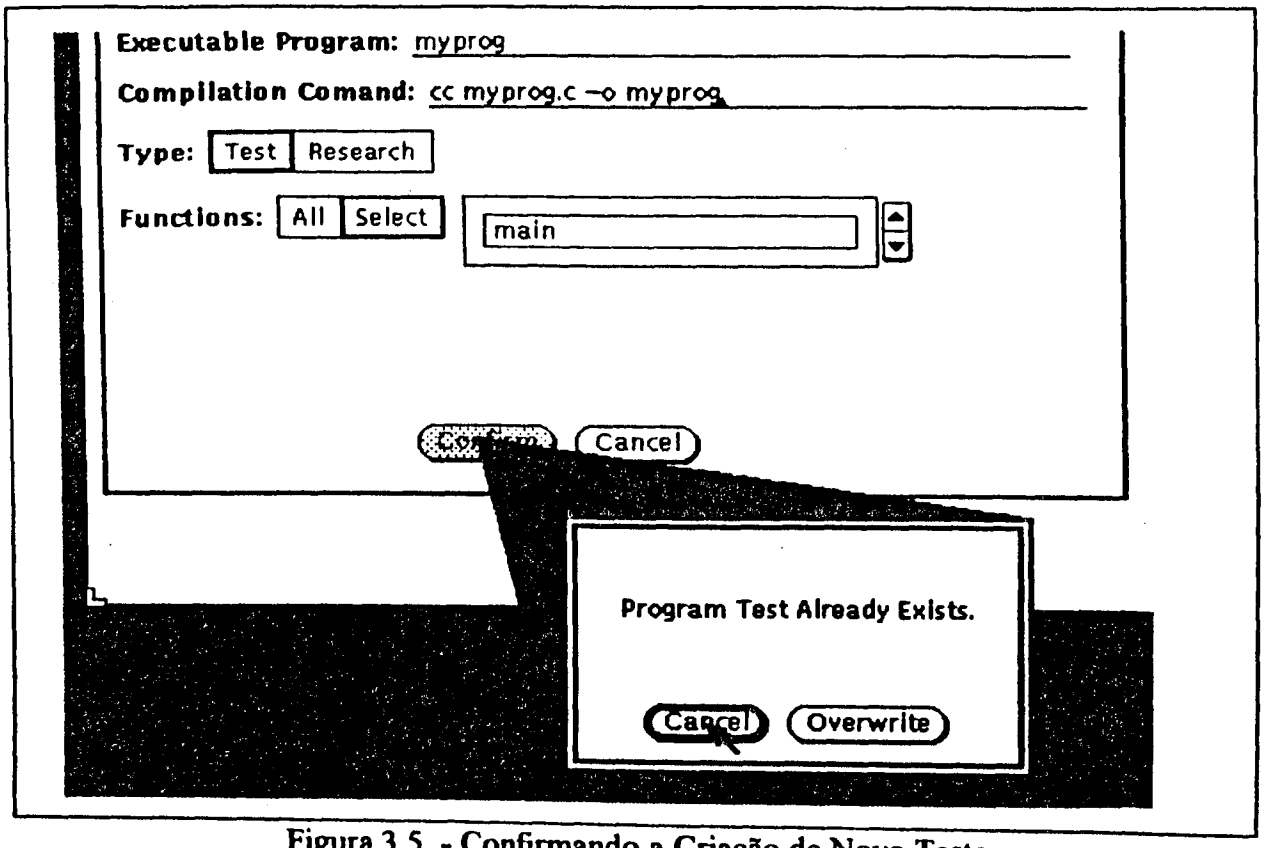


Figura 3.5. - Confirmando a Criação de Novo Teste

3.2. - Retomando um Teste

Para voltar a um teste já iniciado anteriormente o usuário deve utilizar a opção *Load*. Nesse caso, conforme mostra a Figura 3.6., deve-se informar o diretório onde estão os arquivos de trabalho e o programa a ser testado e fornecer o nome dado ao teste que se deseja carregar.

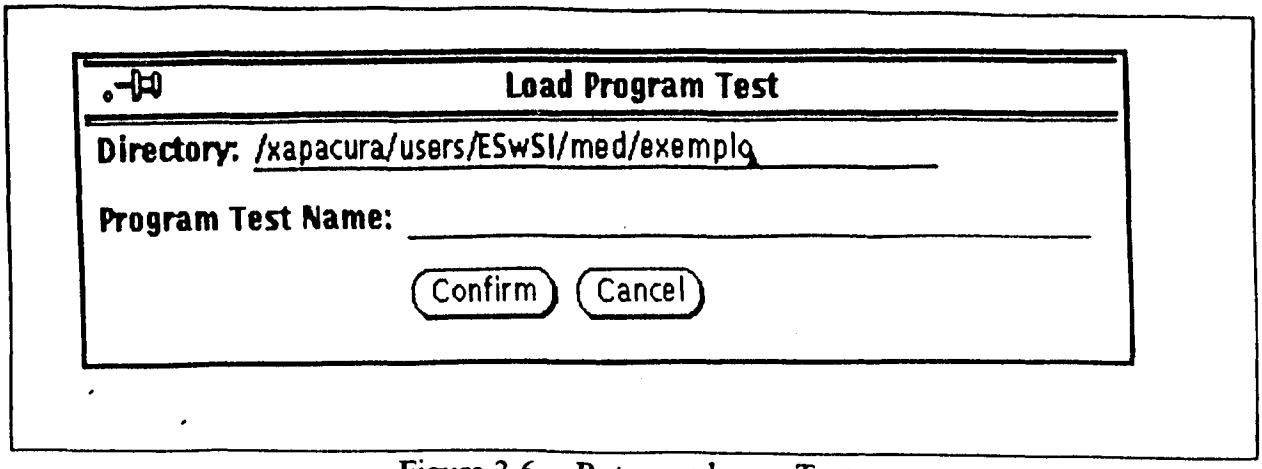


Figura 3.6. - Retomando um Teste

Ao carregar um teste, o Proteum faz uma cópia de todos os arquivos de trabalho, criados pelo Proteum, referentes ao teste. Isso permite que uma nova sessão sobre um teste já iniciado anteriormente possa ser abandonada sem que se destrua o estado anterior do mesmo, caso o usuário não queira guardar os resultados obtidos na sessão corrente.

Quando o usuário escolhe retomar ou criar um teste e já existe um teste carregado, o Proteum encerra o teste que estava carregado, permitindo ao usuário escolher se salva ou não as operações efetuadas durante a sessão (Figura 3.7.). Caso a opção escolhida seja *Abort*, a sessão de teste é desfeita copiando-se de volta os arquivos de "back-up" criados no início da sessão. Já com a opção *Save*, as operações efetuadas são efetivadas e o teste que estava carregado é encerrado. A seleção do botão *Cancel* faz com que se retorne normalmente ao teste que já estava sendo conduzido.

As cópias estão no mesmo diretório dos arquivos de trabalho e têm os mesmo nomes destes, trocando-se a última letra do sufixo por um "%", ou seja:

myprog.PT%
myprog.TC%
myprog.IO%
myprog.IN%
myprog.MU%

3.3. - Salvando uma Sessão

Durante uma sessão de teste pode-se salvar as operações realizadas, escolhendo para isso a opção *Save* do menu mostrado na Figura 3.2..

Nessa operação o Proteum copia novamente os arquivos de trabalho, atualizados, sobre os arquivos de "back-up". Assim, mesmo que, ao se abandonar o teste, escolha-se a opção *Abort*, garante-se que as operações realizadas até o momento do "salvar", foram efetivadas.

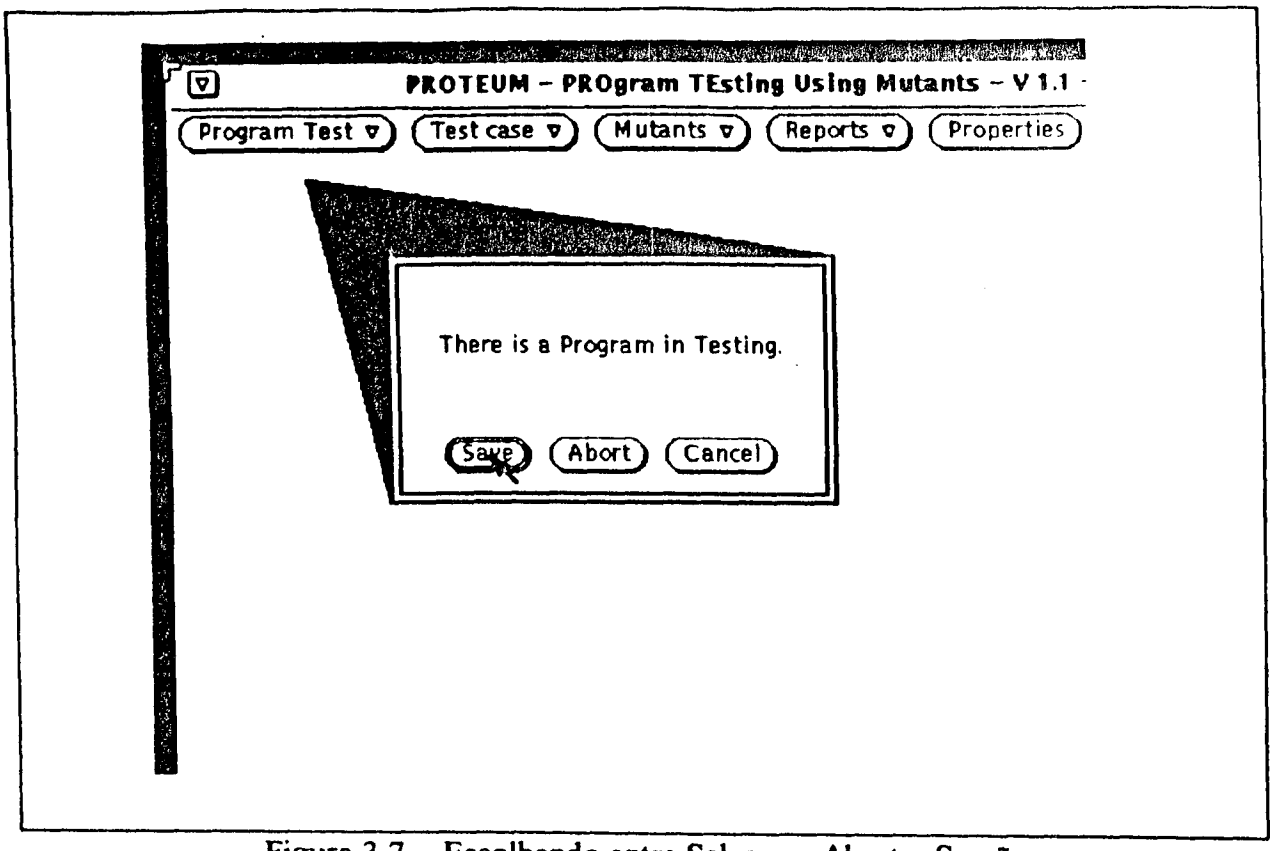


Figura 3.7. - Escolhendo entre Salvar ou Abortar Sessão

4. - Abandonar o Sistema

Para sair do Proteum pode-se selecionar o botão *Quit* no menu principal ou no menu da janela principal, mostrado na Figura 4.1.. Em ambos os casos, o Proteum verifica se existe um teste em andamento e em caso afirmativo mostra novamente o menu *Save/Abort/Cancel*.

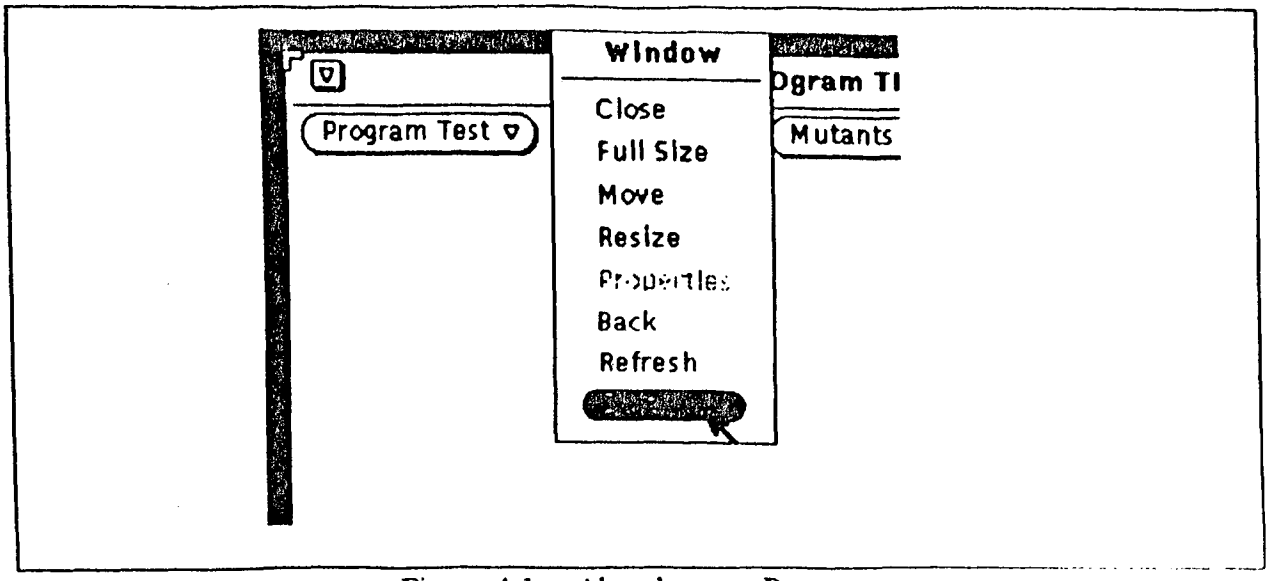


Figura 4.1. - Abandonar o Proteum

Algumas operações não podem ser realizadas enquanto outras ainda estão acontecendo. Por exemplo, não é possível sair do Proteum enquanto estão sendo executados os mutantes. O procedimento correto nesse caso é interromper a execução dos mutantes, como será visto adiante, e depois então abandonar o ambiente.

Além desta, outras combinações não são permitidas. Portanto, se em dado instante algum menu "não responde" à ativação deve ser porque existe alguma operação pendente que deve ser concluída antes de uma nova operação ser iniciada.

5. - O Conjunto de Casos de Teste

Basicamente, o que caracteriza um teste, além do programa sendo testado, é uma base de dados que guarda informações sobre os mutantes gerados e seus respectivos estados (morto, vivo, etc), e sobre os casos de teste que devem ser empregados no teste.

Para manipular o conjunto de casos de teste, o Proteum possui 6 operações que podem ser escolhidas através do menu mostrado na Figura 5.1.: *Add* para incluir casos de teste; *View*, para consultar os casos de testes existente; *Delete*, para excluir casos de teste e *Import* para copiar casos de teste já definidos em outros testes ou outras ferramentas.

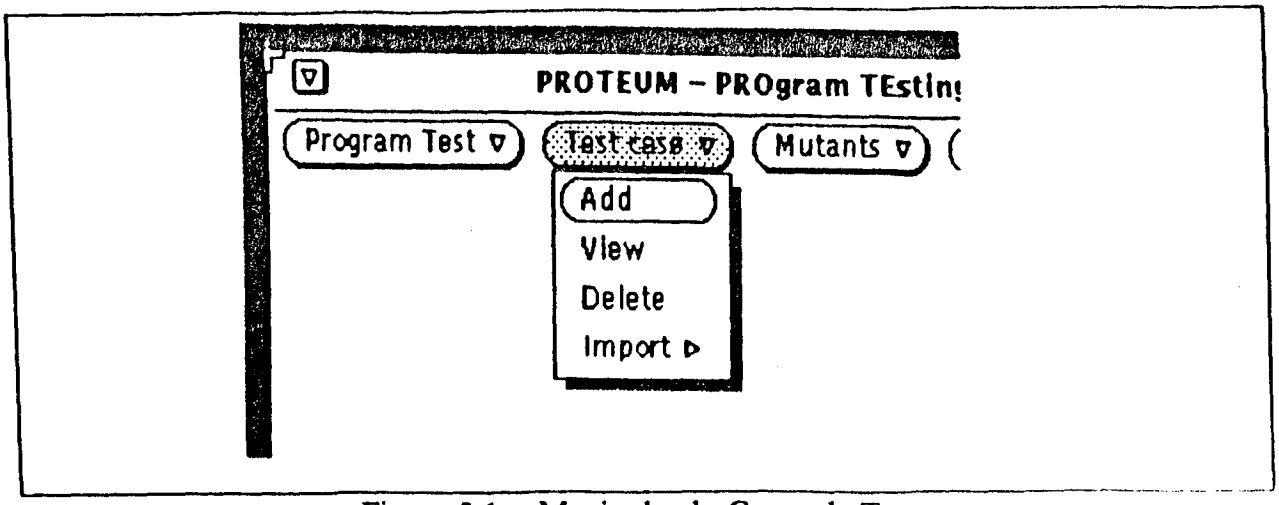


Figura 5.1. - Manipulando Casos de Teste

5.1. - Incluindo Casos de Teste

A inclusão de casos de teste no Proteum é feita de maneira bastante simples para o usuário. Primeiramente, ele fornece os parâmetros do programa, ou seja, os dados que são fornecidos ao programa quando este é chamado na linha de comando do sistema operacional. Por exemplo, no comando abaixo, o programa *cd* está sendo invocado com o parâmetro */usr/bin*.

```
cd /usr/bin
```

Uma vez fornecidos os parâmetros iniciais, o Proteum inicia a execução do programa em teste e o usuário interage com este, fornecendo os dados necessários em tempo de execução e vendo, na tela, os resultados produzidos pelo programa. O Proteum captura e guarda os dados digitados e as saídas obtidas, caracterizando assim o caso de teste. Somente entradas tipos texto, ou seja, os dados digitados no teclado são considerados como entrada. Entradas via mouse e saídas/entradas gráficas não são tratadas pelo Proteum.

Cabe ao usuário avaliar se as saídas produzidas são as esperadas. Caso não sejam, um erro foi revelado no programa e deve ser corrigido.

Inicialmente, no processo descrito acima, o Proteum apresenta um painel onde o testador deve fornecer os parâmetros iniciais, como mostrado na Figura 5.2. Se o programa, ou um caso de teste específico, não necessita de parâmetros, este painel pode ser deixado em branco, sendo confirmado apenas teclando-se *<return>*.

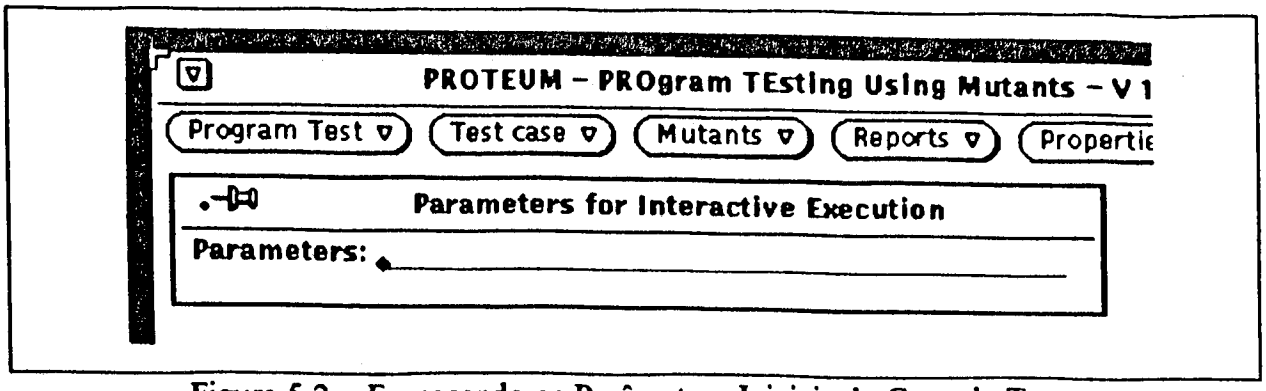


Figura 5.2. - Fornecendo os Parâmetros Iniciais do Caso de Teste

Em seguida é iniciada a execução numa janela que simula um terminal, onde o usuário interage com o programa. Nessa janela é apresentado o comando correspondente à execução do caso de teste e um lembrete que para abortar a inclusão do caso de teste o testador pode precionar as teclas <*CTRL*> e <C> simultaneamente

A Figura 5.3. mostra um exemplo da inclusão de um caso de teste. Nota-se, olhando o comando correspondente, que não fói fornecido nenhum parâmetro inicial. Em seguida, o programa inicia a execução e fica esperando que o usuário digite um número. Nesse ponto, deve-se digitar os dados solicitados para que a execução prossiga

Após fornecer todos os dados, ao final da execução, o usuário pode avaliar se o resultado apresentado está correto ou não, e escolher entre incluir ou não o caso de teste. Para isso, o Proteum pede a confirmação, como mostra a Figura 5.4.

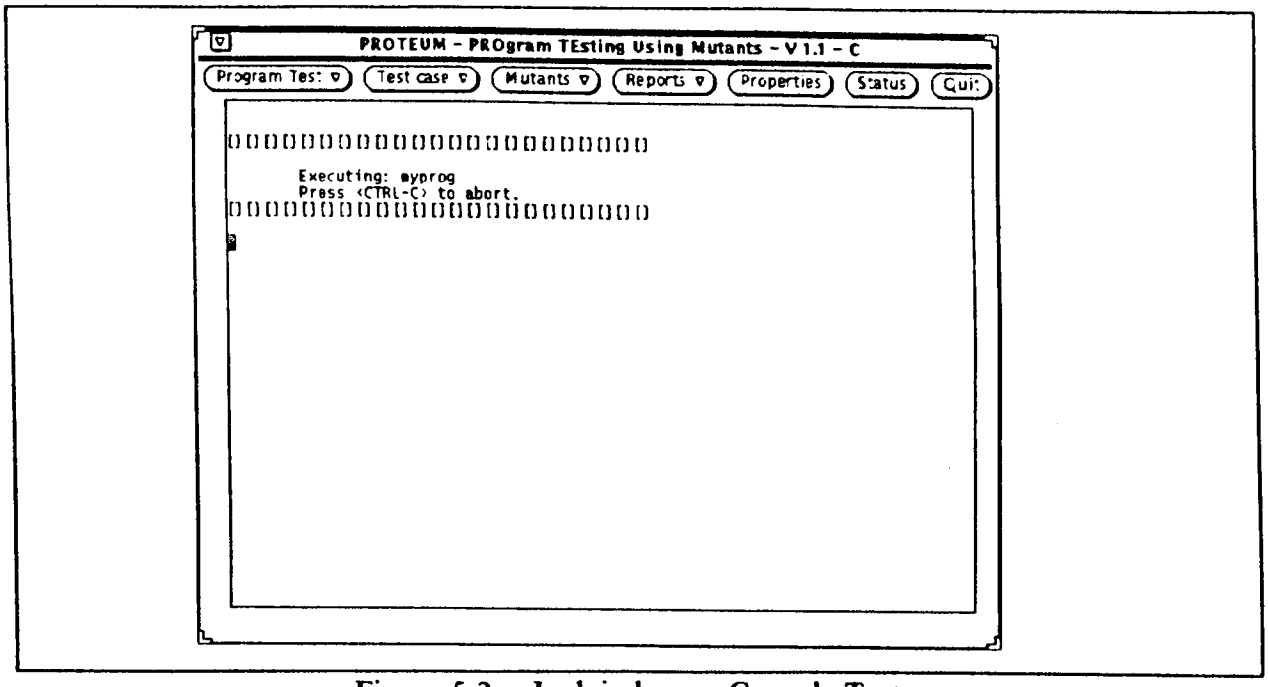


Figura 5.3. - Incluindo um Caso de Teste

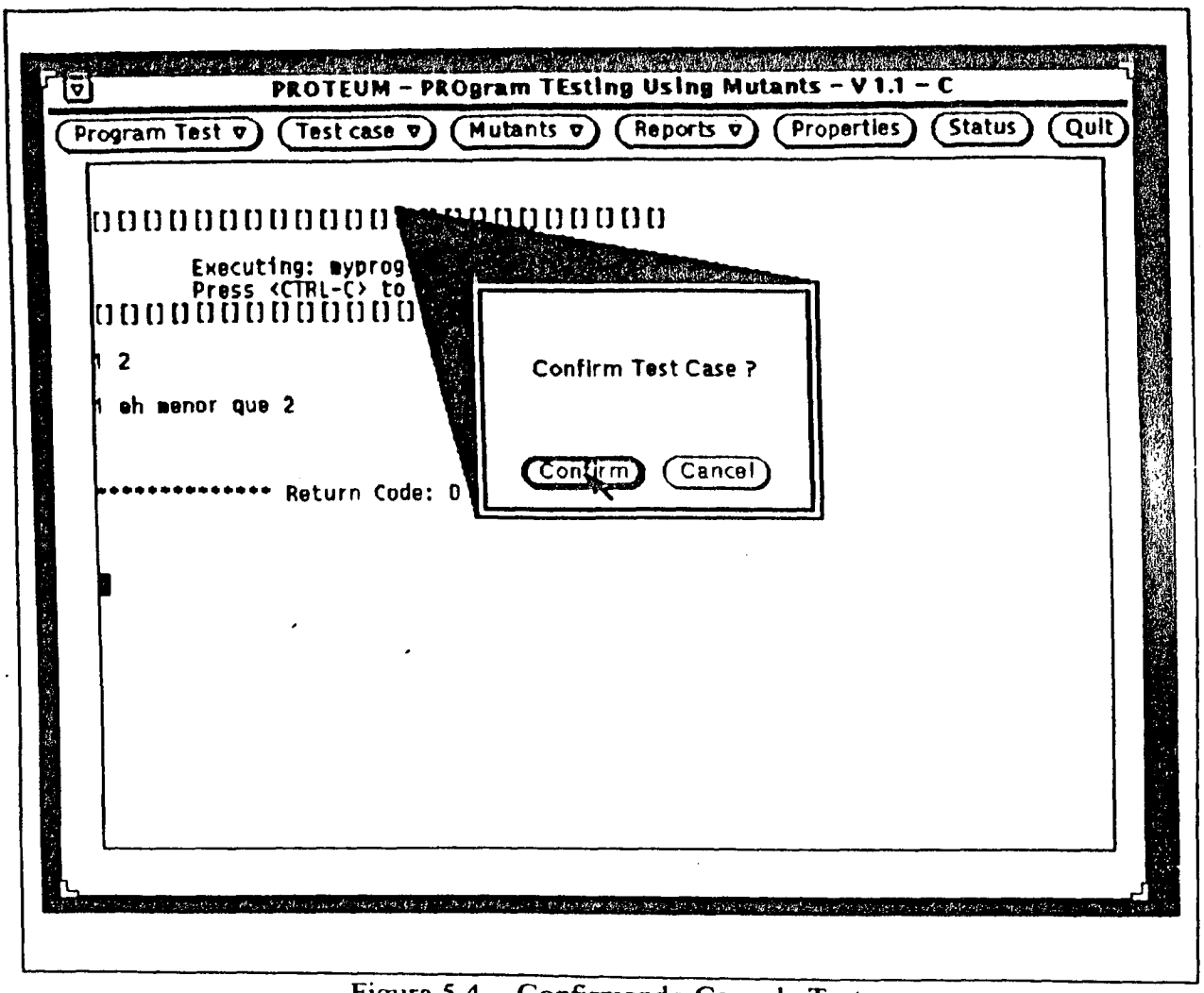


Figura 5.4. - Confirmando Caso de Teste

5.2. - Consultando o Conjunto de Casos de Teste

Uma vez incluídos casos de teste, o usuário pode visualizá-los através da opção *View* do menu de casos de teste. O Proteum exibe um painel como o mostrado na Figura 5.5., onde é possível identificar os principais pontos que caracterizam cada caso de teste.

O primeiro campo indica qual o número do caso de teste. Através desse campo o usuário tem acesso direto a qualquer caso de teste, apenas digitando o seu número. Se o número digitado ultrapassa o último caso de teste disponível, o último caso de teste é mostrado.

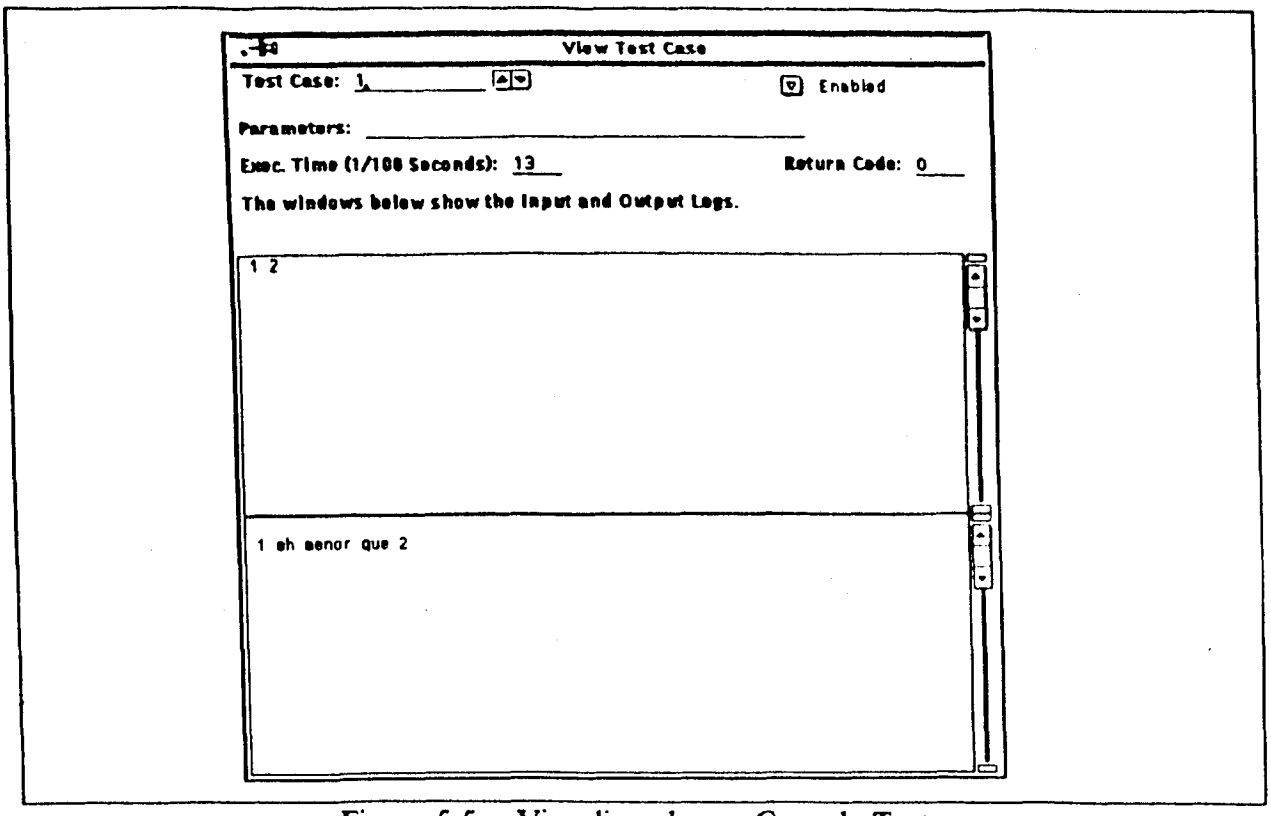


Figura 5.5. - Visualizando um Caso de Teste

O usuário pode também "navegar" seqüencialmente através dos casos de teste selecionando um dos campos com a seta para cima ou para baixo, indicando próximo caso de teste ou caso de teste anterior, respectivamente.

Os demais campos apresentam as características do caso de teste propriamente ditas. O campo *Parameters* apresenta os parâmetros iniciais fornecidos pelo usuário para o caso de teste.

O campo *Exec Time* mostra quanto tempo de CPU, em centésimos segundos, ocupou a execução do programa com esse caso de teste. Este valor é utilizado para que se possa eliminar um mutante que entra num laço infinito, ou seja, se o tempo de execução de um mutante sobre esse caso de teste ultrapassa muito o tempo de execução do programa original, então pode-se considerar esse mutante como morto. Essa característica será tratada mais adiante, quando tratar-se da execução dos mutantes.

O código de retorno, ou como vê-se no exemplo, o *Return Code*, é um valor que indica qual o motivo que causou o fim do programa, e pode ser visto como um indicativo de erro ou sucesso na execução do mesmo. Este é outro parâmetro que pode ser usado para comparar o comportamento do programa original e de seus mutantes. Por isso, também é armazenado para caracterizar um caso de teste.

As subjanelas, embaixo do painel, apresentam os registros de entrada e saída do caso de teste. Em outras palavras, a janela de cima exibe aquilo que o usuário digitou em tempo de execução do programa e a janela de baixo mostra os resultados produzidos pelo programa.

No canto superior direito do painel existe uma lista com duas opções mutuamente exclusivas que o usuário pode escolher: *Enable/Disable*. Se seleciona a segunda, o usuário estará desabilitando o caso de teste, ou seja, a próxima vez em que executar os mutantes, o Proteum não irá considerar esse caso de teste na execução. E mais que isso, se um mutante estiver marcado como morto por esse caso de teste, ele será "resuscitado" e executado com os demais casos de teste habilitados (é bom observar que isso só acontece quando o usuário explicitamente ordenar a execução dos mutantes) A operação não implica na remoção do caso de teste do conjunto disponível, podendo um caso de teste desabilitado ser habilitado simplesmente escolhendo-se a opção *Enable* no painel de visualização do caso de teste, como mostrado na Figura 5.6.

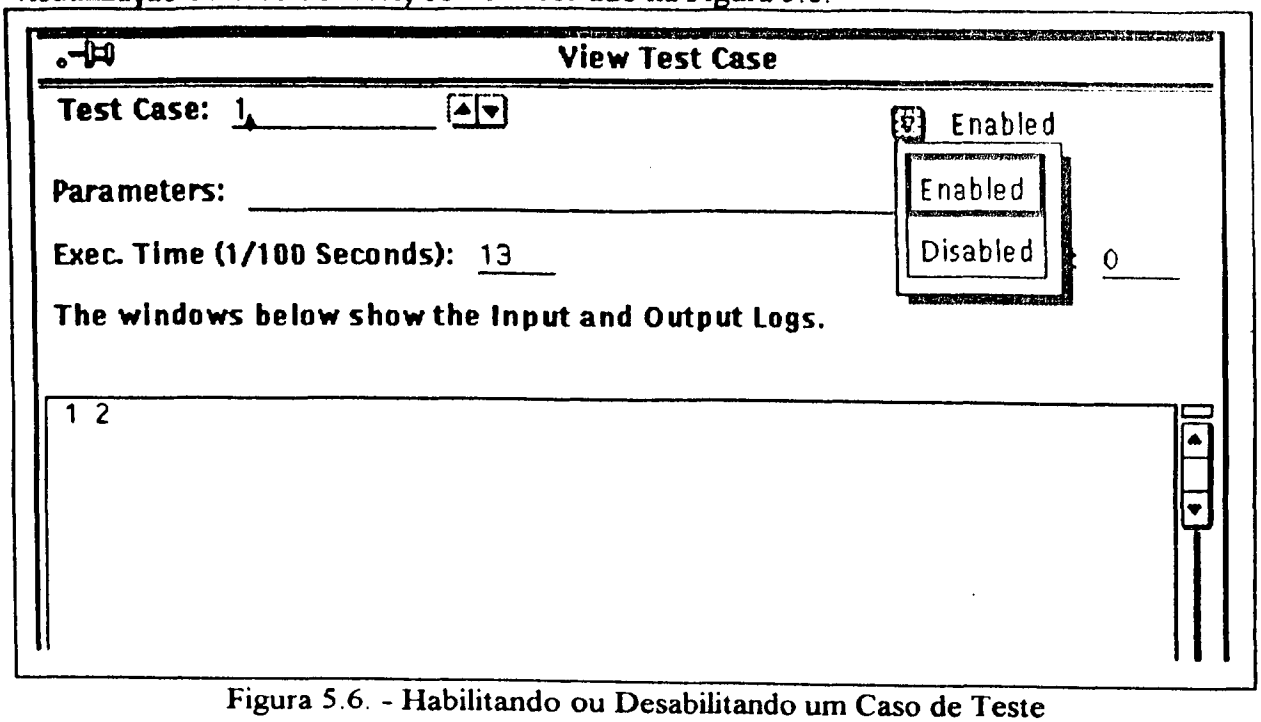


Figura 5.6. - Habilitando ou Desabilitando um Caso de Teste

5.3. - Eliminando Casos de Teste

É possível também excluir fisicamente casos de teste da base de dados. Para isso basta escolher a opção *Delete* (da Figura 5.1.) e o Proteum apresenta o painel da Figura 5.7., onde o usuário informa o número dos casos de teste que deseja eliminar.

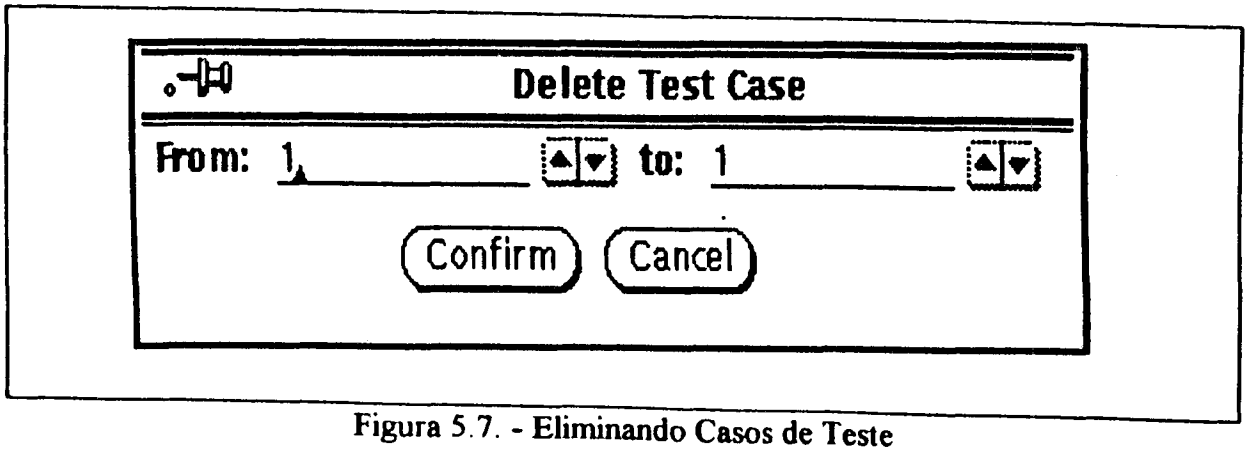


Figura 5.7. - Eliminando Casos de Teste

Uma vez eliminados os casos de teste, os demais, que permanecem no conjunto, são renumerados. Por exemplo: um conjunto tem 10 casos de teste numerados de 1 a 10. Se o usuário escolher eliminar os casos de teste de 4 a 7, então obtém-se um conjunto numerado de 1 a 6, sendo que o caso de teste que antes era número 8 passou a ser 4, o que era 9 passou a ser cinco e o 10 passou a ser 6.

Ao retirar-se fisicamente um caso de teste do conjunto, não é possível recuperá-lo, a não ser através da operação de inclusão.

5.4. - Importando Casos de Teste

Muitas vezes, pode-se desejar utilizar conjuntos de casos de teste já definidos em outro lugar, permitindo assim, além da economia de tempo e esforço, que conjuntos de casos de teste já avaliados para outros testes sejam reavaliados no Proteum

A atual versão do Proteum permite que o usuário copie casos de teste de três possíveis fontes, ou seja: de outros testes do próprio Proteum, de testes da POKE-TOOL, ou de arquivos ASCII. É o que mostra a Figura 5.8.

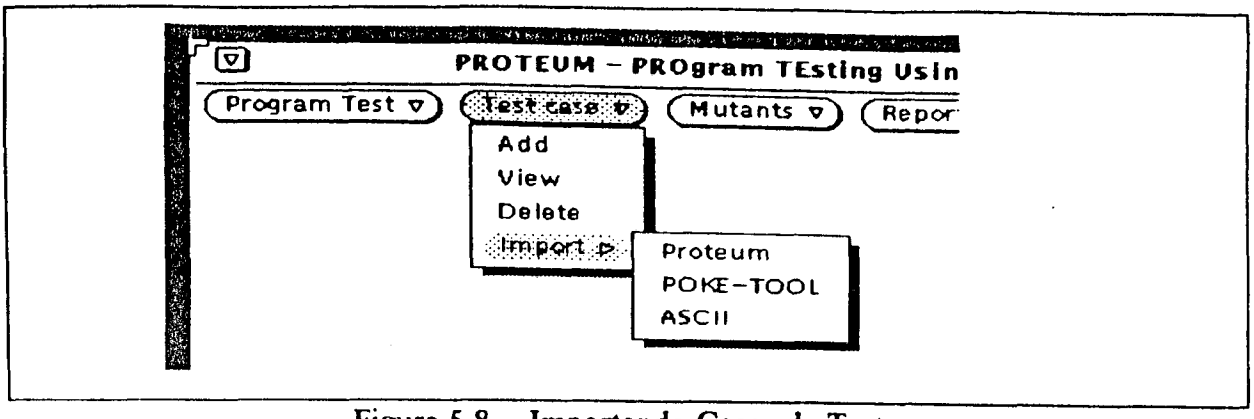


Figura 5.8. - Importando Casos de Teste

5.4.1. - Casos de Teste do Proteum

Às vezes, deseja-se realizar mais de um teste, com características diferentes, para um mesmo programa. Assim, não é necessário realizar diversas vezes o trabalho de incluir o mesmo conjunto de casos de teste. Basta definí-lo uma vez, para um teste e nos demais simplesmente importá-lo.

Para isso, o Proteum solicita ao testador que forneça o nome e o diretório do teste do qual o conjunto de casos de teste deve ser copiado. Os casos de teste são adicionados aos que já existem, como se fossem incluídos um a um. Se algum dos casos de teste não interessa, basta que o usuário o remova após importá-lo.

Um exemplo é mostrado na Figura 5.9.

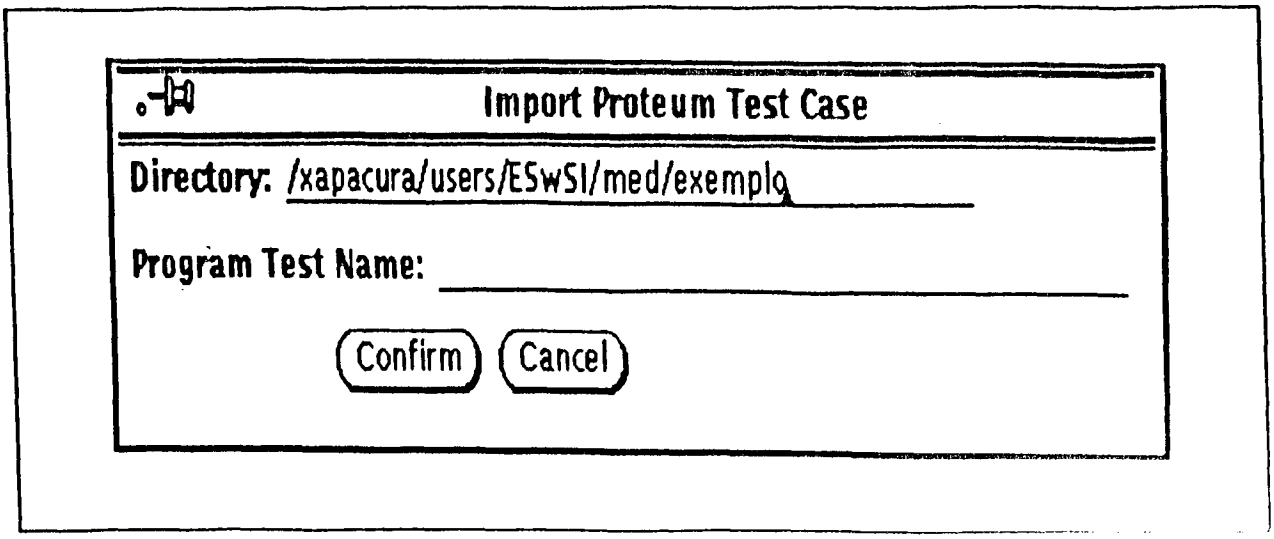


Figura 5.9. - Importando Casos de Teste do Proteum

5.4.2. - Casos de Teste da POKE-TOOL

A **POKE-TOOL** é uma ferramenta que apóia o teste estrutural de programa utilizando os critérios **Potenciais Usos** [MAL91,CHA91]. Para importar os casos de teste definidos para a POKE-TOOL, o procedimento é basicamente o mesmo definido acima.

A diferença é que, como mostra a Figura 5.10., deve-se fornecer apenas o diretório onde estão armazenados os arquivos com os casos de teste da POKE-TOOL.

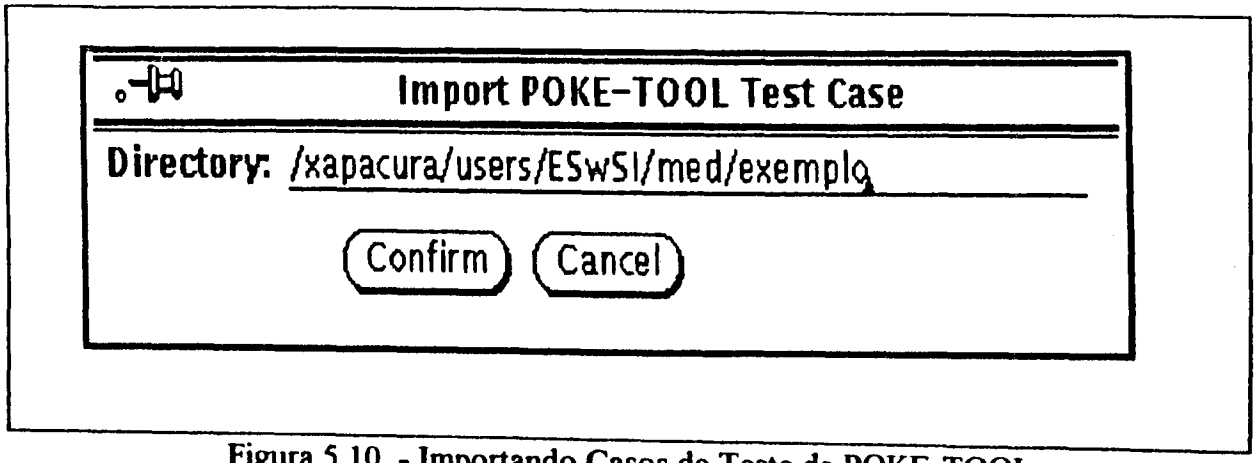


Figura 5.10. - Importando Casos de Teste da POKE-TOOL

5.4.3. - Casos de Teste de Arquivos ASCII

Conforme mostra a Figura 5.11. podem ser importados casos de teste de arquivos no formato ASCII, que contém as entradas correspondentes aos casos de teste. Se o usuário

fornece como nome do arquivo, por exemplo, o nome *casoteste*, e valores de 1 a 10, então, serão utilizados os arquivos com nomes *casoteste1*, *casoteste2*, ..., *casoteste10*.

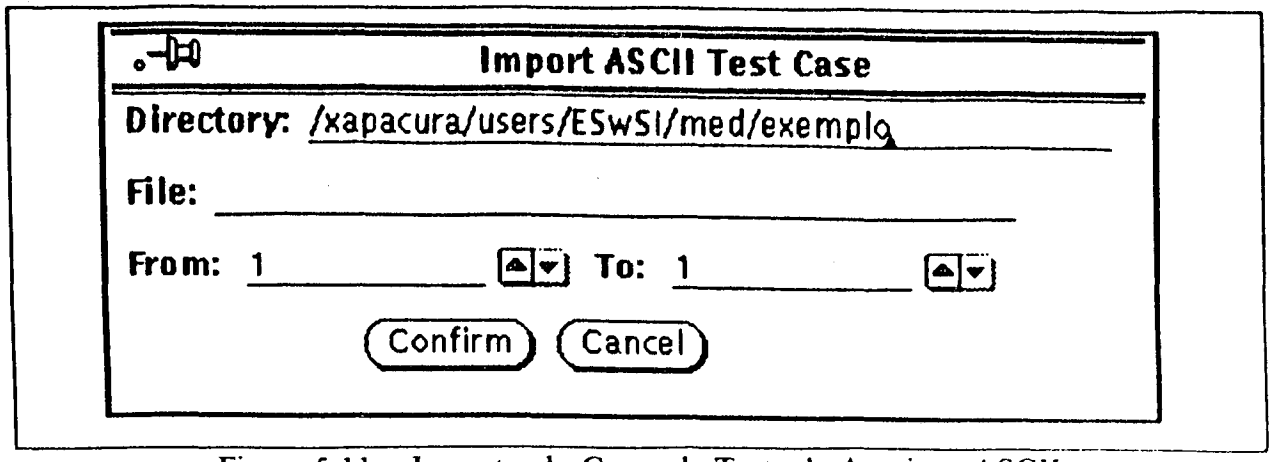


Figura 5.11. - Importando Casos de Teste de Arquivos ASCII

6. - Trabalhando com os Mutantes

A Figura 6.1. mostra quais as operações que se podem realizar sobre a base de dados que representa os mutantes os quais configuram a parte central da atividade de teste. É através dessas operações que se avalia o conjunto de casos de teste.

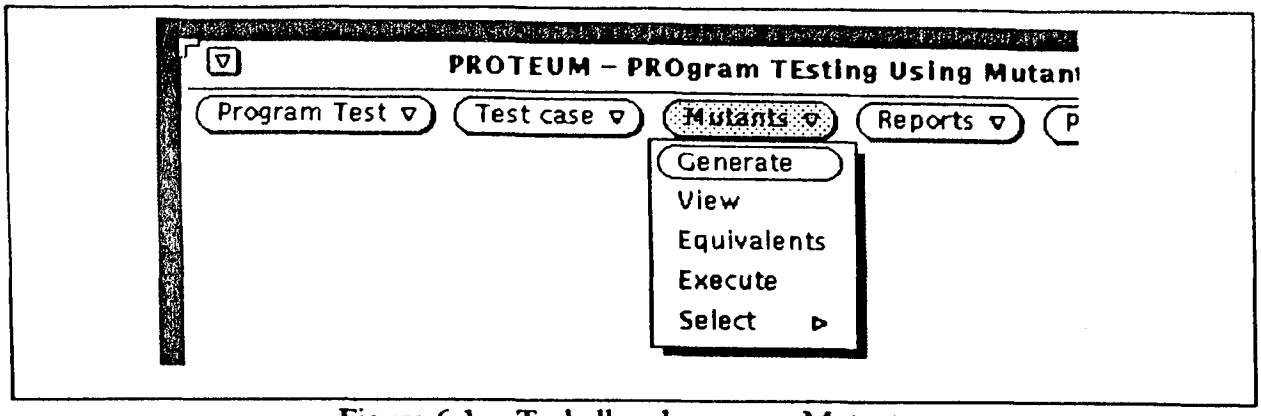


Figura 6.1. - Trabalhando com os Mutantes

6.1. - Gerando os Mutantes

A primeira operação que se deve efetuar é a geração dos mutantes. Nessa operação, através da opção *Generate*, o usuário escolhe quais mutantes devem ser gerados ou, mais precisamente, quais operadores de mutação devem ser aplicados sobre o programa em teste para a criação dos mutantes.

A Figura 6.2. mostra um painel onde o usuário pode escolher uma das 4 classes de operadores de mutação. Selecionando um dos botões de classes, o Proteum mostra uma lista (Figura 6.3.) com todos os operadores. Para cada operador são mostrados dois valores: a quantidade de mutantes gerados e a percentagem de geração escolhida pelo testador. Se já foram gerados os mutantes de um determinado operador, esses valores aparecem acinzentados, e o usuário não pode mais alterá-los. Caso contrário, o testador pode selecionar a porcentagem de geração, individualmente para cada operador.

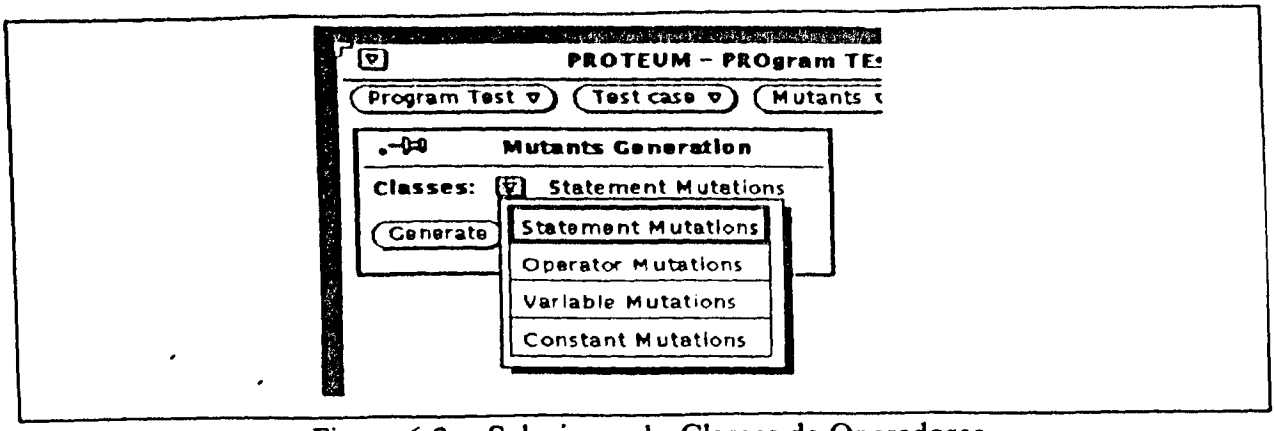


Figura 6.2. - Selecionando Classes de Operadores

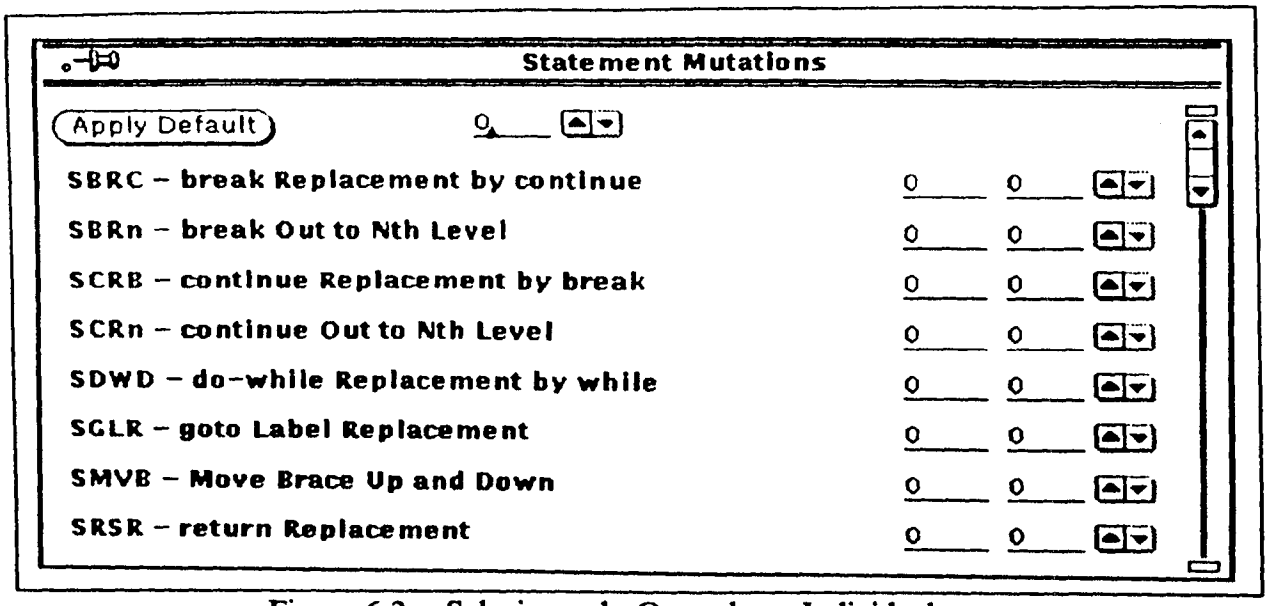


Figura 6.3. - Selecionando Operadores Individualmente

No topo de cada janela de classe existe também um campo em que o testador pode fornecer um valor default para a classe. Ao alterar-se o valor padrão de uma classe e acionar-se o botão *apply default*, alteram-se todas as percentagens dos operadores que ainda estão habilitados, ou seja, que ainda não geraram mutantes. Por exemplo, se escolhe-se o valor 100 para a classe *Statement Mutations*, isso significa que, para cada operador dessa classe, deseja-se gerar 100% do número total de mutantes. Mas se escolhe-se para os operadores *SDWD*, *SGLR* e *SMVB* o valor 0 e para o operador *SRSR* o

valor 50, então todos os operadores da classe geram 100% dos mutantes mas os operadores *SDWD*, *SGLR* e *SMVB* não geram nenhum mutante e o operador *SRSR* gera 50% dos seus mutantes. Se alterar-se, o valor padrão de 100 para 50%, e pressionar-se novamente o botão *apply default*, então todos os valores individuais dos operadores passam para 50, inclusive aqueles alterados individualmente..

Para iniciar a geração dos mutantes, o usuário aciona o botão *Confirm* e o Proteum passa a mostrar, para cada função do programa, a contagem dos operadores que vão sendo aplicados.

Esta operação pode ser repetida diversas vezes. Numa segunda vez, o usuário só poderá gerar mutantes com os operadores com os quais ainda não tenha gerado, ou seja, para os quais tenha escolhido valor de 0% na primeira vez. É o que mostra a Figura 6.4. Nota-se que os operadores que já geraram os mutantes anteriormente aparecem desativados no painel, ou seja, o usuário não pode alterar o seu valor que representa a porcentagem escolhida anteriormente para esse operador. Por exemplo, para os operadores de comandos, vemos que todos estão desativados, menos os operadores *SDWD*, *SGLR* e *SMVB* que ainda não geraram mutantes e para os quais o usuário pode escolher uma porcentagem qualquer.

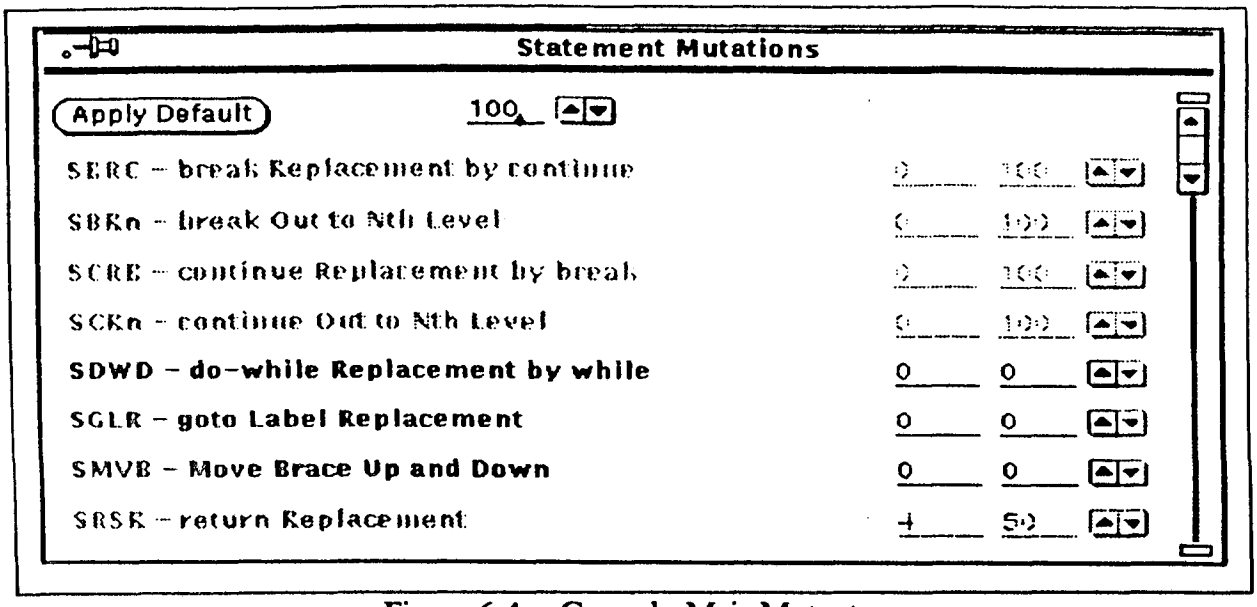


Figura 6.4. - Gerando Mais Mutantes

6.2. - Executando os Mutantes

A terceira opção do menu de mutantes permite ao usuário iniciar a execução dos mutantes. Com essa operação, os mutantes gerados são executados utilizando-se os casos de teste definidos pelo usuário e que estiverem habilitados. Se um mutante apresenta, para algum dos casos de teste, resultados diferentes do programa original, costuma-se dizer que esse mutante está **morto**. Por outro lado, se os resultados apresentados por um mutante coincidem com os resultados do programa em teste, diz-se que ele está **vivo**.

Se o usuário definiu o tipo do teste (ver operação de criação de um teste) como *Research*, então cada mutante é executado com todos os casos de teste, independentemente de algum caso de teste matar ou não o mutante. Dessa forma, obtém-se informação relacionando todos os mutantes e todos os casos de teste. Nesse caso, se o usuário efetua a operação de execução e depois inclui novos casos de teste, ele deve re-executar os mutantes, obrigando todos os mutantes a serem executados com todos os novos casos de teste.

Por outro lado, se o tipo do teste é o convencional, ou seja, tipo *Test*, somente os mutantes vivos são executados. Por exemplo: se o conjunto de casos de teste utilizado possui 5 casos de teste e um mutante é morto ao ser executado com o caso de teste número um, então ele não será executado com os casos de teste 2 a 5 pois já está morto. O mesmo ocorre ao incluir-se novos casos de teste; somente os mutantes que não foram mortos em execuções anteriores serão executados com os casos de teste recém incluídos.

O usuário pode também selecionar sub-conjuntos de mutantes que deseja executar. Através da operação *Select* (ver Seção 6.4), pode-se determinar quais mutantes devem ser considerados na operação de execução (são chamados mutantes **ativos**) e quais não devem ser considerados (mutantes **inativos**). No cálculo do escore de mutação apenas os mutantes ativos são tomados em consideração (mais detalhes na Seção 7).

A operação de desabilitar um caso de teste também pode ter efeito sobre o estado dos mutantes. Tomando o exemplo acima, se desabilitar-se o caso de teste número 1, na próxima operação de execução dos mutantes, aquele mutante morto por esse caso de teste deverá ser executado com os demais casos de teste, até que seja morto ou até que todos os casos de teste tenham sido tentados. Neste caso, o estado do mutante muda, passando a ser considerado vivo novamente.

É importante notar que, para manter a integridade entre o conjunto de casos de teste e o estado dos mutantes, deve-se realizar a operação de executar os mutantes. Assim, ao incluir, excuir, habilitar ou desabilitar casos de teste é aconselhável executar essa operação.

Como mostra a Figura 6.5., o usuário tem a possibilidade de interromper a execução dos mutantes, acionando o botão *Cancel*. Nesse caso, somente os mutantes que foram executados até o momento da interrupção serão considerados no cálculo do **escore de mutação**. Mais detalhes serão fornecidos adiante quando será explicada a opção *Status*.

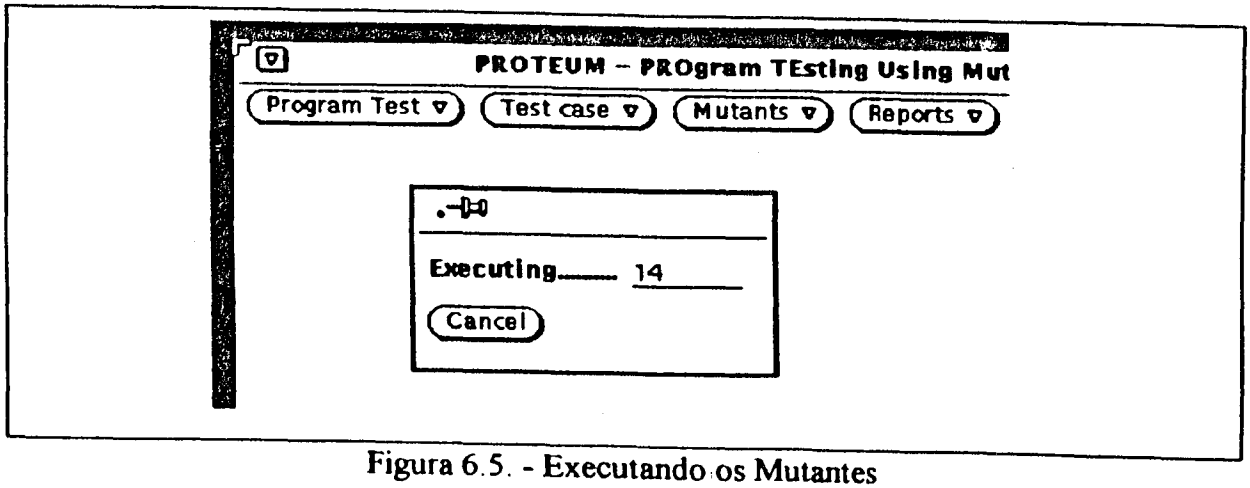


Figura 6.5. - Executando os Mutantes

6.3. - Visualizando os Mutantes

Uma vez executados os mutantes, o usuário provavelmente desejará analisá-los, principalmente com o intuito de verificar porque alguns deles não morreram.

Assim como é feito para os casos de teste, o usuário pode visualizar um mutante fornecendo seu número ou então "navegar" seqüencialmente através do conjunto de mutantes gerados, como mostra a Figura 6.6.

A figura mostra um painel onde os mutantes são apresentados. No topo, do lado direito do painel existe uma lista em que o usuário pode escolher o tipo de mutantes que deseja visualizar. Assim, se desejar, por exemplo, ver apenas os mutantes que continuam vivos, o usuário deve "desligar" os botões *Dead*, *Equivalent* e *Anomalous*; então, enquanto estiver navegando pelos mutantes só serão exibidos aqueles com status de vivo. Pode também escolher se deseja ver apenas os mutantes ativos ou também os inativos se desejar ver apenas os ativos, deve desligar o botão *Inactive* nessa lista

Do lado esquerdo, o primeiro campo do painel mostra o número do mutante É através desse campo que o usuário escolhe o número do mutante a ser exibido. Se o usuário escolhe o número de um mutante que não existe ou cujo status não coincide com aqueles habilitados na lista descrita acima, então o Proteum exibe o mutante mais próximo do número escolhido e que tenha o status correto. Por exemplo: o painel está exibendo o mutante número 8 e na lista de tipos estão selecionados apenas os botões *Live* e *Inactive*; então o usuário digita o valor 15. Se o mutante não estiver vivo, o próximo mutante com este status será apresentado, se existir algum.

O segundo campo mostra qual o status do mutante, que pode ser *Live*, *Dead* ou *Anomalous*. Para o caso de mutantes mortos, tem-se a indicação de qual foi o motivo que levou um mutante a ser marcado como morto. Esses motivos podem ser: saída produzida pelo mutante difere da saída do programa em teste (*stdout*), código de retorno difere do código de retorno do programa em teste (*return code*) ou tempo de execução do mutante é muito maior que o tempo de execução do programa original (*timeout*). No caso de testes do tipo *Research*, mais de um caso de teste pode ter matado o mutante; neste caso, é apresenatada a causa relacionada com o primeiro caso de teste que matou o mutante.

Em seguida vem a indicação se o mutante está ativo ou inativo. O quarto campo é um botão, o qual o usuário pode ligar ou desligar, indicando se o mutante é ou não equivalente ao programa em teste. Portanto, essa operação de visualização serve também para marcar os mutantes como equivalentes. É bom lembrar que os mutantes marcados como equivalentes não serão mais executados, ou seja, o Proteum não tentará mais matar esses mutantes. Para o caso de mutantes mortos ou anômalos, esse campo aparece desabilitado, não permitindo que sejam marcados como equivalentes.

Na atual versão do Proteum, um mutante é considerado anômalo quando possui alguma construção sintaticamente incorreta, ou seja, quando a aplicação do operador de mutação gerou um programa com erros sintáticos. Embora isso ocorra raramente, em alguns casos, ainda são gerados mutantes anômalos.

View Mutant

Mutant: 8

Types to Show: Alive | Dead | Anomalous | Equivalent | Inactive

Status: Active: Alive

☐ Equivalent

Operator: SRSR – return Replacement

Original Program.

Execute

```
main()
{
int     a,b;

  scanf("%d %d", &a, &b);
  if (a > b)
        printf("\n%d eh maior que %d", a, b);
  else
  if (a < b)
        printf("\n%d eh menor que %d", a, b);
  else
        printf("\n%d eh igual a %d", a, b);
  return 0;
}
```

Mutant Program.

Execute

```
main()
{
int     a,b;

  scanf("%d %d", &a, &b);
  if (a > b)
        return 0;
  else
  if (a < b)
        printf("\n%d eh menor que %d", a, b);
  else
        printf("\n%d eh igual a %d", a, b);
  return 0;
}
```

Figura 6.6. - Analisando os Mutantes

O campo seguinte contém o nome do operador de mutação que gerou o mutante. Na parte inferior do painel, duas janelas exibem o código fonte do programa original e o do mutante para que o usuário possa compará-los e verificar qual alteração foi feita na criação do mutante. O ponto (ou o primeiro ponto, caso existam mais que um) onde a alteração foi feita no programa original e o ponto correspondente no mutante são automaticamente exibidos nas janelas, ou seja, o usuário não precisa buscar por todo o texto do mutante a alteração efetuada. Sobre cada uma dessas janelas há um botão *Execute*. Ao acionar um desses botões, o usuário inicia a execução do programa correspondente, interagindo com ele de maneira semelhante à inclusão de casos de teste. Com esse procedimento, o usuário pode experimentar dados de entrada e verificar se eles são ou não eficientes para matar determinado mutante.

6.4. - Selecionando Sub-conjuntos de Mutantes

Através da opção *Select* o usuário pode selecionar sub-conjuntos do conjunto de mutantes gerados. Em outras palavras, o usuário pode determinar que alguns mutantes serão desativados, não sendo considerados na condução do teste. Mutantes inativos não são executados e não são considerados no cálculo do escore de mutação.

Existem dois critérios para selecionar mutantes: através de porcentagens aleatórias para cada operador de mutação (opção *By Operator*) e através da restrição do número de mutantes por bloco do programa (opção *By Block*).

6.4.1. - Seleção por Operador de Mutação

Escolhendo a opção de seleção por operador, o usuário tem à sua disposição um painel igual àquele utilizado na geração dos mutantes. Nele deve ser fornecido, por classe ou individualmente para cada operador de mutação, uma porcentagem de seleção.

Como resultado, o Proteum seleciona aleatoriamente entre os mutantes gerados a porcentagem indicada, para cada operador de mutação. Por exemplo, se o usuário gerou 50% dos mutantes para os operadores da classe de mutação sobre comandos (*Statment Mutations*) e seleciona 80% dos mutantes dessa mesma classe, então permanecerão ativos apenas 40% do total de mutantes correspondentes a esta classe (0.5 x 0.8 = 0.4).

É importante notar que, para um dado operador de mutação, gerar X% dos mutantes e depois selecionar Y% deles não é equivalente a gerar Y% e selecionar X%; o conjunto de mutantes que permanecem ativos nesses dois casos, não é necessariamente o mesmo. Isso só acontece quando X (ou Y) é igual a 100%. Ou seja, se o usuário gera 100% dos mutantes de determinado operador e em seguida seleciona, por exemplo, 30% desses mutantes, permanecerão ativos os mesmos mutantes que seriam criados escolhendo-se 30% como porcentagem de geração para este operador.

6.4.2. - Seleção por Blocos[1]

Outra opção para minimizar o número de mutantes a serem considerados no teste é através da escolha de quais pontos do programa devem ser aplicados os operadores de mutação. Mais precisamente, através da opção *By Block*, o usuário pode determinar o número máximo de vezes que um operador de mutação pode ser aplicado em cada bloco do programa.

Ao escolher esta opção, o usuário deve fornecer, através do painel mostrado na Figura 6.7., o número máximo (N) de mutantes para cada operador de mutação, por bloco do programa. Neste caso, o Proteum seleciona dentre os mutantes gerados, apenas N mutantes de um determinado tipo, por bloco.

Figura 6.7. - Selecionando Mutantes por Bloco de Programa

7. - Verificando o Andamento do Teste

A opção *Status* no menu principal do Proteum permite ao usuário ter acesso a informações sobre o andamento do teste. A Figura 7.1. mostra quais são essas informações.

A mais importante é o escore de mutação (*Mutation Score*), que indica o quanto o conjunto de casos de teste está próximo da adequação para o programa em teste, em relação ao conjunto de mutantes gerados ou, na verdade, em relação aos mutantes ativos e executados.

[1] Ainda não implementado.

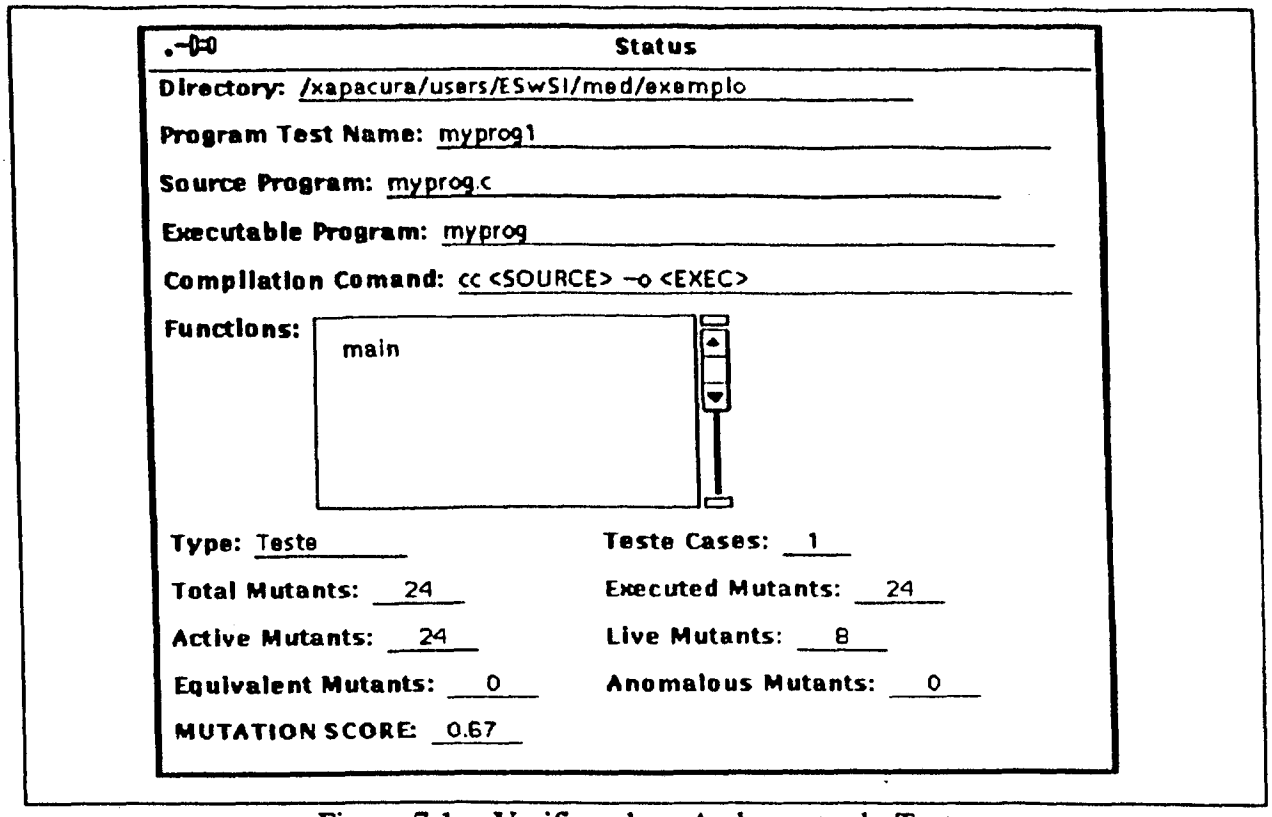


Figura 7.1. - Verificando o Andamento do Teste

Como pode-se ver, existe no painel de status um campo com o número de mutantes executados, que não necessariamente é igual ao número total de mutantes. Isso pode acontecer porque o usuário pode interromper a execução dos mutantes antes do final, fazendo com que parte dos mutantes não esteja refletindo o estado real em relação aos casos de teste. Portanto, somente aqueles mutantes que foram atualizados na última operação de execução dos mutantes podem ser considerados no cálculo do escore de mutação, que por isso é calculado da seguinte maneira:

$$ms = \frac{\#executados - \#inativos - \#vivos - \#anomalos - \#equivalentes}{\#executados - \#anomalos - \#equivalentes}$$

Como pode-se notar, o número de mutantes inativos é subtraído do numero de mutantes gerados, ou seja, o número de mutantes inativos exibido no painel diz respeito apenas aos mutantes executados, e não ao conjunto total de mutantes.

O escore varia entre 0 e 1. O objetivo é conseguir casos de teste que obtenham um escore o mais próximo possível de 1, ou seja, que faça com que o número de mutantes vivos se aproxime de 0.

8. - Criando Relatórios

Outra maneira de verificar o andamento do teste é através de relatórios que o Proteum produz. Atualmente, o único tipo de relatório gerado é sobre os casos de teste.

8.1. - Criando Relatórios sobre Casos de Teste

Selecionando o botão *Report* e a opção *Test Case*, o usuário visualiza um painel onde ele pode escolher as informações que deseja ver no relatório. Por exemplo, com as escolhas mostradas na Figura 8.1., obtém-se o relatório da Figura 8.2.. Escolhendo-se todas as opções do relatório, obtém-se o relatório da Figura 8.3..

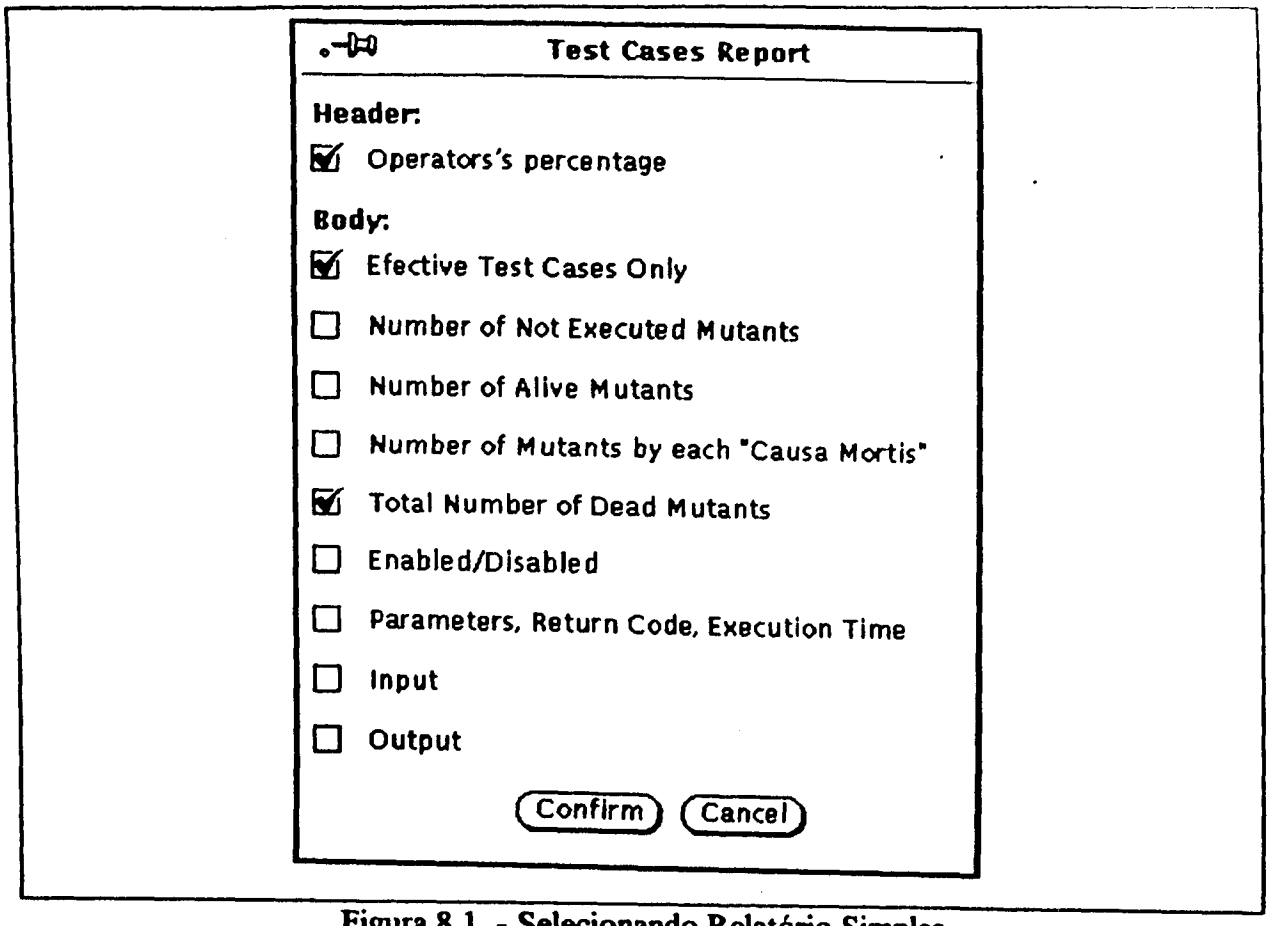


Figura 8.1. - Selecionando Relatório Simples

```
/xapacura/users/ESwSI/med/exemplo/myprog1.TMP

[][][][
```

```
/xapacura/users/ESwSI/med/exemplo/myprog1.TMP

[]
[]   EQUIVALENT MUTANTS: 0
[]
[]   MUTATION SCORE:   0.79
[]
[]   OPERATORS:
[]   SBRC 100%      SBRn 100%      SCRB 100%      SCRn 100%
[]   SRSR  50%      SSDL 100%      STRI 100%      STRP 100%
[]   SMTC 100%      SSWM 100%      SMTT 100%      SWDD 100%
[]
[][][][
```

9. - Configurando o Proteum

A opção *Properties* do menu principal permite ao usuário configurar algumas variáveis do ambiente. O painel mostrado na Figura 9.1. mostra quais são essas variáveis.

O primeiro campo, *Default Directory*, determina qual é o diretório padrão, ou seja em que diretório o usuário deseja trabalhar. O diretório que o usuário escolher aparece como padrão nos demais painéis onde o nome de um diretório deva ser fornecido pelo usuário. É o caso, por exemplo do painel da Figura 3.6., onde o usuário deve fornecer um diretório e o nome do teste a ser carregado. Ao acionar esse painel, o campo *Directory* vem sempre preenchido com o diretório padrão; porém, nada impede que o usuário altere esse campo.

No início, quando o Proteum é carregado, o diretório padrão é o diretório corrente da janela onde o Proteum foi invocado.

A segunda variável que pode-se configurar é o valor de "timeout" para execução dos mutantes. Quando um mutante é criado, pode-se introduzir alguma alteração que faça com que o mutante, ao ser executado, entre num laço sem fim. Por esse motivo, utiliza-se o tempo de execução como um parâmetro para diferenciar o programa original dos mutantes. Se o tempo de execução de um mutante, com determinado caso de teste, ultrapassar várias vezes o tempo de execução do programa original com o mesmo caso de teste, então esse mutante é considerado morto. O valor fornecido pelo usuário determina quantas vezes o tempo de execução do mutante deve ser superior ao do programa original para que seja considerado morto. Inicialmente, o valor utilizado é 5. Por exemplo, se o programa original tem um tempo de execução de 2 centésimos de segundo, então o tempo de timeout dos seus mutantes será de 10 centésimos de segundo. Um valor muito baixo para o timeout pode fazer com que mutantes que não estão num laço infinito sejam considerados mortos; por outro lado, um valor muito alto irá certamente fazer aumentar o tempo total de execução dos mutantes.

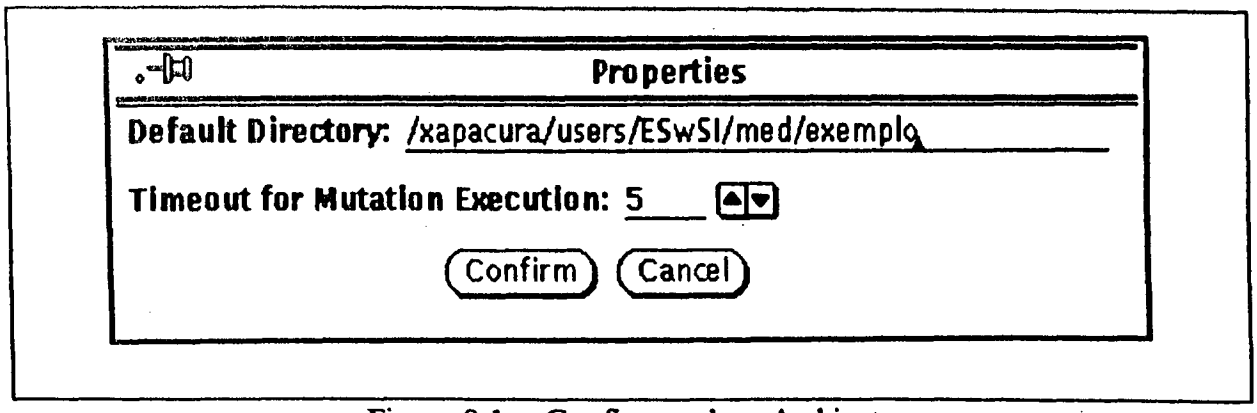


Figura 9.1. - Configurando o Ambiente

10. - Diferenças em Relação à Versão 1.0 - C

As principais diferenças da atual versão do Proteum em relação à versão anterior [DEL93] são:

- A versão atual trata o conjunto completo das estruturas da linguagem C. Para isso, utiliza o pré-processador CPP do ambiente Unix.
- Foram implementadas novas estratégias de minimização do número de mutantes. Estas estratégias podem ser executadas de maneira dinâmica, ou seja, mesmo que todos os mutantes tenham sido gerados, pode-se desativar e ativar conjuntos de mutantes através da opção *Select*.
- Foi implementada a execução interativa dos mutantes para facilitar ao usuário escolher casos de teste efetivos para matar um mutante.
- A performance na execução dos mutantes foi melhorada, através da diminuição do número de mutantes que precisam ser compilados. Na versão atual, vários mutantes são reunidos num único programa que é compilado apenas uma vez.
- A eliminação de mutantes através de timeout tornou-se mais precisa. Agora o tempo de execução de um caso de teste não é mais medido em segundos mas sim em centésimos de segundo e leva em consideração apenas o tempo de CPU utilizado (não mais o tempo total de execução). Com isso melhorou-se também a performance da execução dos mutantes.
- Na visualização dos mutantes tem-se a informação de qual foi a causa da morte de um mutante (timeout, código de retorno ou saída produzida). No caso de testes do tipo *Research*, é exibida a causa relacionada ao primeiro caso de teste que matou o mutante.
- O número máximo de casos de teste, na versão atual passou a ser 512.

Referências

[CHA91] Chaim, M.L., POKETOOL - Uma Ferramenta para Suporte ao Teste Estrutural de Programas Baseado em Análise de Fluxo de Dados, Dissertação de Mestrado, DCA/FEE/UNICAMP, Abr 1991.

[DEL93] Delamaro, M.E., Proteum - Um Ambiente de Teste Baseado na Análise de Mutantes, Dissertação de Mestrado, ICMSC-USP, Out 1993.

[DEM78] DeMillo, R.A., Lipton, R.J., Sayward, F.G., "Hints on Test Data Selection: Help for the Practicing Programmer", Computer, vol. 11(4), Abr 1978.

[MAL91] Maldonado, J.C., Critérios Potenciais Usos: uma Contribuição ao Teste Estrutural de Software, Tese de Doutorado, FEE/UNICAMP, Campinas - S.P., 1991.

[MAR90] Marshall, A.C., Hedley, D., Riddell, I.J., Hennell, M.A., "Static Dataflow-aided Weak Mutation Analysis (SDAWM)", Information and Software Technology, vol 32, No. 1, Jan/Fev 1990.