PROTEUM/IM: Uma Ferramenta de Apoio ao Teste de Integração

Márcio Eduardo Delamaro
Instituto de Física de São Carlos - USP

José Carlos Maldonado Elisa Yumi Nakagawau
Instituto de Ciências Matemáticas de São Carlos - USP
Cx Postal 668, 13560-970
São Carlos - SP
{med,jcmaldon,elisa}@icmsc.sc.usp.br

1 Introdução

A PROTEUM/IM é uma ferramenta de auxílio à atividade de teste. Ela apóia a aplicação do critério Mutação de Interface [2]. Esse critério, baseado em mutações, busca exercitar as interações entre as unidades de um programa, ou seja, trata-se de um critério interprocedimental, sendo assim indicado para o teste de integração de software. Dados o programa **P** e o conjunto de casos de teste **T**, cuja adequação deseja-se avaliar, são considerados programas $\mathbf{M_1}$, $\mathbf{M_2}$, ..., $\mathbf{M_n}$ (chamados de **mutantes de P**), semelhantes a **P**, mas com pequenos desvios sintáticos, como por exemplo, a troca de uma expressão "$a = b + 1$" por "a = c + 1". Essas alterações sintáticas são introduzidas em certos pontos de **P** de maneira que os mutantes representem erros de integração, caracterizados por valores incorretos sendo trocados entre duas unidades do programa. O conjunto de teste **T** é avaliado pela sua capacidade em distinguir **P** de seus mutantes, ou seja, pela sua capacidade em mostrar que um mutante se comporta de maneira diferente do programa original **P**.

Para isso, **P** e seus mutantes são executados com os casos de teste em **T**. Mutantes que apresentam resultados idênticos a **P** (chamados de mutantes "vivos") ou são equivalentes a **P** ou indicam uma deficiência no conjunto **T** que não possui casos de teste que consigam distingui-los. Já os mutantes que apresentam comportamento diverso de **P** para algum caso de teste em **T**, são ditos "mortos". Quanto maior o número de mutantes mortos por **T**, melhor sua qualidade para o teste de **P**, sob o ponto de vista da Mutação de Interface. O grau de adequação de **T** é dado pelo **escore de mutação**:

$$MS(P,T) = \frac{\#mutantes\ mortos}{\#mutantes\ não\ equivalentes}$$

Para aplicação do critério Mutação de Interface numa dada conexão f-g, são seguidos os seguintes passos:

1. definição de um conjunto de casos de teste **T**;
2. execução de **P** com os casos de teste em **T**;
3. geração dos mutantes correspondentes à conexão f-g;
4. execução dos mutantes com os casos de teste de **T**; e
5. análise dos mutantes vivos.

A Ferramenta *PROTEUM/IM* auxilia a aplicação do critério, executando de maneira automatizada algumas dessas tarefas e auxiliando o testador a executar outras. Ela possui dois modos de interação. O primeiro é diretamente na linha de comando da shell do sistema ou em scripts. O segundo é de maneira transparente para o testador, através de uma interface gráfica que cuida de invocar os programas que compõem a ferramenta. Essas duas formas de se conduzir uma sessão de teste são descritas nas seções seguintes.

2 Interface Gráfica

A interface gráfica da *PROTEUM/IM* permite ao testador sem experiência explorar e aprender os conceitos sobre o teste baseado em mutação, Mutação de Interface e sobre a própria ferramenta. Ela também fornece melhores recursos para se visualizar o conjunto de casos de teste e os mutantes, facilitando dessa maneira algumas tarefas como a identificação de mutantes equivalentes. Uma visão geral das operações que podem ser realizadas através da interface é mostrada na Figura 1.

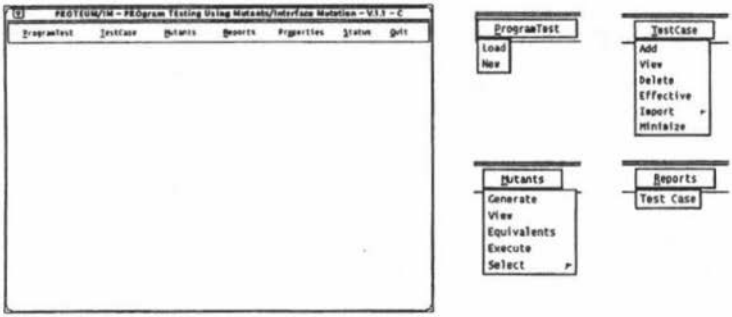

Figura 1: Operações disponíveis na interface da *PROTEUM/IM*

O primeiro passo necessário para que se conduza um sessão de teste é identificar qual base de teste deseja-se manipular. Caso trate-se de uma nova base de teste (que ainda não foi iniciada), deve-se então criá-la. Para isso o testador utiliza a operação *Program Test/New* e deve fornecer algumas informações, entre elas um nome que identificará a base de teste.

Uma vez identificada a base de teste a ser utilizada na sessão, operações podem ser realizadas, por exemplo, para construir o conjunto de teste a ser avaliado. A primeira dessas operações permite que o testador adicione interativamente casos de teste ao conjunto. Com essa funcionalidade, ativada através da operação *Test Case/Add*, o testador deve fornecer os dados de entrada para o caso de teste. A ferramenta executa o programa em teste com esses dados e exibe os resultados para o testador. O testador então faz o papel de oráculo e decide se o caso de teste deve ser incluído ou não. A ferramenta se encarrega de capturar os dados fornecidos pelo testador, bem como os resultados produzidos, caracterizando dessa maneira o caso de teste.

Outra maneira de incluir casos de teste é através de sua importação de arquivos que descrevem esses casos de teste. Existem três tipos de arquivos que podem ser importados: 1) outra base de casos de teste da própria *PROTEUM/IM*; 2) uma base de casos de teste da ferramenta de teste POKE-TOOL [1]; e 3) um par de arquivos ASCII que descrevem os parâmetros de linha de comando e os dados da entrada padrão.

Casos de teste podem ser retirados fisicamente da base de dados através da operação *Test Case/Delete*. A *PROTEUM/IM* também permite a exclusão lógica de casos de teste. Quando isso ocorre o caso de teste não é fisicamente removido da base mas apenas temporariamente

desabilitado, o que implica que não será usado nas próximas operações de execução dos mutantes. O testador pode também examinar o conjunto de casos de teste vigente. Isso é feito através da operação *Test Case/View*.

O ponto central no teste baseado em mutações é a manipulação dos mutantes. Os operadores de mutação implementados na PROTEUM/IM estão descritos em [4]. Para cada operador pode ser selecionada uma porcentagem de geração que representa a porcentagem de mutantes que serão efetivamente criados para aquele operador. Pode-se determinar também o número máximo de mutantes criados pelo operador em cada ponto de mutação.

Uma vez gerados os mutantes e criado o conjunto de casos de teste a ser avaliado, resta executar os mutantes com os casos de teste e comparar os seus resultados com os do programa original. A PROTEUM/IM executa um mutante com cada caso de teste até que um deles mate o mutante ou até que todos os casos de teste sejam usados com esse mutante. Passa-se então para o próximo mutante.

Após a execução dos mutantes, o testador pode analisar os mutantes vivos e deve decidir quais são equivalentes e quais não são e, nesse último caso, pode ainda selecionar novos casos de teste para matar tais mutantes.

O estado corrente de uma sessão pode ser obtido através de relatórios fornecidos pela ferramenta. O relatório resumo é exibido através da operação *Status*. Também existe um relatório com informações sobre o comportamento dos casos de teste em relação aos mutantes. O testador pode escolher quais informações devem ser exibidas nesse relatório.

A operação *Properties* permite ao testador configurar algumas características da ferramenta. Pode-se configurar o diretório corrente, onde deve estar o programa a ser testado e onde é criada a base de teste, e o valor de "timeout" para que um mutante seja distinguido através de seu tempo de execução.

Finalmente, a operação *Quit* faz com que o ambiente da interface gráfica seja abandonado.

3 Scripts de Teste

Executar uma sessão de teste através da interface gráfica é, provavelmente, mais fácil, porém menos flexível que através da chamada direta dos programas da PROTEUM/IM. A interface depende de constante interação do testador, pelo menos para passar-se de uma certa operação para a próxima. O uso de scripts de teste que chamam diretamente os programas da PROTEUM/IM pode evitar tal problema mas exige um esforço de programação e completo domínio, tanto dos conceitos sobre o teste baseado em mutação e o critério Mutação de Interface, quanto dos próprios programas que compõem a ferramenta.

As mesmas operações efetuadas através da interface gráfica podem ser efetuadas através da chamada direta dos programas. Uma completa descrição dos programas e todos os parâmetros pode ser encontrada no manual do usuário da PROTEUM/IM[3].

A criação de uma base de teste pode ser feita utilizando-se o programa *test-new*. As mesmas informações que devem ser fornecidas à interface gráfica devem ser passadas como parâmetros para esse programa. Por exemplo, para criar-se no diretório corrente uma base de teste chamada *myprog-1*, cujo programa fonte a ser testado é *myprog.c*, cujo executável é *myprog* e cujo comando de compilação é "*gcc myprog.c -o myprog -w*", pode-se utilizar o programa *test-new* da seguinte maneira:

```
test-new -D . -S myprog -E myprog -C "myprog.c -o myprog -w" myprog-1
```

Para incluir casos de teste interativamente pode-se usar o programa *tcase-add*. Os parâmetros de linha de comando, se existirem, são passados como parâmetros para o *tcase-add* e, assim como na interface gráfica, inicia-se a execução do programa em teste para que o testador forneça os

dados lidos através da entrada padrão. Para adicionar um caso de teste na base de teste *myprog-1* criada acima, pode-se utilizar o comando

```
tcase-add -p "-l -g 10 file1" myprog-1
```

Para a geração de mutantes há um outro programa. Ele não só aplica os operadores de mutação, criando os descritores dos mutantes, mas também se encarrega de incluir esses descritores na base de mutantes. Os parâmetros para esse programa são os nomes dos operadores de mutação para os quais deseja-se gerar mutantes e a porcentagem e número máximo de mutantes por ponto de mutação. Por exemplo:

```
muta-gen -I-DirVarRepPar 10 2 myprog-1
```

Esse comando irá criar mutantes para o operador "I-DirVarRepPar". A geração para esse operador fica limitada a, no máximo, 2 mutantes em cada ponto de mutação e além disso, entre esses mutantes selecionam-se apenas 10% que serão efetivamente gerados.

A execução dos mutantes é feita através do programa *exemuta*. Basta fornecer o nome da base de teste e o programa inicia a execução dos mutantes. Por exemplo, os comandos abaixo executam inicialmente os mutantes de 101 a 200, depois de 201 até o último mutante e depois de 0 a 100.

```
exemuta -exec -f 101 -t 200 myprog-1
exemuta -exec -f 201 myprog-1
exemuta -exec -t 100 myprog-1
```

O programa *report* cria o mesmo relatório citado na sessão sobre a interface gráfica. A escolha das informações que devem aparecer no relatório é feita através de opções passadas para o programa. O relatório é armazenado num arquivo ao invés de ser mostrado ao testador, como na interface gráfica.

4 Plataforma

A ferramenta PROTEUM/IM apóia o teste de programas em C. Está implementada para rodar em ambientes UNIX, tendo sido portada para os sistema SunOS4, SunOS5 e Linux. A interface gráfica é baseada nos programas Tcl/Tk [5].

Referências

[1] M. L. Chaim. *POKE-TOOL - Uma Ferramenta para Suporte ao Teste Estrutural de Programas Baseado em Análise de Fluxo de Dados.* Dissertação de Mestrado, DCA/FEE/UNICAMP, Campinas - SP, abril 1991.

[2] M. E. Delamaro. *"Mutação de Interface: Um Critério de Adequação Inter-procedimental para o Teste de Integração".* Tese de doutorado, IFSC - USP, São Carlos - SP, em preparação.

[3] M. E. Delamaro and J. C. Maldonado. "Proteum/IM - User's Guide". Relatório Técnico em preparação, ICMSC - USP, São Carlos - SP, abril 1997.

[4] M. E. Delamaro and J. C. Maldonado. Teste de Integração: Projeto de Operadores para o Critério Mutação de Interface. In *Submetido ao XI SBES*, Fortaleza - CE, Outubro 1997.

[5] E. F. Johnson. *Graphical Applications with Tcl & Tk.* M&T Books, New York, 1996.