PROTEUM/FSM :

ESPECIFICAÇÃO DE UMA FERRAMENTA PARA APOIAR A VALIDAÇÃO DE MÁQUINAS DE ESTADO FINITO PELO CRITÉRIO ANÁLISE DE MUTANTES

Sandra Camargo Pinto Ferraz Fabbri[1]
Marcio Eduardo Delamaro[2]
José Carlos Maldonado[3]
Paulo Cesar Masiero[3]

1 Doutoranda do IFQSC - USP - São Carlos; Profª do Depto de Computação - UFSCar - Cx.P 676

2 Doutorando do IFQSC - USP - São Carlos - Cx.P 668

3 Professores do. ICMSC - USP - São Carlos - Cx.P 668

End. Eletrônico - JCMALDON@ICMSC.USP.BR

RESUMO

Pesquisas têm sido conduzidas objetivando a caracterização de critérios de geração de seqüências de teste para apoiar atividades de validação de especificações baseadas em Máquinas de Estado Finito (MEF) [CHO78, FUJ91, NAI81, SAB88, GON70]. Mais recentemente, Fabbri et alli exploraram a utilização dos conceitos e princípios da Análise de Mutantes, originalmente aplicada para o teste de programas, para a validação de especificações baseadas em MEF. Este artigo tem por objetivo discutir a especificação de uma ferramenta de teste baseada no critério Análise de Mutantes para apoiar a validação de MEF. São apresentados, resumidamente, os conceitos do critério Análise de Mutantes, os conceitos de MEF e alguns resultados obtidos da aplicação manual desse critério na validação de uma MEF [FAB93]. A especificação da Proteum/FSM é fortemente baseada na ferramenta multilinguagem denominada Proteum [DEL93], que apóia, na versão atual, o teste de programas implementados em C, pelo critério Análise de Mutantes e no ambiente Statsim [MAS91] que fornece recursos para edição, especificação e simulação de Statechart [HAR87].

ABSTRACT

Researches have been conducted aiming at establishing test sequences generation to support validation activities of Finite State Machine (FSM) based specifications [CHO78, FUJ91, NAI81, SAB88, GON70]. More recently, Fabbri et alli have explored

the use of principles and concepts of Mutation Analysis, originally used to program testing, to validate FSM based specifications. This paper presents the specification of a testing tool, based on the Mutation Analysis criterion to support validation of FSM. The Mutation Analysis and FSM concepts and some results obtained with the manual application of this criterion to validate a FSM [FAB93] are presented briefly. The Proteum/FSM specification is strongly based on the multilanguage tool named Proteum [DEL93] that supports, in the actual version, C program testing, by the Mutation Analysis criterion and on the Statsim [MAS91]environment that supports edition, specification and simulation of Statechart [HAR87].

1. INTRODUÇÃO

Durante o desenvolvimento de software um conjunto de atividades denominado Garantia de Qualidade de Software deve ser realizado, no qual a atividade de Teste é extremamente relevante, porém onerosa [PRE92]. O Teste possui como principal objetivo revelar a presença de erros e é a atividade final de validação e verificação dentro do processo de desenvolvimento de software.

Com a evolução da tecnologia de hardware, observa-se um crescente uso de sistemas de software nas mais variadas áreas, provocando uma demanda cada vez maior por sistemas reativos, cuja característica principal é reagir a estímulos internos/externos. Pelo fato das conseqüências de falhas desses sistemas poderem envolver riscos à vida humana, perdas econômicas, etc., as atividades de teste e validação dos mesmos tornam-se ainda mais críticas e fundamentais, além de mais difíceis devido às características desses sistemas, como a concorrência, comunicação e sincronização.

Um ponto bastante importante no desenvolvimento de sistemas reativos é a especificação e validação do seu aspecto comportamental. Várias técnicas de especificação são utilizadas para esse fim, podendo-se citar, dentre as técnicas gráficas, Máquinas de Estado Finito (MEF) [GIL62], Redes de Petri [PET81] e Statechart [HAR88]. Também, vários mecanismos de teste e validação têm sido propostos para auxiliar o desenvolvimento de tais sistemas [LEV87, CHO78, BOA92].

Devido a aspectos de produtividade e qualidade, a utilização de ferramentas na atividade de teste de software é de extrema importância e necessidade; deve-se observar ainda, que a atividade de teste e validação do aspecto comportamental dos sistemas reativos é bastante deficiente quanto à disponibilidade de ferramentas que a apóiem [DAV88]. Assim, o objetivo deste trabalho, dado que Fabbri et alli [FAB93] obtiveram evidências de que o critério Análise de Mutantes é um mecanismo complementar de teste e validação de sistemas especificados através de MEF, é discutir os principais aspectos

da especificação de uma ferramenta de teste, denominada Proteum/FSM, para apoiar o teste e validação de sistemas especificados através de MEF, com base no critério Análise de Mutantes. A Proteum/FSM é derivada de uma outra ferramenta, denominada Proteum [DEL93], que apóia o teste e validação de programas implementados em C, também com base no critério Análise de Mutantes. A Proteum/FSM, para a especificação e simulação das MEFs, utiliza recursos e conceitos disponíveis no ambiente Statsim [MAS91], que apóia a edição, especificação e simulação de Statechart. A especificação e implementação da Proteum/FSM visa a fornecer facilidades para a avaliação do custo de aplicação do critério Análise de Mutantes para o teste e validação de especificações baseadas em MEF, assim como a comparação entre esse critério e critérios de geração de seqüências de teste, através da condução de estudos empíricos.

Na Seção 2 apresenta-se um breve resumo dos resultados obtidos em [FAB93]; na Seção 3 apresentam-se a ferramenta Proteum para o teste de programas sequenciais em C e uma visão geral do ambiente Statsim, com ênfase nos módulos de edição e simulação de Statechart; na Seção 4 apresentam-se aspectos da especificação da ferramenta Proteum/FSM para o teste de especificações basedas em MEFs e, na Seção 5 apresentam-se as conclusões e desdobramentos deste trabalho.

2. ANÁLISE DE MUTANTES BASEADA EM MEF

A Análise de Mutantes [DEM78] é um dos critérios da técnica de teste baseada em erros que tem por objetivo aumentar a confiança de que um programa P, que se quer testar, esteja correto. Para isso, o programa P é executado com um conjunto de casos de teste T que também é usado para executar os mutantes de P - programas gerados a partir de P que possuem uma alteração sintática. As alterações sintáticas aplicadas em P formam o conjunto de operadores de mutação e refletem os erros típicos e comuns cometidos no processo de programação. Ao executar-se cada mutante com o conjunto de casos de teste T, ou o comportamento do mutante é diferente do de P - e então diz-se que o mutante está morto - ou o comportamento do mutante é igual ao de P - e então diz-se que o mutante está vivo. Neste último caso, ou o mutante é equivalente a P (e portanto não haverá nenhum caso de teste capaz de diferenciá-lo de P), ou o conjunto de casos de teste deve ser melhorado para que o comportamento de P e do mutante possa ser distinguido. Aplicando-se esse procedimento iterativamente, na tentativa de matar o maior número possível de mutantes não equivalentes, está-se melhorando a qualidade do conjunto de casos de teste e, portanto, a confiança de que P esteja correto. Tal procedimento é avaliado através do escore de mutação, que corresponde à taxa: (nº de mutantes mortos) / (nº total de mutantes - nº de mutantes equivalentes).

MEF é uma técnica usada na especificação do aspecto comportamental de sistemas. Segundo Fujiwara [FUJ91], uma Máquina de Estado Finito M determinística pode ser representada por uma quíntupla (X,E,Y,T,O), onde:

X: conjunto de entradas, x.
E: conjunto de estados S_i, incluindo um estado S_0 chamado estado inicial.
Y: conjunto de saídas, y, incluindo a saída nula (-).
T: função de transição, X × E -> E.
O: função de saída, X × E -> Y.

A máquina M é dita completamente especificada se para cada estado de M existe uma transição para cada símbolo de entrada em X. A máquina M é fortemente conectada se para cada par de estados (S_i,S_j) existe uma seqüência de entrada que faz M transicionar de S_i para S_j. A máquina M é dita minimal se o número de estados em M é menor ou igual ao número de estados para qualquer máquina M', que seja equivalente a M, isto é, que responda com uma saída idêntica para cada seqüência de entrada.

Para validação de uma MEF, existem vários métodos de geração de seqüências de teste, como por exemplo, os métodos W, Wp e UIO, dentre outros [CHO78, FUJ91, GON70, NAI81, SAB88]. No entanto, para a aplicação desses métodos, requer-se que a MEF satisfaça uma série de requisitos, tais como, ser minimal, completamente especificada, fortemente conectada, etc. o que, na prática, normalmente não acontece, tornando as seqüências de teste menos eficazes.

Assim, considerando-se esse problema e também que as técnicas de teste funcional, estrutural e baseada em erros são complementares [PRE92], aplicou-se a Análise de Mutantes na validação de uma MEF, extraída de [GAB90] - Figura 2.1, utilizando-se para teste seqüências geradas pelo método VT [NAI81] e pelo método W [CHO78], também propostas em [GAB90]. Deve-se observar que a MEF utilizada não satisfazia todos os requisitos exigidos pelos referidos métodos.

Para a aplicação do critério no contexto de MEF foi necessário definir-se um conjunto de operadores de mutação correspondente aos erros típicos cometidos por um especificador. Da aplicação desses operadores sobre a especificação original obtém-se, semelhantemente ao obtido para programas, os mutantes de ordem k (ou k-mutante), onde k é o número de operadores aplicados. Então, mutantes de ordem 1 são aqueles gerados pela aplicação de um operador de mutação sobre a especificação original; de ordem 2, aqueles gerados pela aplicação de 2 operadores e assim por diante. Tal conjunto compôs-se, preliminarmente, dos seguintes operadores: falta de arco, alteração do estado inicial, evento trocado, evento faltando, evento extra, estado extra, estado faltando, saída trocada e saída faltando.

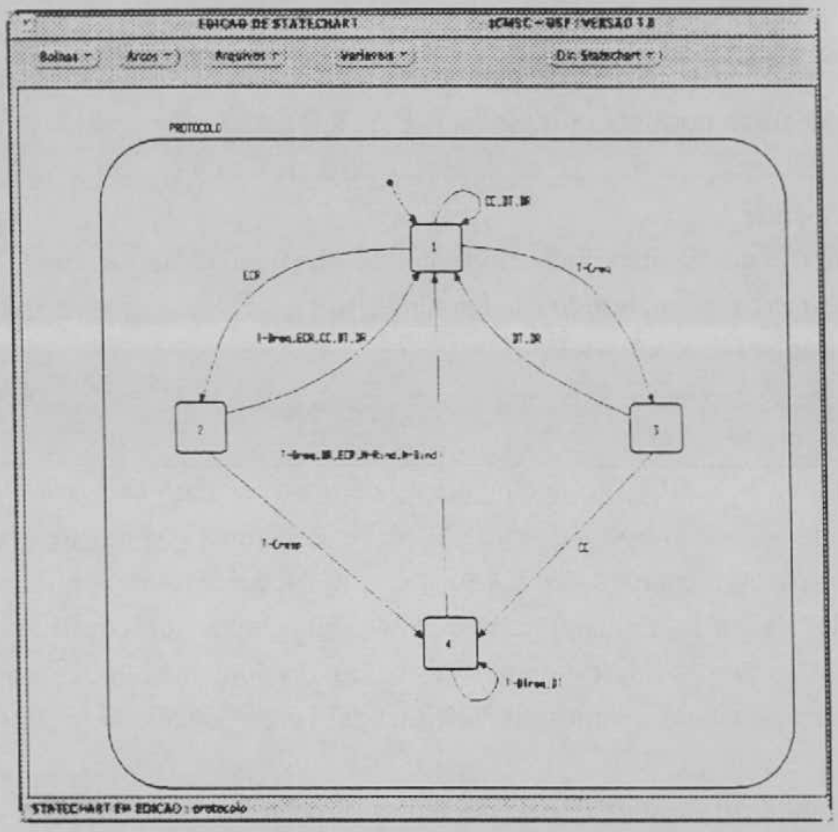


Figura 2.1 - MEF que especifica um Protocolo de Transporte

Os resultados obtidos da aplicação do critério com base nesses operadores foram provenientes de uma análise do comportamento da MEF original e das MEFs mutantes sob dois pontos de vista - a seqüência de estados e a saída gerada, onde pôde-se observar que a última levou a um melhor escore de mutação. A decisão de que informação utilizar para os resultados é um ponto relevante para a aplicação da Análise de Mutantes e que deverá ser explorado em estudos posteriores.

Além disso, observou-se que para as MEFs mutantes de ordem 1, o escore de mutação foi, de um modo geral, relativamente alto, exceto para operadores que exploraram características não satisfeitas pela MEF, como é o caso do operador evento-trocado que explora a não completude da máquina. Ou seja, mesmo para MEFs mutantes de ordem 1, as seqüências geradas pelos métodos mencionados anteriormente não produziram um escore de mutação de 100%, o que poderia ser esperado, no caso do método W, se todos os requisitos exigidos pelo método fossem satisfeitos pela MEF.

O aspecto complementar do critério Análise de Mutantes torna-se ainda mais evidente ao gerar-se mutantes de ordem 2. Esse é o caso do exemplo que segue - Figura 2.2, onde na implementação, para efeito de simplicidade e manutenibilidade, a condição de erro foi isolada, criando-se um estado extra (mutante de ordem 1) e, sobre essa máquina mutante, ao aplicar-se outro operador de mutação, arco faltando, obteve-se

uma MEF mutante de ordem 2 que, se analisada apenas para exemplo, com base no estado extra, fornece um escore de mutação de 25%, que é bastante baixo. Isso significa que em determinadas situações, como é o caso, as seqüências de teste se tornam ainda menos eficazes.

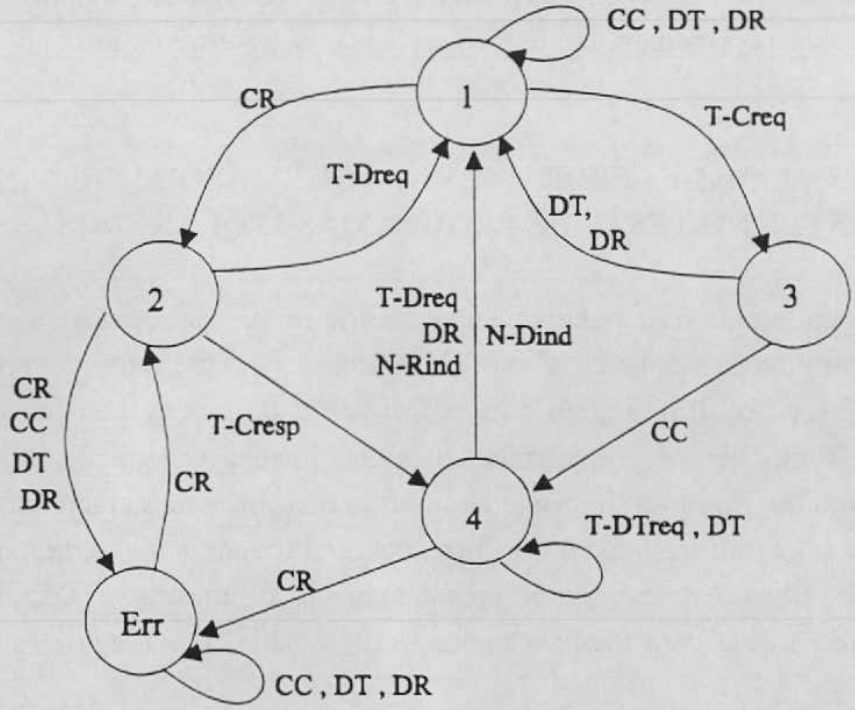


Figura 2.2 - Mutante de Ordem 2 da MEF da Fig. 2.1

Assim, a implementação de uma ferramenta de suporte ao critério Análise de Mutantes é fundamental para viabilizar a realização de outros experimentos empíricos, tornando possível então a validação (ou não) das evidências e hipóteses presentes, assim como a avaliação do custo de aplicação desse critério no contexto de MEF. Na seção seguinte, apresentam-se a ferramenta de teste Proteum [DEL93] e uma visão geral do ambiente Statsim [MAS91], nos quais a especificação da Proteum/FSM é baseada.

3. A FERRAMENTA PROTEUM E O AMBIENTE STATSIM

O Grupo de Engenharia de Software do ICMSC-USP, dentro de suas linhas de pesquisa, vem desenvolvendo o ambiente Statsim [MAS91], que apóia a especificação de sistemas através da técnica Statechart, e ferramentas que apóiam a atividade de teste e validação de sistemas, como por exemplo, as ferramentas POKE-TOOL [MAL91] e Proteum [DEL93].

Considerando-se que a técnica Statechart é uma extensão de MEF e que o critério Análise de Mutantes mostrou-se, num primeiro estudo, ser uma forma complementar de teste no contexto de MEF, para definir a ferramenta discutida neste trabalho, procurou-se reutilizar os recursos já disponíveis. Com esse intuito, são utilizados o ambiente Statsim, restringindo-se a técnica Statechart a MEF, de forma a possibilitar a edição e simulação de sistemas baseados nesta última técnica, como também a ferramenta Proteum, que apóia o critério Análise de Mutantes. Em seguida, são apresentadas as principais características dessas duas ferramentas.

3.1. PROTEUM - UMA FERRAMENTA QUE APÓIA A ANÁLISE DE MUTANTES NO TESTE DE PROGRAMAS SEQÜENCIAIS

Proteum é uma ferramenta que apóia o critério de teste Análise de Mutantes para o teste de programas sequenciais em C. Uma da características mais relevantes dessa ferramenta é ser **multilinguagem** pois possui partes do código construídas de maneira genérica, de forma que não fique vinculada a uma linguagem específica, podendo assim, ser configurada para outras linguagens de programação. Esta característica facilita sua instanciação para outras linguagens, fazendo do Proteum uma ferramenta propícia à reusabilidade. Ela foi desenvolvida em um trabalho de mestrado [DEL93] e apóia as atividades necessárias para a aplicação desse critério através das seguintes tarefas:

- definição de casos de teste;
- execução do programa em teste;
- geração de mutantes;
- execução dos mutantes;
- análise dos mutantes vivos;
- cálculo do escore de mutação.

Simplificadamente, a ferramenta mantém uma base de dados com informação sobre casos de teste e sobre os mutantes e seus estados. O usuário fornece o programa a ser testado e atua sobre essa base de dados editando o conjunto de casos de teste a ser utilizado, gerando os mutantes, executando-os e eliminando aqueles mutantes equivalentes e também solicitando relatórios sobre o estado do teste que está sendo conduzido.

A ferramenta foi desenvolvida de forma a permitir o trabalho com **sessões de teste**, através das funções *Criar Teste* e *Carregar Teste*, permitindo que o usuário inicie o teste de um programa, encerre a atividade quando desejado e depois retome-a do ponto onde havia parado; para isso os estados intermediários da aplicação do teste são guardados na base de dados para que possam ser recuperados posteriormente.

A definição de um conjunto de casos de teste T, na ferramenta, fica por conta do testador, através das funções *Incluir Casos de Teste* e *Excluir Casos de Teste*, embora seja possível apoiá-la de forma automatizada, com base no critério Análise de Mutantes [DEM91]. O Proteum permite, então, que o conjunto de casos de teste seja modificado dinamicamente, possibilitando a adição e remoção de casos de teste, à medida em que eles se mostrem pouco efetivos na eliminação dos mutantes, podendo, então ser substituídos por outros.

A função *Habilitar ou Desabilitar Casos de Teste* facilita ainda mais a seleção de bons casos de teste, pois permite que estes sejam excluídos logicamente (desabilitados) ou reincluídos no conjunto de casos de teste que está sendo utilizado, sem que seja preciso realizar as tarefas de inclusão e exclusão dos mesmos.

A geração dos mutantes é feita através da função *Gerar Mutantes*, que está baseada no conjunto de operadores de mutação definido. Cada operador modela uma classe específica de erro e, para que o testador possa escolher qual classe ele deseja priorizar, o Proteum permite a seleção dos operadores que serão utilizados na geração dos mutantes. Além disto, tal geração pode ser feita em etapas, permitindo ao testador lidar com o problema de construir casos de teste adequados, de forma incremental.

A execução e comparação dos resultados dos mutantes com o programa em teste é realizada através da função *Executar Mutantes*, de forma automática, sem a intervenção do testador. A execução dos mutantes reflete a qualidade do conjunto de casos de teste usado naquela execução, ou seja, o estado dos mutantes é coerente ao conjunto de teste habilitado naquele momento, fazendo com que sejam reconsiderados os mutantes que haviam sido mortos por casos de teste desabilitados na execução atual.

Finalmente, a função *Marcar Mutante Equivalente* permite que o usuário compare o programa fonte original com o mutante fonte a fim de decidir sobre a equivalência dos programas, já que esse é , em geral, um problema indecidível e portanto, difícil de ser automatizado [BUD81].

Outras características relevantes do Proteum são:

- possui **Interface** amigável : como a atividade de teste é uma das que mais consomem tempo e recursos no processo de desenvolvimento de software, é fundamental que uma ferramenta de teste, tentando facilitar a tarefa do testador, possua uma interface amigável. A interface do Proteum com o usuário foi construída utilizando-se o XView (X-Window-system-based Visual/Integrated Environment for Workstations) [HEL91], uma biblioteca para construção de interfaces gráficas interativas que rodam sob o sistema Openwindow em uma estação Sun. O resultado é uma interface gráfica onde o

usuário pode manipular janelas, ícones e um "mouse" para selecionar as operações que deseja realizar [DEL93].

- permite o **Teste de Subprogramas** : embora o critério Análise de Mutantes seja mais indicado para o teste de unidade, a ferramenta permite testar uma ou algumas funções que compõem a unidade, fornecendo-se seus nomes ao iniciar-se o teste.

- é um **Ambiente Compilado** : a execução do programa original e dos mutantes é feita através de código executável, reduzindo o tempo de execução dos mutantes, o que a torna mais eficiente.

Na arquitetura da Proteum (Figura 3.1.1) identificam-se três grandes módulos :

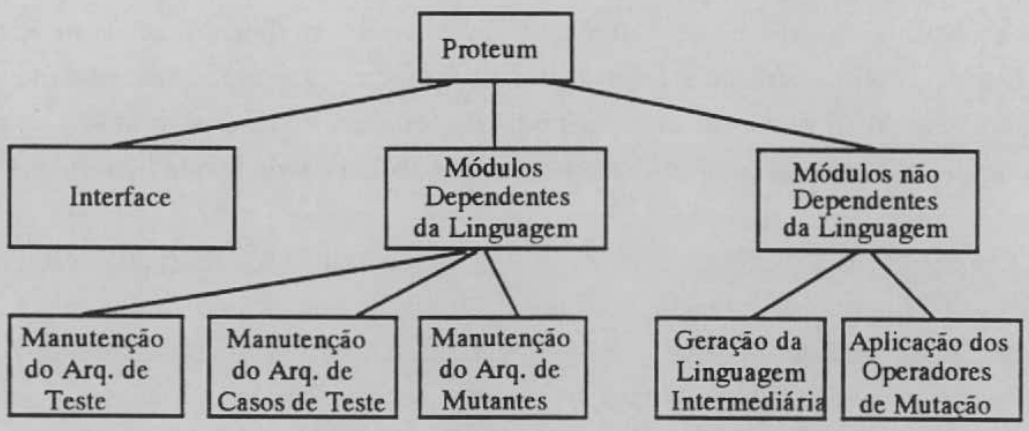


Figura 3.1.1 - Módulos do Proteum

- Interface : possui as características já citadas, é baseada em janelas e, pelo fato de seus módulos serem completamente dependentes do ambiente, eles foram mantidos isolados dos demais módulos da ferramenta.

- Módulos Dependentes da Linguagem : são aqueles que devem ser alterados para que a ferramenta seja instanciada para apoiar o teste de programas na linguagem alvo.

- Módulos não Dependentes da Linguagem : são aqueles que não dependem da linguagem alvo como, por exemplo, a criação de novo teste, a edição de casos de teste e a execução dos mutantes.

3.2. STATSIM - UM AMBIENTE QUE APÓIA A UTILIZAÇÃO DA TÉCNICA STATECHART PARA A ESPECIFICAÇÃO DE SISTEMAS

O ambiente Statsim [MAS91] apóia a técnica Statechart [HAR87], utilizada na especificação do aspecto comportamental de sistemas reativos. Os principais módulos desse ambiente são: um editor gráfico de statecharts, um analisador para a linguagem de especificação de statecharts (LES) e ferramentas para a simulação da especificação.

Os módulos de interesse para este trabalho são o *Editor Gráfico de Statechart*, que possibilita a especificação do sistema, de forma gráfica (como exemplo de utilização desse módulo, vide Figura 2.1 que foi elaborada no ambiente Statsim através desse Editor); o *Analisador da LES*, que realiza a análise sintática quando a especificação do sistema é feita de forma textual e o *Simulador*, que permite a simulação da especificação, quer ela esteja representada graficamente ou textualmente.

Como a técnica Statechart é uma extensão de MEF, o ambiente Statsim possui recursos para a especificação de todos os aspectos que diferenciam essa técnica das MEFs, como os aspectos de hierarquia, concorrência, comunicação e história. Então, a utilização desse ambiente no contexto de MEF, deve ser feita de maneira apropriada, restringindo-o para esta outra técnica. Para tanto, definiu-se um subconjunto da LES, voltado para MEF, de forma que o ambiente passa a editar e simular especificações de sistemas baseadas nesta técnica. Ressalta-se que os módulos não são afetados por essa restrição. A Figura 3.2.1 mostra a especificação do Protocolo, apresentado anteriormente na Figura 2.1, através da LES restrita ao contexto de MEF.

```
estado protocolo ou
subestados estado1 def,estado2,estado3,estado4
estado estado1 atomo
estado estado2 atomo
estado estado3 atomo
estado estado4 atomo ;
evento cc origem estado1 destino estado1
evento cc origem estado2 destino estado1
evento cc origem estado3 destino estado4
evento dt origem estado1 destino estado1
evento dt origem estado2 destino estado1
evento dt origem estado3 destino estado1
evento dt origem estado4 destino estado4
evento dr origem estado1 destino estado1
evento dr origem estado2 destino estado1
evento dr origem estado3 destino estado1
evento dr origem estado4 destino estado1
evento ecr origem estado1 destino estado2
evento ecr origem estado2 destino estado1
evento ecr origem estado4 destino estado1
evento T_Dreq origem estado2 destino estado1
evento T_Dreq origem estado4 destino estado1
evento T_Creq origem estado1 destino estado3
evento T_Cresp origem estado2 destino estado4
evento T_DTreq origem estado4 destino estado4
evento N_Rind origem estado4 destino estado1
evento N_Dind origem estado4 destino estado1 ;
```

Figura 3.2.1 - Especificação do Protocolo da Fig. 2.1 através da LES restrita a MEF

Através da Figura 3.2.1 pode-se verificar que o conjunto das declarações básicas da LES, quando restrita a MEF, é formado pela descrição dos estados e dos eventos. Na descrição dos estados, para o exemplo em questão, está declarado o estado *protocolo* que é do tipo *ou* , composto pelos estados *estado1*, *estado2*, *estado3* e *estado4*, que por sua vez são todos *átomos*, isto é, indecomponíveis, tendo como estado inicial - *def* - o *estado1*. Na descrição dos eventos tem-se o nome do evento seguindo a palavra *evento*, o nome do estado origem daquele evento, seguindo a palavra *origem* e o nome do estado destino daquele evento, seguindo a palavra *destino*. Além dessas declarações, o subconjunto da LES restita a MEF, possui algumas outras que não foram utilizadas no exemplo em questão.

Um trecho da simulação desse Protocolo, quando especificado textualmente através da LES, é apresentado na Figura 3.2.2. A simulação ajuda a validar preliminarmente o modelo aumentando a confiança em que ele esteja correto ou próximo disso. O teste do modelo pela Análise de Mutantes complementa essa validação.

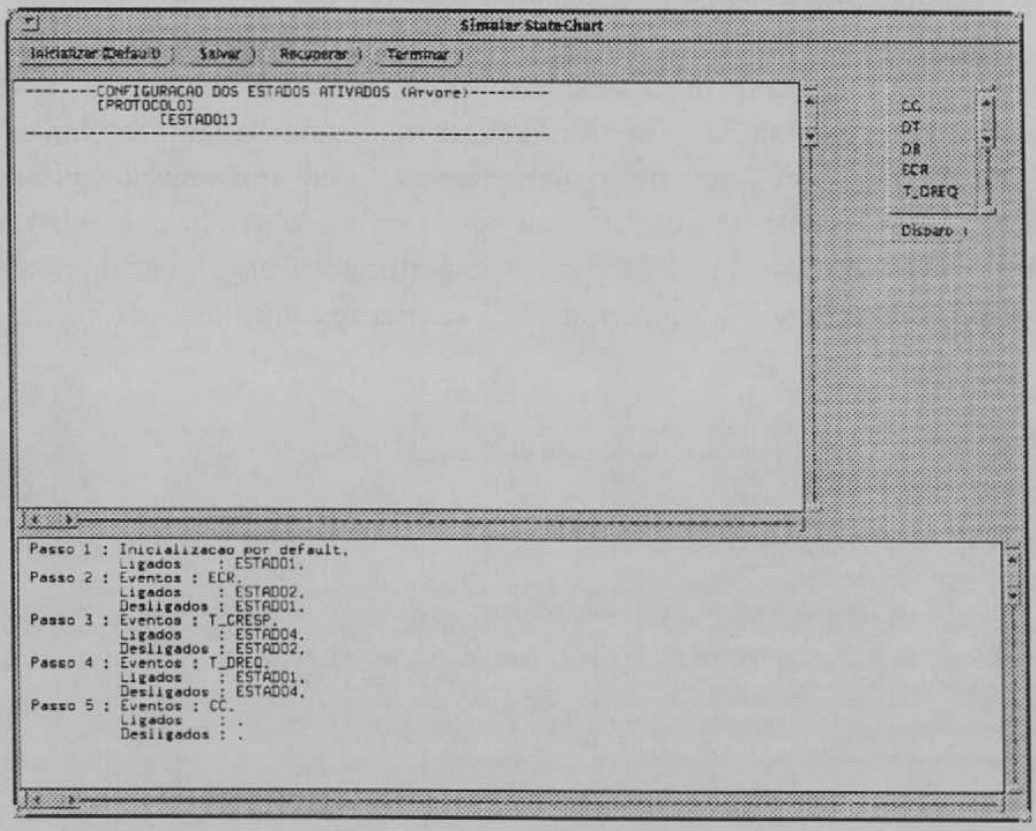


Figura 3.2.2 - Trecho da Simulação do Protocolo

Na simulação, as informações apresentadas para o usuário são: um quadro com a relação dos eventos declarados na especificação LES da MEF, para que o usuário possa dispará-los à medida do desejado; um quadro com a configuração dos estados que estão

ativados e um outro quadro com a seqüência de execução da MEF, que mostra qual o estado ativado (ligado) num determinado passo, qual o estado que foi desativado (desligado) naquele passo e qual o evento que ocorreu.

Da mesma forma, se a especificação for feita graficamente, ela também pode ser simulada, salientando-se que, nesse caso, o sistema produz a LES correspondente a esta especificação. Ou seja, de qualquer maneira, tem-se disponível a especificação do sistema através da LES, que é o "programa" em teste na ferramenta Proteum/FSM. Na simulação, o sistema gera um arquivo Log, que contém a seqüência de execução do Statechart (ou da MEF), o qual é utilizado como uma das formas de comparação entre o comportamento da MEF original e das MEF's mutantes.

4. PROTEUM/FSM - UMA FERRAMENTA PARA APOIAR A ANÁLISE DE MUTANTES NO CONTEXTO DE MÁQUINAS DE ESTADO FINITO

Os aspectos funcionais, operacionais e de implementação da Proteum/FSM, apresentados em seguida, são praticamente os mesmos disponíveis na Proteum/versão C, pois a ferramenta apóia o mesmo critério, apesar de contextos diferentes.

4.1. ASPECTOS FUNCIONAIS

As funções disponíveis na Proteum/FSM são aquelas necessárias para apoiar o critério Análise de Mutantes no contexto de MEF. Os principais pontos que a diferenciam da Proteum são: a função *Editar Especificação de MEF*, que é realizada fundamentalmente através dos recursos disponíveis no ambiente Statsim e a função *Gerar Seqüências de Teste*, que corresponde a um trabalho de mestrado em andamento [NAG94], cujo objetivo é gerar seqüências típicas para o teste de MEF. O Diagrama de Fluxo de Dados da Figura 4.1.1. mostra as funções da Proteum/FSM.

4.2. ASPECTOS OPERACIONAIS

Em seguida, os aspectos operacionais da Proteum/FSM são exemplificados utilizando-se as telas que compõem a interface com o usuário.

Na Figura 4.2.1a tem-se o Menu Principal com uma nova Sessão de Teste sendo criada, através da seleção das opções *FSM Test* e *New*.

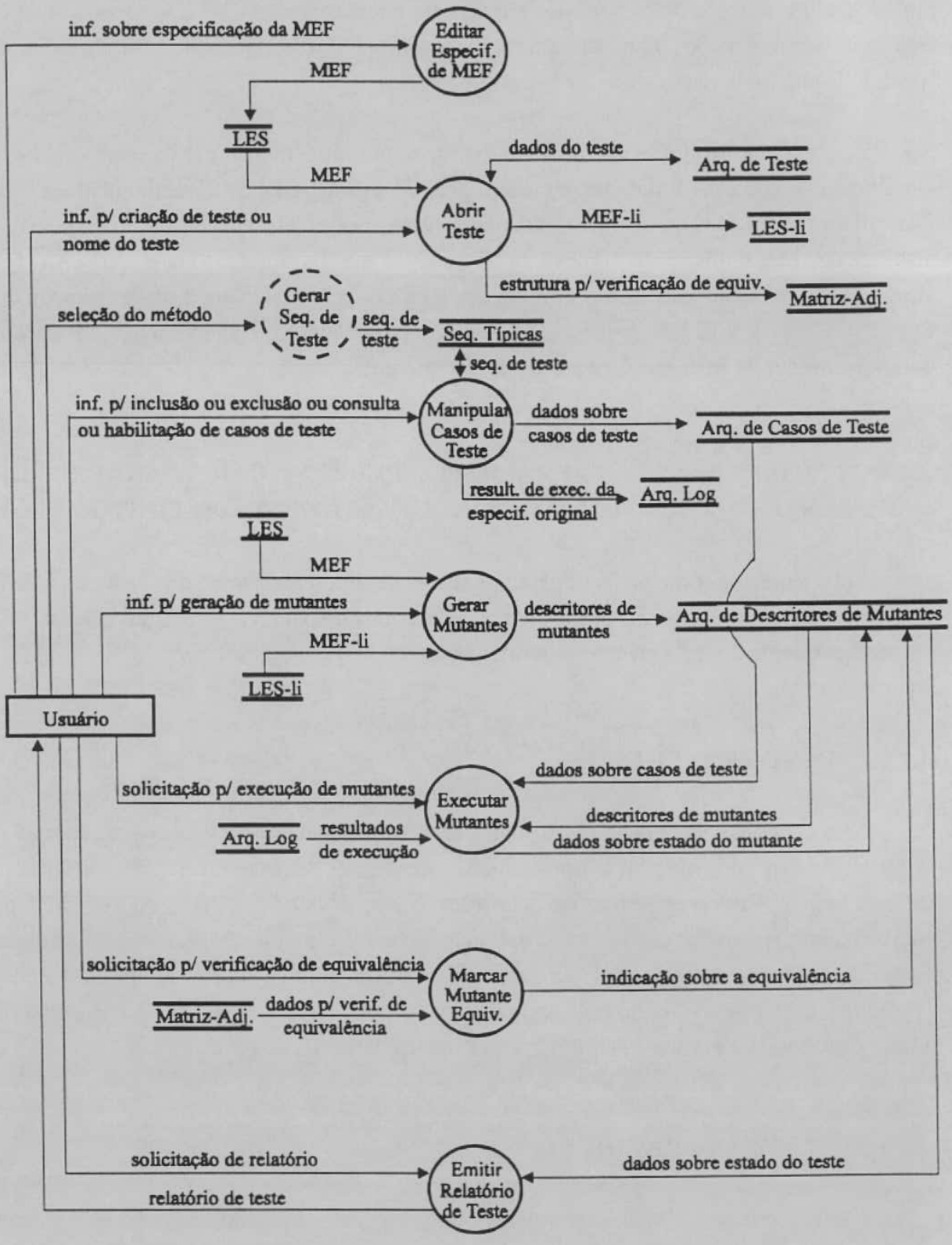


Figura 4.1.1 - DFD da Ferramenta Proteum/FSM

Em resposta à seleção de tais opções, abre-se uma janela - Figura 4.2.1b - onde são fornecidos os dados necessários para a criação de uma nova sessão de teste, como o diretório onde está armazenada a descrição da MEF que será testada e o nome que se deseja dar ao teste que está sendo criado. Ao se confirmar tais dados selecionando-se o botão *Confirm*, será chamado o módulo "*Analisar*" (do ambiente Statsim), o qual além de analisar a LES fornecida irá gerar a LES-li (LES estendida) e as estruturas de dados que serão usadas na implementação do algoritmo de equivalência de MEF [GIL62].

A opção *Generate* da opção *Mutants* do Menu Principal - Figura 4.2.1c - irá abrir uma janela com a relação das classes de erros para MEFs, de acordo com [CHO78], segundo as quais, os operadores de mutação para MEFs foram agrupados - Figura 4.2.1d. Nesse instante, pode-se escolher a porcentagem de mutantes que se deseja gerar de cada uma dessas classes e, selecionando-se os botões correspondentes a tais classes, é possível ser ainda mais criterioso no tipo de erro que se deseja explorar, escolhendo-se porcentagens individuais para cada operador separadamente. Ao selecionar-se o botão *Confirm*, é dado início à geração dos mutantes. Essas facilidades viabilizam o estabelecimento e o estudo de métodos/critérios alternativos da Análise de Mutantes para MEF, a exemplo dos estudos conduzidos por Mathur & Wong [MAT93].

Depois de gerados os mutantes, para que eles possam ser executados, deve-se criar casos de teste, isto é, seqüências de entradas que testem o comportamento da MEF. A opção *Test Case* do Menu Principal trata desse aspecto - Figura 4.2.1e. A opção *Add* permite que casos de teste sejam adicionados na Base de Dados; isso pode ser feito manualmente ou em batch, de forma que o usuário forneça cada seqüência de teste ou crie um arquivo com várias seqüências quando isso for necessário, como é o caso das seqüências geradas pelo método W [CHO78], que são compostas por várias subseqüências. A opção *View* permite que se veja os casos de teste armazenados na Base de Dados. A opção *Delete* permite que os casos de teste sejam removidos da Base de Dados. A opção *Generate* permite que sejam geradas seqüências de teste segundo os métodos utilizados para validação de MEFs [CHO78, FUJ91, NAI81, SAB88, GON70]; esse módulo está sendo desenvolvido por um trabalho de mestrado [NAG94] e será acoplado à ferramenta. A opção *Import* permite que casos de teste sejam importados de outras ferramentas.

Quando é escolhida a opção *Add*, descrita anteriomente, a ferramenta inicia a simulação da MEF original. Nesse ponto, cabe ao testador analisar o resultado dessa simulação para poder confirmar a inclusão desse caso de teste, caso o comportamento da MEF esteja de acordo com o esperado, ou cancelar a inclusão, caso contrário.

Se confirmada a inclusão, pode-se então executar os mutantes, ou seja, aplicar as seqüências de teste às MEFs mutantes. Para tanto, deve ser selecionada a opção *Execute*, que é uma das opções de *Mutants* do Menu Principal. A ferramenta inicia a

execução dos mutantes com os casos de teste que estão habilitados no momento, não repetindo a execução para casos de teste que já tenham sido usados em outra sessão de teste e que estejam habilitados naquela sessão. O resultado da execução dos mutantes é analisado através da comparação da saída gerada por eles com a saída gerada pela MEF original, ou então, pela seqüência de estados percorridos durante a execução. À medida em que os mutantes vão sendo executados, a ferramenta exibe o número do mutante e, como essa tarefa pode ser demorada, o usuário pode cancelar a execução através do botão *Cancel*, sendo mantidas as informações pertinentes aos mutantes já executados até então.

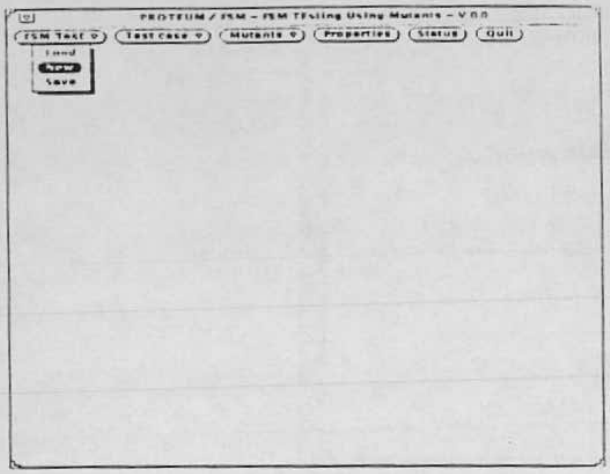
Quando desejado e, particularmente após a execução de mutantes, com um conjunto de casos de teste selecionado, pode-se verificar o estado do teste através da opção *Status* do Menu Principal. A tela apresentada - Figura 4.2.1f - possui uma série de informações acerca desse teste, como por exemplo, o número de mutantes gerados até então, o número de mutantes equivalentes, o número de mutantes que permanecem vivos com base no conjunto de casos de teste utilizado na execução, o número de mutantes que foram executados com aquele conjunto de casos de teste e o escore de mutação.

O usuário pode encerrar a sessão de teste em andamento através da opção *Quit*. Então, ele pode escolher entre salvar as operações executadas na sessão de teste - através do botão *Save*, ou ignorá-las, não salvando nada sobre a sessão realizada - através do botão *Abort*, ou então continuar a sessão de teste - através do botão *Cancel* - Figura 4.2.1g.

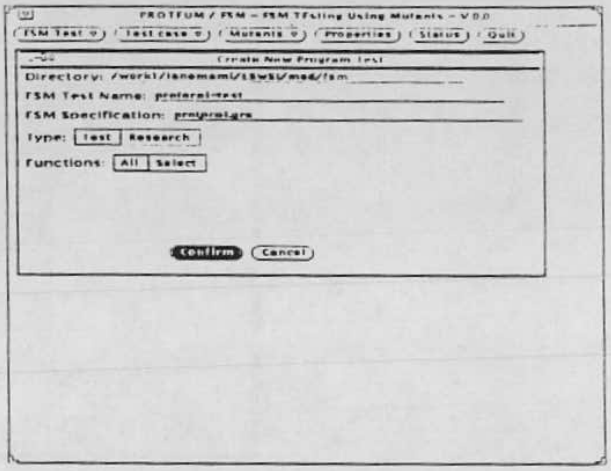
4.3. ASPECTOS DE IMPLEMENTAÇÃO

A implementação da ferramenta Proteum/FSM consiste, essencialmente, na integração e adequação dos módulos já disponíveis e operacionais da ferramenta Proteum e do ambiente Statsim.

Quanto aos módulos do ambiente Statsim que serão utilizados, isto é, o Editor Gráfico, o Analisador e o Simulador, as alterações são simples e imediatas. O primeiro aspecto foi caracterizar um subconjunto da LES para tratar especificações baseadas em MEF, como ilustrado no exemplo fornecido anteriormente. Ressalta-se que esta restrição é totalmente transparente a esses módulos, uma vez que cada componente básico de um Statechart é, em essência, uma MEF. Outro aspecto que já está sendo implementado é a saída gerada quando da execução de uma MEF, para que o comportamento da MEF original possa ser comparado aos das MEFs mutantes através desta informação além da seqüência de execução, cuja informação já é gerada, atualmente, dentro do ambiente.

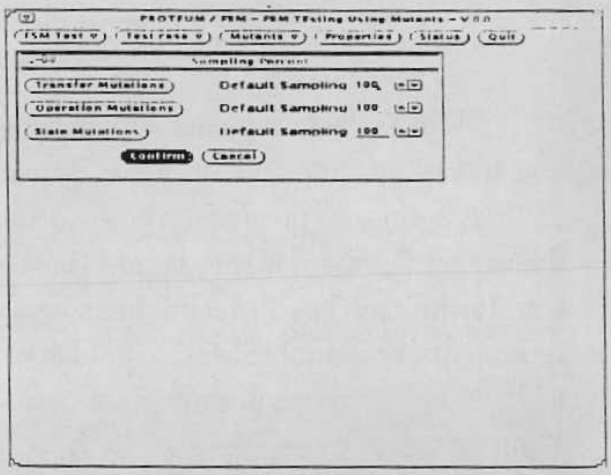


a - Menu Principal e Opção para Criar Teste

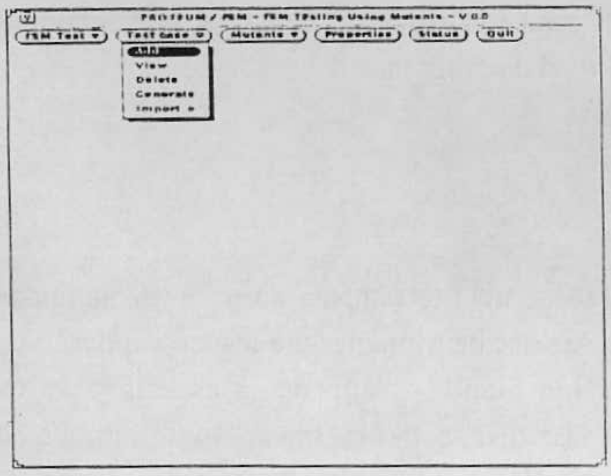


b - Confirmando a Criação de um Teste

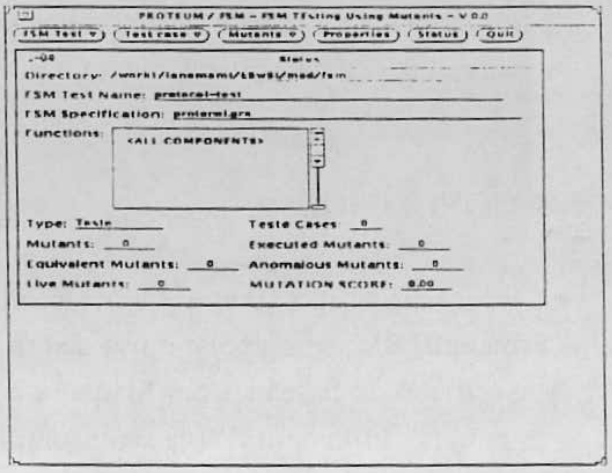
c - Menu de Mutantes



d - Menu para Geração de Mutantes

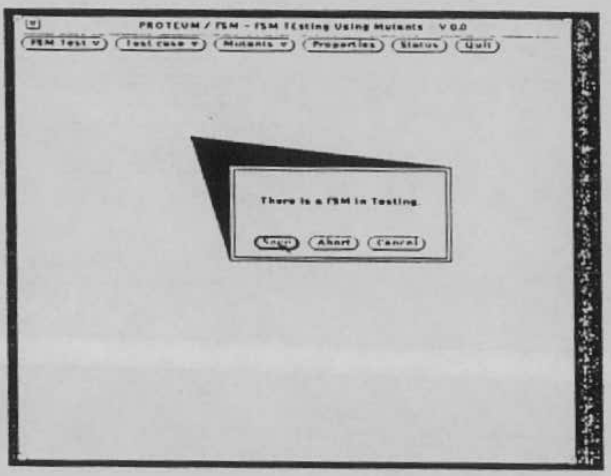


e - Menu de Casos de Teste



f - Status do Teste

Figura 4.2.1 - Telas da Interface da Proteum/FSM



g - Confirmação para sair da ferramenta Proteum/FSM

Figura 4.2.1 - Telas da Interface da Proteum/FSM (continuação)

Quanto à ferramenta Proteum, de uma maneira bem simplificada, para que ela seja usada no contexto de MEF, torna-se necessário "instanciá-la" para a linguagem LES. Ou seja, é preciso fazer com que rode sob a Proteum não um programa na linguagem C, mas um "programa" (uma especificação de uma MEF) na linguagem LES. Um dos módulos da Proteum que será usado integralmente, com pequenas alterações na ferramenta Proteum/FSM, é o módulo de Interface. Como este módulo fica totalmente isolado, ele já foi instanciado para o contexto de MEF, como mostram os exemplos da Figura 4.2.1.

Atualmente, a ferramenta Proteum/FSM encontra-se num estágio bastante avançado de sua especificação, e sua implementação é, basicamente, uma questão de integração da ferramenta Proteum com módulos já disponíveis no ambiente Statsim; a parte nova a ser desenvolvida corresponde à geração das MEFs mutantes, que seguirá a mesma linha da geração de mutantes para o nível de programas.

5. CONCLUSÕES

Este artigo apresentou a especificação de uma ferramenta de teste, denominada Proteum/FSM, para apoiar o uso do critério Análise de Mutantes, no teste e validação de especificações baseadas em Máquina de Estado Finito. Atualmente, a especificação da ferramenta Proteum/FSM já se encontra em fase final, como foi mostrado neste trabalho e a implementação está em andamento. Esta deverá estar pronta em curto espaço de tempo, dado que estão sendo reutilizados vários módulos já implementados de duas ferramentas já existentes: Proteum e Statsim.

Ferramentas que apóiem o teste de software nas diversas fases de seu desenvolvimento são de extrema importância e necessidade, pois aumentam a produtividade e a qualidade durante esse processo. Nesse sentido, a ferramenta Proteum/FSM possibilitará o aumento da produtividade e qualidade já nas fases iniciais de desenvolvimento do software, permitindo o teste de seu aspecto comportamental. Além disso, sua implementação viabilizará a realização de estudos empíricos visando a constatação (ou não) das hipóteses e evidências que têm dado suporte a esta pesquisa, obtidos em [FAB93]. Salienta-se que a aplicação manual do critério Análise de Mutantes, como foi feito no exemplo citado neste trabalho, é impraticável devido a aspectos de tempo, de falha humana, dentre outros, o que torna inviável conclusões e resultados mais significativos e de maior generalidade.

Uma vez constatado o aspecto complementar do critério Análise de Mutantes no contexto de MEF, ou mesmo sua aplicação, através da ferramenta Proteum/FSM, como forma opcional de teste de especificações baseadas nessa técnica, tem-se como objetivo de desdobramento deste trabalho estender esses resultados para outras técnicas de especificação de sistemas reativos, como Statechart e Redes de Petri (para esta última técnica, tal estudo já está sendo conduzido manualmente).

7. BIBLIOGRAFIA

[BOA92] Boaventura, I.A.G. *Propriedades Dinâmicas de Statecharts*, Dissertação de Mestrado, ICMSC-USP, São Carlos, 1992.

[BUD81] Budd, T.A. *Mutation Analysis: Ideas, Examples, Problems and Prospecties*, Computer Program Testing, North-Holand Publishing Company, 1981.

[CHO78] Chow, T.S. *Testing Software Design Modeled by Finite-State Machines*. IEEE Transactions on Software Engineering, SE(4(3)), pp. 178-187, 1978.

[DAV88] Davis, A.M. *A Comparison of Techniques for the Specification of External System Behavior*, Communication of the ACM, Vol. 31, N. 9, sep. 1988.

[DEL93] Delamaro, M.E. *Proteum - Um Ambiente de Teste Baseado na Análise de Mutantes*, Dissertação de Mestrado, ICMSC-USP, 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), pp.34-41, 1978.

[DEM91] DeMillo, R.A.; Offut, A.J. *Constraint-Based Automatic Test Data Generation*, IEEE, Transactions on Software Engineering SE(17(9)), 1991.

[FAB93] Fabbri, S.C.P.F.; Maldonado, J.C.; Masiero, P.C.; Delamaro, M.E. *Análise de Mutantes Baseada em Máquinas de Estado Finito*, in Anais do 11º Simpósio Brasileiro de Redes de Computadores, Campinas 1993.

[FUJ91] Fujiwara,S.; Bochmann,G.V.; Khendek,F.; Amalou,M.; Ghedamsi,A., *Test Selection Based on Finite State Models*, IEEE Trans. on Software Eng., Vol. 17, N. 6, June 1991.

[GAB90] Gabos, D.; Stiubiener, S. *Aspectos de Metodologia de Geração de Seqüências de Teste para Protocolos de Comunicação de Dados*, in Anais 8º Simpósio de Redes de Computadores, 1990.

[GIL62] Gill, A. *Introduction to the Theory of Finite-State Machines*. New York, McGraw-Hill, 1962.

[GON70] Gonenc, G. *A Method for the Design of Fault-Detection Experiments*, IEEE Trans. Comput., vol C-19, pp 551-558, june 1970.

[HAR87] Harel, D., *Statecharts: A Visual Formalism for Complex Systems*, Science of Computer Programming, 1987.

[HAR88] Harel, D., *On Visual Formalisms*, Communications of the ACM, Vol. 31, N. 5, pp. 514-530, 1988.

[HEL91] Heller, D., XView Programming Manual, O'Reilly & Associates, Canadá, 1991.

[LEV87] Levesson, N.G.; Stolzy, J.L. *Safety Analisys using Petri Nets*. IEEE Transactions on Software Engineering, SE(13(3)), pp. 386-397, 1987.

[MAL91] Maldonado, J.C. *Critérios Potenciais Usos: Uma Contribuição ao Teste Estrutural de Software*, Tese de Doutorado, FEE/Unicamp, Campinas - SP, 1991.

[MAS91] Masiero,P.C.; Fortes,R.P.M.; Batista Neto,J.E.S., *Edição e Simulação do Aspecto Comportamental de Sistemas de Tempo Real*, Anais do XI Congresso Nacional da SBC, XVIII SEMISH, Santos, pp. 45-61,5-9 Agosto 1991.

[MAT93] Mathur, A.P.; Wong, W.E *Evaluation of the Cost of Alternate Mutation Strategies*, in anais do VII Simpósio Brasileiro de Engenharia de Software, Rio de Janeiro, 1993.

[NAG94] Nagazato, K.K. Dissertação de Mestrado, ICMSC-USP, em andamento.

[NAI81] Naito, S.; Tsunoyama, M. *Fault Detection for Sequential Machines by Transition-Tours*, in Proceedings FTCS (Fault Tolerant Comput. Systems), pp 238-243, 1981.

[PET81] Peterson, J.L. *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1981.

[PRE92] Pressman, R.S. *Software Engineering - A Practitioner's Approach*, (3rd edition), McGraw-Hill, 1992.

[SAB88] Sabnani, K.K.; Dahbura, A.T. *A Protocol Testing Procedure*, Comput. Networks and ISDN Syst., Vol. 15, N. 4, pp. 285-297, 1988.