Integrando injeção de falhas e testes formais na validação da tolerância a falhas

Eliane Martins
DCC - UNICAMP
Cid. Univ.Zeferino Vaz CP 6065
13087-970 Campinas - SP
e-mail: eliane@dcc.unicamp.br

ABSTRACT

This paper presents a test approach which integrates fault injection and formal testing. *Fault injection* is an experimental technique in which a system is validated in presence of special inputs: the faults. The term *formal testing* is used here to designate a functional test based on a formal specification of the system. This allow the selection of test inputs, i.e., faults and activations (inputs that activate system funcionalities), to be based on this formal specification. The approach is centered on the realization of statistical testing. One important aspect of this type of test concerns test size assessment: how many inputs are necessary to achieve validation objectives ? This is the main focus of the study presented here.

RESUMO

Este artigo apresenta uma estratégia de teste que integra a injeção de falhas e o teste formal.A *injeção de falhas* é uma técnica experimental na qual um sistema é validado em presença de entradas especiais: as falhas. O termo *teste formal* é usado aqui para designar o teste funcional baseado na especificação formal do sistema. Deste modo as entradas de teste, que são as falhas e as ativações (entradas que ativam as funcionalidades do sistema), são selecionadas a partir desta especificação. A estratégia de testes é baseada na realização de testes estatísticos. Um aspecto importante nos testes estatísticos diz respeito ao tamanho dos testes: quantas entradas são necessárias para atingir os objetivos da validação? Este é o enfoque principal do estudo aqui apresentado.

PALAVRAS-CHAVE

testes funcionais - injeção de falhas - testes estatísticos - estimação

I. INTRODUÇÃO

Sistemas computacionais são constituídos de uma série de componentes de hardware, de software e outros, que podem falhar eventualmente. A presença de falhas em um ou mais componentes podem causar a ocorrência de erros, os quais podem levar o sistema a apresentar um defeito[1], se o serviço fornecido não está de acordo com o que foi especificado.

Um sistema tolerante a falhas tem a capacidade de fornecer um serviço conforme à especificação, mesmo em presença de falhas em alguns de seus componentes.

Uma das dificuldades no desenvolvimento de sistemas tolerantes a falhas diz respeito à validação dos mesmos, pois além das funcionalidades normais, deve-se levar em conta também os mecanismos de tolerância a falhas (MTF), acionados quando ocorrem falhas/erros no sistema.

A **validação** é uma atividade que visa garantir a confiança na capacidade do sistema em fornecer o serviço espedicificado, e engloba dois aspectos [Laprie 92]:

- **eliminação de falhas**, que envolve a *verificação*, o diagnóstico e a correção;
- **previsão de falhas**, que visa obter, por *avaliação*, medidas que permitam caracterizar o comportamento do sistema em presença de falhas.

O uso de técnicas formais de verificação (como prova de programas), é altamente desejável para garantir a correção do sistema, em particular, de seus MTF. Assim como o uso de métodos analíticos (modelos de Markov, ...) para a avaliação oferecem medidas mais exatas da eficiência desses mecanismos. No entanto, devido à complexidade de tais sistemas, elas são limitadas a partes do mesmo ou ainda a um número reduzido de falhas.

---

[1] Uma falha ("fault") é a causa direta de um erro. Um erro é a parte do estado do sistema que pode levar a um defeito ("failure"). Não existe ainda um consenso quanto à terminologia a ser usada em português; os termos usados aqui estão baseados em [Leite 87].

Estas técnicas devem então ser complementadas pela validação experimental. A **injeção de falhas** é uma técnica que vem sendo muito utilizada na validação de mecanismos de tolerância a falhas pois permite validá-los em presença de entradas especiais: **as falhas**.

A injeção de falhas pode ser aplicada em diferentes fases do desenvolvimento de um sistema. Neste estudo, enfoca-se o seu uso na fase de testes.

Nos testes, duas atividades apresentam especial dificuldade:

- a geração das entradas de teste: dado um conjunto de entradas geralmente muito grande, (e na injeção de falhas isso se torna então mais crítico, pois, trata-se de um teste a dupla entrada: as falhas e as entradas funcionais), como escolher um subconjunto apropriado aos objetivos da validação?
- a análise dos resultados: como determinar se os resultados dos testes estão corretos (o conhecido problema do oráculo)? como obter estimativas com boa precisão?

Neste estudo é apresentada um estratégia de teste que tem um duplo objetivo:

① tratar de forma integrada a geração das falhas e das entradas funcionais do sistema. O interesse desta integração se deve ao fato de que o comportamento do sistema em presença de falhas varia conforme a atividade que ele estejq desempenhando no momento da ocorrência das falhas;

② englobar os dois objetivos da validação.

Para isso, uma das características da estratégia proposta é a integração de testes formais com a injeção de falhas. O que se entende aqui por testes formais são os testes baseados em uma especificação formal do sistema, ou de seus mecanismos de tolerância a falhas. Com isso, o aspecto ① é satisfeito.

Uma outra característica da estratégia proposta é o uso de testes estatísticos, nos quais as entradas de teste são geradas aleatoriamente de acordo com uma distribuição de probabilidades associada ao domínio de entrada [Laprie 92]. Os testes estatísticos foram escolhidos pois permitem que os dois objetivos da validação sejam tratados (aspecto ②).

Uma das dificuldades na realização de testes estatísticos diz respeito à determinação do número de testes a serem realizados. Será mostrado neste estudo como determinar o tamanho dos testes de acordo com os objetivos da validação.

O artigo está organizado da seguinte forma: na Seção II é feita uma breve descrição do método de injeção falhas, com suas diferentes formas de aplicação e seus atributos; a Seção III apresenta a forma de injeção enfocada neste estudo, que é a injeção de falhas implementada por software. A Seção IV apresenta a estratégia proposta, enfocando a determinação do tamanho dos testes. A Seção V conclui o texto e apresenta também as direções para trabalhos futuros.

II. O MÉTODO DE INJEÇÃO DE FALHAS

Os itens a seguir apresentam uma visão geral do método de injeção de falhas, pois o objetivo é simplesmente fornecer uma base para o que vai ser discutido a seguir. Para aqueles que desejem uma apresentação mais detalhada dos aspectos básicos, pode ser consultada a referência [Martins 93b]; para uma visão mais aprofundada, [Arlat 92] é a referência mais adequada.

II.1. Motivação

A importância da validação dos mecanismos de tolerância a falhas (MTF) de um sistema se deve a duas razões principais:

- a presença de falhas de projeto/implementação nesses mecanismos pode levar à deficiências no comportamento dos mesmos quando em presença de falhas para as quais eles foram projetados para tratar; essas deficiências podendo levar o sistema a não mais fornecer o serviço correto (i.e, conforme à especificação);
- a eficiência dos mecanismos de tolerância a falhas influi fortemente em medidas como a confiabilidade do sistema, por exemplo.

Um parâmetro muito usado para quantificar a eficiência dos MTFs é o **fator de cobertura** (ou cobertura da tolerância a falhas) [Bouricius 69], o qual geralmente é definido em termos probabilísticos como:

$$c = \Pr\{\text{serviço correto é fornecido} \mid \text{uma falha é ativada}\}$$

Diversos estudos mostram a influência da cobertura da tolerância a falhas sobre medidas da segurança no funcionamento [Bouricius 69, Arnold 73, ...]. É sabido que variações de milésimos no fator de cobertura podem causar uma variação de um fator de 10 no MTTF ("Mean Time To Failure"), uma das medidas da confiabilidade/disponibilidade de sistemas. Sua avaliação é, em consequência, um aspecto de suma importância na validação da tolerância a falhas.

Dois parâmetros que têm influência sobre a cobertura são a **dormência** de falhas e a **latência** de erros. A **dormência** de falhas é definida como o intervalo de tempo entre a ocorrência de uma falha e sua ativação na forma de erro. A **latência** de um erro é o intervalo de tempo entre a ativação da falha e sua detecção pelos mecanismos de tolerância a falhas.

Os MTFs são construídos de acordo com as classes de falhas que o sistema deve tolerar (**hipóteses de falhas**), estabelecidas nas fases iniciais do desenvolvimento do sistema. Portanto, a cobertura que eles fornecem é fortemente ligada aos tipos de erros aos quais eles são confrontados. Um erro não pode ser detectado enquanto uma falha não for ativada. A presença de falhas dormentes no sistema pode violar as hipóteses de falhas, levando à ocorrência de um defeito. Erros não detectados podem se propagar através do sistema, gerando outros erros, ou seja, uma latência muito elevada pode levar ao defeito dos MTFs, dado que eles podem se confrontar com situações de erro que não foram levados em conta na sua concepção.

II.2. Formas de aplicação

As falhas que afetam a confiabilidade de um sistema podem ser divididas em [Arlat 92]: **falhas de hardware**, que "imitam" a consequência de falhas de hardware sobre o sistema, e **falhas de software**,que representam as falhas de projeto/implementação do software, designadas também como falhas de concepção.

Sendo as falhas de hardware o interesse deste estudo, este texto se limita a apresentar as formas de injeção deste tipo de falhas.

As falhas de hardware podem ser aplicadas de diferentes formas, dependendo do nível de abstração utilizado para representar o sistema (modelo ou protótipo) e do nível de aplicação das

falhas (físico ou lógico) [Arlat 92]. As formas de injeção mais empregadas são a simulação de falhas, a injeção física de falhas e a injeção de falhas por software, apresentadas a seguir.

Na **simulação de falhas**, falhas lógicas são introduzidas em um modelo do sistema. Este modelo pode representar o comportamento ou a estrutura (em termos de portas e/ou transistores) do sistema. As falhas são injetadas no modelo com o intuito de analisar o processo de ativação de falhas e de propagação de erros, a fim de se avaliar o comportamento dos MTFs. As falhas podem ser aplicadas por módulos especialmente construídos para este fim [Czeck 90, Goswami 91, ...], ou através de mutações do modelo original [Jenn 92].

A **injeção física de falhas** consiste em aplicar as falhas físicas a um protótipo do hardware (e/ou do software) do sistema. As falhas são injetadas geralmente sobre os pinos de circuitos integrados [Crouzet 82, Damm 88, Arlat 90, Madeira 93, ...], mas pode-se também submeter os componentes ao bombardeamento por íons pesados [Gunneflo 89] ou ainda à aplicação radiações eletromagnéticas [Kopka 88]. Neste caso os testes visam principalmente estudar o comportamento de MTFs implementados por hardware, mas pode-se também testar software tolerante a falhas [Martins 93a]. As falhas são aplicadas por um dispositivo construído especialmente para este fim, e que interage com o sistema em teste.

Na **injeção de falhas implementada por software**, falhas lógicas são introduzidas em um protótipo do software (e/ou do hardware) do sistema, com o objetivo de emular a consequência de falhas físicas. Esta técnica apresenta as seguintes vantagens, em relação às outras duas técnicas apresentadas:

- a facilidade de realização, pois não necessita de equipamento especial de hardware, como é o caso na injeção física de falhas,
- a controlabilidade e observabilidade do sistema durante os testes é mais fácil, como no caso da simulação, sem apresentar o tempo elevado de processamento desta última.

Esta técnica é adequada para a validação de mecanismos de tolerância a falhas implementados por software, por ter a capacidade de injetar condições de erro específicas que permitam ativar esses mecanismos, o que não pode ser garantido na injeção física de falhas.

A injeção de falhas implementada por software, a qual passaremos a nos referir simplesmente como *injeção por software*, é a técnica abordada neste estudo, e voltaremos a ela com mais detalhes na seção III.

II.3. Caracterização dos testes

A injeção de falhas é caracterizada por um domínio de entrada e um domínio de saída. O *domínio de entrada* corresponde a um conjunto de falhas a injetar **F** e um conjunto de ativações **A** que especifica as entradas destinadas a exercitar o sistema e, em consequência, ativar as falhas injetadas. O *domínio de saída* é caracterizado por um conjunto **R** de dados coletados e por um conjunto de medidas **M** derivadas da análise e tratamento dos conjuntos **F**, **A** e **R**. Os conjuntos **FARM** constituem então os principais atributos da injeção de falhas.

III. INJEÇÃO DE FALHAS POR SOFTWARE NA VALIDAÇÃO DE SISTEMAS

São apresentados nesta seção alguns trabalhos significativos sobre injeção por software. Para sintetizar a apresentação, os trabalhos considerados foram analisados segundo os seguintes aspectos: o sistema alvo, os objetivos da validação, o tipo de falhas injetadas, a geração das falhas e o nº de experiências realizadas.

A Tabela III.1 resume as informações mencionadas para cada um dos trabalhos estudados. As duas primeiras colunas da tabela apresentam a referência principal, o nome do organismo, seguido do nome da ferramenta (em itálico), se for o caso. As outras colunas são relativas aos aspectos mencionados acima, os quais são apresentados resumidamente nos itens que se seguem.

Tabela III.1. Características de algumas ferramentas de injeção de falhas por software.

Referência	Organismo (*ferramenta*)	Sist. Alvo	Objetivos	Tipo de Falhas	Geração das falhas	Número de exper.
[Segall 88a,b]	Un. Carn. Mell., IBM, System/Tech. Dev. Corp. (*FIAT*)	aplic. distrib.	aval.	falhas mem.	— estat.	≈ 130.000
[Winfrey 89]	Un. de Columbia	mecan. de det. e recuper. de erros	verif.	falhas comunic.	funcional determ.	—
[Avresky 91]	Lab. d'Autom. et d'An. Syst.	protoc. toler. falhas	verif.	falhas comunic.	funcional determ.	—
[Dilenno 91]	IBM, System/Tech. Dev. Corp.	arquit. distrib.	aval. verif.	falhas mem., process., comunic.	modelo de falhas determ. estat.	300 (testes determinísticos)
[Echtle 92]	Un. de Dortmund (*EFA*)	protoc. toler. falhas	verif.	falhas comunic.	estrut. determ.	—
[Kanawati 92]	Un. do Texas at Austin (*FERRARI*)	aplic. toler. falhas	aval. verif.	falhas mem., process., outras	— estat.	≈ 600.000
[Rosenberg 93]	Un. de Michigan (*SFI*)	arquit. distrib.	aval. verif.	falhas mem., process., comunic.	— estat.	≈ 10.000
[Lovric 93]	Un. de Dortmund (*ProFI*)	arquit. distrib.	aval. verif.	falhas process.	—	≈ 90.000
[Kao 93]	Un. de Illinois at Urb.-Champ. (*FINE*)	aplic. em ambiente Unix	aval.	falhas mem., process., outras (1)	— estat. determ.	≈ 500

Observações:
- "—": não fornecido
- "(1)": além de falhas de hardware, a ferramenta é capaz também de injetar falhas de software.

III.1. O sistema alvo

Nesta coluna pode-se ter uma idéia da aplicabilidade da injeção por software, indo desde a validação de mecanismos específicos até à validação de arquiteturas distribuídas. A maioria das ferramentas, mesmo as que foram construídas para testes de sistemas específicos, são aplicáveis a uma vasta gama de sistemas.

III.2. Os objetivos da validação

Neste caso é indicado se a validação teve por objetivos a verificação, a avaliação ou ambos.

Na maioria dos estudos, o objetivo visado é a avaliação de parâmetros tais como o fator de cobertura e a latência. Esses parâmetros podem ser usados tanto na construção de modelos analíticos para o cálculo da confiabilidade ou do desempenho, como em [Kao 93], quanto na validação de modelos pré-existentes, como em [Rosenberg 93]. Nestes estudos a verificação também pode ser considerada, mas como aspecto secundário.

Os estudos que abordam unicamente a verificação, por outro lado, visam principalmente exercitar o modelo do sistema, e não consideram o aspecto avaliação.

III.3. Tipos de falhas consideradas

Todas as ferramentas consideradas neste estudo são voltadas para a injeção de falhas de hardware[2], com exceção de FINE [Kao 93], que também leva em conta as falhas de software.

Os tipos de falhas considerados mais comumente para injeção seriam: *falhas de memória* (alterações no conteúdo da memória, podendo ocorrer tanto no segmento de programa quanto no de dados); *falhas de processador* (alterações no conteúdo de registradores, ou em bits de instruções, ou no endereço de destino de uma instrução de desvio, ou ainda em temporizadores); *falhas de barramento* (alterações tanto nas linhas de endereçamento quanto nas linhas de dados, podendo afetar bits em instruções ou nos dados transmitidos através do barramento); *falhas de comunicação* (alteração, perda, retardo ou duplicação de mensagens que circulam nos meios de comunicação).

III.4. Geração das falhas

Como em outras técnicas de teste, a determinação das falhas a serem injetadas envolve dois aspectos [Laprie 92]: estabelecimento de um critério de seleção e geração das falhas segundo o critério adotado. O critério de seleção pode ser estabelecido com base:

- nos modos de falha do sistema: [Dilenno 91], apresenta uma análise dos modos de falha do sistema, dos seus efeitos e da sua criticalidade (FMECA, de "Failure Modes, Effects and Criticality Analysis"), realizada nas várias fases do ciclo de

[2] Em vez de falhas de hardware deveria ser empregado erros de software, de acordo com as definições de falhas e erros [Laprie 92]. No entanto elas são tratadas aqui como falhas de hardware para significar que a ativação das mesmas é a causa de erros no software, seja no código, seja nos dados.

desenvolvimento, a fim de identificar os modos de falha que tenham maior probabilidade de causar defeitos graves no sistema. Esses modos de falhas são induzidos através de injeção de falhas;

- na estrutura do programa: em [Echtle 92], as falhas são escolhidas de forma a cobrir ao máximo o código do programa em teste;
- na especificação do sistema: em [Winfrey 89] e [Avresky 91] as falhas são escolhidas de forma a cobrir ao máximo o modelo funcional do sistema.

De acordo com o critério adotado, a geração das falhas pode ser **determinística**, na qual as falhas a serem injetadas são determinadas a partir de uma escolha seletiva; ou **estatística**, na qual as falhas são selecionadas de acordo com uma distribuição de probabilidades associada ao domínio de entrada.

III.5. Número de testes realizados

Da tabela III.1 pode-se constatar que o número de testes é geralmente elevado, especialmente quando se realizam testes estatísticos. Essa quantidade é devida em parte às possibilidades de combinação dos atributos que podem ser usados para definir as falhas. Além da variação dos atributos, outro fator que contribui para a grande quantidade de testes é o modo de ativação (c.f. II.3), devido a sua influência sobre parâmetros tais como o fator de cobertura, por exemplo. Uma forma de definição dos testes baseado nos atributos das falhas é apresentado em IV.4.

IV. Estratégia de teste proposta

IV.1. Princípios

A análise apresentada na seção III permite constatar que na maioria dos trabalhos sobre injeção de falhas por software, o objetivo visado é a avaliação: as falhas injetadas são escolhidas da forma a serem representativas das falhas que possam ocorrer em fase operacional. Quanto aos modos de ativação, geralmente são escolhidos de forma a representar o perfil operacional do sistema; nestes casos, não se dispunha de um modelo, estrutural ou funcional, que orientasse a geração dos testes, ou ainda, que servisse de referência para determinar se os resultados estavam corretos ou não. Devido à ausência dessa referência de corretude (ou oráculo), a solução empregada consiste geralmente em comparar os resultados de uma execução sem falhas com os resultados obtidos durante as injeções. Por outro lado, nos poucos trabalhos onde se dispunha de um modelo, os testes foram determinísticos, o que inviabilizou o aspecto avaliação.

A estratégia proposta neste estudo pressupõe a existência de um modelo funcional do sistema. O uso desse modelo na injeção por software apresenta as seguintes vantagens:

- permitir que, a partir desse modelo, sejam geradas entradas de teste que exercitem efetivamente os MTFs, a obtenção dessas entradas podendo inclusive ser feita de forma semi-automática;
- permitir a monitoração da cobertura dos testes obtidos com relação a esse modelo;

- resolver o problema do oráculo, pois o modelo formal serve como referência;
- facilitar o diagnóstico e a correção de erros.

A estratégia proposta tem por objetivo cobrir os dois aspectos da validação, quais sejam a eliminação de falhas e a previsão de falhas. Neste texto os principais pontos considerados na abordagem podem ser resumidos pelas seguintes questões:

i. como gerar os testes, usando o modelo formal, de maneira a cobrir os dois aspectos da validação?
ii. na eliminação de falhas, como determinar o número adequado de testes a realizar?
iii. na previsão de falhas: como determinar o número adequado de testes a realizar, de forma a obter boas estimativas dos parâmetros desejados? como obter estas estimativas?

Estes pontos serão tratados nos itens a seguir.

IV.2. Geração das entradas de teste

Do que foi apresentado em IV.1, tem-se que o critério de seleção é baseado no modelo funcional do sistema ou dos seus mecanismos de tolerância a falhas em teste. Quanto à geração dos testes, pode-se considerar a realização de testes determinísticos ou estatísticos.

Os testes determinísticos são mais adequados para a eliminação de falhas, pois a escolha seletiva das entradas de teste pode permitir cobrir ao máximo o modelo do sistema. Devido à esta escolha seletiva, as estimativas obtidas são tendenciosas, o que dificulta o segundo aspecto da validação.

Os testes estatísticos são muito utilizados no teste do hardware, e ultimamente vêm chamando a atenção da comunidade de software. Estudos mostram que eles têm um grande potencial para a validação [Thévenod 91a, Thévenod 91b, ...]: dependendo da distribuição associada ao domínio de entrada, servem tanto para verificação quanto para a avaliação. Uma dificuldade nos testes estatísticos está em determinar a distribuição adequada aos objetivos da validação, pois a escolha desta distribuição influi no número de testes a serem realizados.

Em suma, ambos os tipos de teste apresentam suas vantagens e suas limitações. A fim de explorar melhor as potencialidades de cada um, sugere-se adotar a estratégia proposta em [Thévenod 91a], que consiste em combiná-los da seguinte forma:

Passo 1. *Realização de testes estatísticos para a verificação*:
As entradas de teste são escolhidas aleatoriamente, com uma probabilidade de entrada que permita reduzir o número de testes necessários para cobrir o modelo do sistema, de acordo com o critério de seleção adotado. Espera-se desse modo revelar um grande número de falhas de concepção.

Passo 2. *Realização de testes determinísticos para a verificação* :
Testes adicionais são realizados para selecionar as entradas que não puderam ser testadas no Passo 1, devido à probabilidade delas ser insuficiente para o número de testes realizados.

Passo 3. *Realização de testes estatísticos para a avaliação* :

As entradas são escolhidas aleatoriamente, com uma probabilidade que reflita a distribuição encontrada durante a vida operacional (*perfil operacional*). Dessa forma, as medidas da eficiência dos MTFs podem ser consideradas como representativas dos parâmetros "reais". Neste passo ainda podem ser encontradas novas falhas de concepção, as quais devem ser diagnosticadas e corrigidas.

A estratégia proposta é baseada nos testes estatísticos. Um aspecto importante na realização deste tipo de teste é relativo à determinação do número de testes a serem realizados segundo os objetivos da validação, o que será tratado nos itens a seguir.

IV.3. Eliminação de falhas

Na eliminação de falhas os testes têm por objetivo exercitar ao máximo os MTFs. Como um teste exaustivo é geralmente impossível, um critério de seleção é estabelecido para determinar um subconjunto de elementos do modelo que vão ser exercitados durante os testes. Esse subconjunto deve ser finito, de forma a que seus elementos tenham a chance de ser executados pelo menos uma vez.

Assim, por exemplo, em um teste funcional a partir de um modelo na forma de Máquina de Estados Finitos (MEF) um critério de seleção pode ser: "cobrir todos os estados", e o subconjunto correspondente é {estados da MEF}.

Seja C um critério de seleção e S_C o subconjunto de elementos que deverão ser exercitados durante os testes de acordo com esse critério. Para a determinação de N, o número de testes a serem realizados, utiliza-se a **qualidade do teste com relação a um critério C**, Q_N, definida como a probabilidade de que cada elemento de S_C seja exercitado pelo menos uma vez durante a realização dos testes [Thévenod 91a]. Desse modo, procura-se um valor de N tal que cada elemento de S_C tenha uma probabilidade mínima igual a Q_N de ser exercitado.

A determinação de N com base na qualidade do teste é válida tanto para os testes estatísticos quanto para os determinísticos [Thévenod 92a]. Nestes, as entradas de teste são escolhidas a priori, de tal modo que cada elemento de S_C seja executado pelo menos 1 vez[3]; tem-se então: $Q_N = 1$.

Nos testes estatísticos as entradas são escolhidas aleatoriamente; por isso não se pode garantir, dado um número finito de testes, que cada elemento de S_C seja executado pelo menos uma vez, ou seja, $Q_N < 1$. Portanto nos testes estatísticos é importante determinar N para que se tenha:

$$\forall k \in S_C \quad p_k \geq Q_N$$

para um dado Q_N, onde $p_k = P\{k \text{ seja exercitada ao menos 1 vez}\}$.

Para determinar N, é preciso primeiramente estabelecer p_k, $\forall k \in S_C$. Esta probabilidade é obtida em função da combinação de entradas que permitam exercitar o elemento k.

[3] Geralmente essa escolha é feita de modo que cada elemento seja executado exatamente 1 vez, para reduzir o número de testes necessários.

Por exemplo, Sidhu e Chang mostram um teste estatístico funcional baseado em MEF para protocolos de comunicação onde S_C = {estados} e a probabilidade associada a cada estado é obtida a partir das probabilidades associadas às transições [Sidhu 89].

Se as probabilidades p_k puderem ser aproximadas por um valor constante, a relação entre N e Q_N é dada pela expressão (1) abaixo [Thévenod 91a]:

$$Q_N = 1 - (1-p)^N \qquad (1)$$

onde $p = \min\{p_k, k \in S_C\}$.

A justificativa deste resultado é simples; dado que p é a probabilidade de exercitar o elemento menos provável, cada elemento tem uma probabilidade maior ou igual a $1 - (1-p)^N$ de ser exercitado ao menos uma vez.

Assim, dado um valor de Q_N , o valor mínimo de N para atingir essa qualidade do teste é dado por:

$$N = \ln(1 - Q_N) / \ln(1 - p) \qquad (2)$$

Se a probabilidade de ocorrência de um elemento k depende não somente da entrada de teste fornecida mas também do estado interno do sistema, p_k não pode ser aproximada por um valor constante. Neste caso, se n_k é o número necessário de testes para que o elemento k ocorra pelo menos uma vez com a probabilidade Q_N , então N é dado por [Thévenod 91a]:

$$N = \max\{n_k\} \qquad (3)$$

Os valores de n_k podem ser obtidos analiticamente usando-se, por exemplo, modelos de Markov. Uma outra alternativa consiste em realizar experiências prévias e analisar os resultados obtidos afim de determinar que parâmetros são mais interessantes de serem considerados para acelerar a ocorrência de erros/defeitos, como é proposto em [Chillarege 93].

Em resumo, a realização de testes estatísticos visando a eliminação de falhas envolve dois aspectos principais:

i. escolha da distribuição de probabilidades associada ao domínio de entrada , que vai servir para determinar as probabilidades p_k de se exercitar cada elemento de S_C ;

ii. determinação do número de testes N que permita atingir uma dada qualidade de teste Q_N dadas as probabilidades p_k inferidas em (i).

Resultados experimentais obtidos em diversos estudos mostraram a adequabilidade dos testes estatísticos na eliminação de falhas, quer o critério de seleção seja estrutural [Thévenod 91b], quer seja funcional [Thévenod 92a e b]. Estes trabalhos mostram que os testes estatísticos podem ser altamente eficazes na revelação de falhas de concepção, desde que se escolha a distribuição de testes apropriada.

IV.4. IV4. Previsão de falhas

Na previsão de falhas, o objetivo dos testes estatísticos é obter medidas da eficiência dos MTFs. Um dos parâmetros que geralmente é obtido em testes experimentais é o fator de cobertura. Na avaliação o que se deseja então é estimar este (ou outro) parâmetro com boa precisão.

A título ilustrativo, será apresentada uma abordagem para a obtenção de estimativas da cobertura a partir dos testes por injeção de falhas. Para simplificar a apresentação, a dedução das expressões utilizadas para a estimativa da cobertura não é apresentada. A referência [Martins 92] apresenta estes detalhes. Os conceitos referentes à teoria da estimação apresentados aqui estão baseados em [Grais 90].

Nos testes por injeção de falhas as entradas são caracterizadas pelos conjuntos **F** e **A** (c.f. II.3). Portanto o domínio de entrada, **G**, pode ser definido como o produto cartesiano desses dois conjuntos:

$$\mathbf{G} = \mathbf{F} \times \mathbf{A}$$

Um caso de teste por injeção de falhas é um elemento $g \in \mathbf{G}$, caracterizado por: $g = \langle f,a \rangle$ onde $f \in \mathbf{F}$ e $a \in \mathbf{A}$.

O conceito de fator de cobertura (c.f. II.1) pode ser formalizado da seguinte maneira: seja Y um predicado que caracterize o comportamento do sistema em presença de falhas, tal que: Y=1 se o sistema se comporta conforme o esperado, e Y=0 em caso contrário. A cobertura do sistema pode ser expressa como:

$$c(\mathbf{G}) = P\{Y=1 \mid \mathbf{G}\} \qquad (4)$$

Y é uma variável aleatória podendo assumir os valores 0 ou 1 para cada elemento de **G**;.o fator de cobertura, c(**G**), pode ser visto como $E\{Y \mid \mathbf{G}\}$, o valor esperado de Y na população **G**. Expressando este valor em termos de cada elemento $g \in \mathbf{G}$, tem-se:

$$c(G) = \sum_{g \in G} y(g) \bullet P(g|G) \qquad (5)$$

onde:

y(g)=1 se Y=1 quando o sistema é submetido a g e 0 em caso contrário;

$P\{g \mid \mathbf{G}\}$ é a probabilidade de ocorrência associada a cada elemento $g \in \mathbf{G}$.

Seja $T = \{g_1, \ldots, g_N\}$ o conjunto de testes aplicado ao sistema, escolhido por amostragem aleatória a partir de **G**. A qualidade das estimativas a serem obtidas para c(**G**) a partir do conjunto de testes T depende, dentre outros fatores, da maneira como é realizada a escolha dos elementos de T. Considera-se neste estudo que cada $g_i \in T$ seja escolhido com base na sua probabilidade de ocorrência, e que a escolha de um novo elemento independe do elemento escolhido anteriormente.

Dado que c(**G**) é o valor esperado da variável aleatória Y na população **G**, estimar c(**G**) é portanto um problema de estimação da média de uma população. Dadas as considerações acima sobre a escolha dos elementos de T, deduz-se que a estimativa apropriada[4] para c(**G**), designada por $\hat{c}(G)$, é dada pela expressão abaixo:

$$\hat{c}(G) = \frac{1}{N} \bullet \sum_{i=1}^{N} y(g_i) \tag{6}$$

Uma vez obtida uma estimativa, é importante determinar a precisão da mesma. Uma forma de fazê-lo consiste em definir um intervalo, $(\hat{c}(G)-\Delta c,\ \hat{c}(G)+\Delta c)$, tal que:

$$P\{(\hat{c}(G)-\Delta c \leq c(\mathbf{G}) \leq \hat{c}(G)+\Delta c)\} = (1-\alpha) \tag{7}$$

A expressão (7) deve ser interpretada como: $(1-\alpha)$ é a probabilidade de que o intervalo aleatório $(\hat{c}(G)-\Delta c,\ \hat{c}(G)+\Delta c)$ contenha c(**G**). Tal intervalo é denominado **intervalo de confiança** para c(**G**), com **coeficiente de confiança** $(1-\alpha)$, ou intervalo de confiança $100(1-\alpha)\%$. Assim, se $\alpha = 5\%$, tem-se 95% de chance de que o valor c(**G**) se encontre no intervalo obtido.

Para encontrar Δc será considerado apenas o caso em que N é suficientemente grande de modo que seja possível utilizar a aproximação pela distribuição normal, ou seja, Y é uma variável aleatória normal cujos parâmetros são: a média, c(**G**), e o desvio padrão:

$$\sigma = \sqrt{c(G) \bullet (1 - c(G))} \tag{8}$$

Para um coeficiente de confiança pré-fixado e igual a $(1-\alpha)$, o intervalo de confiança é dado por:

$$\Delta c = \left| c(G) - \hat{c}(G) \right| = z_{\alpha/2} \bullet \sigma / \sqrt{N} \tag{9}$$

onde $z_{\alpha/2}$ pode ser obtido das tabelas de distribuição normal.

A partir das expressões (8) e (9) pode-se deduzir o valor de N para que se tenha uma precisão que seja no mínimo igual a k% de c(**G**) (a precisão é pré-fixada em termos relativos), para um coeficiente de confiança também pré-fixado, que é dado pela relação:

$$z_{\alpha/2} \bullet \sqrt{\frac{c(G) \bullet (1 - c(G))}{N}} \leq k \bullet c(G) \tag{10}$$

o que leva a:

$$N \geq \frac{z_{\alpha/2}^2}{k^2} \bullet \frac{1 - c(G)}{c(G)} \tag{11}$$

---

[4] Por apropriada entende-se uma estimativa não tendenciosa e de variância mínima, para maiores detalhes sobre o assunto, recomenda-se a consulta à literatura existente sobre o assunto.

De (11) vê-se que para uma precisão e um coeficiente de confiança pré-fixados, o valor de N depende de c(**G**), que é o parâmetro procurado. Na prática, conhecendo-se a ordem de grandeza de c(**G**) pode-se determinar N. Esta ordem de grandeza pode ser obtida de forma analítica ou empírica (usando-se resultados de experiências prévias).

Em resumo, o teste por injeção de falhas visando a previsão de falhas envolve a realização dos seguintes passos:

i. escolha da distribuição de probabilidades associada ao domínio de entrada que reflita o perfil operacional do sistema, para que as estimativas obtidas sejam representativas dos valores reais dos parâmetros procurados;
ii. determinação do número de testes N que permita atingir um dado coeficiente de confiança $(1-\alpha)$ e uma dada precisão nas estimativas obtidas, a partir dos testes realizados segundo a distribuição de probabilidades estabelecida em (i).

IV.5. Caracterização do domínio de entrada

Para a realização de testes nos quais a escolha das entradas é feita segundo a distribuição de probabilidades associada ao domínio de entrada, é de fundamental importância a escolha de uma distribuição apropriada, para que as estimativas obtidas sejam representativas dos valores reais. A escolha desta distribuição é uma das dificuldades dos testes estatísticos, agravada aqui pelo fato de que o domínio de entrada é duplo: como determinar $P\{g \mid \mathbf{G}\}$, para cada elemento $g = \langle f, a \rangle$?

Uma alternativa, inspirada no trabalho desenvolvido em [Martins92] para a injeção física de falhas, consiste em considerar que f e a são selecionados de maneira independente com probabilidades $P\{f \mid \mathbf{F}\}$ e $P\{a \mid \mathbf{A}\}$, suas respectivas probabilidades de ocorrência, de modo que $P\{g \mid \mathbf{G}\} = P\{f \mid \mathbf{F}\} \times P\{a \mid \mathbf{A}\}$. Para isso deve-se proceder da seguinte forma:

i. determinam-se as probabilidades $P\{f \mid \mathbf{F}\}$ para todo $f \in \mathbf{F}$. Seleciona-se $f \in \mathbf{F}$ de acordo com sua probabilidade de ocorrência, $P\{f \mid \mathbf{F}\}$;
ii. determinam-se as probabilidades $P\{a \mid \mathbf{A}\}$ para todo $a \in \mathbf{A}$. Seleciona-se $a \in \mathbf{A}$ de acordo com essa probabilidade e injeta-se uma falha f em algum instante aleatório após essa ativação.

Na prática, caracteriza-se o conjunto **F** de acordo com os atributos das falhas cujas distribuições possam ser definidas de maneira independente. Assim, por exemplo, para um determinado tipo de falhas, por exemplo, falhas de memória, podemos definir os atributos: (a) *localização* da falha no sistema ("heap", área de pilha, ...), (b) a *multiplicidade* da falha (quantos bits ou bytes são afetados simultaneamente), (c) a *posição* afetada dentro da localização escolhida, (d) a *natureza* da falha (zera 1 bit, seta 1 byte, ...), (e) a *característica temporal* (permanente, temporária). Dessa forma, $P\{f \mid \mathbf{F}\}$ pode ser expressa como o produto de várias probabilidades condicionais, as quais definem as distribuições de probabilidades de cada atributo dentro de sua categoria:

$$\begin{aligned} P\{f \mid \mathbf{F}\} &= P\{\text{local} \mid \text{conjunto\_localização}\} \\ &\times P\{\text{mult} \mid [1..M]\} \\ &\times P\{\text{posição} \mid \text{conjunto\_posição}[\text{local, mult}]\} \end{aligned}$$

$\times$ P{valor | conjunto_natureza[mult]}

$\times$ P{temp | conjunto_caract_temporais}

Nesse exemplo pode-se perceber que para cada tipo de falha tem-se a realização de um certo número N de testes. Se for levada em conta a injeção de diversos tipos de falha (falha de processador, de comunicação, ...), esses tipos de falha acarretam o particionamento de **F** e em conseqüência, de **G**, em subconjuntos disjuntos. Neste caso, para se obter as estimativas de c(**G**) deve-se considerar outras técnicas de estimação, que levem em conta este particionamento [Martins 92].

V. Conclusões e perspectivas

Neste estudo foi apresentada uma estratégia de teste integrando o método de injeção de falhas com os testes formais. Por teste formal entenda-se aqui um teste funcional, onde as entradas de teste (englobando as falhas e os casos de teste que vão ativar o sistema) são obtidas a partir de um modelo formal do sistema.

A estratégia proposta é centrada na realização de testes estatísticos, no qual as entradas de teste são obtidas com base em uma distribuição de probabilidades associada ao domínio de entrada, que no caso, é um espaço constituído de falhas-ativações.

As vantagens dessa estratégia com relação a outros estudos utilizando injeção de falhas são: (a) o uso de um modelo do sistema, que serve tanto para orientar a geração dos testes quanto para a verificação dos resultados; e (b) a utilização de testes estatísticos, o que permite cobrir os dois aspectos da validação (verificação e avaliação).

O fato de estar baseada na injeção por software permite que esta estratégia seja usada desde cedo na fase de testes, pois não é preciso dispor do hardware em que o sistema vai residir. Por exemplo, nos testes unitários, cada componente do software pode ser testado separadamente, utilizando-se o modelo definido na fase de Projeto Detalhado. Nos testes de integração, utiliza-se um modelo global, representando a interação entre os componentes, e assim por diante. Desse modo tem-se como vantagem suplementar o fato de que o teste por injeção de falhas pode ser usado desde cedo em complemento às outras técnicas de teste utilizadas.

A estratégia apresentada serve de base para um ambiente de testes por injeção de falhas que está sendo desenvolvido em um projeto conjunto entre o DCC-Unicamp e o INPE. Nesta versão inicial está sendo considerado um modelo do sistema baseado em Máquina de Estados Finitos. Vários pontos restam a ser estudados para a realização deste ambiente:

- como definir um modelo que englobe falhas e ativações, tal que sua complexidade não inviabilize o seu uso para a geração dos testes?
- como obter um oráculo a partir deste modelo, que possa ser usado para determinar se os resultados observados nos testes estão corretos ou não?

Um outro ponto de pesquisa em aberto é referente ao caso em que os testes não revelam falhas, ou seja, o sistema não apresentou um comportamento errôneo durante a realização dos testes. A avaliação, em especial, é afetada neste caso, pois qual a confiança que se pode ter nas estimativas obtidas? Uma alternativa estudada em [Martins 92] propõe uma técnica da teoria de

amostragem, denominada de *estratificação a posteriori*, para reduzir os erros de precisão nas estimativas. Outras alternativas são apresentadas em [Thévenod 91a].

REFERÊNCIAS

[Arlat 90] J.Arlat, M.Aguera, L.Amat, Y.Crouzet, J.-C.Fabre, J.-C.Laprie, E.Martins, D.Powell. Fault injection for dependability validation - a methodology and some applications. *IEEE Transactions on Software Engineering*, vol. 16, fev. 1990.

[Arlat 92] J.Arlat. Fault injection for the experimental validation of fault tolerant computer systems. *Relatório interno do LAAS*, nº 92489, Outubro de 1992.

[Arnold 73] Thomas F.Arnold. The concept of coverage and its effect on the reliability model of a repairable system. *IEEE Transactions on Computers*, C-22(3), 1973, pp. 251-254.

[Avresky 91] D.R.Avresky, J.Arlat, J.-C.Laprie, Y.Crouzet. Guiding the process of fault injection for testing fault tolerance. *Relatório interno LAAS nº 91-351*. Dezembro 1991.

[Bouricius 69] W.G.Bouricius, W.C.Carter, P.R.Schneider. Reliability modeling techniques for self-repairing computer systems. *Proc. 24th. National Conference of ACM*, 1969.

[Chillarege 93] R.Chillarege. The art of failure acceleration to design fault-injection experiments. *IEEE Intl. Workshop on Fault and Error Injection for Dependability Validation of Computer Systems*, Gotemburgo, Suécia, 1993.

[Crouzet 82] Y.Crouzet, B.Decouty. Measurements of fault detection mechanisms efficiency: results. *Proc. FTCS-12*, Santa Monica, CA, USA, jun 1982.

[Czeck 90] E.W.Czeck, D.P.Siewiorek. Effects of transient gate-level faults on program behavior. *Proc. FTCS-20*, Newcastle upon Tyne, Inglaterra, jun 1990.

[Damm 88] A.Damm. Experimental evaluation of error-detection and self-checking coverage of components of a distributed real-time system. Tese de doutorado, Tech. Univ. Viena, 1988.

[Dilenno 91] T.R.Dilenno, D.A.Yaskin, J.H.Barton. Fault tolerance testing in the Advanced Automation System. *Proc. FTCS-21*, Montreal, Canadá, jun 1991.

[Echtle 92] K.Echtle, M.Leu. The EFA fault injector for fault-tolerant distributed system testing. Anais do IEEE Workshop on Fault Tolerant Parallel and Distributed Systems, Amherst, MA, EUA, 1992.

[Goswami 91] K.K.Goswami, R.K.Iyer, "DEPEND: a simulation based environment for system level dependability analysis". Relatório da Univ. of Illinois at Urbana-Champaign nº CRHC-UIUC, 1991.

[Grais 90] B.Grais *Techniques Statistiques-2: Méthodes Statistiques*. Dunod, 1990, cap. 6.

[Gunneflo 89] U.Gunneflo, J.Karlsson, J.Torin. Evaluation of error detection schemes using fault injection by heavy-ion radiation. *Proc. FTCS-19*, Chicago, IL, USA, 1989.

[Jenn 92] E.Jenn, J.Arlat, "Implementation of fault injection in VHDL". Relatório Interno nº LAAS92266, julho de 1992. (em francês)

[Kanawati 92] G.A.Kanawati, N.A.Kanawati, J.A.Abraham. FERRARI: A Tool for the Validation of System Dependability Properties. *Proc. FTCS-22*, Boston, MA, USA, 1992.

[Kao 93] W.Kao, R.K.Iyer, D.Tang. FINE: A fault injection and monitoring environment for tracing the Unix system behavior under faults. *IEEE Transactions on Software Engineering*, 19(11), 1993.

[Kopka 88] B.Kopka. Etude et validation d'une redondance homogène d'ordre deux à décalage temporel pour des applications à haut niveau de sécurité. Tese de doutorado, Nancy, França, 1988.

[Laprie 92] J.-C.Laprie. Sûreté de fonctionnement: concepts de base et terminologie. *Dependable Computing and Fault Tolerance*. Springer Verlag, 1992.

[Leite 87] Julius C.B.Leite, Orlando G.Loques F°. Software. *II SCTF*, cap. 4 do mini-curso intitulado: Introdução à Tolerância a Falhas, Campinas, SP, 1987.

[Lovric 93] T.Lovric, K.Echtle. ProFI: Processor Fault Injection for dependability validation. *IEEE Intl. Workshop on Fault and Error Injection for Dependability Validation of Computer Systems*, Gotemburgo, Suécia, 1993.

[Madeira 93] H.Madeira, F.Moreira, M.Rela, P.Furtado, J.G.Silva. Pin-level fault injection for dependability validation: some research results at the University of Coimbra. *IEEE Intntl. Workshop on Fault and Error Injection for Dependability Validation of Computer Systems*, Gotemburgo, Suécia, 1993.

[Martins 92] E.Martins. Validation de systèmes répartis par injection de fautes. Tese de doutorado, ENSAE, 1992.

[Martins 93a] E.Martins. Teste de protocolos tolerantes a falhas por injeção de falhas. *11° SBRC*, Campinas, Maio de 1993.

[Martins 93b] E.Martins. Injeção de falhas na validação experimental da tolerância a falhas. *V SCTF*, mini-curso, S.José dos Campos, Outubro de 1993.

[Rosenberg 93] H.A.Rosenberg, K.G.Shin. Software Fault Injection and its Application in Distributed Systems. *Proc. FTCS-23*, Toulouse, França, 1993.

[Segall 88a] Z.Segall, D.Vrsalovic, D.P.Siewiorek, D.Yaskin, J.Kownacki, J.Barton, R.Dancey, A.Robinson, T.Lin. FIAT-Fault Injection based Automated Testing environment. *Proc. FTCS-18*, Tokyo, Japão, jun 1988.

[Segall 88b] Z.Segall, J.Barton, D.Vrsalovic, D.P.Siewiorek, R.Dancey, A.Robinson. Fault injection based automated Testing: practice and examples. *Proc.8° Digital Avionics System Conference*, San Jose, EUA, 1988.

[Sidhu 89] D.P.Sidhu, C.S.Chang. Probabilistic testing of protocols. *ACM SIGCOMM'89*, Austin, Texas, USA, set. 1989.

[Thévenod 91a] P.Thévenod-Fosse. From random testing of hardware to statistical testing of software. *IEEE Comp'Euro91*, Bolonha, Itália, 1991.

[Thévenod 91b] P.Thévenod-Fosse, H.Waeselynck. An investigation of software statistical testing. Relatório LAAS n° 91.003, 1991.

[Thévenod 92a] P.Thévenod-Fosse, H.Waeselynck. On functional statistical testing designed from software behavior models. *3° IFIP Intl. Working Conference on Dependable Computing for Critical Applications (DCCA-3)*, Palermo, Itália, 1992.

[Thévenod 92b] P.Thévenod-Fosse, H.Waeselynck. STATEMATE applied to statistical software testing., Relatório LAAS n° 92.483, dezembro de 1992.

[Winfrey 89] T.L.Winfrey, G.E.Kaiser. Testing reliable distributed applications through simulated events. *Proc. FTCS-19*,1989.