Um Sistema de Padrões para Injeção de Falhas por Software

Nelson G. M. Leme
nelsongm@dcc.unicamp.br

Eliane Martins
eliane@dcc.unicamp.br

Cecília M. F. Rubira
cmrubira@dcc.unicamp.br

Universidade Estadual de Campinas (UNICAMP)
Instituto de Computação (IC)

Resumo:
Sistemas computacionais tolerantes a falha tem se tornado cada vez mais numerosos e importantes. Consequentemente, testá-los e verificar se eles comportam-se adequadamente na presença de falhas também ficou mais importante. Para isso, uma técnica que tem se mostrado muito útil é a de Injeção de Falhas. Esta técnica gera falhas dentro do sistema sob teste e observa seu comportamento. A maioria das atuais ferramentas que usam essa técnica operam sob condições muito restritas, havendo portanto a necessidade de se desenvolver ferramentas que sejam mais genéricas e customizáveis. Este trabalho propõe criar um Sistema de Padrões para Injeção de Falhas por Software, de tal maneira que seja mais fácil para os desenvolvedores criarem novos programas e ferramentas que façam injeção de falhas usando os padrões descritos nesse sistema. Esses padrões irão descrever, independentemente de linguagem e ambiente, a arquitetura e estruturas necessárias para realizar injeção de falhas.

Abstract:
Fault-tolerant systems has become more numerous and important. As a consequence, testing and verifying them in the presence of faults has also become more important. A technique that has proved to be very useful for testing software systems is Fault Injection. This technique produces faults inside the system under test and observes its behaviour. Most of the existent tools for fault injection are designed to work under very restrictive conditions, and so there is a need for fault injection tools which can be more generic and customized. This project intends to create a System of Patterns for Software Fault Injection, aiming at to easy the task of developers to create new fault injection programs and tools by using the patterns described in the system. These patterns will describe, independently from language and environment, the architecture and structures needed to perform fault injection.

I. Introdução

Atualmente sistemas computacionais tolerantes a falhas tem se tornado cada vez mais importantes e numerosos. Esse tipo de sistema deve, mesmo na presença de falhas, dar uma resposta no mínimo esperada [Lap95]. Para tanto já se desenvolveram diversas técnicas, para se construir Mecanismos de Tolerância a Falhas (MTFs) dentro de tais sistemas, de tal forma que estes últimos possam apresentar um comportamento adequado. Deve-se assegurar que esses MTFs possam tratar as falhas dentro de um sistema, porque, caso contrário, isso poderá causar danos materiais, humanos e/ou financeiros.

Dessa maneira, também tornou-se importante testar sistemas tolerantes a falhas, para se verificar se seus MTFs reagem adequadamente às falhas. Assim, no desenvolvimento de um sistema tolerante a falha, tem-se dedicado tempo e recursos ao teste de seu MTF. Uma das técnicas que tem se mostrado mais interessantes para fazer isso é a de *Injeção de Falhas*, a qual será melhor explicada nas seções seguintes. Entretanto, pode-se dizer rapidamente que essa abordagem procura produzir ou simular algumas das falhas possíveis de ocorrer no sistema, durante uma execução do mesmo, e observá-lo para verificar se este reage como esperado. Como pode se perceber, essa técnica permite testar nas condições mais próximas das reais o funcionamento de um MTF.

Outro conceito que tem se revelado interessante, para o desenvolvimento de sistemas em geral, é o de *padrões*, que são formas de documentar soluções recorrentes para problemas que ocorrem com freqüência no desenvolvimento de sistemas dentro de um determinado domínio de aplicação. Esse é um conceito muito prático, porque evita que os desenvolvedores tenham que ficar formulando soluções que já foram encontradas anteriormente: eles podem se basear em trabalho já feito para sistemas semelhantes. Por causa dessa praticidade, padrões vem se tornando cada vez mais populares.

Tendo esses dois conceitos em mente, quando do início deste trabalho, examinou-se alguns dos problemas que ocorriam nessas áreas. Especialmente, observou-se a área de Injeção de Falhas, onde foram

percebidas certas necessidades, descritas na seção a seguir. E procurou-se uma solução para suprir essas necessidades, o que se tornou o objetivo do presente trabalho e é descrito na seção III. Em seguida, na seção IV e V são expostas as duas bases teóricas deste trabalho: Injeção de Falhas e Padrões. Na seção VI mostra-se uma arquitetura genérica proposta para injeção de falhas, e por fim na seção VII são indicadas algumas conclusões e mostra-se o trabalho presentemente em andamento.

II. Motivação

Como foi dito na seção anterior, a Injeção de Falhas tem se mostrado um recurso útil para testar sistemas tolerantes a falhas. Para realizar esses testes, foram desenvolvidas ferramentas de injeção de falhas: essas ferramentas tomam o sistema a ser testado, produzem ou simulam as falhas possíveis, e observam o sistema, verificando seu comportamento. Tais ferramentas podem ser um auxílio valioso no teste de sistemas. Mesmo sistemas que não precisam de tolerância a falhas podem se beneficiar do uso de tais ferramentas: elas podem dar uma estimativa de que danos esses sistemas poderiam causar se ocorrer uma falha. Portanto, há uma clara necessidade de ferramentas de injeção de falhas. Dessa forma, vários centros de pesquisa e universidades começaram a desenvolvê-las. Pode-se citar algumas mais conhecidas, tais como FERRARI [KKA92], Fiesta [KJA98], FINE [KIT93], Orchestra [DJM97], Xception [CMS95], FIRE [Ros98], etc. Contudo, estas ferramentas tem se revelado de uso muito específico para um dado ambiente. Isto é, são construídas para testarem sistemas dentro de condições restritas, tais como: se eles são sistemas distribuídos; se são sistemas de tempo real; se rodam sobre o sistema operacional Linux; se rodam sobre determinados tipos de processadores; etc. Assim, essas ferramentas não suprem toda a demanda por testes usando injeção de falhas.

Entretanto, para desenvolver-se uma ferramenta para algum tipo de ambiente que ainda não conta com injeção de falhas, o trabalho do desenvolvedor tem que começar do zero. Isto é, a cada vez que é desenvolvida uma ferramenta ou programa para injeção de falhas, deve-se formular qual seria a arquitetura para se fazer tal programa, quais estruturas seriam necessárias para se fazer injeção de falhas, e outros problemas relacionados. Esses problemas já foram resolvidos antes, nas ferramentas existentes, porém não há documentação sobre isso. Há, portanto, a necessidade de se generalizar as arquiteturas e estruturas empregadas nas ferramentas existentes. Consolidando-se esse conhecimento, não haveria a necessidade de se estar sempre resolvendo os mesmos problemas no desenvolvimento. Esse conhecimento teria que estar expresso na forma mais independente de linguagem e ambiente possível, para que pudesse ser útil nos mais diversos ambientes onde fosse necessário. No entanto, tal conhecimento ainda não foi documentado, e assim, o desenvolvimento de programas de injeção de falhas tem que começar sempre resolvendo problemas que já foram solucionados antes.

III. Objetivo

Neste trabalho, procura-se solucionar o problema descrito na seção anterior. A forma mais genérica e independente de linguagem e plataforma para descrever a forma de fazer injeção de falhas é criando um Sistema de Padrões para Injeção de Falhas. Um Sistema de Padrões é um conjunto de padrões relacionados que dão soluções para problemas no desenvolvimento de um sistema dentro de um dado domínio de aplicação. Por exemplo, um sistema de padrões para inteligência artificial irá dar soluções para problemas no desenvolvimento de programas de inteligência artificial.

Quando vai se desenvolver um programa para fazer injeção de falhas, alguns dos problemas que surgem são os seguintes. Qual a arquitetura de um programa para injeção de falhas? Quais estruturas ele precisa para fazer a injeção, a monitorização do sistema, a ativação do sistema sob teste, etc? As soluções para esses problemas estariam descritas dentro de um Sistema de Padrões para Injeção de Falhas. Dessa maneira, o desenvolvedor não teria que novamente formular essas estruturas, bastaria tomar o Sistema de Padrões, no qual elas já estariam expostas.

Além disso, o presente trabalho se propõe também a criar uma nova ferramenta de injeção de falhas. Essa ferramenta apresentará a arquitetura e estruturas descritas no sistema de padrões desenvolvido, e, dessa forma, poderá comprovar a real utilidade deste último. Também, o sistema de padrões poderá ser modificado e melhorado, de acordo com a experiência obtida no seu uso no desenvolvimento de uma ferramenta de injeção de falhas. Igualmente, pretende-se que a ferramenta desenvolvida não seja apenas um exemplo de implementação do sistema de padrões mas também uma ferramenta de injeção de falhas perfeitamente funcional, tendo utilidade por si própria.

IV. Injeção de Falhas

Como já foi falado, Injeção de Falhas tem se mostrado uma abordagem muito útil para se avaliar o comportamento de sistemas na presença de falhas. O conceito em si é o seguinte: procura-se produzir ou simular algumas das falhas que se julgam possíveis de ocorrer durante a vida útil de um sistema sob teste, o passa-se a observar esse sistema para se avaliar seu comportamento na presença dessas falhas. Por exemplo, para se testar o funcionamento de um caixa bancário automático, simula-se uma falha em sua memória, e observa-se se ele irá realizar operações bancárias incorretas, ou se irá ter o comportamento esperado, que é mostrar uma mensagem dizendo que o caixa está indisponível. Considera-se aqui "falha" como sendo o componente ou elemento anômalo dentro do sistema, cuja execução irá acarretar valores errôneos no mesmo, que são os "erros", e estes por sua vez irão determinar um comportamento fora do especificado do sistema, que é o "defeito" [AAA+90, CIP95, HTI97].

Atualmente, existem três modalidades de injeção de falhas:

- *Injeção de falhas por hardware:* nesta modalidade, usa-se um circuito especial para produzir as falhas no sistema sob teste. Pode-se citar como exemplo, um chip de um processador espera um sinal "um" em um de seus pinos. Coloca-se um circuito para interceptar esse sinal e colocar em seu lugar um sinal "zero", e verifica-se como reagirá o sistema.
- *Injeção de falhas por software:* já nesta forma de injeção de falhas, utiliza-se código associado ao sistema sob teste para simular a presença de falhas no mesmo. Por exemplo, um trecho de código poderia ser chamado a cada vez que o sistema sob teste escrevesse em uma certa posição de memória. Esse trecho de código, logo em seguida que o sistema tivesse escrito nessa posição, iria escrever "zero" na posição, dessa forma simulando que a posição de memória teria ficado presa em zero (*stuck-at-zero*). Como na modalidade anterior, observa-se o sistema sob teste para conferir se ele conseguirá reagir adequadamente a essa falha.
- *Injeção de falhas por simulação:* também chamada de injeção de falhas em tempo de projeto. Essa forma de injeção é feita ainda na etapa do projeto do sistema sob teste, em oposição às formas anteriores, que são feitas durante o teste do sistema. Nela, toma-se o projeto do sistema e simula-se o que ocorreria nele caso ocorresse uma falha. Através disso, pode-se determinar se o projeto do sistema é adequado para tratar das possíveis falhas que possam ocorrer.

Das modalidades de injeção de falhas descritas acima, a injeção de falhas por software tem se mostrado mais vantajosa ao invés da injeção por hardware, porque é mais barata, já que não exige circuitos especialmente desenvolvidos, que em geral são caros. Além disso, é mais versátil, já que é mais fácil adaptar código para fazer injeção de falhas em um outro tipo de sistema do que o originalmente previsto, do que tentar fazer o mesmo com circuitos. E também é mais controlável: por exemplo, uma técnica para simular falhas em memória por hardware é bombardear o circuito com radiação; porém, usando essa técnica, é difícil controlar quais posições serão afetadas, enquanto isso é possível através da injeção por software. Comparada com injeção por simulação, a injeção por software é mais próxima das condições de execução do sistema sob teste. A injeção por software irá tomar o produto final do desenvolvimento e trabalhará sobre essa versão final, enquanto a simulação irá lidar apenas com o projeto do sistema. Depois do projeto, uma complexidade extra será adicionada na codificação, e essa complexidade poderá mudar a resposta do sistema às falhas.

Por causa dessas vantagens, a injeção de falhas por software tem se popularizado. Assim, têm sido escritas diversas ferramentas que usam injeção de falhas por software. E também, de acordo com isso, neste trabalho irá se utilizar a injeção de falhas por software.

V. Padrões

Assim como Injeção de Falhas, Padrões constituem outro conceito que surgiu recentemente e tem sido considerado muito útil para o desenvolvimento. Trata-se do seguinte: ao se examinar programas dentro de certo domínio de aplicação, pode-se observar que eles tiveram problemas comuns no seu desenvolvimento, e que para esses problemas eles deram soluções semelhantes, que se mostraram mais eficientes. Essas soluções seriam, então, documentadas de uma forma definida e independente de linguagem de programação. Dessa forma quando um desenvolvedor fosse criar um programa nesse mesmo domínio, em vez de ter que novamente formular a mesma solução, ele apenas teria que consultar um catálogo de padrões para encontrar um padrão que já lhe resolvesse o problema [GHJ+94, BMR+96].

Padrões são documentados de uma forma definida, e podem ser estruturados da seguinte forma. Primeiramente mostra-se o contexto onde o padrão pode ser empregado, e depois fala-se mais especificamente

do problema que o padrão pretende resolver, discutindo-se também as forças, ou elementos que devem ser balanceados numa possível solução. Em seguida mostra-se a solução proposta pelo padrão, e em seções seguintes expõe-se a estrutura dessa solução e sua dinâmica de execução. Após isso fala-se como pode-se implementar o padrão num sistema. Por fim, mostram-se alguns usos conhecidos do padrão, algumas variantes do mesmo e discute-se as vantagens e problemas advindos do uso deste [BMR+96].

Com relação à diferentes tipos de padrões, no presente trabalho consideraremos dois tipos [BMR+96]:

- *Padrões de Arquitetura:* esses padrões são utilizados numa fase inicial do projeto, quando está se discutindo qual será a arquitetura do sistema sendo desenvolvido, e portanto lidam com um aspecto mais global do sistema. Eles pretendem resolver um problema inicial do desenvolvimento, que é como se irá arquitetar o sistema que está se criando.
- *Padrões de Projeto:* já esses padrões são usados numa fase posterior do projeto, para resolver problemas localizados dentro do mesmo. Por exemplo, numa determinada parte do projeto, há um problema de como dividir a realização de uma tarefa. Há um padrão de projeto denominado *master-slave* (mestre-escravo) que propõe uma solução: um objeto mestre divide a tarefa em sub-tarefas a serem realizadas por objetos escravos.

Há um conceito relacionado com o de padrões que é o de *Sistema de Padrões*. Sistemas de padrões consistem em conjuntos de padrões relacionados, que resolvem problemas que ocorrem no desenvolvimento de sistemas dentro de um mesmo domínio de aplicação. Dentro do sistema, os padrões são organizados seguindo um dado esquema de classificação, de forma que um desenvolvedor possa rapidamente localizar o padrão que resolva o problema que ele tem em mãos. Dessa maneira, o desenvolvedor pode economizar muito tempo no desenvolvimento do sistema [BMR+96].

VI. Arquitetura de Software Genérica para Injeção de Falhas

Procurando determinar-se uma arquitetura genérica para programas que façam injeção de falhas, primeiramente efetuou-se um estudo das ferramentas de injeção de falhas já existentes. Foram pesquisadas diversas delas, e examinou-se suas arquiteturas e estruturas para injeção de falhas. Algumas das ferramentas que foram observadas foram: FERRARI [KKA92], FINE e DEFINE [KIT93], SOFIT [AvT95], Orchestra [DJM97], Xception [CMS95], Fiesta [KJA98], IPA [Voa98], FIRE [Ros98] e ComFIRM [BLW99]. A partir desse estudo, procurou-se generalizar a arquitetura e estruturas das ferramentas, de forma a obter-se um padrão comum. Um primeiro padrão que surgiu refere-se à arquitetura dessas ferramentas, que é o padrão de arquitetura "Injetor de Falhas" que propõe uma arquitetura para sistemas que façam injeção de falhas (*Fig. 1*). Desenvolvedores, quando precisarem criar tais sistemas, poderão tomar esse padrão para determinarem como pode ser sua arquitetura.

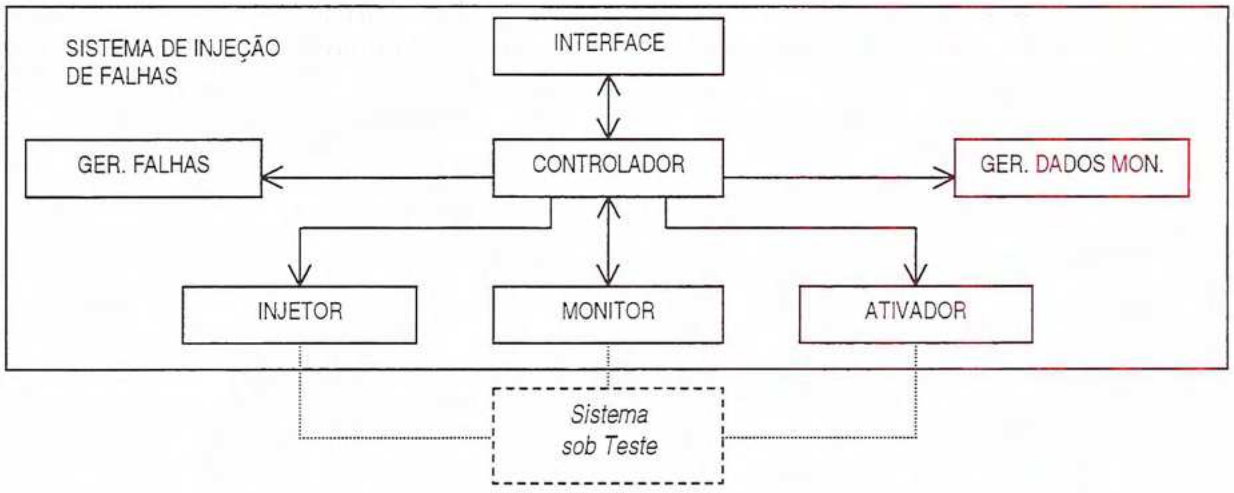

Fig.1: Padrão de Arquitetura "Injetor de Falhas", descrito segundo a notação UML.

Note que, no diagrama, "Sistema sob Teste" indica o sistema onde estão sendo injetadas as falhas, e não faz parte do sistema de injeção de falhas. O padrão "Injetor de Falhas" especifica que um sistema de injeção de falhas se divide em cinco subsistemas, que possuem as seguintes responsabilidades:

- *Injetor:* é o subsistema que realiza a injeção de falhas propriamente dita, ou seja, irá produzir ou simular as falhas dentro do sistema sob teste.
- *Monitor:* este subsistema monitora o sistema sob teste, para verificar seu comportamento quando da ocorrência das falhas. Irá portanto obter dados sobre a execução do sistema alvo.
- *Ativador:* antes de injetar as falhas, é necessário que o sistema sob teste esteja ativo e executando alguma tarefa: este subsistema realiza isso.
- *Controlador:* trata-se do subsistema que coordena e controla a execução dos demais componentes.
- *Interface:* realiza a interface com o usuário, isto é, recolhe deste último dados para fazer a injeção de falhas e depois mostra os resultados.

Além desses subsistemas, o padrão inclui mais dois que servem como repositórios de dados para o programa:

- *Gerenciador de falhas:* armazena os dados sobre as falhas a serem injetadas no sistema alvo. O *controlador* solicita esses dados e os repassa para o *injetor*.
- *Gerenciador de dados monitorados:* os dados obtidos pelo *monitor* e passados para o *controlador*, são armazenados no *gerenciador de dados monitorados*. Este pode guardá-los tais como estão (como um *log file*) ou então consolidar os dados. Pode também elaborar estatísticas com eles.

Nessa arquitetura, esses subsistemas se relacionam da seguinte forma. O *controlador* sabe da existência de todos os demais subsistemas. Estes últimos não sabem da existência de mais nenhum subsistema na arquitetura, exceto *interface* e *monitor*, que conhecem a existência do *controlador*.

VII. Trabalhos em Andamento e Contribuições

Atualmente, está se trabalhando na formulação de padrões de projeto referentes aos subsistemas do padrão de arquitetura "Injetor de Falhas" (seção VI). Posteriormente, irá se iniciar o projeto de uma ferramenta de injeção de falhas baseada na arquitetura e padrões descritos neste trabalho.

Espera-se que o presente trabalho, quando terminado, irá consolidar os conhecimentos sobre a forma de fazer injeção de falhas. A partir dessa base, outros pesquisadores poderão estender a injeção de falhas, tornando-a mais poderosa ou mais ampla. Além disso, este trabalho supre uma necessidade de injeção de falhas, tanto da parte de pesquisa quanto da de mercado. Um pesquisador que esteja desenvolvendo uma arquitetura tolerante a falhas, pode, com este sistema de padrões, rapidamente criar um programa de injeção de falhas que irá validar a arquitetura que criou. Por outro lado, um desenvolvedor comercial pode mais facilmente criar um programa para fazer injeção de falhas talhado especificamente para o ambiente que está trabalhando e testar o sistema que está desenvolvendo, para verificar como ele reage na presença de falhas (se, por exemplo, ele não pode causar danos inesperados).

VIII. Referências Bibliográficas

[AAA+90] Arlat, J.; Aguera, M.; Amat, L.; Crouzet, Y.; Fabre, J. C.; Laprie, J. C.; Martins, E.; Powell, D. "Fault Injection for Dependability Validation – A Methodology and some Applications". *IEEE Transactions on Software Engineering*, 16 (2), Fevereiro/1990, pp. 166-182.

[AvT95] Avresky, D. R.; Tapadiya, P. K. "A Method for Developing a Software Based Fault Injection Tool". *Texas A&M University, Department of Computer Science, Technical Report 95-021*, Texas, EUA, 1995.

[BLW99] Barcelos, Patrícia P. A.; Leite, Fábio O.; Weber, Taisy Silva. "Implementação de um Injetor de Falhas de Comunicação". *Anais do SCTF'99 – VIII Simpósio de Computação Tolerante a Falhas*, Campinas, Brasil, Julho/1999, pp. 225-239.

[BMR+96] Buschmann, Frank; Meunier, Regine; Rohnert, Hans; Sommerlad, Peter; Stal, Michael. *A System of Patterns: Pattern-Oriented Software Architecture*. John Wiley & Sons, Chichester, EUA, 1996.

[ClP95] Clark, Jeffrey; Pradhan, Dhiraj. "Fault Injection: A Method for Validating Computer-System Dependability". *IEEE Computer*, Junho/1995, pp. 47-56.

[CMS95] Carreira, J.; Madeira, H.; Silva, J. G. "Xception: Software Fault Injection and Monitoring in Processor Functional Units". *5th IFIP International Working Conference on Dependable Computing for Critical Applications*. Urbana-Champaign, EUA, 1995, pp. 135-149.

[DJM97] Dawson, S.; Jahanian, F.; Mitton, T. "ORCHESTRA: A Fault Injection Environment for Distributed Systems". Disponível na Wide World Web em: www.eecs.umich.edu.

[GHJ+94] Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, EUA, 1994.

[HTI97] Hsueh, Mei-Chen; Tsai, Timothy; Iyer, Ravishankar. "Fault Injection Techniques and Tools". *IEEE Computer*, Abril/1997, pp. 75-82.

[KIT93] Kao, Wei-lun; Iyer, Ravishankar; Tang, Dong. "FINE: A Fault Injection and Monitoring Environment for Tracing the UNIX System Behavior under Faults". *IEEE Transactions on Software Engineering*. Volume 19, número 11, Novembro 1993, pp. 1105-1118.

[KJA98] Krishnamurthy, N.; Jhaveri, V.; Abraham, J. "A Design Methodology for Software Fault Injection in Embedded Systems". *Proc of the 1998 IFIP International Workshop on Dependable Computing and its Applications*. Johannesburg, South Africa, Jan. 12-14, 1998, pp. 237-248.

[KKA92] Kanawati, N.; Kanawati, G.; Abraham, J. "FERRARI: A Tool for the Validation of System Dependability Properties". *Proc. FTCS-22*. IEEE CS Press, Los Alamitos, EUA, 1992, pp. 336-344.

[Lap95] Laprie, Jean-Claude. "Dependability – Its Attributes, Impairments and Means". *Predictability Dependable Computing Systems (B. Randell, J.-C. Laprie, H. Kopetz, B. Littlewood, eds)*. Springer, Berlin, Alemanha, 1995, pp. 3-18.

[Ros98] Rosa, Amanda. *Uma Arquitetura Reflexiva para Injetar Falhas em Aplicações Orientadas a Objetos*. Dissertação de Mestrado, UNICAMP, Campinas, Brasil, 1998.

[Voa98] Voas, Jeffrey. "Certifying Off-the-Shelf Software Components". *IEEE Computer*. Junho/1998, pp. 53-59.