Uma Arquitetura de Injetor de Falhas Orientada a Aspectos para Validação de Sistemas de Comunicação de Grupo

Karina Kohl Silveira, Taisy Weber

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

{kohl, taisy}@inf.ufrgs.br

***Resumo**: Sistemas de comunicação de grupo são blocos de construção eficientes para o desenvolvimento de sistemas distribuídos com características de dependabilidade. A injeção de falhas é uma abordagem que permite acelerar a ocorrência de erros e defeitos em um sistema para que seja possível a validação das suas propriedades de dependabilidade, assim como a avaliação do impacto dos mecanismos de detecção e remoção de erros no desempenho do sistema. Esse artigo apresenta uma arquitetura de injetor de falhas baseada em orientação a aspectos para a validação de sistemas de comunicação de grupo. É mostrado como se pode evitar a intrusão espacial do injetor de falhas, utilizando-se o paradigma de orientação a aspectos.*

1. Introdução

Um sistema tolerante a falhas caracteriza-se pela capacidade de fornecer o serviço especificado, mesmo na ocorrência de falhas. No desenvolvimento destes sistemas, surgem problemas relacionados à validação dos mesmos e além das suas funcionalidades normais, os mecanismos de tolerância a falhas devem ser testados. A injeção de falhas é uma técnica de validação e pode ser definida como a introdução intencional de falhas em um sistema para observar seu comportamento [ARL90].

Neste trabalho pretende-se apresentar uma alternativa para injeção de falhas utilizando-se recursos do paradigma de orientação a aspectos [KIC97]. Uma tecnologia relativamente nova, baseada em reflexão computacional e que surgiu para aumentar a expressividade do paradigma de orientação a objetos. A opção por aspectos surgiu ao se observar à possibilidade de evitar a intrusão espacial do injetor de falhas. Aspectos possibilitam a interceptação e até mesmo modificação de atributos, construtores e métodos do programa a ser validado sem alteração do código fonte.

O principal objetivo deste artigo é propor a injeção de falhas baseada no paradigma orientado a aspectos para a validação de mecanismos de *membership* de um protocolo de comunicação de grupo. Esse tipo de protocolo trabalha com visões de um grupo e quando é identificado *crash* ou alguma falha de comunicação, o membro é isolado e é criada uma nova visão sem a presença do membro com problemas. O injetor se baseia na possibilidade de interceptação das mensagens enviadas entre os membros do grupo, simulando falhas de comunicação e *crash*, sem a necessidade de alterar o código fonte do protocolo de *membership*. As falhas injetadas permitem verificar a criação de novas visões refletindo a exclusão de membros.

Entre ferramentas de injeção de falhas de comunicação podem ser citadas CSFI, ORCHESTRA, ComFIRM e a extensão da ferramenta Jaca. A ferramenta CSFI

(*Communication Software Fault Injection*) [CAR95] foi umas das primeiras desenvolvidas para injeção de falhas de comunicação e o objetivo principal era o de avaliar o impacto de falhas em sistemas paralelos. A versão existente de CSFI foi implementada para um sistema *transputer* T805. ORCHESTRA [DAW96] foi desenvolvido para teste de dependabilidade de protocolos distribuídos. A ferramenta ComFIRM (*Communication Fault Injection through OS Resources Modification*) [BAR2000] propõe-se a injetar apenas falhas de comunicação e fica situada no núcleo do sistema operacional Linux. A ferramenta Jaca, foi estendida recentemente para permitir falhas de comunicação [JAC2004] e é utilizada para a validação de aplicações orientadas a objetos escritas em Java. O maior objetivo de Jaca é de injetar falhas utilizando características de programação de alto nível durante a execução do programa, corrompendo valores de atributos, parâmetros de métodos ou valores de retorno. Jaca está baseada em reflexão computacional. A proposta deste artigo diferencia-se de Jaca por usar orientação a aspectos.

Este artigo está organizado da seguinte forma: a seção 2 apresenta alguns conceitos relacionados ao artigo. Na seção 3 é apresentada a injeção de falhas baseada em *software*, uma arquitetura genérica para injetores de falhas e a arquitetura baseada em aspectos com seu modelo de falhas, as regras de injeção de falhas e uma discussão sobre intrusividade.

2. Conceitos Relacionados

A injeção de falhas pode ser definida como a introdução intencional de falhas em um sistema para observar seu comportamento [ARL90]. Acelerar a ocorrência de erros e defeitos é uma abordagem eficaz para a validação das propriedades de dependabilidade de um sistema assim como para avaliar o impacto dos mecanismos de detecção e remoção de erros no desempenho do sistema.

O tipo de injeção de falhas apresentada nesse trabalho é conhecido como SWIFI (*Software-implemented fault injection*). Nessa abordagem as falhas são injetadas no *software* através de *software*, corrompendo o código ou os dados das aplicações. Falhas são injetadas durante a execução da aplicação alvo e um monitor associado ao injetor fornece dados sobre o comportamento desse sistema em presença das mesmas. A análise desses dados indica se o sistema atende à especificação, ou seja, se realmente mascara ou recupera-se das falhas a que se propôs na fase de projeto [HSU97].

A interferência da injeção de falhas na aplicação alvo, também chamada de perturbação ou intrusão, pode ser temporal e espacial. Na intrusão temporal o tempo de execução é aumentado devido às atividades do injetor de falhas junto à aplicação alvo. Já a intrusão espacial, refere-se à modificação do código da aplicação alvo [BAR2001].

Sistemas de comunicação de grupo [GAR99] são blocos de construção poderosos para o desenvolvimento de sistemas distribuídos tolerantes a falhas. A comunicação de grupo é um meio de oferecer comunicação multiponto a multiponto, pela organização dos processos em grupos. Um sistema de comunicação de grupo orientado a visões, provê serviços de *membership* e também de *multicast* confiáveis. Um serviço de *membership* mantém uma lista dos processos de um grupo que estão correntemente ativos e conectados. A saída de um serviço de *membership* é chamada de visão. Os serviços de *multicast* confiáveis entregam as mensagens aos membros da visão atual.

A programação orientada a aspectos é um paradigma de programação relativamente novo, introduzido por Kickzales [KIC97], e é baseado em decomposição aspectual. A decomposição aspectual complementa a decomposição funcional e tenta superar a limitação de decomposição funcional de capturar e representar funcionalidades que cruzam o sistema. Após a separação do sistema em construções funcionais, a decomposição aspectual é

aplicada ao projeto para se capturar interesses cruzados [PAP2004]. Um aspecto é uma unidade modular de implementação cruzada. O aspecto encapsula comportamentos que afetam várias classes em módulos reusáveis.

Linguagens orientadas a aspectos utilizam cinco elementos principais para modularizar *crosscutting concerns*: *join points, pointcuts, introductions, advices* e *aspects*. Um *join point* é um ponto bem definido no fluxo do programa (por exemplo, chamadas de métodos e construtores). Um *point cut* seleciona um *join point* em particular filtrando um subconjunto de todos *join points* baseados em critérios definidos. Os critérios podem ser nomes explícitos de funções ou nomes de funções especificados por coringas. As *introductions* permitem que novos métodos ou campos sejam inseridos a uma classe existente. Um *advice* é usado para definir o código adicional que deve ser executado nos *joint points*.

3. Injeção de Falhas baseada em Aspectos

O injetor proposto é implementado como sendo um aspecto do sistema de comunicação de grupo e onde o principal *concern* é efetivamente a comunicação, pois o injetor irá trabalhar com a interceptação das chamadas de métodos de envio e recepção de mensagens.

O objetivo é a validação do comportamento do sistema de comunicação de grupo na presença de falhas de comunicação e *crash*. O resultado esperado é que o serviço de *membership* instale uma nova visão quando perceber que um determinado membro do grupo não está se comportando da forma esperada, isto é, reconheça a falha injetada como sendo uma situação errônea e "cumpra" sua especificação.

Como o paradigma de orientação a aspectos trabalha no meta nível, o injetor proposto permite que seja definido um roteiro de injeção de falhas sem que exista a necessidade de análise ou alteração do código fonte da aplicação.

3.1. Arquitetura

A arquitetura proposta é baseada na arquitetura genérica sugerida por Hsueh [HSU97]. O ambiente é basicamente constituído de um sistema alvo, um injetor de falhas, uma biblioteca de falhas, um gerador de carga, uma "biblioteca de carga", um controlador, um monitor, um coletor de dados e um analisador de dados.

O injetor de falhas injeta falhas no sistema alvo assim como executa comandos enviados pelo gerador de carga. O monitor rastreia a execução dos comandos e inicia a coleta de dados quando necessária. O coletor de dados realiza a coleta *on line* de dados e o analisador de dados, que pode estar *off line*, realiza o processamento de dados e a análise. O controlador controla o experimento. Em função desse modelo, será apresentada a arquitetura para o injetor proposto.

Usando o paradigma de orientação a aspectos e seus elementos básicos, foi possível o agrupamento de alguns elementos da arquitetura proposta por Hsueh. Mais de uma funcionalidade de cada elemento de Hsueh se encontra em um único elemento da orientação a aspectos. A Figura 1 apresenta a arquitetura para o injetor de falhas baseado em programação orientada a aspectos.

Como o objetivo do injetor é a validação de um sistema de comunicação de grupo, então o gerador de carga para o sistema alvo, nada mais é que qualquer aplicação distribuída construída sobre o sistema de comunicação de grupo a ser validado. Por esse motivo, o gerador não está representado na arquitetura, mas sim externamente a ela.

O controlador e o injetor de falhas de Hsueh se tornaram um único elemento. Esse será implementado como um elemento *advice* da orientação a aspectos. Como já foi citado, um *advice* é responsável pelo disparo da lógica adicional que será executada. Além disso, é o *advice* que reconhece o momento que um determinado método deve ser interceptado. Portanto essas duas funcionalidades cumprem os papéis delegados ao controlador e ao injetor, pois é o controlador que é responsável por identificar o momento da injeção e o injetor deve realizar a injeção.

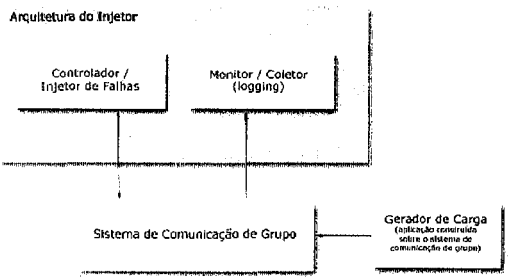

Figura 1 – Arquitetura do injetor baseado em aspectos

O monitor e o coletor também foram agrupados em um único elemento. O monitor é responsável por identificar que uma falha será injetada e dispara o coletor, que irá realizar o *log* das injeções realizadas. O monitor/coletor será implementado como um único *advice*, pelo mesmo motivo que o controlador e o injetor foram. A atividade de *log* é um dos principais exemplos citados na literatura para o uso de orientação a aspectos.

3.2. Modelo de Falhas

Aplicações que utilizam técnicas de tolerância a falhas são construídas para suportar um dado conjunto de falhas, dentro do modelo de falhas especificado. Portanto, considerando a técnica de injeção de falhas, é importante a identificação do modelo de falhas tolerado para que o procedimento de validação tenha sentido. No contexto desse trabalho, o escopo de falhas a ser utilizado compreende as falhas de comunicação, descritas abaixo e definidas por Cristian [CRI91]:

- Falha de *crash* de processo: ocorre uma parada prematura do processo no sistema. A partir da falha, o processo não realiza qualquer função. Antes da ocorrência da falha o processo apresentava comportamento correto.
- Falha por omissão de envio: um processo falha por intermitentemente omitir o envio de mensagens;
- Falha de temporização: um processo falha pela violação do limite de tempo exigido para a execução de uma determinada função (por exemplo, não enviar o *ack* de recebimento de uma mensagem).

4. Intrusividade

A intrusividade é uma característica indesejada, pois pode influenciar nas medições obtidas e gerar resultados que não condizem com a realidade. Um injetor de falhas com uma

intrusão muito grande no sistema pode mascarar erros e defeitos. Quanto menor a intrusão gerada pelo injetor, melhores e mais corretos os resultados obtidos.

O injetor proposto não apresenta intrusão espacial no código do sistema alvo. O paradigma de orientação a aspectos permite que os métodos do sistema sejam interceptados e então seja adicionada a lógica adicional para alteração de métodos e campos. Porém essa lógica não está no código fonte da aplicação, que não precisa sequer ser conhecido, e sim no código do injetor de falhas e será ativado em tempo de execução. Porém a intrusão temporal não é possível de ser evitada, pois código adicional tem que ser executado.

5. Metodologia

O injetor será acionado toda a vez que alguma rotina de comunicação for chamada. Por exemplo, se um membro do grupo enviar uma mensagem, a chamada de envio poderá ser interceptada e atrasada, simulando uma falha de temporização, ou simplesmente não ser enviada, simulando uma falha de omissão ou de *crash*. Nesse caso as chamadas aos métodos de envio e recepção correspondem ao conceito de *joint point* na orientação a aspectos.

A figura do *advice* no paradigma orientado a aspectos é o que irá permitir a execução da lógica de injeção de falhas. É o comportamento que será inserido nos *join points*. Além disso, as *introductions* permitirão que novos métodos ou campos sejam inseridos a uma classe existente, sendo útil no momento de interceptação de mensagens, podendo corrompê-las.

Um dos principais serviços de um sistema de comunicação de grupo é o *membership* que compreende o controle das visões, pois são elas que mantêm a integridade do sistema distribuído, evitando que membros falhos interfiram na computação As visões contêm os identificadores de todos os membros ativos e conectados em um determinado momento e quando algum membro falha o serviço de *membership* identifica e isola o membro falho, instalando uma nova visão. Para validar se o sistema de comunicação de grupo realmente atua da forma especificada, o injetor irá interceptar chamadas de envio ou recepção de mensagens e fazer com que não seja entregues ou recebidas, o que será entendido pelo serviço de *membership* como uma falha no membro que enviou a mensagem ou que não recebeu a mensagem. Porém não é aconselhável que todas as mensagens sejam alvo do injetor, o que impossibilitaria a execução normal do programa afetando até mesmo o experimento de injeção de falhas. O modelo de disparo das falhas será tratado no próximo tópico.

5.1. Disparo das Falhas

O disparo ou a ativação das falhas deve estar relacionado a alguma condição que possa ser definida em termos espaciais ou temporais. Uma falha disparada espacialmente torna-se ativa quando o sistema alcança determinado estado. Por exemplo, uma falha pode ser ativada após a quinta mensagem ser enviada. Uma falha disparada temporalmente deve ser ativada após um determinado período de tempo [BAR2001].

A partir disso, optou-se pelo disparo espacial de falhas, realizado em três momentos:

- Quando um determinado número (gerado aleatoriamente) de membros estiver presente na visão, permitindo a simulação de *crash* ao atingir o número de membros definido;
- Quando um contador de mensagens enviadas ou recebidas por um determinado membro atingir um número gerado aleatoriamente, pode-se omitir a próxima mensagem a ser

enviada simulando uma falha de omissão ou atrasar a confirmação da próxima mensagem recebida, simulando uma falha de temporização;

- Quando um contador de mensagens enviadas por um determinado membro atingir um valor fixo, permitindo a inserção de falhas de forma determinística, de crash, omissão ou temporização, caso nenhum dos casos anteriores seja atingido.

6. Considerações Finais

O artigo apresenta uma arquitetura para um injetor de falhas baseada no paradigma de orientação a aspectos. Esse injetor tem como objetivo a validação de sistemas de comunicação de grupo, que são ferramentas valiosas para a construção de sistemas distribuídos que necessitam de características de dependabilidade. O injetor segue a filosofia da abordagem SWIFI, onde um software injeta falhas no sistema alvo, corrompendo código ou dados. O modelo de falhas que a arquitetura leva em consideração são as falhas de *crash*, omissão e temporização.

A principal vantagem oferecida pela orientação a aspectos na construção de um injetor de falhas é a possibilidade de interceptação e alteração de métodos de um sistema, em tempo de execução, sem a necessidade de alteração do código original da aplicação. Dessa forma, é possível eliminar a intrusão espacial do injetor de falhas no código do sistema alvo. Essa propriedade se torna ainda mais importante quando o código da aplicação não está disponível, impossibilitando alterações e recompilação.

Como última consideração, é proposto como trabalho futuro a extensão da arquitetura baseada em aspectos apresentada nesse artigo, para que seja utilizada na validação de outros sistemas, distribuídos ou não, além dos sistemas de comunicação de grupo.

Referências

[ARL90] ARLAT, J. Et al. *Fault Injection for dependability Validation: a methodology and some applications*. IEEE Transactions on Software Engineering, v. SE-16, n.2, Fevereiro 1990.

[BAR2001] BARCELOS, P.P.; WEBER, T. *INFIMO – Um Toolkit para Experimentos de Intrusão de Injetores de Falhas*. Tese de doutorado. UFRGS. 2001.

[CAR95] CARREIRA, J., SILVA, J.G *Assessing the Effects of Communication Faults on Parallel Application*. Proceedings of IPDS'95. Abril 1995.

[CRI91] CRISTIAN, F. *Understanding Fault-Tolerant Distributes Systems*. Communication of the ACM, v.34, n.2, Fevereiro 1991.

[DAW96] DAWSON, S. JAHANIAN, F., MITTON, T. *ORCHESTRA: A Probing and Fault Injection Environment for Testing Protocol Implementation*. Proceedings of IPDS'96. Setembro 1996.

[HSU97] HSUEH, M., TSAI, T., IYER, R. *Fault Injection Techniques and Tools*. IEEE Computer, v. 30, issue 4, Abril1997.

[JAC2004] JACQUES, G., MORAES, R., WEBER, T., MARTINS, E. *Validando sistemas distribuídos desenvolvidos em Java utilizando injeção de falhas de comunicação por software*. V Workshop de Testes e Tolerância a Falhas. Maio 2004

[KIC97] KICKZALES, G. *Aspect Oriented Programming*. Proceedings of the European Conference on Object-Oriented Programming. Junho 1997

[PAP2004] PAPAPETROU, O., PAPADOPOULOS, G. *Aspect Oriented Programming for a component-based real life application: A case study*. Proceedings of the 2004 ACM Symposium on Applied Computing. Março 2004.