Uma Arquitetura de Testes para Validação de Computadores de Bordo

Cláudia Santos da Silva
Nandamudi L. Vijaykumar
Eliane Martins
claudia@dea.inpe.br, vijay@lac.inpe.br, eliane@ic.unicamp.br

Resumo

Este artigo apresenta um estudo de arquiteturas de testes para auxiliar na validação de Computadores de Bordo visando a aplicação de testes de conformidade. Em sistemas computacionais utilizados em satélites, ou aplicações espaciais, a principal característica é a confiabilidade, portanto, é necessário investir no processo de desenvolvimento do sistema, principalmente nos testes. Este trabalho está baseado no projeto ATIFS (Ambiente de Testes por Injeção de Falhas por Software). O ATIFS é uma plataforma que possui um conjunto de ferramentas que apóiam as atividades de geração, execução e análise de resultados de testes. O estudo que está sendo realizado visa configurar a arquitetura ferry clip para multi-partes de modo a apoiar testes de computadores de bordo, utilizando a ferramenta FSoFIST (Ferry-clip with Software Fault-Injetion Support Tool) do ATIFS. Uma avaliação de arquiteturas multi-partes é discutida.

Abstract

This article presents a study of architectures of tests to assist in the validation of On-Board Computers concentrating on applying to conformance tests. In computational systems used in satellites, or space applications, the main characteristic is the reliability, therefore, it is necessary to invest in the process of development of the system, mainly in the tests. This work is based on ATIFS (Environment of Tests with Software Fault Injection) project. ATIFS is a platform that consists of a set of tools that support the activities of generation, execution and analysis of results of tests. The study that is being carried out is to configure the ferry clip architecture for mutli-parts in order to support tests of on-board computers, using the FSoFIST tool (Ferry-clip with Software Fault-Injetion Support Tool) of the ATIFS. An evaluation of architectures multi-parts is discussed.

1. Introdução

A operação de sistemas de tempo real se diferencia de outras formas de processamento de dados pelo envolvimento explícito do requisito de tempo. Sistemas computacionais de tempo real tem de responder a estímulos externos dentro de um tempo finito especificado, ou seja, a corretude dos sistemas não depende somente do resultado lógico computacional, mas também do tempo que os resultados são produzidos [6].

Neste tipo de aplicação, o computador interfaceia diretamente com outros equipamentos e é utilizado em geral para seu monitoramento e controle. Neste caso, o computador é parte de um sistema de engenharia maior. Sistemas computacionais que apresentam estas características são conhecidos como *Sistemas de Tempo Real Embarcados*. Como exemplo de aplicação para sistemas computacionais de tempo real embarcados podemos mencionar controle de tráfego, controle industrial, telecomunicações, aviônica e satélites.

A principal característica de sistemas computacionais utilizados em satélites, ou aplicações espaciais é a confiabilidade, ou seja, o sistema deve ter uma alta probabilidade de funcionar de acordo com o especificado durante um intervalo de tempo pré-definido.

Como não é possível prevenir completamente a ocorrência de falhas que possam provocar o mau funcionamento do sistema, torna-se imprescindível também o uso de alguns mecanismos de tolerância a falhas, que façam com que o sistema continue operando (de forma degradada ou não) mesmo após a ocorrência de falhas no sistema. Além disso, é necessário investir no processo de desenvolvimento do sistema (especificação, projeto, implementação e testes), ou seja, no processo de prevenção de falhas.

Com a constante evolução tecnológica os sistemas embarcados em satélites têm evoluído e mudado de plataforma com uma freqüência cada vez maior ficando

inviável o desenvolvimento de um equipamento de teste para cada aplicação.

Assim, com o objetivo de investir no ciclo de desenvolvimento do sistema, principalmente nos testes, a proposta do trabalho é viabilizar a adaptação de uma arquitetura de teste de software para computadores de bordo concentrando nos testes de conformidade.

A Seção 2 apresenta o conceito de testes de conformidade e introduz o padrão desenvolvido pela ISO para dar suporte a este tipo de testes. A Seção 3 apresenta a arquitetura *Ferry Clip* desenvolvida para dar suporte às metodologias de testes definidas pela ISO. O projeto ATIFS e a ferramenta FSoFIST, desenvolvida utilizando a arquitetura proposta para teste de protocolo denominada *ferry clip* são brevemente apresentados na Seção 4. A Seção 5 descreve a arquitetura de testes do Microsatélite Franco-Brasileiro (FBM) que será utilizado como estudo de caso. A Seção 6 apresenta alguns estudos realizados visando a realização de testes multi-partes, isto é, para várias conexões em paralelo, o que é um requisito inerente para um sistema de teste de computadores de bordo. Finalmente, a Seção 7 apresenta a conclusão do artigo e direcionamento para atividades futuras.

2. Testes de Conformidade

Testes de conformidade têm por objetivo determinar se uma dada implementação satisfaz os requisitos definidos em sua especificação. No contexto do Modelo de Interconexão para Sistemas Abertos (ou OSI, de *Open Systems Interconection*), foi desenvolvido um padrão para os testes de conformidade do modelo OSI. Trata-se do padrão IS9646: "*OSI Conformance Testing Methodology and Framework*". O padrão define a metodologia, a estruturação e especificação de seqüências de testes, além de procedimentos a serem seguidos durante os testes. O intuito é permitir que os resultados de testes realizados por diferentes grupos sejam comparáveis e reprodutíveis, evitando assim a duplicação de esforços. O padrão não estabelece como os testes devem ser gerados, mas define um arcabouço para a estruturação dos testes, bem como a forma de especificá-los. O padrão define também as arquiteturas de apoio à execução dos testes. Segundo o padrão as arquiteturas (também chamadas de métodos) de testes devem basear-se numa metodologia abstrata de testes, cujo principal objetivo é a especificação dos Pontos de Controle e Observação (PCO) das entradas e saídas da implementação em testes (IUT).

A arquitetura de testes pode ser descrita em termos de [7]: (i) acessibilidade aos PCOs, (ii) contexto de teste, que é o ambiente que a IUT está embutida durante os testes, (iii) testadores, que são associados aos PCOs, e são chamados de testador superior (UT) e testador inferior (LT).

3. Arquitetura Ferry Clip

A arquitetura *Ferry Clip* [2] foi desenvolvida com o objetivo de dar suporte às metodologias de testes definidas pela ISO. A idéia básica é permitir que os dados sejam trocados entre o Sistema em Teste e o Sistema de Teste de maneira transparente de modo a permitir que tanto o Testador Superior quanto o Testador Inferior residam junto ao sistema de teste, simplificando a sincronização entre os testadores e minimizando a quantidade de software residente ao sistema em teste. Esta arquitetura foi escolhida para estudos, pois apresenta a facilidade de portar para diferentes plataformas, facilidade de adaptar a diferentes implementações e facilidade no acréscimo de novas funcionalidades, ou seja, expansão da arquitetura com mínima interferência.

Os principais componentes da arquitetura *ferry* são dois *ferry-clips*, chamados de *Ferry* Ativo (*Active Ferry* – AF) e o *Ferry* Passivo (*Passive Ferry* – PF) que trocam dados entre si através de um protocolo de comunicação denominado Canal do *Ferry*. A Figura 1 apresenta a arquitetura *ferry-clip*.

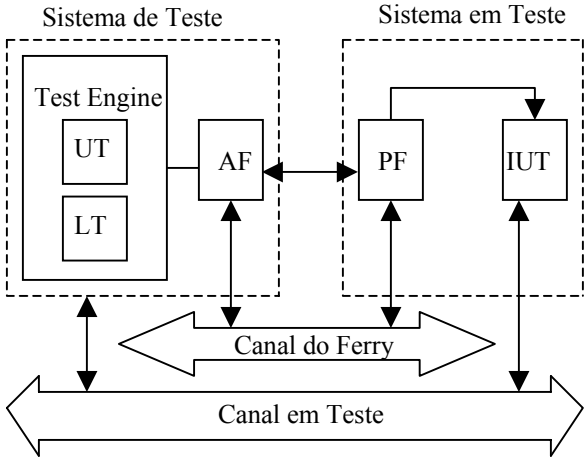

Figura 1 – Arquitetura ferry clip

O Sistema de Teste é formado pelo AF, responsável por iniciar a conexão com o PF e por transferir os dados dos testadores para o sistema em teste e pelo *Test Engine* que agrupa as funções do Testador Superior e do Testador Inferior. O Sistema em Teste é formado pelo PF, quem efetivamente transfere os dados enviados para a IUT e pela Implementação em Teste (IUT).

4. ATIFS

O ATIFS é um projeto conjunto entre Instituto da Computação da Unicamp e o Instituto Nacional de Pesquisas Espaciais (INPE) cujo objetivo é dar suporte à estratégia de teste por injeção de falhas e também suporte a outros tipos de testes, tais como testes de conformidade [5].

As principais características do ambiente ATIFS são:

- Suporte às atividades do processo de testes, tais como: desenvolvimento e geração de casos de teste, execução dos testes, seguido da análise dos resultados, compreendendo tanto a verificação do comportamento do sistema quanto a avaliação da eficiência de seus mecanismos de tolerância a falhas;
- Integração dos diversos subsistemas que realizam estas atividades, integração esta relativa à apresentação (o usuário interage com os diferentes subsistemas de maneira similar), ao controle (os subsistemas utilizam serviços uns dos outros) e os dados (dados definidos por um subsistema podem ser usados por outros sem maiores dificuldades);
- Extensibilidade, isto é, facilidade de se modificar ou acrescentar subsistemas ao ambiente.

A arquitetura do ATIFS está dividida em um conjunto de subsistemas, cada qual responsável por uma atividade de teste. Cada subsistema é constituído por uma ou mais ferramentas.

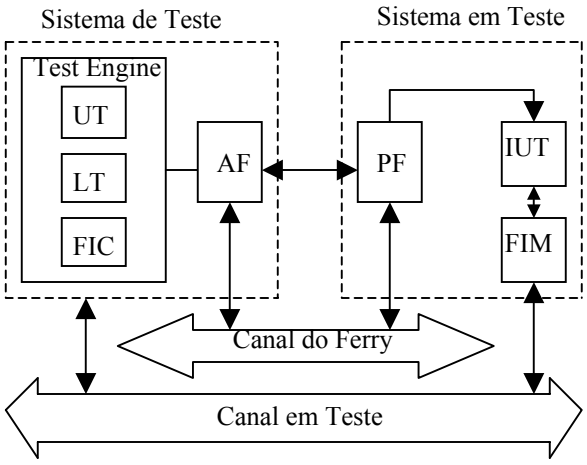

Figura 2 – Arquitetura de FSoFIST

A ferramenta FSoFIST foi projetada, dentro do ambiente ATIFS, para dar suporte à realização de testes em protocolos de comunicação e foi desenvolvida através da extensão da arquitetura *Ferry clip* [1]. A idéia desta extensão é permitir a injeção de falhas em protocolos que possuam mecanismos de recuperação e tolerância a falhas. Esta modificação consistiu em acrescentar um injetor de falhas junto à IUT e um mecanismo simples de controle deste injetor junto ao Sistema de Teste. O mecanismo de troca de mensagens é o mesmo utilizado para enviar dados à IUT. A Figura 2 apresenta a arquitetura da FSoFIST.

5. Estudo de Caso

Esta Seção descreve a arquitetura do Equipamento de Testes (ETBPC) do Computador dos Experimentos Brasileiros (BPC) do Microsatélite Franco-Brasileiro (FBM), que será utilizado como estudo de caso para este trabalho.

O equipamento de testes do BPC é responsável por prover os recursos necessários para desenvolvimento e testes do hardware e software do BPC [3]. A Figura 3 apresenta esta arquitetura. O software do Equipamento de Testes está distribuído no PC de teste 1, que contém a placa de desenvolvimento do *Transputer*, no PC de Teste 2, que contém a placa AD/DA, no PC de Teste 3 e no Simulador do computador principal do satélite FBM, denominado OBC (*On Board Computer*).

O software do PC de teste 1 tem como objetivo simular a operação do OBC, possibilitando a execução dos testes funcionais do BPC, antes da integração com o OBC real. Ele é implementado utilizando um ambiente de desenvolvimento integrado para a linguagem de programação *Occam*, que permite editar, compilar, linkar, configurar e carregar os programas na placa de desenvolvimento e no BPC.

O software do PC de teste 2 está dividido em vários programas com a finalidade de simular as entradas analógicas do BPC, ler os arquivos que contém pacotes de telemetrias recebidos do BPC, verificar a consistência dos dados, extrair os pacotes de telemetria dos experimentos, verificar a consistência dos dados transmitidos pelo BPC, armazenar em um banco de dados e fornecer recursos para sua visualização.

O software do PC de teste 3 tem como objetivo simular o protocolo de comunicação BPC/APEX através da linha de comunicação serial RS422.

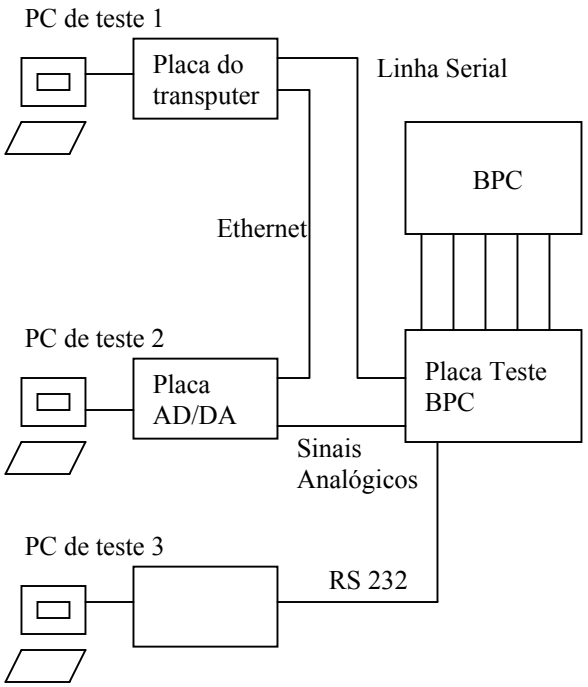

Figura 3 – Arquitetura de teste do FBM

A arquitetura de teste do BPC é específica para a aplicação dos testes deste projeto. Comparando-se esta arquitetura com aquela utilizada na ferramenta FSoFIST verifica-se a possibilidade de utilização da arquitetura da FSoFIST para testar computadores de bordo. Entretanto, atualmente a FSoFIST comporta apenas um canal de comunicação entre os *ferrys* Ativo e Passivo. Para atender as necessidades de teste de computadores de bordo faz-se necessário aumentar o número destes canais. Baseando-se na arquitetura utilizada no FBM, estudos estão sendo realizados para configurar a arquitetura *ferry clip* para testes multi-partes.

6. Configurações de Arquiteturas Ferry Clip Multi-Partes

Esta seção apresenta um estudo das configurações da arquitetura *ferry* para testes multi-partes, conforme descrito em [4]. A Figura 4 apresenta um exemplo de arquitetura distribuída multi-partes. Um dos nós representa a IUT e os outros nós representam implementações denotadas por CI1 a CI5. Nesta configuração o nó CI4 é utilizado como IUT. Em função da própria natureza dos sistemas distribuídos multi-partes a configuração e análise dos resultados dos testes é dificultada pela restrição existente na observação do comportamento da IUT e dos nós vizinhos durante a comunicação. Além disso, esta configuração não é bem aplicada aos testes de conformidade pela falta de funcionalidade em expor a IUT a comportamentos incorretos entre os pares.

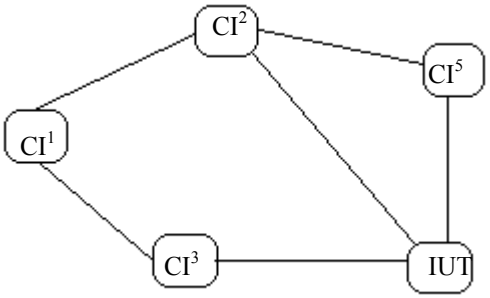

Figura 4 – Teste multi-partes sem equipamento de teste

A Figura 5 apresenta outra configuração desse tipo de sistema, porém introduzindo um sistema de teste, no qual cada nó da rede é conectado a este sistema via canal do *ferry,* possibilitando assim também a realização de testes de interoperabilidade. Cada componente pode ser seletivamente controlado por meio de APSs (Primitivas de Serviço Abstrato) transportadas via o canal do *ferry*. Nesta configuração o sistema de teste tem controle das atividades de comunicação na rede, podendo observar o comportamento da IUT e das implementações dos nós que estão diretamente ligadas a ela. Com isso torna-se mais viável a utilização dessa configuração do que a mostrada na Figura 4 para testes de conformidade, porém há também a dificuldade ou até mesmo, impossibilidade dentro deste ambiente de expor a IUT a um comportamento inválido dos pares.

Se todos os nós da rede são considerados IUTs, então a configuração permite a realização de testes de interoperabilidade.

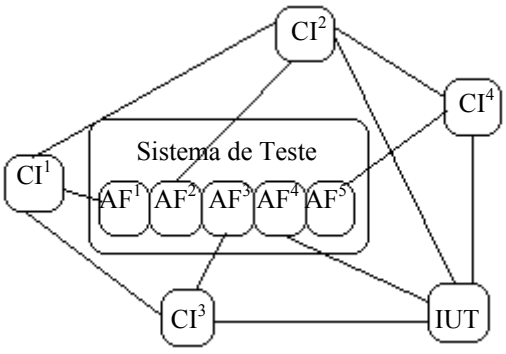

Figura 5 – Teste multi-partes com sistema de teste utilizando ferry

Uma outra configuração deste ambiente de teste, apresentada na Figura 6, mostra a extensão de uma rede virtual modelada dentro do sistema de teste. Isto é realizado simulando a comunicação entre as implementações dos nós no sistema de teste, no qual somente as implementações que são visíveis a IUT são requeridas. No exemplo de rede considerado, as implementações necessárias são a CI2, CI3 e CI5, que mantém três canais de teste com a IUT. A presença do CI1 no exemplo de rede é simulado somente se este influenciar o comportamento das demais implementações.

Nesta configuração o sistema de teste exerce controle total do comportamento da comunicação com a IUT na rede, expondo a IUT ao possível comportamento inválido dos pares.

A vantagem dessa configuração é que pode-se mudar a topologia de testes sem qualquer modificação no ambiente de teste real, uma vez que a diferença entre as topologias visíveis pela IUT é o número de canais de teste.

Mesmo com as vantagens desta configuração, difere muito da arquitetura existente na FSoFST. No projeto da FSoFIST, a IUT comunica-se com o sistema de teste somente via o AF, não existindo outra forma de comunicação das implementações no sistema de teste com a IUT. Como a proposta é adaptar a arquitetura já existente no projeto ATIFS para testes multi-partes, esta arquitetura não será adotada.

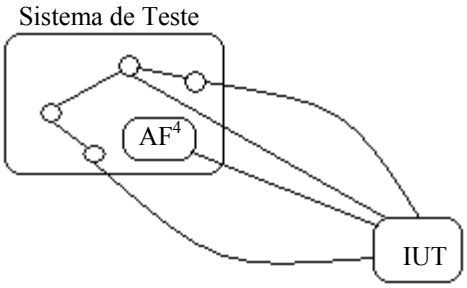

Figura 6 – Configuração multi-partes para testes de conformidade

A Figura 7 apresenta uma configuração da arquitetura multi-partes proposta em [2].

O estudo desta configuração mostra como a arquitetura *ferry clip* pode ser configurada para encontrar os requisitos dentro do escopo da OSI.

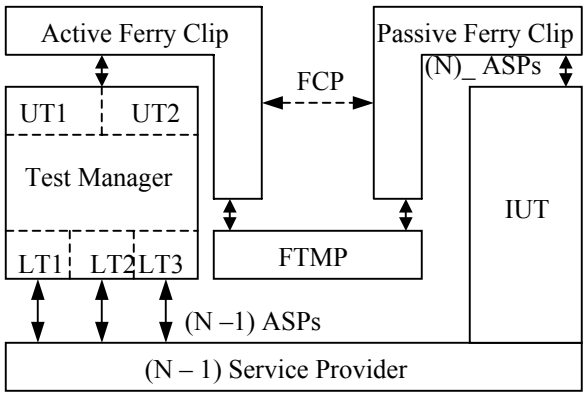

Figura 7– Configuração multi-partes baseado na ferry-clip para camada de transferência de mensagens

Esta configuração também não se aplica ao estudo que está sendo realizado, pois na arquitetura do ATIFS a IUT comunica-se com o sistema de teste apenas via canal do *ferry*, não existindo, portanto, um testador inferior comunicando-se diretamente com a IUT.

7. Conclusão

O artigo descreveu brevemente o conceito de testes de conformidade que têm por objetivo determinar se uma dada implementação satisfaz os requisitos definidos em sua especificação. Apresentou a arquitetura *ferry clip* que além de dar suporte às metodologias de testes definidas pela ISO, simplifica a sincronização entre os testadores.

O ATIFS é um projeto que tem como objetivo dar suporte às estratégias de testes por injeção de falhas e outros tipos de testes como os de conformidade. A arquitetura do ATIFS está dividida em subsistemas no qual são constituídos por ferramentas. Uma dessas ferramentas é a FSoFIST que dá facilidade para realização de teste em protocolo de comunicação e foi desenvolvida através da extensão da arquitetura *ferry clip*.

A arquitetura de testes do FBM é apresentada como estudo de caso para a configuração da arquitetura de testes para validação em computadores de bordo. Estudos foram realizados visando a realização de testes multi-partes, como requerido para este tipo de sistemas, porém até o momento não foi concluído qual seria a melhor configuração da arquitetura proposta no projeto ATIFS para computadores de bordo. Outros estudos ainda serão realizados afim de concluir este trabalho.

Referências

[1] Araújo, M. R. R. "fsofist – Uma ferramenta para teste de protocolos tolerantes a falhas" – Dissertação de Mestrado Instituto da Computação – Unicamp, out 2000.

[2] Chanson, S. T; Voung, S.; Dany, H. "Multi-party and interoperability testing using the Ferry Clip approach", Computer communications vol 15, no 3, April 1992.

[3] FBM-XX-BC-07-5001-INPE, " French-Brasilian Microsatellite FBM", dez 2001.

[4] Fischer, K. P.; GmbH, T. "Position Statement to IWPTS 1989" - Participant's Proceedings 2 nd International Workshop on Protocol Test Systems, Berlim (West), Germany. October 3-6, 1989.

[5] Martins, E. "ATIFS: um Ambiente de Testes baseado em Injeção de Falhas por Software" – Relatório Técnico DCC-95-24 – UNICAMP, dez 1995.

[6] Stankovic, J. A.; Ramamritham K. "Hard Real-Time Systems" – IEEE Computer Society Press.

[7] Tretmans, J.; Belinfante, A.. "Automatic testing with formal methods". Proc. EuroStar'99:7th. European Conference on Software testing, In proceedings of the Conference on Software Testing, Analysis and Review. EuroStar'99, November, 1999.