"Validação de Microssistemas para Aplicações Críticas"

Mário Zenha-Rela e João Carlos Cunha

Laboratório de Informática e Sistemas,
Instituto Pedro Nunes,
Coimbra, Portugal
{mzrela, jcunha}@ipn.pt

Sumário

A sociedade moderna depende de modo vital dos seus sistemas informáticos: desde servidores utilizados em grandes instituições financeiras a microssistemas dedicados para detecção de intrusão, passando por computadores de bordo de veículos automóveis, o correcto funcionamento destes sistemas é essencial para salvaguardar vidas humanas e prevenir perdas económicas ou riscos ambientais.

A garantia do correcto funcionamento destes sistemas informáticos é uma tarefa exigente dada a sua complexidade. Assim, enquanto que em sistemas de grande porte as soluções passam essencialmente por abordagens transaccionais (certificação de que as operações efectuadas o são de forma coerente), tal não é possível em sistemas *embedded*. Efectivamente, nos sistemas que trabalham em tempo-real interagindo com o mundo físico, na maioria dos casos não é possível anular acções de controlo erróneas e repeti--las correctamente. Esta situação é particularmente grave quando do sistema em causa podem depender vidas humanas (p.ex. sistemas médicos de suporte à vida ou controlo de ABS – *sistema anti-bloqueio de travões*).

O Laboratório de Informática e Sistemas do Intituto Pedro Nunes (lis.IPN) trabalha há vários anos no desenvolvimento e validação de sistemas *embedded* para aplicações críticas nos domínios do controlo industrial, sistemas médicos, computadores de bordo para satélites, aeronáutica e sector automóvel. Na presente comunicação serão apresentadas algumas das abordagens utilizadas nestes domínios, com ênfase na validação da robustez de sistemas *embedded* a perturbações de origem física (ruído eléctrico ambiental, vibrações e partículas subatómicas).

1 Introdução

Desde há muito que a injecção de perturbações tem sido vista como um dos principais actores da avaliação da confiabilidade de sistemas informáticos. Esta abordagem é utilizada habitualmente para avaliar e medir a eficácia dos mecanismos de detecção, tolerância e recuperação de erros.

A injecção de perturbações visa um aspecto que não é adequadamente resolvido pelas técnicas mais tradicionais de teste, pois ao invés de se debruçar sobre o funcionamento correcto dos sistemas, visa antecipar o comportamento anómalo de um sistema sob circunstâncias inesperadas e imprevistas. Sendo o número de comportamentos anómalos virtualmente infinito, este é um problema encarado com algum desconforto e geralmente alvo de duas atitudes extremadas. Na maioria das situações são assumidas simplificações irrealistas que não são capazes de suportar as perturbações mais triviais. Por outro lado, quando as avarias podem ter consequências dramáticas são tomadas precauções extremas que aumentam os custos de forma desnecessária.

A injecção de perturbações fornece ao projectista valores quantitativos que lhe permitem calcular o risco de ocorrência de avarias e a sua gravidade, mas também ao engenheiro de desenvolvimento uma ferramenta insubstituível para avaliar a qualidade do produto desenvolvido.

Muitos injectores foram já propostos, com diferentes tecnologias e abordagens sobre a forma mais adequada de emular um ambiente real de perturbações no funcionamento de um computador. Estas abordagens variam entre perturbações físicas, baseadas em *software*, mistas e/ou simulações. Uma descrição detalhada destas diversas abordagens pode ser encontrada em (Hsueh *et al*, 1997). A principal questão é que diferentes abordagens conduzem a diferentes resultados, uma vez que as diversas abstracções do sistema a ser avaliado variam consideravelmente. Conclui-se assim que as várias abordagens são complementares, mais do que alternativas. Por exemplo, embora a injecção baseada em *software* tenha um grau de precisão e eficácia inultrapassáveis, é inadequada para em determinados instantes de funcionamento do sistema (p.ex. imediatamente após o arranque do sistema).

Cientes desta diversidade, as ferramentas desenvolvidas em diversas parcerias envolvendo o lis.IPN abarcam metodologias muito diversas. Na verdade, estas acompanham a própria evolução dos computadores, desde os uniprocessadores de 8 bits até aos mais sofisticados processadores modernos (Pentium e PowerPC), passando pelos Transputers T800 e T9000 utilizados na década de 90.

Na próxima secção apresentamos os conceitos de base da injecção de perturbações, como funcionam as diversas abordagens, quais as respectivas vantagens e inconvenientes, bem como os resultados que é possível obter.

Seguidamente é descrito um ambiente típico de injecção e as etapas por que passa uma campanha de injecção.

A secção 4 é o núcleo deste trabalho: nele são apresentadas as várias ferramentas que ao longo dos últimos anos têm sido desenvolvidas em parceria com o lis.IPN. Concluímos o texto com uma breve reflexão sobre o papel deste laboratório como catalisador dos diversos interessados nos resultados da investigação aplicada ao domínio das tecnologias da informação e comunicação.

2 Avaliação da Confiabilidade através da Injecção de Perturbações

2.1 O que é a confiabilidade?

A **confiabilidade** é a propriedade dos sistemas informáticos que traduz a confiança que se pode justificadamente depositar no seu correcto funcionamento. Este atributo qualitativo do sistema pode ser quantificado através de diferentes medidas, sendo as mais significativas as seguintes:

> **Fiabilidade** – consiste na probabilidade de um sistema cumprir correctamente o seu serviço, sem interrupções, durante o tempo em que está a realizar uma missão. Por exemplo, a fiabilidade do controlador de voo de um avião comercial encontra-se geralmente na ordem dos 99,999% significando que, probabilisticamente, num voo em cada 100.000 ocorre uma avaria do controlador[1]. Numa sonda espacial não tripulada a exigência de fiabilidade já não é tão elevada, podendo situar-se na ordem dos 95% a 99%.
>
> **Disponibilidade** – consiste na probabilidade de um sistema se encontrar operacional em determinado instante, podendo ocorrer interrupções momentâneas no serviço. Esta medida é especialmente importante nos casos em que os sistemas se encontram continuamente em operação e interrupções esporádicas não comprometem irremediavelmente a sua missão. Por exemplo, a Administração Federal de Aviação Americana (FAA) exige que os sistemas de posicionamento global por satélite (GPS) utilizados pelos aviões civis tenham uma disponibilidade mínima de 99,9%, o que significa que podem estar inoperacionais no máximo durante 3.6 segundos por hora.
>
> **Segurança no funcionamento** – consiste na probabilidade de um sistema executar correctamente ou terminar o seu serviço de forma a não comprometer a segurança de pessoas ou provocar consequências catastróficas para o ambiente.

[1] Não significa isto que o avião caia pois, em último caso, o piloto está sempre pronto a assumir o seu controlo.

2.2 A injecção de perturbações

A injecção de perturbações é um método que tem vindo a ser cada vez mais utilizado na avaliação da confiabilidade dos sistemas informáticos. Consiste na introdução deliberada e controlada de perturbações nos sistemas, simulando o tipo de perturbações a que estes estão sujeitos durante a sua fase operacional, tais como radiações electromagnéticas, picos de corrente ou mesmo *bugs* de *software*. Desta forma consegue-se avaliar até que ponto o sistema em teste consegue tolerar estas perturbações (isto é, cumprir o seu serviço correctamente apesar de ter sido sujeito a perturbações), e dar importantes indicações aos projectistas e engenheiros de sistemas quanto à necessidade de melhorar a robustez de determinados módulos de *software* ou *hardware*.

Por exemplo, durante uma experiência de avaliação da confiabilidade de um sistema de controlo de temperaturas de um forno industrial, injectam-se tipicamente alguns milhares de perturbações (3000 a 5000) e obtêm-se resultados probabilísticos sobre o número de perturbações que são toleradas pelo sistema, que activam um mecanismo de alerta de avaria, ou que induzem o sistema em erro sem activar nenhum alerta (este último caso é o pior possível, pois pode levar o forno a produzir cerâmica demasiado frágil sem que ninguém se aperceba). Os resultados da experiência permitem ainda a obtenção de informação pormenorizada sobre o comportamento do sistema perante as perturbações, podendo, por exemplo, identificar-se qual o tipo de perturbações a que ele é mais sensível, ou quais os módulos que necessitam de ser reforçados.

As informações obtidas através das experiências de injecção de perturbações dão indicações extremamente valiosas sobre a forma mais eficaz de aumentar a confiabilidade do sistema, com o menor custo, identificando os locais onde se podem adicionar módulos redundantes de *software* ou *hardware* que garantem a continuidade do serviço correcto quando outros módulos manifestam comportamentos errados. Para além disso, a injecção de perturbações tem a capacidade adicional de desmascarar falhas do sistema, alertando para erros no seu projecto ou desenvolvimento.

A secção seguinte descreve um ambiente genérico utilizado na injecção de perturbações.

3 Arquitectura de um ambiente de injecção de perturbações

3.1 O ambiente de injecção

Um ambiente de testes de injecção de perturbações é constituído tipicamente pelos seguintes actores (ver Figura 1):

1. o **sistema-alvo**, isto é, o sistema cuja confiabilidade se pretende avaliar. Este consiste num sistema computacional, que pode ir desde um multi-processador a um sistema *embedded*, passando por um computador pessoal. Em alguns casos, como os sistemas de controlo, os testes são efectuados enquanto o controlador executa a sua missão de controlo de um processo físico ou simulado (neste caso pode utilizar-se um outro computador que recebe as acções de controlo e produz resultados, simulando as reacções do objecto controlado).
2. o **injector**, que consiste num módulo de *hardware* ou *software* embutido no sistema-alvo, e que é responsável por introduzir as perturbações e monitorizar o seu comportamento. Em alguns sistemas mais recentes é possível injectar as perturbações utilizando recursos existentes no próprio sistema-alvo sem necessidade de nenhum módulo embutido (ver na secção seguinte as ferramentas BSCAN4FI, GooFI ou *mCrash*). Nestes casos, o módulo de injecção encontra-se no próprio sistema de gestão das experiências.

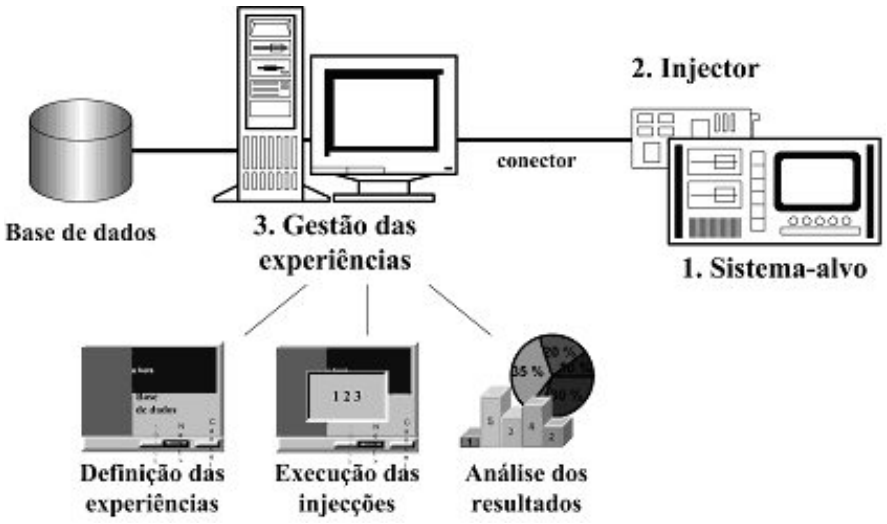

Figura 1 *– Arquitectura de um ambiente de injecção de perturbações*

3. o sistema de **gestão das experiências** é um computador que se encontra ligado ao módulo injector através de uma ligação TCP-IP, série ou outra, e executa uma aplicação com 3 funções essenciais: 1) definir as perturbações a injectar em cada experiência; 2) controlar a injecção de cada perturbação e

recolher os resultados; 3) analisar os resultados. A sub-secção seguinte irá descrever com mais pormenor estas 3 funções.
É através deste ambiente que o experimentador, através de uma ambiente gráfico e auxiliado por diversas ferramentas de *software*, define, controla e analisa os resultados das experiências de injecção de perturbações. Todos os dados e resultados relacionados com as experiências são armazenados numa base de dados, permitindo uma posterior análise com o nível de detalhe pretendido.

3.2 Passos de uma experiência de injecção

Uma experiência completa de injecção de perturbações é constituída por uma série de passos que se descrevem de seguida:

Definição das experiências – o experimentador, através de um ambiente gráfico adaptado especialmente para a plataforma alvo, configura uma série de parâmetros que orientam a geração automática e aleatória de um determinado número (normalmente alguns milhares) de definições de perturbações. Esta informação é armazenada na base de dados da experiência e define o modelo de perturbações, através de parâmetros como:

- **instante de activação** – condição que provoca a injecção da perturbação, como um prazo ou a execução de uma determinada instrução;

- **localização** – definição de um endereço de memória, registo, ou unidade funcional que será perturbada;

- **tipo** – forma como é perturbado o alvo, como alteração do valor de um ou mais bits (*bit-flip*), colocação dos bits a zero (*stuck-at-0*), ou introdução de um atraso temporal.

Execução das injecções – o controlador das experiências, de forma automática, recolhe da base de dados as definições das perturbações, uma a uma, e envia-as para o injector. Este, baseando-se na informação recebida, espera pela ocorrência da condição de injecção e introduz a perturbação. no sistema-alvo. Seguidamente envia para o controlador das experiências informações sobre o estado do alvo no momento da injecção e o seu comportamento posterior. Todos estes dados são armazenados na base de dados.

Análise dos resultados – todos os resultados das experiências são posteriormente analisados, questionando a base de dados. Podem obter-se informações como a percentagem de perturbações que conduziram o sistema à produção de resultados errados, à produção de resultados correctos, ao accionamento de um alarme de erro ou à

sua paragem descontrolada (*crash*). Pode também analisar-se os resultados a um nível de detalhe maior, tal como analisar passo-a-passo o que aconteceu após a injecção de uma perturbação específica.

4 Técnicas e ferramentas de injecção de perturbações

Nesta secção apresentamos as diversas técnicas de injecção de perturbações que foram desenvolvidas em parcerias envolvendo o Laboratório de Informática e Sistemas do Instituto Pedro Nunes (lis.IPN). Para cada ferramenta será apresentada o respectivo modelo de erros, a(s) tecnologia(s) utilizada(s), o sistema-alvo e a natureza da aplicações a que se destina.

4.1 RIFLE

Este injector efectua perturbações nos pinos do processador do sistema alvo. Ele pode ser adaptado a um grande número de sistemas e mesmo em circuitos integrados que não o processador, tais como a memória e controladores de periféricos. No entanto, dado que o processador é o principal componente dos sistemas computacionais e que por seu intermédio é possível emular a maioria das perturbações noutros componentes, a utilização desta ferramenta limitou-se até este momento ao processador.

Essencialmente, o RIFLE é composto por uma placa de circuito impresso colocada entre os pinos do processador e a placa-mãe em que se encontra colocado. Esta placa, análoga à utilizada por analisadores lógicos e ICE's (*in-circuit emulators*), permite ao injector monitorizar todos os sinais dos pinos do processador. Quando ocorre a condição de injecção, o(s) sinais lógico(s) são alterados de acordo com o modelo de perturbação que se pretende efectuar (usualmente um *bit-flip*), continuando a sua execução normal. A partir desse momento o injector passa a modo *trace*, ou seja, recolhe toda a actividade do sistema-alvo (incluindo as saídas) para posterior análise de comportamento.

A injecção é determinística (é possível reproduzir exactamente a perturbação efectuada em diferentes experiências) e permite determinar se efectivamente a perturbação modificou o estado do sistema alvo ou não (pode perturbar-se um sistema sem que a perturbação tenha um impacto efectivo no alvo).

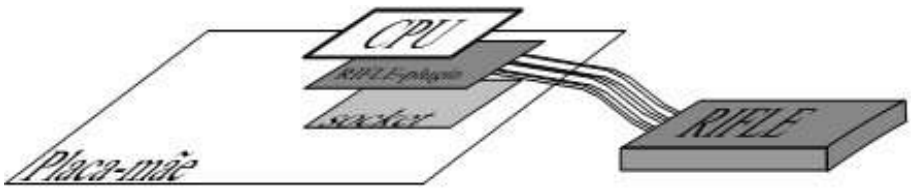

Figura 2 *– Arquitectura física do RIFLE*

A interface com o experimentador permite a definição de conjuntos de perturbações 'a pedido', permitindo definir campanhas de injecção com fins bem dirigidos. Por exemplo, se o experimentador pretender avaliar a capacidade da lógica de protecção de memória, pode gerar exclusivamente erros de endereçamento.

Esta técnica de injecção de perturbações apresenta contudo algumas limitações. Por um lado, é necessário desenvolver um módulo de adaptação aos diferentes alvos utilizados (*plug-in* para Z80, Motorola 68000,...) apesar da maior parte da lógica estar numa placa distinta e reutilizável. Por outro lado, a miniaturização crescente dos processadores e consequente inclusão de inúmeras funcionalidades dentro do chip (*system-on-a-chip*), tem reduzido progressivamente a actividade do sistema que é possível monitorizar ao nível dos pinos. O uso de *caches* internas e tecnologias sofisticadas (como por exemplo a execução especulativa) impossibilitam mesmo a utilização desta técnica. Assim, ela está restrita a sistemas mais baratos e mais simples, ainda que com grande quota de mercado, tais como a maioria dos controladores industriais.

Este injector foi utilizado para a validação do controlador de bordo da actual geração de satélites Brasileiros de comunicações.

4.2 FTMPS

Uma extensão do injector RIFLE foi utilizada para a avaliação de sistemas multi-processador. Embora a filosofia de perturbações por *bit-flip* fosse idêntica, o facto de se utilizar como processador um Transputer INMOS T800 em placas de 8 processadores integrados em computadores que podiam ir até aos 65535 (2^{16}) Transputers, correndo um sistema operativo compatível Unix, conferiu-lhe uma natureza bem distinta do RIFLE original. Estes sistemas eram fabricados pela empresa alemã *Parsytec* e destinavam-se a aplicações para cálculo intensivo tais como análise quantum-cromodinâmica, projecto genoma humano, análise de aerodinâmica de helicópteros pela *British Aerospace* e avaliação de potencial petrolífero de dados geológicos pela empresa *Shell*.

No caso FTMPS (*Fault-Tolerance for Massively Parallel Systems*) o potencial de adaptação do *plug-in* para sistemas-alvos concretos foi levado ao extremo não só porque a arquitectura dos Transputers apresentava características únicas (a comunicação embutida no processador) como a utilização de um sistema operativo sofisticado (*Parix*©) obrigou a uma redefinição do papel do código residente no sistema-alvo e toda a lógica de sincronismo. Concretamente, o código residente foi integrado no núcleo do sistema operativo para poder interagir sem interferências com a lógica de sincronismo de execução do sistema-alvo com o injector, bem como inclusão

de código de recolha de informação do comportamento do alvo após a injecção das perturbações. Note-se que num sistema paralelo com milhares de processadores apenas um pequeno número está directamente ligado ao exterior.

4.3 Xception©

A generalizada miniaturização da electrónica digital, a integração de inúmeras funções dentro dos *chips* e a utilização de técnicas sofisticadas de aumento do desempenho, tornaram praticamente impossível manter a abordagem baseada em perturbações nos pinos.

Porém, os fabricantes de *hardware* debatiam-se com os mesmos problemas e começaram a incluir nos seus processadores lógica de controlo e monitorização, essencialmente vocacionadas para *debug* e avaliação de desempenho.

Esta nova ferramenta de injecção de perturbações consistiu basicamente em colocar toda a lógica de monitorização e injecção dentro do próprio processador utilizando este *hardware* de *debug* embutido.

Efectivamente, ao invés de monitorizar externamente, por exemplo, o acesso a um determinado endereço, configura-se o próprio *hardware* de controlo do processador para provocar um *breakpoint* nesse endereço. A partir daí o processador vai executar código residente com o modelo da perturbação a efectuar (previamente definido) e retorna à execução normal já com a injecção efectuada. O próprio injector passa a utilizar as funcionalidades de *trace* presentes no processador, enviando para o exterior essa informação para análise posterior. Foram desenvolvidos injectores usando esta abordagem para vários processadores Motorola da família PowerPC que dispunham desta infra-estrutura de *debug* e monitorização.

Este injector foi desenvolvida inicialmente por investigadores do lis.IPN que posteriormente fundaram a empresa *Critical Software SA,* tornando-o um bem sucedido produto comercial sob a designação de Xception©. Este injector foi utilizado pelo Jet Propulsium Laboratory (JPL) da Agência Espacial Americana (NASA).

4.4 RT-Xception

O injector Xception© original que entretanto se desenvolveu como um produto comercial de sucesso para aplicações informáticas típicas não era o mais adequado para aplicações de tempo real, tais como controladores industriais, devido ao *overhead* temporal do código residente no núcleo.

Assim, foi desenvolvida de raiz uma nova ferramenta que, baseando-se na mesma filosofia, não tem praticamente qualquer impacto nas temporizações

do sistema alvo, Cunha et al (1999). Os princípios básicos por detrás desta ferramenta consistiram em realizar *on-line* apenas as operações estritamente necessárias e reescrever o núcleo do injector (residente no alvo) em linguagem *assembly*. Adicionalmente, todo o *log* de execução passou a ser armazenado no próprio sistema-alvo e apenas transferido para o exterior após o fim da experiência de injecção. Se porventura os resultados da perturbação corrompem este *log* os resultados dessa experiência são simplesmente ignorados. Uma vez que a injecção visa sobretudo estudos de natureza estatística, descartar algumas experiências não tem consequências significativas.

Esta ferramenta apresenta ainda como característica específica ter sido desenvolvido para um processador Pentium© comum, pois muitos sistemas industriais são computadores pessoais adaptados. Este projecto foi desenvolvido em parceria com a empresa Californiana *MicroDigital* que disponibilizou o acesso ao código fonte do seu núcleo de tempo-real, o SMX©, sob um acordo de não divulgação (NDA). Foi assim possível utilizar um verdadeiro núcleo operativo comercial e não um protótipo académico. Por seu lado, esta empresa teve acesso exclusivo aos dados de robustez do seu núcleo (e identificação de deficiências), o que lhes proporcionou inegáveis vantagens competitivas.

Utilizando esta ferramenta foram efectuados diversos estudos da robustez de aplicações de controlo industrial, daí resultando a definição de um modelo de avaria inovador, o modelo de **erro admissível** (*fail-bounded error model*), Cunha *et al* (2001).

Este modelo foi reconhecido pela comunidade científica no domínio da computação confiável, tendo sido adoptado por diversas equipas de investigação nomeadamente da Suécia (Chalmers) e Estados Unidos (Illinois).

4.5 BScan4FI

Este projecto mantém a orientação de injectar perturbações sem utilizar *hardware* adicional, por recurso a infraestruturas presentes no sistema-alvo. Desta vez o acesso é feito utilizando a infraestrutura de varrimento periférico (*boundary-scan*) que obedece à norma IEEE1149.1 utilizando a interface JTAG. Esta insfraestrutura foi inicialmente concebida para teste de circuitos lógicos em regime estacionário e consiste em interligar os pinos dos circuitos integrados numa cadeia de varrimento (*shift register*), de tal forma que é possível deslocar o estado desses pinos de e para o exterior (sem necessariamente afectar o valor presente nos pinos). Uma das características distintivas desta tecnologia consiste na utilização de apenas quatro pinos (*serial in, serial-out, clock* e *modo*) para aceder ao interior de dispositivos lógicos bastante complexos, desde circuitos de lógica discreta a processadores como o PowerPC.

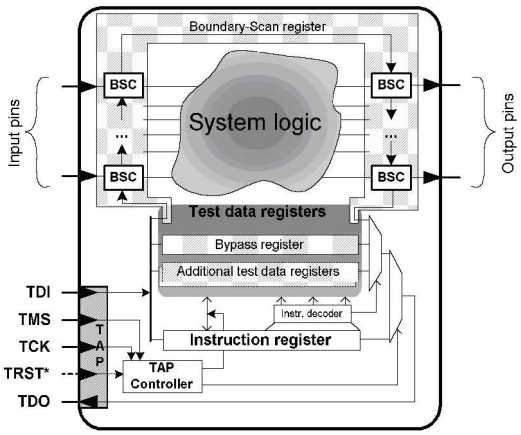

Figura 3 *- Arquitectura de uma infra-estrutura de varrimento periférico*

Limitar a injecção de perturbações aos pinos seria apenas uma forma (na verdade muito eficaz) de ultrapassar as limitações dos injectores baseados em perturbações ao nível dos pinos (tal como no RIFLE). Porém, a maioria dos fabricantes estendeu a norma IEEE 1149.1 para além do estritamente *standard*, criando novas cadeias de varrimento internas, mais ligadas às funcionalidades dos circuitos integrados. Assim, por exemplo num processador, há cadeias internas específicas para os registos de inteiros, registos de vírgula flutuante, para a lógica de controlo, relógios internos e mesmo aspectos não visíveis ao nível externo como as *caches* e os diversos andares da *pipeline*. Compreende-se assim o interesse desta infraestrutura para efeitos de injecção de perturbações: por seu intermédio torna-se possível afectar e monitorizar o funcionamento do sistema-alvo com um nível de controlo e precisão jamais atingidos.

Por outro lado, interligando os portos JTAG de diversos chips é possível aceder aos diversos circuitos integrados de um sistema completo utilizando os mesmos 4 pinos.

A utilização desta técnica em aplicações críticas representa a resposta a um dos grandes problemas do teste de integração: com efeito, o código de instrumentação necessário para efectuar os testes finais têm de permanecer no sistema testado, uma vez que retirá-los depois do teste representa uma modificação do sistema testado e como tal os valores de confiabilidade obtidos podem já não ser válidos. Assim, ao utilizar o próprio *hardware* do sistema alvo, estamos a garantir que todos os testes efectuados são efectivamente representativos do perfil de funcionamento em missão.

É esta a abordagem utilizada no injector BScan4FI (*Boundary-scan for Fault Injection*) desenvolvido em parceria entre o lis.IPN, a empresa *Critical Software SA* e os Institutos Politécnicos do Porto e de Coimbra.

A primeira versão do BScan4FI foi construída para o processador TSC695F da ATMEL, uma implementação da arquitectura Sparc v7. É um processador RISC típico, construído com tecnologia à prova de radiações, uma vez que se destina a ser usado pela Agência Espacial Europeia (ESA) no segmento espacial. O TSC695F também foi adoptado pelas agências espaciais Chinesa e Brasileira.

4.6 GooFI-*slice*

A ferramenta de injecção de perturbações 'GooFI' (Aidemark *et al*, 2001) foi desenvolvida pelo *Departamento de Computer Engineering* da Universidade de Chalmers na Suécia no âmbito de uma parceria com as empresas SAAB e Volvo.

Este injector utiliza uma abordagem semelhante ao BScan4FI, mas utiliza a infra-estrutura *Nexus*, uma evolução da norma IEEE1149.1. Esta infra-estrutura é mais sofisticada que a suportada pela norma original, mas apenas processadores mais modernos e poderosos a suportam, nomeadamente o powerPC MPC565.

Sendo uma ferramenta muito flexível, e na sequência da colaboração de há largos anos entre os laboratórios de investigação de Coimbra e de Chalmers, está a ser desenvolvida conjuntamente uma extensão do GooFI para permitir injecção de perturbações ao nível dos interfaces entre módulos de um programa a correr sobre as plataformas que suportam este processador. Como se depreende, este injector passa assim, não só a suportar modelos de erros que emulam perturbações no *hardware*, mas também erros no *software* (*bugs*).

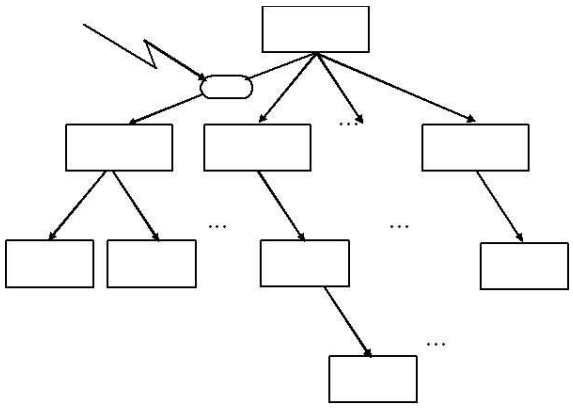

Figura 4 *– Localização das perturbações pelo injector GooFI-slice*

Esta ferramenta destina-se prioritariamente a avaliar a confiabilidade do sistema de *break-by-wire* de automóveis Volvo, podendo ser utilizado em outros sistemas de elevada criticalidade.

4.7 mCrash

As competências adquiridas pelo lis.IPN ao longo dos últimos anos permitem-lhe agora avançar para áreas emergentes. Uma destas áreas é o domínio da computação móvel (computadores portáteis, telemóveis e PDA's).

Um dispositivo móvel é para todos os efeitos um sistema *embedded* centrado na comunicação, o que lhe confere características únicas de ubiquidade de funcionamento e aplicações possíveis.

Está actualmente a ser desenvolvida uma ferramenta (*mCrash*) que representa um novo passo na integração da injecção de perturbações via *hardware* (por varrimento periférico) e via *software* (por injecção de perturbações na API – interface de programação das aplicações – do sistema operativo).

Esta ferramenta destina-se a avaliar a confiabilidade de dispositivos móveis para aplicações de natureza crítica. Não se trata aqui de criticalidade clássica, como tolerância a perturbações ou robustez, mas no domínio da segurança. Note-se que a generalização dos dispositivos móveis irá levar à sua utilização como cartão de débito e crédito, chave de edifícios, automóveis e mesmo como cartão de eleitor. Essencialmente, estes dispositivos irão interagir com os inúmeros sistemas informáticos que nos rodeiam, desde uma simples fotocopiadora a destrancar a porta do automóvel, agindo como identificador e representante o seu proprietário humano. Num tal cenário, a contaminação por vírus e *worms* pode ser dramática para o seu utilizador e para a sociedade em geral. Este estudo terá um interesse particular nos esforços de generalização da adopção de pagamentos electrónicos utilizando dispositivos móveis.

Os sistemas alvos iniciais são o núcleo operativo Symbian (utilizado em telemóveis Nokia, Sony-Ericsson e Panasonic), o Windows CE e o Windows Mobile. A primeira plataforma de *hardware* escolhida é baseada no processador ARM9.

4.8 Resumo

Apresenta-se abaixo um quadro com as principais características das ferramentas de injecção de perturbações desenvolvidas.

Ferramenta	*Tecnologia*	*Aplicação/clientes*	*Plataformas*
RIFLE	*Bit-flip* nos pinos	Controlo industrial, Agência Espacial Brasileira	Zilog Z80, Motorola 68000©
FTMPS	*Bit-flip* nos pinos	Computação maciça paralela, *Parsitec GmBH*	INMOS T800 *(transputer)*
Xception©	*Hardware* de debug do processador	NASA-JPL©	Motorola PowerPC
RT-Xception	*Hardware* de debug + *in-line assembly*	MicroDigital SMX© Real Time kernel	Intel Pentium©
BScan4FI©	IEEE 1149.1 *(JTAG interface)*	Agência Espacial Europeia (ESA)	Atmel TSC695F (sparcV7)
GooFI©-slice	IEEE-ISTO 5001-1999 *(Nexus standard)*	Volvo cars&trucks *(brake by wire)*	Motorola PowerPC MPC565
mCrash	*Software* + JTAG	Dispositivos móveis, pagamento electrónico.	ARM9

***Tabela 1**. Resumo das ferramentas desenvolvidas e tecnologias utilizadas.*

5 Conclusão

Na presente comunicação foram apresentadas algumas das abordagens utilizadas pelos investigadores do Laboratório de Informática e Sistemas do Intituto Pedro Nunes (lis.IPN) ao longo de vários anos na validação da robustez de sistemas críticos a perturbações de origem física (ruído eléctrico ambiental, vibração e partículas subatómicas) e, mais recentemente, de erros de programação.

Estas competências foram adquiridas como fruto da actividade de migração dos resultados da pesquisa efectuada nos laboratórios de investigação da Universidade de Coimbra, para empresas nacionais e estrangeiras. Esta actividade tem originado diversas empresas *spin-off* que prosseguem a exploração comercial dos produtos desenvolvidos nestes projectos. Complementarmente, muitos dos engenheiros participantes nestes projectos são frequentemente integrados nos quadros das empresas envolvidas, que podem assim utilizar todo o potencial da formação avançada dos seus quadros.

Estes projectos são o resultado da cooperação bem sucedida entre os laboratórios de investigação da Faculdade de Ciências e Tecnologia da Universidade de Coimbra, a incubadora de empresas do Instituto Pedro Nunes (IPN) e as empresas nacionais e estrangeiras abertas à inovação. O Laboratório de Informática e Sistemas (lis) actua assim como o catalisador de

convergência de interesses da investigação aplicada no domínio da Informática.

Figura 5 *– o lis.IPN como catalisador de investigação aplicada.*

Referências:

Mei-Chen Hsueh, T. K. Tsai, R. Iyer, (1997), "Fault Injection Techniques and Tools", *IEEE Computer*, 30(4), pp. 75-82.

J.C. Cunha, M.Z. Rela, J.G. Silva, (1999), "Can Software Implemented Fault-Injection be used on Real-Time Systems?", *the Third European Dependable Computing Conference (EDCC-3)*, Prague, Czech Republic, September 1999, Springer-Verlag, Lecture Notes in Computer Science series, Hlavicka, J., Maehle, E., Pataricza, A., (Eds.), Volume 1667, ISBN 3-540-66483-1, pp 209-226

J.C. Cunha, R. Maia, M.Z. Rela, J.G. Silva, (2001), "A Study of Failure Models in Feedback Control Systems", *the International Conference on Dependable Systems and Networks (DSN'2001)*, Göteborg, Sweden, July 2001, IEEE Computer Society Press, ISBN 0-7695-1101-5, pages 314-323.

J. Aidemark, J. Vinter, P. Folkesson, and J. Karlsson, (2001) , "GOOFI: Generic Object-Oriented Fault Injection Tool", *Proc. International Conference on Dependable Systems and Networks (DSN 2001)*, Gothenburg, Sweden, July 2001