

TTCN-3 TOOLS INTEROPERABILITY BETWEEN JAVA AND C/C++ PLATFORMS

Ricardo Rezzano ¹ Ariel Sabiguero ¹ César Viho²

¹Instituto de Computación, Facultad de Ingeniería, Universidad de la República
J. Herrera y Reissig 565, Montevideo, Uruguay
{rrezzano,asabigue}@fing.edu.uy

²IRISA / Dionysos
Campus Universitaire de Beaulieu
35042 Rennes CEDEX, France
viho@irisa.fr

05/06/2008

Work Motivation

Context

Dual APIs - Proposal

Dual APIs - Implementation

Final remarks

TTCN-3 an abstract language

TTCN-3 Abstraction

- ▶ TTCN-3 Abstract Test Suites
- ▶ TTCN-3 Interfaces and Adapters
- ▶ Reuse of ATS and Implementations

Different Platforms

- ▶ Platforms Implementations
- ▶ Implementations nature determine the platform
- ▶ Lack of reuse between different platforms

Project Context Activities and Tools

TTCN-3 Activities and Goals

- ▶ TTCN-3 Experience and Know How
- ▶ Develop picoTTCN-3 Compiler
- ▶ Experiment and Propose enhances to TTCN-3 promoting its use

Go4IT Project Activities

- ▶ Use of different platforms to facilitate IPV6 Testing Activities
 - ▶ T3DevKit and Codec Generator in C/C++
 - ▶ Testing Tools based in Java/Testing Tech technology
- ▶ Development of free TTCN-3 Compiler and Tools

The wrapper based solutions

How to enable reuse between platforms ?

- ▶ Using JNI for interoperability between C/C++ and Java
- ▶ JNI Wrapper based solutions concept

Some Experiences

- ▶ T3DevKit Wrapper for Java (H. Martínez - A. Sabiguero)
 - ▶ Testing Tools Java Wrapper for Go4IT IPV6 Kit
 - ▶ Code Generator Wrapper for Java Platform
- ▶ Testing Tools Java Wrapper for Go4IT C/C++ compiler
- ▶ Modeling external activities in PERL
- ▶ Mapping to new Languages (C#)

Dual APIs solution - Definition and Technology

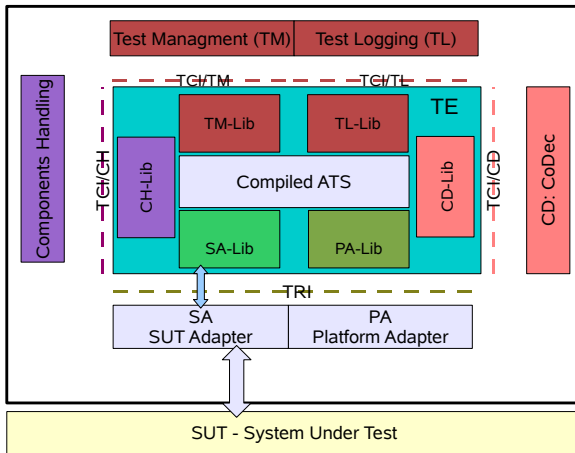
Generalizing the Solution

- ▶ Integrating the solution to the TTCN-3 tools
- ▶ The Dual APIs solution

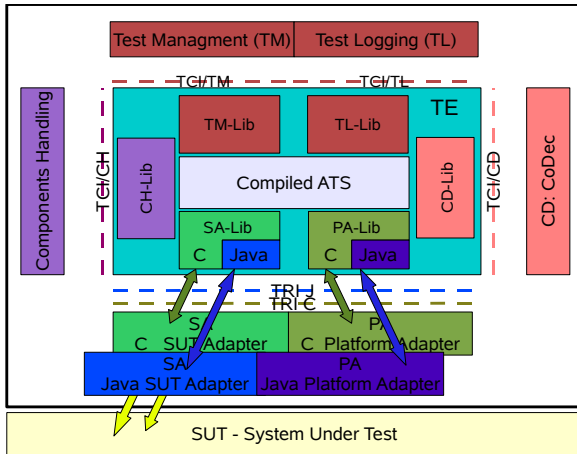
Solution Features

- ▶ Keep the implementations transparently to TTCN-3 specifications
- ▶ Extending Abstraction to the implementations between platforms
- ▶ JNI Technology a proved solution

Simple APIs - Go4IT Design



Dual APIs - Design Integration Architecture



Prototype Implementation

Solution Prototype

- ▶ Go4IT TTCN-3 Compiler and Tools in C/C++
- ▶ Implementing TRI Interoperability with Java
 - ▶ SA Library
 - ▶ PA Library
- ▶ Enable the Go4IT platform to use Java implementations and libraries

Implementation Description

Implementation Details

- ▶ Adapting T3RTS
- ▶ Interface Names
- ▶ APIs to modify
 - ▶ SALib
 - ▶ PALib

Implementations features

- ▶ Platform dynamical resolution
- ▶ Configuration

Implemented Examples - DNS Tester

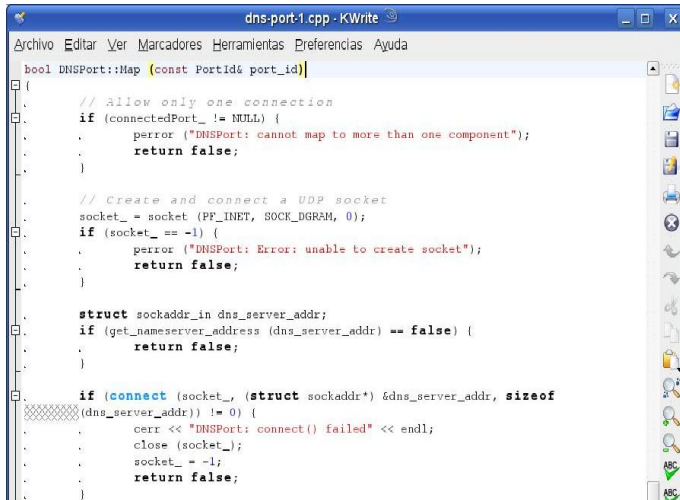
Example Details

- ▶ DNSTester ATS, reused from Go4IT
- ▶ DNSTester ETS core, implemented in C/C++
- ▶ Parts in different platforms
 - ▶ SA Library in Java
 - ▶ PA Library in C/C++

Extra Work Implementation

- ▶ How to configure - XML

Source Code Map - C++



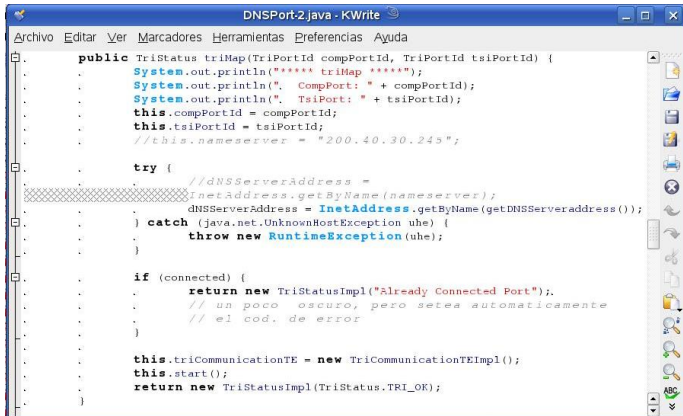
```
bool DNSPort::Map (const PortId& port_id)
{
    // Allow only one connection
    if (connectedPort_ != NULL) {
        perror ("DNSPort: cannot map to more than one component");
        return false;
    }

    // Create and connect a UDP socket
    socket_ = socket (PF_INET, SOCK_DGRAM, 0);
    if (socket_ == -1) {
        perror ("DNSPort: Error: unable to create socket");
        return false;
    }

    struct sockaddr_in dns_server_addr;
    if (get_nameserver_address (dns_server_addr) == false) {
        return false;
    }

    if (connect (socket_, (struct sockaddr*) &dns_server_addr, sizeof
(dns_server_addr)) != 0) {
        cerr << "DNSPort: connect() failed" << endl;
        close (socket_);
        socket_ = -1;
        return false;
    }
}
```

Source Code Map - Java



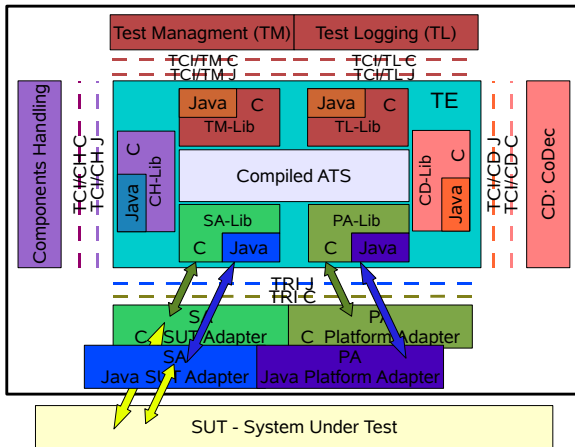
```
public TriStatus triMap(TriPortId compPortId, TriPortId tsiPortId) {
    System.out.println("***** triMap *****");
    System.out.println("  CompPort: " + compPortId);
    System.out.println("  TsiPort: " + tsiPortId);
    this.compPortId = compPortId;
    this.tsiPortId = tsiPortId;
    //this.nameserver = "200.40.30.245";

    try {
        //dNSServerAddress =
        InetAddress.getByAddress(nameServer);
        dNSServerAddress = InetAddress.getByAddress(getDNSServeraddress());
    } catch (java.net.UnknownHostException uhe) {
        throw new RuntimeException(uhe);
    }

    if (connected) {
        return new TriStatusImpl("Already Connected Port");
        // un poco oscuro, pero setea automaticamente
        // el cod. de error
    }

    this.triCommunicationTE = new TriCommunicationTEImpl();
    this.start();
    return new TriStatusImpl(TriStatus.TRI_OK);
}
```

Fianl Solution Design



Results and conclusions

- ▶ This new interpretation of the standard will allow the reuse of tools and libraries between different platforms and suppliers easily and transparently.
- ▶ Promote the effort and partnering to its construction in different and new fields.
- ▶ It will allow the users of the tool to use the best services and solutions provided by each platform, instead of having to choose beforehand the platform of a project as a whole.

Pros and Cons

Pros

- ▶ Transparent use of different platforms components
- ▶ Enhancing TTCN-3 Implementations Abstraction
- ▶ Easily configurable

Cons

- ▶ Dual Implementations are more complex
- ▶ Test Architect should master in TTCN-3, C/C++ and Java

Summary and Remarks

To do

- ▶ Provide a full DUAL API TRI and TCI implementation
 - ▶ SA Library (done)
 - ▶ PA Library (partially done)
 - ▶ TM Library (to do)
 - ▶ CD Library (to do)

Native Interop in others Compilers, Tools, and Languages

- ▶ Dual Libs
 - ▶ Java Compilers using C/C++ code implementations and Libs
 - ▶ C/C++ Compilers using Java code implementations and Libs
- ▶ Apply same idea for others Platforms Languages, e.g. C#

Thank you for your time

Questions?

<http://www.fing.edu.uy/~asabigue/publi/rrezzanot3uc2008paper.pdf>