Metodologia de Teste para Sistemas com DRM

Nuno Miguel Fernandes Dantas de Faria

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores,
orientada pela Professora Doutora Maria Teresa Andrade
e co-orientada pelo Professor Doutor Jorge Mamede

(O Presidente do Júri, Professor Doutor Francisco Restivo)

Faculdade de Engenharia da Universidade do Porto
Departamento de Engenharia Electrotécnica e de Computadores
Rua Roberto Frias, s/n, 4200-465 Porto, Portugal

Março de 2008

Metodologia de Teste para Sistemas com DRM

Nuno Miguel Fernandes Dantas de Faria

Trabalho realizado no âmbito da disciplina de Dissertação, do 1º semestre, do 5º ano, do Mestrado Integrado de Engenharia. Electrotécnica e de Computadores, Ramo Sistemas Telecomunicações, Electrónica e Computadores da Faculdade de Engenharia da Universidade do Porto.

Faculdade de Engenharia da Universidade do Porto
Departamento de Engenharia Electrotécnica e de Computadores
Rua Roberto Frias, s/n, 4200-465 Porto, Portugal

Março de 2008

Resumo

O projecto Metodologias de Teste para Sistemas com DRM tem como objectivo a implementação de um conjunto de componentes de teste que possam ser usados para validar o funcionamento dos elementos desenvolvidos no projecto “Serviço de VoD (Video on Demand) usando DRM (Digital Rights Management)”. Este projecto foi desenvolvido no âmbito da disciplina de Projecto, Seminário ou Trabalho Final de Curso pelos alunos Pedro Oliveira e Vítor Barbosa e realizado nas instalações da Unidade de Telecomunicações e Multimédia do INESC Porto.

Numa primeira fase do projecto, e uma vez que este projecto é uma continuação de um anterior, realizou-se uma análise promenorizada do projecto anterior, tendo em vista uma melhor compreensão dos requisitos deste novo projecto. Nesta fase especificaremos a arquitectura do sistema, sistema esse alvo dos componentes de teste definidos. Será também descrito o projecto VISNET II, em particular à sua arquitectura DRM.

A segunda fase do projecto inicia-se com a abordagem à tecnologia TTCN, em particular a tecnologia TTCN-3, utilizada para definir os componentes de teste, bem como o software adoptado responsável pela implementação desses componentes.

A implementação de componentes de teste é um dos últimos passos do projecto e tem como objectivo testar a comunicação entre diferentes módulos do sistema.

O último passo do projecto está associado a um estudo sobre a viabilidade de implementação de possíveis componentes de teste, numa arquitectura DRM específica, como a arquitectura do projecto VISNET II. No caso de ser viável essa implementação, este projecto poderia ter uma funcionalidade futura para, a partir do software utilizado na implementação dos testes, interagir no projecto VISNET II como ferramenta responsável pela comunicação entre os diferentes módulos do sistema, numa arquitectura DRM. Essa arquitectura DRM pretende gerir os direitos digitais de conteúdos audiovisuais durante todo o seu ciclo de vida e será baseada nos trabalhos desenvolvidos anteriormente e tendo em conta a existência de outras iniciativas que especifiquem um sistema DRM.

Tendo em vista os clientes deste projecto, as palavras de ordem são facilidade de utilização e fiabilidade do software adoptado.

Agradecimentos

Gostaríamos de agradecer ao INESC Porto – Instituto de Engenharia de Sistemas e Computação do Porto – todo o equipamento e local de trabalho disponibilizados, tal como a ajuda, disponibilidade e empatia de todos os colaboradores da UTM – Unidade de Telecomunicações e Multimédia. Agradecemos ainda a importante orientação do Engenheiro Jorge Mamede e da Engenheira Maria Teresa Andrade para a realização deste projecto.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

ASN.1 – Abstract Syntax Notation One

ATS – Abstract Test Suite

CD – Codecs

CH – Component Handling

CL – Core Language

DMP – Digital Media Project

DRM – Digital Rights Management

DSS – Darwin Streaming Server

GFT – Graphical Presentation Format

HTTP – Hypertext Transfer Protocol

IDL – Interface Description Language

IMAP – Internet Message Access Protocol

IP – Internet Protocol

IUT – Implementation Under Test

JDBC – Java Database Connectivity

JNDI – Java Naming and Directory Interface

JSP – JavaServer Pages

JWS – Java Web Service

J2EE – Java 2 Enterprise Edition

LPD – License Provider Device

MTC – Main Test Component

NCSA – National Center for Supercomputing Applications

NNTP – Network News Transfer Protocol

OSI – Open Systems Interconnection

PA – Platform Adapter

PHP – Hypertext Preprocessor

POP3 – Post Office Protocol

PTC – Parallel Test Component

REL – Rights Expression Language

RTP – Real Time Protocol

RTSP – Real Time Streaming Protocol

SA – System Adapter

SOAP – Simple Object Access Protocol

SNMP – Simple Network Management Protocol

SUT – System Under Test

STF – Special Task Force

TA – Test Adapter

TCI – Test Control Interface

TE – Test Executable

TM – Test Management

TRI – Test Runtime Interface

TTCN – Tree and Tabular Combined Notation

TTCN-1 – Tree and Tabular Combined Notation Edition 1

TTCN-2 – Tree and Tabular Combined Notation Edition 2

TTCN-3 – Testing and Test Control Notation

URL – Uniform Resource Locator

WSDD – Web Service Deployment Descriptor

WSDL – Web Service Description Language

XML – eXtensible Markup Language

Capítulo 1

1. Introdução

1.1 Enquadramento

O trabalho descrito neste relatório insere-se no âmbito da disciplina de Dissertação do Mestrado Integrado em Engenharia Electrotécnica e de Computadores (MIEEC), Ramo Sistemas Telecomunicações, Electrónica e Computadores (TEC). O projecto decorreu nas instalações do INESC[1] Porto sob proposta dos investigadores INESC Jorge Mamede e Maria Teresa Andrade e sob supervisão da professora doutora Maria Teresa Andrade da Faculdade de Engenharia da Universidade do Porto (FEUP).

1.2 Motivação

A Internet tornou-se na infra-estrutura de telecomunicações por excelência. Surgiu em 1969, de um projecto do Departamento de Defesa dos Estados Unidos, que tinha como objectivo a interligação de computadores utilizados em centros de investigação com fins militares.

Apresenta diversas vantagens:

- Interactividade: O utilizador não é passivo da informação, pode escolher como quer vê-la e dar uma resposta directa;
- Produtividade: Há a possibilidade de realizar comércio electrónico, intercâmbio de informação, uso de dispositivos e recursos remotos (sistemas de *e-learning*);
- Actualidade: Os documentos na rede actualizam-se continuamente.
- Economia: A informação a que se tem acesso, desde qualquer parte do mundo, está no computador de cada um, de uma forma rápida e idêntica à original;

2:48 2:48

[1] Instituto de Engenharia de Sistemas e Computadores

- Globalidade: Uma vez que se entra na rede, tem-se acesso a toda a informação e aos recursos que lá se encontram.

O desafio que se revela importante nesta altura é a antecipação das necessidades dos consumidores, melhorando as respostas aos seus interesses, através de mais e melhores serviços e funcionalidades disponibilizadas. As empresas que disponibilizam serviços através da Internet precisam assim de ser inovadoras, de apresentar e apostar em soluções capazes de trocar informação de forma organizada e estruturada, dinâmica, de baixo custo e fiável, oferencendo soluções personalizáveis e atractivas para o consumidor não descurando a eficácia, eficiência e agradabilidade em termos em termos de usabilidade do sistema criado.

A expansão contínua da Internet a nível de apostas em novas aplicações multimédia que proporcionem a visualização de conteúdos digitais, criou e tornou alguns problemas evidentes, destancando-se a cópia ilegal desses mesmos conteúdos.

Surge assim o mecanismo Digital Rights Management (DRM), com o objectivo de solucionar o problema, baseado no conceito de criação de licenças associadas aos conteúdos digitais, as quais deverão ser adquiridas pelo consumidor de forma a poder visualizar o vídeo ou conteúdo digital.

Actualmente existem diferentes mecanismos de DRM, projectados por diferentes empresas, contudo, apresentam características em comum:

- Detectam quem acede a cada conteúdo digital, quando e sob quais condições, e reportam essa informação ao proprietário do conteúdo;
- Autorizam ou negam de maneira irrefutável o acesso ao conteúdo, de acordo com as condições estabelecidas pelo proprietário ou fornecedor do conteúdo;
- Quando autorizam o acesso, impõem algumas condições restritivas estabelecidas pelo proprietário da obra.

Com o intuito de proporcionar uma melhor inteligíbilidade, associada ao princípio de funcionamento de uma arquitectura DRM específica, surge a proposta Metodologia de Teste para Sistemas com DRM revelando-se extremamente atraente aos olhos do autor deste documento pelo seu carácter inovador e interactivo. Inovador porque proporciona um desenvolvimento de componentes de teste sobre um sistema, utilizando a tecnologia TTCN-3, uma tecnologia relativamente recente e do nosso ponto de vista, inovadora. Interactivo pois a partir de ferramentas de software de teste, é possível definir e controlar a execução dos testes.

1.3 Objectivos

Este projecto foi lançado com o propósito de implementar um conjunto de componentes de teste em sistemas DRM, utilizando uma tecnologia de descrição de testes formal (isto é, com fundamentação matemática).

Como objectivo adicional, projectou-se um possível componente de teste, considerando a possibilidade de utilização da mesma metodologia, após uma fase prévia de estudo sobre a viabilidade da mesma, numa arquitectura DRM específica, como a arquitectura do projecto VISNET II.

1.4 Estrutura do Relatório

Este documento encontra-se estruturado em 6 capítulos dos quais o primeiro é constituido por esta introdução ao trabalho.

No segundo capítulo é feita uma análise sobre o projecto “Serviço de VoD (Video on Demand) usando DRM (Digital Rights Management)”. É abordado detalhadamente a configuração base do servidor, apresentando as ferramentas instaladas, enumerando também as necessidades do cliente para aceder à totalidade dos serviços. Será também descrito o projecto VISNET II, em particular à sua arquitectura DRM.

No terceiro capítulo é feita uma abordagem concisa sobre a tecnologia TTCN, em particular a tecnologia TTCN-3, utilizada na implementação de componentes de teste. Neste capítulo será também efectuada uma abordagem sobre a Testing Technologies, empresa responsável pela implementação do software utilizado bem como o próprio software responsável pela implementação de componentes de teste, denominado TTworkbench Professional.

No quarto capítulo é apresentada, numa fase inicial, a arquitectura de teste adoptada. Numa fase posterior, é dado início à implementação prática dos componentes de teste efectuados sobre o sistema.

O quinto capítulo apresenta um estudo sobre a viabilidade de implementação de possíveis componentes de teste, numa arquitectura DRM específica, como a arquitectura do projecto VISNET II. Neste capítulo será também apresentado uma possível implementação de um componente de teste, na arquitectura VISNET II, que puderá ser desenvolvido numa fase posterior ao término do projecto.

O derradeiro capítulo tira algumas conclusões sobre a comparação entre as proposições iniciais e os resultados obtidos.

Capítulo 2

2. Projectos com DRM

Pretende-se com este projecto implementar um conjunto de componentes de teste que possam ser usados para validar o funcionamento dos elementos desenvolvidos no projecto "Serviço de VoD (Video on Demand) usando DRM (Digital Rights Management)".

2.1 "Serviço VoD usando DRM"

Este projecto está associado a um serviço de Video On Demand usando DRM desenvolvido no âmbito da disciplina de Projecto, Seminário ou Trabalho Final de Curso pelos alunos Pedro Oliveira e Vítor Barbosa realizado nas instalações da Unidade de Telecomunicações e Multimédia do INESC Porto.

O objectivo final deste serviço visa proporcionar ao cliente a possibilidade de visualizar filmes desde que possua licença para isso. Deste modo, é condição necessária para o acesso ao serviço a utilização de um computador com ligação à Web, um reprodutor multimédia com codec MPEG-4 (por exemplo, QuickTime) e por fim um Web Browser. Existe também uma solução de acesso alternativa cuja opção não dispensa os requisitos anteriormente definidos, que se resume a uma aplicação cliente que permite, não só as mesmas funcionalidades definidas pela solução do Web Browser, mas também a possibilidade de ver um filme armazenado localmente, não descuidando as verificações necessárias e que eram o objecto central deste projecto.

A arquitectura do serviço é do tipo cliente – servidor, sendo que o servidor é constituído por um repositório de bitstreams de vídeo MPEG-4, uma aplicação de streaming de vídeo (Darwin Streaming Server), uma outra para disponibilizar páginas Web dinâmicas on-line (Apache2 e PHP5) e por fim uma base de dados (PostgreSQL) que armazena a informação dos clientes, das suas preferências e licenças bem como toda a informação relacionada com cada filme. No servidor estão também instalados mecanismos de gestão pelo que é possível administrar remotamente a base de dados e a aplicação do Darwin Streaming Server, sendo que a administração desta última permite monitorizar o estado das ligações

estabelecidas e tomar medidas de controlo como, por exemplo, baixar o débito do fluxo de vídeo ou, em caso extremo, terminar a ligação.

O cliente para aceder à totalidade dos serviços necessita ter uma conta pessoal, registando-se na aplicação e apenas poderá visualizar algum dos filmes se possuir licença, a qual poderá ser gerada pela aplicação, via acesso Browser ou Aplicação Cliente. A visualização via Browser far-se-á em tempo real directamente da Web, no entanto também é possível fazer o download do filme, armazená-lo localmente e através da aplicação visualizá-lo desde que se disponha da licença que satisfaça o requisito do filme escolhido.

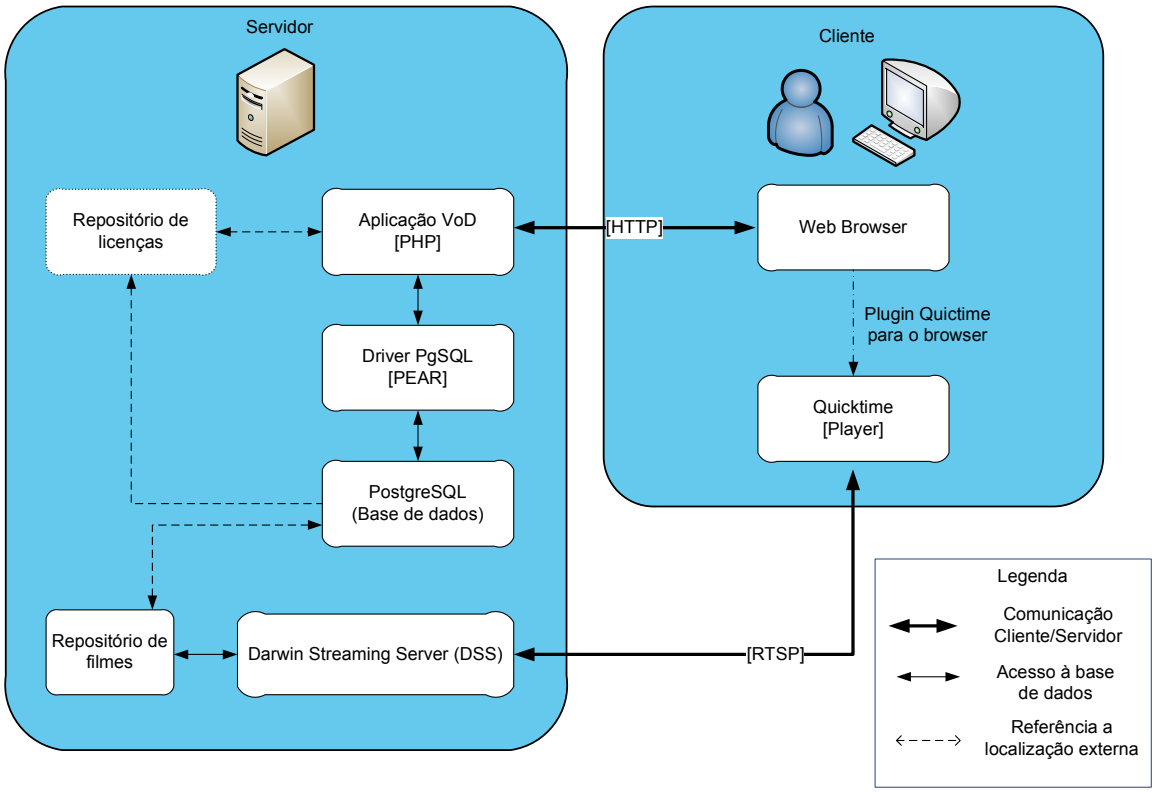


Figura 1 – Arquitectura do sistema de Video on Demand desenvolvido

A Figura 1 ilustra dois tipos de comunicação entre o cliente e o servidor. Numa primeira fase o cliente, por intermédio do seu Web browser, realiza pedidos Hypertext Transfer Protocol (HTTP) ao servidor de modo a navegar pela aplicação Web desenhada em Hypertext Preprocessor (PHP). O PHP é importante, pois no lado do servidor existe a comunicação entre a aplicação Web e a base de dados que contem informações sobre os utilizadores e filmes, permitindo a construção automática de páginas web de acordo com os pedidos dos utilizadores.

Numa segunda fase, escolhido o conteúdo multimédia é possível estabelecer a comunicação entre o software de reprodução do cliente e o Darwin Streaming Server (DSS) de forma a realizar o streaming através do Real Time Streaming Protocol (RTSP). Este protocolo é utilizado para controlo de fluxo ao nível da camada de aplicação. Este protocolo é

desenvolvido para comunicar com protocolos de nível inferior, nomeadamente o Real Time Protocol (RTP), proporcionando um serviço completo de streaming sobre Internet Protocol (IP). O conceito de streaming está associado ao transporte de um meio multimédia em tempo real desde um servidor até um cliente através de uma rede. Através desta técnica, o conteúdo multimédia é apresentado pelo software do cliente à medida que a informação vai sendo entregue, não se chegando a efectuar o download do conteúdo para o disco do utilizador.

Apresenta-se de seguida a arquitectura do sistema, abordando detalhadamente as ferramentas necessárias para construir/configurar o servidor, implementado neste projecto e utilizada agora como sistema alvo de testes. Uma vez que essa máquina, já se encontrava configurada, foi aproveitada como base de trabalho para este projecto.

2.1.1 O Servidor

Com base na análise deste projecto, chegou-se a conclusão que o servidor estaria configurado com uma base de dados, um servidor de HTTP/PHP, um servidor Tomcat (JAVA) necessário para a realização de uma análise detalhada acerca da criação de licenças de acordo com o Digital Media Project (DMP), o esquema de licenças que foi adoptado, a framework Apache Axis, a interface License Provider Device (LPD) e o Darwin Streaming Server, uma aplicação de streaming de vídeo.

2.1.1.1 Sistema Operativo

O primeiro passo na instalação e configuração do servidor foi a escolha do sistema operativo. Um sistema operativo é um programa ou um conjunto de programas cuja função é servir de interface entre um computador e o utilizador.

Optaram por um servidor num sistema operativo Linux, nomeadamente a distribuíçao Ubuntu, uma vez que apresenta um processamento eficiente embora seja menos interactivo que o sistema operativo baseado em Win32. A opção de escolha da distribuição esteve associada ao facto da crescente utilização da distribuição Ubuntu, o que iria facilitar a troca de experiências na resolução de problemas.

2.1.1.2 Servidor Apache HTTP

O passo seguinte foi a instalação de um servidor HTTP, neste caso a escolha recaiu sobre o servidor Apache (Apache Server) [1] dado ser o servidor web livre mais bem sucedido. Foi criado em 1995 por Rob McCool, então funcionário do NCSA (National Center for Supercomputing Applications), Universidade de Illinois. Na última pesquisa efectuada pelo site www.netcraft.com, em Dezembro de 2005, foi comunicado que a utilização do Apache supera 60% de servidores activos no mundo, dado ser uma aplicação de código fonte aberto.

É a principal tecnologia da Apache Software Foundation, responsável por mais de uma dezena de projectos envolvendo tecnologias de transmissão via web, processamento de dados e execução de aplicativos distribuídos.

As suas funcionalidades são mantidas através de uma estrutura de módulos, podendo inclusive o utilizador implementar os seus próprios módulos, utilizando a API do software.

É disponibilizado versões para os sistemas Windows, Novell Netware, OS/2 e para outros sistemas do padrão POSIX (Unix, Linux, FreeBSD).

2.1.1.3 PHP

Uma vez que o servidor Apache HTTP não implementa a tal interactividade que este serviço requer, surge a necessidade de encontrar uma tecnologia que esteja associado ao uso de páginas web dinâmicas. Entre as várias tecnologias que satisfaziam o dinamismo pretendido, a opção recaiu sobre a linguagem PHP [2].

PHP é um acrónimo recursivo para "PHP: Hypertext Preprocessor", caracterizado por uma linguagem de programação de computadores interpretada, livre e muito utilizada para gerar conteúdo dinâmico na Web. Apesar de ser uma linguagem de fácil aprendizagem e utilização em scripts dinâmicos, o PHP é uma linguagem poderosa orientada a objectos cuja sintaxe é similar à linguagem C/C++ e PERL.

Suporta um grande número de bases de dados, tais como Oracle, Sybase, PostgreSQL, InterBase, MySQL, SQLite, MSSQL, Firebird, e protocolos como Internet Message Access Protocol (IMAP), Simple Network Management Protocol (SNMP), Network News Transfer Protocol (NNTP), Post Office Protocol (POP3), HTTP, Lightweight Directory Access Protocol (LDAP), Simple Object Access Protocol (SOAP). É ainda possível abrir sockets e interagir com outros protocolos.

2.1.1.4 Base de dados PostgreSQL

O PostgreSQL [3] é um sistema de gestão de base de dados (SGBD) inicialmente desenvolvido na Universidade de Berkeley, Califórnia. A partir de 1996, o PostgreSQL, consolidou-se como um projecto de código-fonte aberto desenvolvido por uma comunidade de voluntários sob a coordenação do PostgreSQL Global Development Group. Além de doações, o projecto PostgreSQL é financiado pelo patrocínio de diversas empresas, entre as quais se destacam: Fujitsu, Hub.Org, NTT Group, Red Hat, Skype, SRA e Sun Microsystems.

O software tem adquirido prestígio na comunidade Linux, tendo recebido diversas vezes o prémio Linux Journal Editor's Choice de melhor SGBD.

2.1.1.5 phpPgAdmin

O phpPgAdmin [4] traduz-se como uma interface gráfica para a gestão da base de dados remotamente, via protocolo HTTP e apresentação dinâmica via PHP. Deste modo é possível realizar todas a funções sobre a base de dados de maneira intuitiva.

2.1.1.6 Secure Shell

Secure Shell ou SSH é, simultaneamente, um programa de computador e um protocolo de rede que permite a conexão com outro computador, de forma a executar comandos de uma unidade remota. Possui as mesmas funcionalidades do TELNET, com a vantagem da conexão entre o cliente e o servidor ser encriptada.

O SSH faz parte da lista de protocolos TCP/IP que torna segura a administração remota de um servidor Unix.

É possível assim, aceder remotamente ao servidor para operações de manutenção ou administração.

2.1.1.7 Servidor Tomcat

O Tomcat [5] é um servidor de aplicações Java para Web sendo distribuído como software livre e desenvolvido como código-fonte aberto dentro do conceituado projecto Apache Jakarta. É oficialmente apoiado pela Sun como a Implementação de Referência (RI) para as tecnologias Java Servlet e JavaServer Pages (JSP). O Tomcat é robusto e eficiente tornando possível a sua utilização em ambiente de produção.

Tecnicamente, o Tomcat é um container para aplicações Web, cobrindo parte da especificação Java 2 Enterprise Edition (J2EE) com tecnologias como Servlet e JSP, e tecnologias de apoio relacionadas como Realms e segurança, Java Naming and Directory Interface (JNDI) e Java Database Connectivity (JDBC). Tem ainda, a capacidade de actuar como servidor web/HTTP, ou pode funcionar integrado com um servidor web dedicado como o Apache HTTP ou o Microsoft IIS.

Apesar de actualmente não desempenhar um papel relevante no serviço que disponibilizaram, foi fundamental para a compreensão do conceito de licença e como esta pode ser gerada. A sua instalação estava relacionada com o facto de se terem baseado no projecto Chillout do Digital Media Project (DMP), o qual tinha como interface para o License Provider Device (LPD) uma aplicação Web em JSP.

2.1.1.8 Axis

O Apache Axis [6] é uma framework de código aberto, baseado na linguagem JAVA e no padrão eXtensible Markup Language (XML), utilizado para construção de serviços Web no protocolo SOAP, possibilitando a criação de aplicações distribuídas em Java. Além da

versão para Java, existe uma implementação baseada na linguagem C++. O projecto Apache Axis é suportado pela Apache Software Foundation.

O Axis disponibiliza dois modos para "expor" os métodos de uma classe através de serviços Web. O modo mais simples utiliza os arquivos Java Web Service (JWS) do Axis. O outro método utiliza um arquivo Web Service Deployment Descriptor (WSDD), que descreve com detalhe como serão criados os serviços web a partir dos recursos (classes Java) existentes.

Também é possível, através do Axis, gerar automaticamente o arquivo Web Service Description Language (WSDL), que contém a definição da interface dos serviços Web.

Esta framework teve de ser instalada uma vez que o LPD depende dela.

2.1.1.9 LPD

O LPD, License Provider Device, foi uma interface desenvolvida no âmbito do Project Chillout do Digital Media Project (DMP) cuja finalidade se enquadra com a criação de licenças para conteúdos digitais.

O LPD foi utilizado como teste, gerando um número alargado de licenças e permitindo uma análise detalhada sobre as mesmas, dando informações relevantes para desenvolver um criador de licenças via PHP. Assim foi possível integrar o criador de licenças, elaborando um pequeno programa que gera o mesmo output do LPD, na aplicação Web que vinha a ser desenvolvida. Deste modo, a interface que o Chillout propõe veio a tornar-se dispensável.

Por sua vez, o Digital Media Project (DMP) é uma organização cuja missão é promover o desenvolvimento e utilização eficaz de conteúdos digitais respeitando os direitos de autor e de distribuição. Esta associação apoia-se e aposta no desenvolvimento da tecnologia Digital Rights Management (DRM).

O DRM é desenvolvido por diferentes empresas, o que pode dificultar ou tornar impossível a comunicação entre utilizadores de diferentes tecnologias de gestão de direitos de autor. O projecto DMP aposta na interoperabilidade entre diferentes sistemas e estabelece assim a necessidade de criar um DRM universal, partindo da uniformização (criação de um standard) de pequenas funções para atingir esse fim.

2.1.1.10 Darwin Streaming Server

O Darwin Streaming Server (DSS) é um servidor de streaming de código livre, pertencendo ao projecto Open Source Streaming Server da Apple. Este projecto usa os protocolos RTP e RTSP sendo capaz de realizar o streaming dos formatos H.264, MPEG-4 e 3GPP sobre a Internet em tempo real.

O servidor está actualmente na sua versão 5.5.5, existindo versões para várias plataformas como Mac OS, Linux e Windows. A versão utilizada como servidor de streaming foi a 5.5.4.

Após a fase de instalação e configuração do servidor, resta apenas arrancar o serviço. O servidor Darwin é iniciado e, para além disso, é lançado uma aplicação web em Perl que permite a administração do servidor. Para aceder a essa aplicação é necessário estabelecer uma ligação HTTP no porto 1220 da máquina que hospeda a aplicação. Através da aplicação web é possível alterar as configurações do servidor, como permitir ou inibir fluxo na porto 80, definir o número máximo de utilizadores simultaneamente, a largura de banda máxima fornecida, o caminho para o repositório de filmes ou alterar o método de autenticação entre outros. A aplicação permite também obter informações sobre os clientes conectados a um conteúdo audiovisual no servidor assim como conhecer estatísticas sobre o servidor e controlar o seu estado.

O servidor está automaticamente configurado para escutar pedidos RTSP no porto 554 e poderá ser configurado para permitir o fluxo de dados através do porto 80 para ultrapassar possíveis problemas com firewalls.

2.1.2 O Cliente

Como já foi enunciado anteriormente, o cliente para aceder à totalidade dos serviços necessita ter uma conta pessoal, registando-se na aplicação e apenas poderá visualizar algum dos filmes se possuir licença, a qual poderá ser gerada pela aplicação, via acesso Browser ou Aplicação Cliente.

Se o cliente aceder via Browser, como podemos constatar na figura 1, tudo o que o cliente necessita para aceder à aplicação, é um browser e um reprodutor multimédia. Existem vários browsers disponíveis como o Internet Explorer, Mozilla Firefox, Opera. Quanto ao reprodutor multimédia, tanto o Windows Media Player como o QuickTime Player podem ser utilizados, uma vez que apresentam o descodificador MPEG-4, ou com capacidade para reproduzir este tipo de ficheiros. A visualização via Browser far-se-á em tempo real directamente da Web.

Se o cliente aceder via Aplicação Cliente, a reprodução de conteúdos audiovisuais é efectuada localmente, sendo desnecessário o módulo referente ao servidor de streaming. Esta aplicação caracteriza-se por ser um software para instalação local no terminal do utilizador, pretendendo fornecer serviços complementares aos que são já oferecidos via aplicação web.

Foi desenvolvido para o sistema operativo Windows dado ser este o mais utilizado pelos utilizadores comuns. A linguagem de programação escolhida foi o Visual Basic pois é um produto Microsoft cuja finalidade é desenvolver aplicações para Windows.

2.2 VISNET II

O projecto VISNET II [7] [8] – Networked Audio Visual Media Technologies é uma rede de excelência recentemente aprovada para financiamento pela Comissão na sequência do processo de avaliação de propostas submetidas na 1ª call do programa IST do 6º Programa Quadro da CE, na área estratégica IST-2002-2.3.1.8 "Networked audiovisual systems and home platforms".

O projecto tem uma duração prevista de cinco anos e reúne especialistas europeus de renome nas áreas de processamento, análise e codificação de áudio e vídeo, metadata e transmissão em redes heterogéneas.

O plano de trabalhos do VISNET II pretende incentivar a troca de conhecimento e experiências dos investigadores do consórcio nas áreas em que são peritos e onde têm vindo a desenvolver a sua investigação. Visa maioritariamente uma integração dos esforços de investigação e a efectiva divulgação externa de informação e resultados, de forma a garantir um impacto significativo em toda a comunidade científica europeia.

A leitura do artigo “Intermediate Progress Report on DRM-based Content Protection Initial Integration”, deu uma ideia importante da posição do VISNET II sobre o DRM. No VISNET II pretende-se definir uma arquitectura DRM para gerir os direitos digitais de conteúdos audiovisuais durante todo o seu ciclo de vida. Esta arquitectura será baseada nos trabalhos relatados noutros artigos e tendo em conta a existência de outras iniciativas que especifiquem um sistema DRM.

A arquitectura defendida pelo VISNET II é a DMAG-MIPAMS, como se pode constatar na figura seguinte, que é uma utilizada para administrar informação multimédia tendo em consideração direitos digitais e protecção.

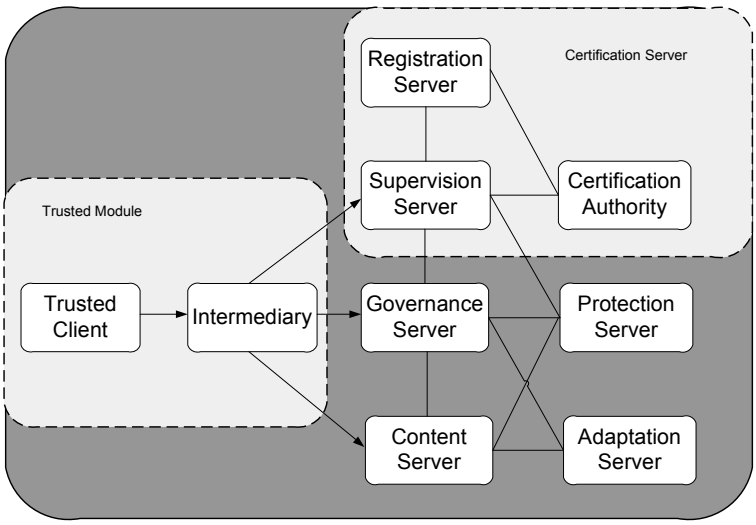


Figura 2 – Arquitectura DMAG-MIPAMS

No contexto da arquitectura DMAG-MIPAMS, define-se dois conceitos importantes: actores e componentes. Um actor é um utilizador (pessoa ou organização) que usa um componente. Um componente é um módulo ou conjunto de software que oferecem funcionalidades relacionadas com DRM e que interagem com outros componentes.

Como é possível ver na figura anterior, a arquitectura está dividida nos seguintes módulos:

2.2.1 Content Server

Este componente permite aos utilizadores explorar e seleccionar conteúdos, fornece o conteúdo que os utilizadores finais seleccionem para visualização, codifica e insere metadata nos conteúdos provenientes dos proprietários;

2.2.2 Adaptation Server

Este componente realiza a adaptação dos conteúdos e dos seus metadados associados, dependendo das condições de transmissão, armazenamento e reprodução. Pode ser incluído como parte integrante do módulo Content Server. A adaptação de metadados pode também envolver a adaptação de licenças. Novas licenças podem ser criadas durante o processo de adaptação, respeitando sempre os termos e condições das licenças originais ou das licenças com conteúdos adaptados;

2.2.3 Protection Server

Este componente realiza a protecção dos conteúdos digitais, através de técnicas de encriptação, faz a descrição e o download das ferramentas de protecção, gera chaves de protecção e armazena as chaves geradas. O processo de prótecção é um processo reversível, uma vez que é possível desproteger um conteúdo inicialmente protegido. Pode ser utilizado pelo Content Server;

2.2.4 Governance Server

Este componente gera as licenças para distribuição ou utilizadores finais, armazena as licenças, traduz as licenças e fornece autorização para as licenças;

Essas licenças foram desenvolvidas segundo a norma MPEG-21 REL [9], apresentada no capítulo 4 deste documento. Uma vez que as licenças necessitavam de um mecanismo de protecção, a UPC-AC implementou um conjunto de ferramentas que proporcionassem essa protecção, baseada em dois mecanismos: Assinatura Digital e Encriptação. Essas ferramentas implementadas incluem funcionalidades para o mecanismo de protecção através de assinatura digital de licenças MPEG-21 REL, proporcionando-lhes integridade e autenticidade, e para o mecanismo de protecção através de encriptação de licenças. Apresenta-se na figura seguinte a arquitectura do módulo responsável pela protecção das licenças,

utilizando no primeiro caso o mecanismo de protecção baseado na assinatura digital e no segundo caso o mecanismo de protecção baseado na encriptação.

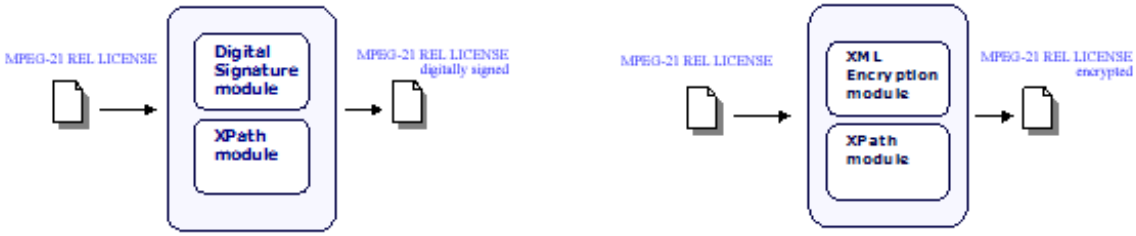


Figura 3 – Arquitectura do Módulo responsável pela protecção de licenças

2.2.5 Certification Server

Está dividido entre o Certification Authority, Registration Server e o Supervision Server envolvendo várias funcionalidades diferentes.

- Certification Authority – Fornece certificados X.509 para as ferramentas instaladas, para utilizadores registados e para os componentes de servidores com diferente arquitectura;
- Registration Server – Usado no registo de actores e ferramentas;
- Supervision Server – Autentica e supervisiona actores e componentes do sistema. É ainda responsável por extrair e registar informações sobre as ferramentas instaladas, recebe e armazena relatórios sobre as acções dos utilizadores;

  Durante os primeiros tempos do projecto VISNET II, foram definidos diferentes mecanismos para proteger a informação enviada e os relatórios dentro de um sistema DRM, em que se destaca o mecanismo implementado pela UPC-AC, um módulo de software que implementa os mecanismos de segurança para Event Reporting. Apresenta duas principais componentes, Digital Signature Module e o Encryption Module. A primeira componente, responsável pela assinatura digital dos relatórios dos acontecimentos, é utilizada na arquitectura DRM para proporcionar integridade e autenticidade na requisição de informação. A segunda componente, responsável pela encriptação dos relatórios dos acontecimentos, é utilizada na arquitectura DRM para restringir o acesso ao conteúdo desses relatórios. Apresenta-se na figura seguinte a arquitectura do módulo de protecção dos relatórios dos acontecimentos ou Event Reports (ERs).

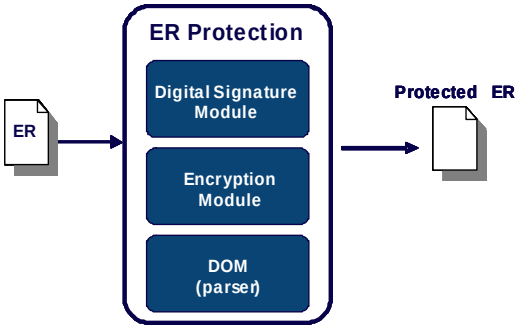


Figura 4 – Arquitectura do Módulo de protecção ER

2.2.6 Trusted Module

Está dividido entre o Trusted Client e o Intermediary envolvendo funcionalidades diferentes.

- Trusted Client – Este módulo é aquele que interage directamente com a aplicação do cliente de forma a reforçar o DRM. Consiste num software de confiança para o sistema, num repositório local de licenças, apresenta informação sobre a protecção e relatórios sobre operações offline.
- Intermediary – Normalmente é parte integrante do módulo Trusted Client mas pode também estar localizado no servidor de modo a reduzir o número de componentes que o módulo no cliente necessita. Pode ser visto como um corrector com quem o cliente comunica para obter autorização e as chaves necessárias para desproteger o conteúdo e dessa forma possibilitar a reprodução do mesmo. Apresenta como principais funcionalidades:
    - Requer certificação para o Supervision Server.
    - Requer verificação para o Supervision Server.
    - Requer autorização online para o Gonvernance Server.
    - Efectua o download da licença proveniente do Gonvernance Server.
    - Envia operações offline para o Supervision Server.
    - Efectua o donwload de ferramentas desprotegidas proveniente do Protection Server.

2.2.7 Caso de Utilização

A seguir, como se pode constatar na figura seguinte, será apresentado um caso de utilização de conteúdo por parte de um utilizador final, que relaciona a arquitectura e o processamento do conteúdo multimédia protegido. Assumimos que neste caso, não é a primeira vez que se reproduz um conteúdo pela aplicação cliente (User Tool), encontrando-se os plugins já certificados pelo sistema.

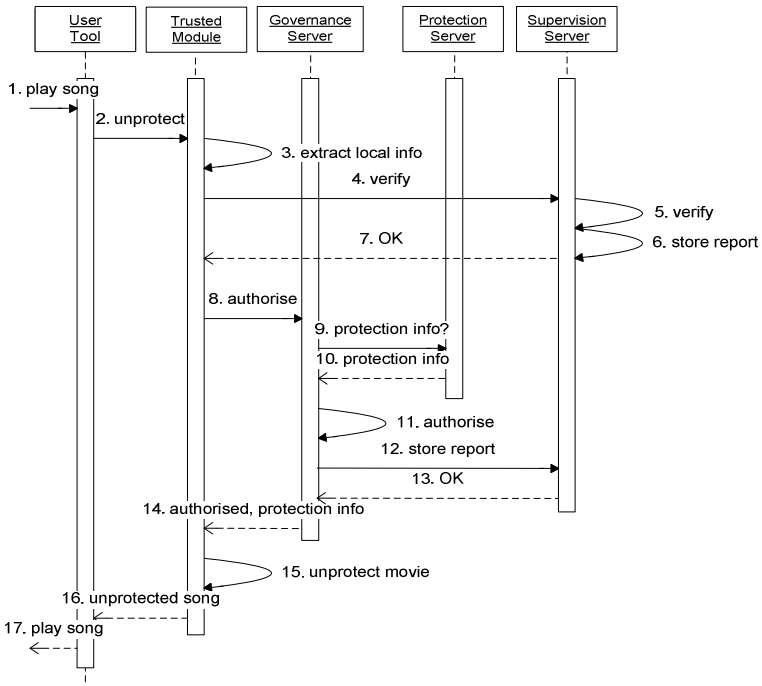


Figura 5 – Caso de utilização da reprodução de um conteúdo protegido

Os passos envolvidos na reprodução de um conteúdo protegido, representados na Figura 5 são os seguintes:

1. O utilizador tenta ouvir a música no seu reprodutor de multimédia.
2. O reprodutor de multimédia necessita enviar o pedido para desproteger o conteúdo a um módulo interno (Trusted Module).
3. O Trusted Module extrai informação local acerca da ferramenta, utilizador e dispositivo que quer aceder ao conteúdo. Verifica também os relatórios acerca das acções realizadas offline, se disponível.
4. O Trusted Module contacta o Supervision Server e envia a informação extraída.
5. O Supervision Server realiza a autenticação do utilizador e das suas ferramentas verificando o seu estatuto e a integridade das mesmas ferramentas. Verifica ainda a lista de acções offline e introduz as acções na respectiva base de dados.

6. Opcionalmente, o Supervision Server guarda um relatório com os resultados da verificação anterior.
7. O Supervision Server notifica o Trusted Module do utilizador com o resultado da verificação.
8. Se a resposta à verificação realizada for positiva, o Trusted Module pede autorização ao Governance Server.
9. O Governance Server determina se a informação necessária para desproteger o conteúdo pretendido está disponível no Protection Server.
10. Se a informação estiver disponível no Protection Server é retornada ao Governance Server.
11. Neste passo é dada a autorização, ou seja, uma licença é criada usando o repositório de licenças.
12. O Governance Server envia a informação sobre a resposta à tentativa de autorização para o Supervision Server.
13. O Supervision Server confirma a recepção e armazenamento do relatório.
14. Neste passo o Governance Server notifica o Trusted Module com a resposta ao pedido de autorização. Se esta for positiva, envia a informação necessária para a reprodução do conteúdo.
15. Se o resultado do pedido de autorização do pedido for positivo, o conteúdo será desprotegido.
16. O conteúdo digital é enviado para a aplicação que fez o pedido.
17. O conteúdo digital é reproduzido.

NOTA: A figura 5 mostra os passos envolvidos na reprodução de um conteúdo protegido e demonstra a necessidade de existir um intermediário entre o sistema de DRM e a aplicação cliente. Esse intermediário, denominado, Trusted Client, como se pode constatar na figura 2, é uma ferramenta reconhecida e licenciada para comunicar com o sistema.

Capítulo 3

3. Tecnologias de Suporte

Uma importante parte do projecto foi dedicada ao estudo de uma tecnologia de testes denominada Tree and Tabular Combined Notation (TTCN), em particular a recente tecnologia Testing and Test Control Notation 3rd Generation (TTCN-3) [12] [13] [14], utilizada na implementação de componentes de teste. Neste capítulo será também efectuada uma abordagem sobre a Testing Technologies, empresa responsável pelo desenvolvimento do software TTworkbench, software esse utilizado para a implementação de componentes de teste. Uma vez que foi utilizado ao longo do nosso projecto, duas versões do TTworkbench, faremos inicialmente uma abordagem sobre a versão inicialmente utilizado, denominado TTworkbench Basic e as razões porque nos levaram a optar por desenvolver os testes noutra versão do TTworkbench, denominado TTworkbench Professional. Ambas as versões do TTworkbench encontram-se descritas na secção 3.2.

Para se perceber melhor o que se acabou de enunciar, neste capítulo serão apresentadas as ideias principais que se devem reter da tecnologia TTCN-3 e softwares mencionados.

3.1 O TTCN

TTCN [10] é uma linguagem de programação dedicada ao teste de protocolos de comunicação e serviços web. De entre os campos de aplicação desta linguagem, destacam-se as comunicações móveis (GSM, 3G, TETRA), wireless LANs (Hiperlan/2), tecnologias Broadband (B-ISDN, ATM), plataformas CORBA e protocolos de Internet como IPv6, SIGTRAN, SIP e OSP.

Actualmente existem 3 versões diferentes de TTCN:

- **TTCN-1** *(Tree and Tabular Combined Notation Edition 1)*: Primeira versão de TTCN, utilizada até ao ano 1998.
- **TTCN-2** *(Tree and Tabular Combined Notation Edition 2)*: Segunda versão de TTCN, utilizada a partir do ano 1998 até o ano 2003.

- **TTCN-3** *(Testing and Test Control Notation)*: Terceira e currente versão de TTCN, utilizada a partir do ano 2003.

Os primeiros trabalhos em TTCN [11] surgiram por volta de 1984, em ISO/IEC JTC 1/ SC 21 e em CCITT SG VII como parte de um trabalho de testes de metodologias e framework, na Open Systems Interconnection (OSI). Esta tecnologia foi normalizada como textos ISO/IEC 9646-3 e CCITT Rec X.292 em 1992 como um dos 7 conjuntos de textos. Desde então, TTCN era frequentemente utilizado para descrever protocolos de testes de conformance, ou seja, testes efectuados em produtos na mesma gama do produto testado (por exemplo: testes de comunicação efectuados entre dois telefones), em organizações normalizadas como ITU-T [30], ISO/IEC, ATM Fórum, ETSI [29] e em indústrias.

Contudo, a primeira edição de TTCN não foi adequadamente concebido para descrever o comportamento concorrente, ou seja, a comunicação entre diferentes componentes a partir do testador ou a partir do Implementation Under Test (IUT) ou entre eles e a necessidade da descrição do comportamento concorrente era indispensável. Neste processo de expansão de funcionalidades, para além do mecanismo concorrente, os conceitos de módulo e empacotar era novamente introduzidos em TTCN para aumentar a reusabilidade e alcançar encapsulamento. Também a manipulação de codificação Abstract Syntax Notation One (ASN.1) foi possível, para além da simples declaração de síntaxe. Uma vez que o TTCN foi sofrendo alterações, surgiu, deste modo, uma nova versão de TTCN (TTCN -2) em ISO/IEC, e em ITU-T, em 1998.

Uma vez que TTCN-2 apresentava algumas limitações em certos tipos de testes, como testes de interoperabilidade (testes efectuados em produtos de gamas diferentes do produto testado, por exemplo, um teste específico num telefone, numa rede), teste de robustez, testes de regressão, testes de sistema e de integração, o que limitou as suas áreas de aplicação, surge a necessidade de encontrar uma nova linguagem de programação, mais poderosa e flexível. A fim de encontrar tal linguagem, a Special Task Force (STF) 133 e 156 de ETSI começaram a trabalhar numa nova versão de TTCN em 1998 e finalizaram em Outubro de 2000.

Surge assim uma nova versão de TTCN, TTCN-3, concebido para implementar todo o tipo de componentes de teste cuja a segunda versão de TTCN não conseguiu implementar.

3.1.1 O TTCN-3

TTCN-3 é a única linguagem internacional de teste normalizada pelo ETSI e ITU-T. Foi criada para específicos e complexos sistemas de teste e é considerada uma solução para diferentes tipos de tecnologias.

Para assegurar uma elevada qualidade de serviços e produtos, o teste deve ser aplicada numa fase primária do ciclo de desenvolvimento do produto ou serviço, o que irá assegurar uma qualidade optimizada em todos os estados do processo.

Com a nova síntaxe, TTCN-3 apresenta um aspecto e uma maneira de funcionar de linguagem de programação moderna. É uma linguagem flexível, fácil de aprender, usar e implementar. As capacidades de TTCN-3 são atractivas aos olhos de um grande número de utilizadores utilizando diferentes tecnologias.

As típicas áreas de aplicação desta tecnologia [21] estão associadas a protocolos, serviços, APIs, módulos de software:

- Session Initiation Protocol (SIP) para Voice over IP (VoIP)
- IPv6 (Core, Mobility, Security)
- Digital Mobile Radio (DMR)
- HiperMAN / WiMax
- 3GPP IP Multimedia Subsystem (IMS)

TTCN-3 é actualmente uma tecnologia bem estabelecida no mercado. É usado em Companhias com um grande alcance de sectores, como por exemplo:

- Fabricantes – Motorola, Siemens, Ericsson, Seagate, Sonus Networks, TI
- Transportadores – Vodafone, O2
- Dispositivos de Teste – Navtel Communications, Alcatel, Tektronix

Tecnologia TTCN-3 versus Tecnologia TTCN-2

A tecnologia TTCN-3 apresenta algumas diferenças comparativamente à tecnologia TTCN-2. Assim sendo, a tecnologia TTCN-3 apresenta as seguintes características:

- Um linguagem de programação mais fácil de aprender.
- Um novo formato gráfico de apresentação.
- Uma completa configuração dinâmica de testes.
- Suporta comunicação síncrona.

- Suporta sistemas de teste distribuídos.
- Interfaces normalizadas de sistemas de teste: TTCN-3 Runtime Interface (TRI) e TTCN-3 Control Interface (TCI).
- Melhor harmonização com uma linguagem formal para descrever mensagens de uma forma abstracta (ASN.1).
- Mecanismo para integrar outros tipos de sistemas, como XML, ASN.1, C.

Standards TTCN-3 Existentes

Apresentam-se de seguida os standards TTCN-3 disponíveis, até à data:

- **ETSI ES 201 873-1: TTCN-3 Core Language (CL)**

TTCN-3 Core Language é um formato de apresentação textual cujo objectivo está associado à especificação de conjuntos de componentes de teste, incluindo síntaxe BNF e operações semânticas. Um editor disponível para gerar TTCN-3 Core Language é o CL Editor integrado na Testing Technologies TTworkbench.

- **ETSI ES 201 873-2: TTCN-3 Tabular Presentation Format (TFT)**

TTCN-3 Tabular Presentation Format representa outra maneira para descrever componentes de teste na tecnologia TTCN-3, usando tabulações.

- **ETSI ES 201 873-3: TTCN-3 Graphical Presentation Format (GFT)**

TTCN-3 Graphical Presentation Format (GFT), tem por base ITU-T Recomendation Z.120. O GFT mostra o conjunto de componentes de teste efectuados de uma maneira mais visual. Um GFT disponível é GFT Editor da Testing Technologies.

- **ETSI ES 201 873-4: TTCN-3 Semantics**

Este standard TTCN-3 descreve a semântica, significado e definição de funções, tipos de dados e templates, que o TTCN-3 Core Language (CL) utiliza na execução de componentes de teste.

- **ETSI ES 201 873-5: TTCN-3 Runtime Interface (TRI)**

TRI é o Runtime Interface no conceito e standard TTCN-3 sendo responsável pela implementação de componentes de teste específicos. Descreve e define a comunicação entre os testes e o sistema a testar no System Adapter (SA). Inclui também uma plataforma Adaptador para aceder a funções externas e integração de conceitos de tempo.

- **ETSI ES 201 873-6: TTCN-3 Control Interface (TCI)**

TCI está associada à parte do controlo do standard TTCN-3. Inclui interfaces normalizadas para Component Handling (CH) e Test Management (TM) e fornece acesso para os Codecs (CD) para codificar/descodificar dados em TTCN-3.

Em fase de conclusão, encontram-se os seguintes standards TTCN-3:

- **ETSI ES 201 873-7: Using ASN.1 with TTCN-3**

Este standard descreve a integração da tecnologia Abstract Syntax Notation One (ASN.1) com o TTCN-3 Core Language (CL).

- **ETSI ES 201 873-8: Using IDL with TTCN-3**

Este standard descreve a integração da tecnologia Interface Description Language (IDL) com o TTCN-3 Core Language (CL).

- **ES 201 873-9: Using XML with TTCN-3**

Este standard descreve a integração da tecnologia eXtensive Markup Language (XML) com o TTCN-3 Core Language (CL).

Implementação de Testes em TTCN-3

Apresenta-se de seguida as diferentes fases para a implementação de um componente de teste, desde a sua especificação, até à sua implementação, passando pela sua compilação:

1 Fase: Especificação do teste

- **Síntaxe TTCN-3**

TTCN-3 oferece 3 formatos diferentes de apresentação para a especificação do teste. A versão TTCN-2 apenas suporta 2 formatos, sendo eles o formato tabular e o formato notação core. O novo formato suportado, o gráfico, permite ao utilizador TTCN-3 específicar complexos testes de uma forma rápida e visual. Apresenta-se de seguida as três interfaces:

Notação Core :

```
testcase myTestcase () runs on MTCType system TSIType
{   mydefault := activate (OtherwiseFail);
    verdict.set(pass);

    :
    connect(PTC_ISAP1:CP_ISAP1,mtc:CP_ISAP1);
    :
    map(PTC_ISAP1:ISAP1, system:TSI_ISAP1);
    :
    PTC_ISAP1.start(func_PTC_ISAP1());
    PTC_MSAP2.start(func_PTC_MSAP2());
    Synchronization();
    all component.done;
    log("Correct Termination");
}
```

Figura 6 – Interface Notação Core

Formato Gráfico :

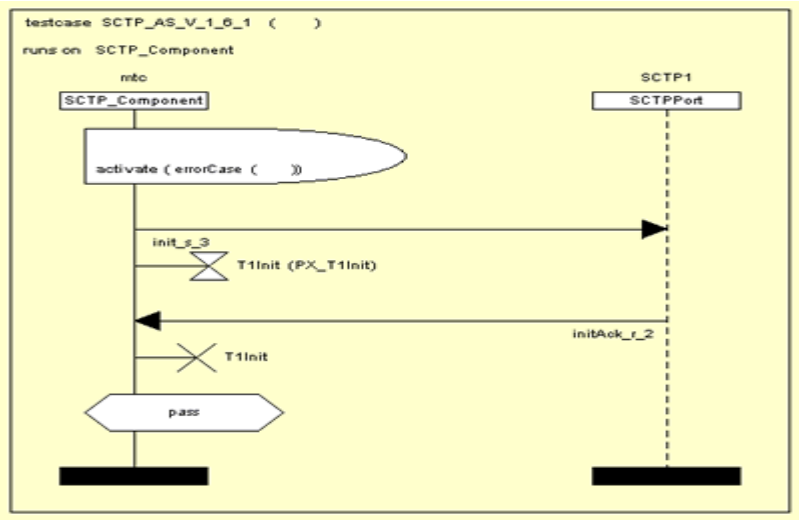
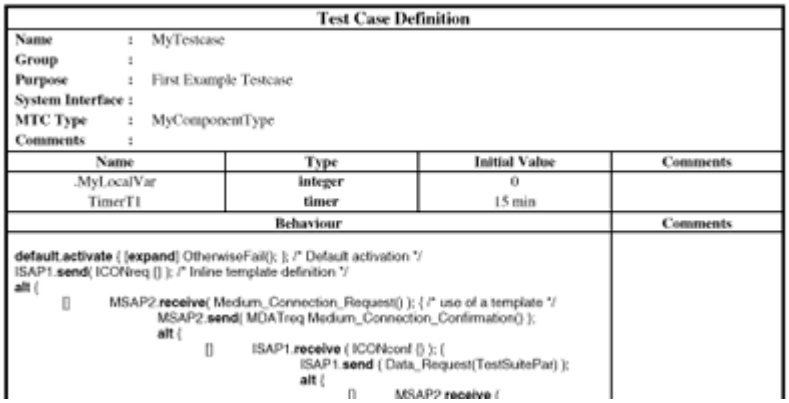


Figura 7 – Interface Gráfica

Notação Tabular:



Test Case Definition

Name	:	MyTestcase
Group	:	
Purpose	:	First Example Testcase
System Interface	:	
MTC Type	:	MyComponentType
Comments	:	

Name	Type	Initial Value	Comments
.MyLocalVar	**integer**	0	
TimerT1	**timer**	15 min	

Behaviour	Comments

```
default.activate ( [expand] OtherwiseFail(); ); /* Default activation */
ISAP1.send( ICONreq {} ); /* Inline template definition */
alt {
        []      MSAP2.receive( Medium_Connection_Request() ); { /* use of a template */
                        MSAP2.send( MDATreq Medium_Connection_Confirmation() );
                        alt {
                                []      ISAP1.receive ( ICONconf {} ); {
                                                ISAP1.send ( Data_Request(TestSuitePar) );
                                                alt {
                                                        []      MSAP2.receive (
```

Figura 8 – Interface Notação Tabular

- **Integração das especificações existentes em TTCN-3**

A tecnologia TTCN-3 é uma linguagem aberta e independente que permite uma interface com outras especificações. Como se pode constatar na figura seguinte, as tecnologias ASN.1, IDL e XML bem como C/C++ e Java foram já implementadas pela Testing Tecnologies para integrar com a Core Notation TTCN-3.

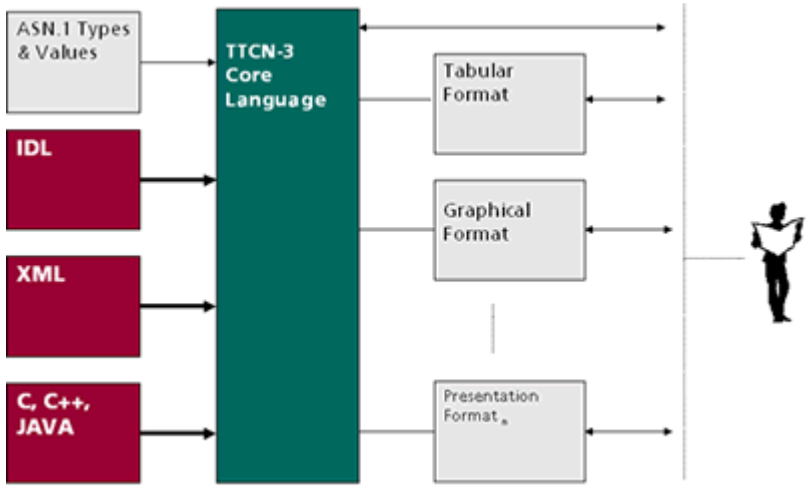


Figura 9 – Tecnologias integradas em TTCN-3

- **Especificação em TTCN-3**

De um modo geral, TTCN-3 estimula um sistema a testar (SUT), compara a resposta observada com a resposta esperada e declara veredictos. Os elementos principais a serem especificados em TTCN-3 são os seguintes:

- **Módulos:** Elemento onde se específica um componente de teste TTCN-3. Os módulos encontram-se estruturados de maneira a inicialmente apresentar uma parte de definições (variáveis) e em seguida uma parte de controlo (casos de teste). Podem importar definições de outros módulos, se necessário.
- **Templates:** Elemento integrante dos módulos onde os dados para teste são descritos.
- **Casos de Teste:** Elemento integrante dos módulos, executados na sua parte de controlo onde são enviados estímulos, através de uma porta, para um sistema a testar. São também estabelecidos veredictos conforme a resposta recebida. A figura 10 demonstra exactamente essa troca de mensagens, num caso de teste.

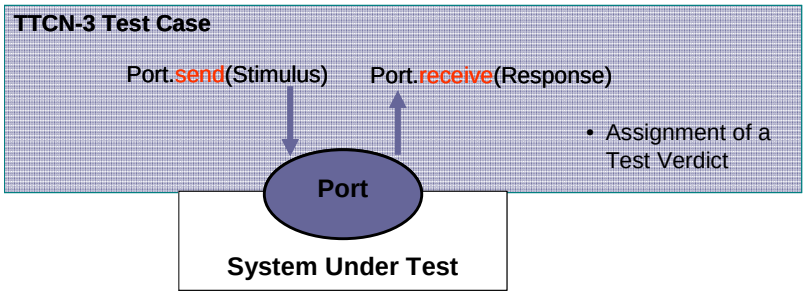


Figura 10 – Troca de mensagens num caso de teste

- **Configurações de Teste:** Conjunto de componentes de teste interligadas, apresentando uma comunicação bem definida entre portas e uma interface que define as fronteiras do sistema de teste. Os componentes de teste podem ser do tipo Main Test Component (MTC), Parallel Test Component (PTC) ou System. Em cada configuração de teste existe apenas um MTC que é criado automáticamente no início de cada execução do caso de teste. Durante a execução do caso de teste, outros componentes podem ser criados dinâmicamente. Esses componentes de teste são os PTCs.

  Apresentado os diferentes tipos de componentes existentes em TTCN-3, abordaremos agora a arquitectura de um sistema de teste, utilizando um componente MTC, na figura 11 e utilizando um componente MTC e dois componentes PTC, na figura 12.

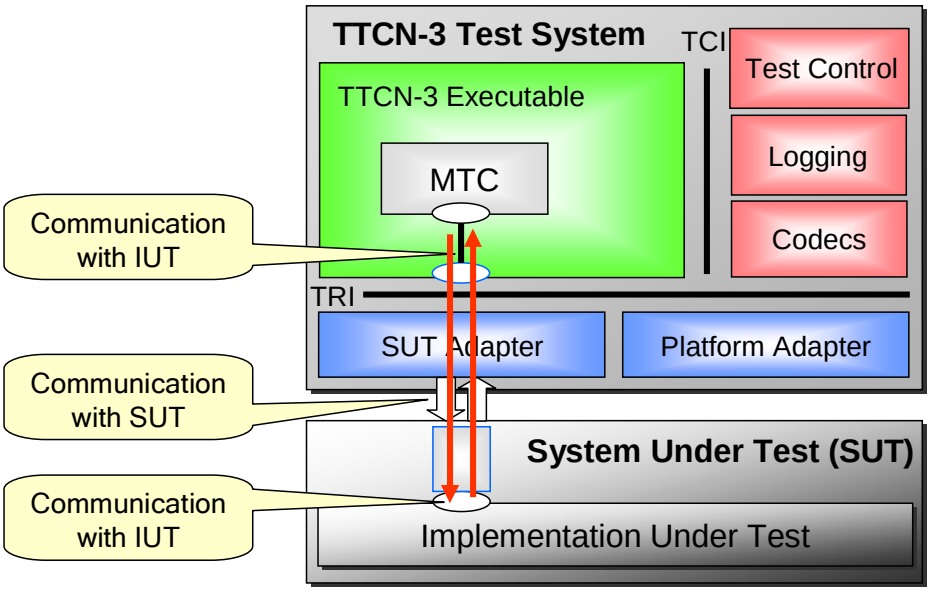


Figura 11 – Arquitectura sistema de teste com componente MTC

Como se pode constatar na figura anterior, esta arquitectura apenas utiliza um componente MTC, que comunica com o IUT (Implementation Under Test). Já em relação à figura 12, podemos constatar que o MTC comunica internamente com cada PTC, que por sua vez comunicam com o IUT. Resta apenas referir que a comunicação entre os componentes e entre os componentes e a interface do sistema de teste é efectuado através de portas, utilizando um modelo de fila de espera First in, first out (FIFO).

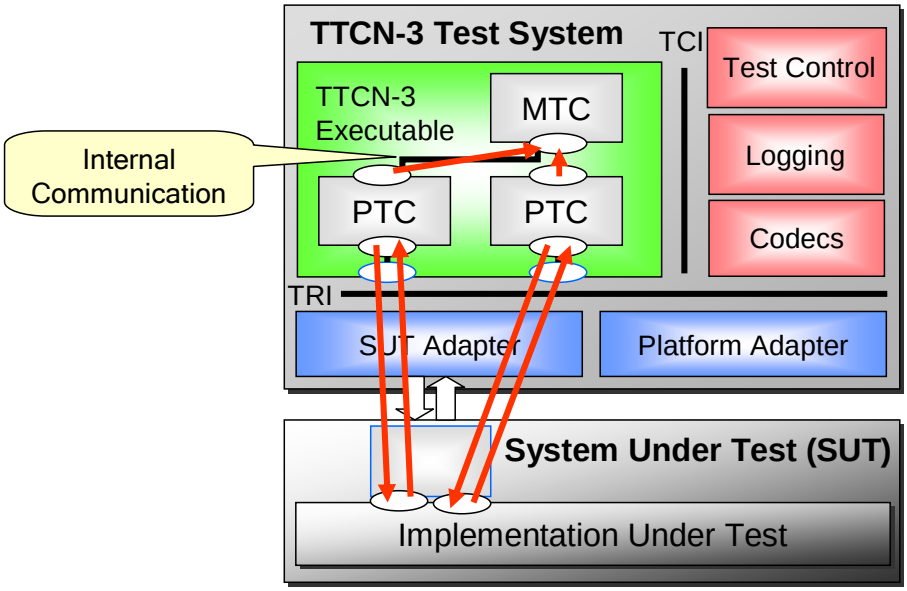


Figura 12 – Arquitectura sistema de teste com componentes MTC e PTC

- **Veredictos de Teste:** Elemento integrante dos módulos que definem o resultado do teste. São retornados pelo caso de teste após a sua execução e podem ser do tipo pass, fail, inconc, none ou error.

2 Fase: Compilação

Depois de específicar um componente de teste em TTCN-3, tem de ser compilado de modo a criar um executável. Normalmente o compilador de TTCN-3 compila de TTCN-3 para Java, C ou C++. Desta forma o ficheiro compilado encontra-se preparado para ser executado sobre o sistema a testar.

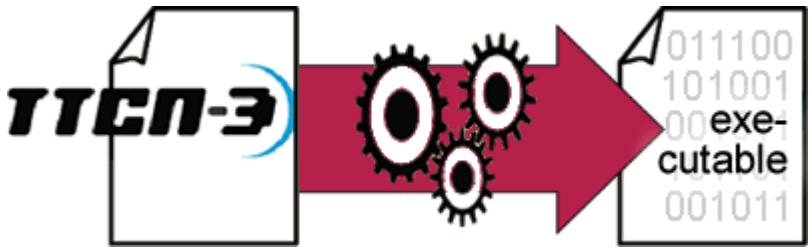


Figura 13 – Processo de Compilação

3 Fase: Implementação

Após se concluir a fase de especificação e compilação de um componente de teste em TTCN-3, é necessário efectuar a sua implementação num sistema existente. Normalmente os componentes de teste são efectuados em dispositivos de teste, PC´s industriais ou pequenos PC´s. Em TTCN-3 as interfaces para controlo e adaptação dos testes para o sistema a testar encontram-se definidos e normalizados.

Esta normalização consiste em duas diferentes partes: uma parte, em que se descreve a estrutura da implementação de um sistema de teste TTCN-3 e a segunda parte para apresentar as especificações das interfaces de controlo TTCN-3.

Como se pode constatar na figura 14, a implementação de um sistema de teste TTCN-3 tem de ter em conta as seguintes entidades:

- **TTCN-3 Executavel (TE):** Entidade responsável pela interpretação ou execução de um módulo TTCN-3.
- **Test Management (TM):** Entidade responsável pela administração do sistema de teste. Responsável pela invocação correcta dos módulos TTCN-3.
- **Coding and Decoding (CD):** Entidade responsável pela codificação e descodificação de dados TTCN-3 de modo a ser compreendido pelo sistema a testar (SUT).
- **Component Handling (CH):** Entidade responsável pela implementação da comunicação entre entidades de sistemas de teste distribuídos e vários componentes de teste

implementados pelo TE. Adapta mensagens ou procedimentos de componentes de teste para uma particular plataforma de execução de um sistema de teste.

- **SUT Adapter (SA):** Entidade responsável pela adaptação de mensagens e procedimentos de comunicação do sistema de teste do SUT para uma particular plataforma. Propaga pedidos de envio e operações sobre o SUT desde o TE até ao SUT notificando o TE sobre a recepção ou não desses pedidos e operações por parte do SUT.
- **Platform Adapter (PA):** Entidade responsável pela implementação de funções externas. Proporciona ao sistema de teste a opção de utilização da funcionalidade de contabilização do tempo de execução do teste.

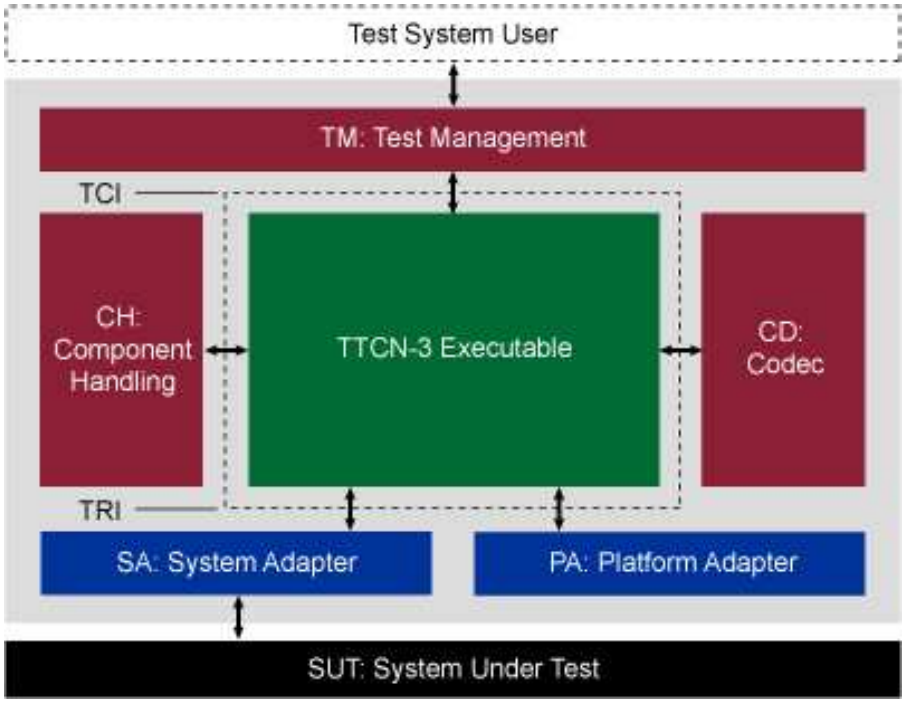


Figura 14 – Arquitectura TTCN-3

Após se ter identificado as entidades presentes numa arquitectura TTCN-3, abordaremos as interacções existentes a nível da TTCN-3 Control Interface (TCI) [22]:

As interfaces são definidas em termos de operação, implementadas como parte de uma entidade e chamadas por outras entidades do sistema de teste. Por cada operação, a especificação de interface define uma estrutura de dados associados, os efeitos no sistema de teste e limitações sobre a operação efectuada. O TCI define as interacções entre as entidades TE e TM, TE e CD, TE e CH. As interacções entre TE e SA e TE e PA estão associadas ao TTCN-3 Runtime Interface (TRI).

A nível do TCI existem as seguintes interfaces:

- **TCI Test Management Interface (TCI-TM):** Esta interface inclui todas as operações necessárias para administrar a execução de componentes de teste.
- **TCI Component Handling Interface (TCI-CH):** Esta interface consiste num conjunto de operações que são necessárias para implementar as comunicações entre os componentes TTCN-3 num sistema de teste centralizado ou distribuído. Inclui operações para criar, iniciar e parar componentes de teste, estabelece a conexão entre componentes de teste TTCN-3 e administra componentes de teste e seus veredictos.
- **TCI Coding/Decoding Interface (TCI-CD):** Esta interface inclui todas as operações necessárias para recuperar ou aceder aos codecs, utilizando o codificador TTCN-3, responsável pelo envio de dados codificados e o descodificador TTCN-3 responsável pela recepção dos dados descodificados.

Por sua vez, a nível do TTCN-3 Runtime Interface (TRI) existem as interfaces triComunnication e triPlatform que corresponde a comunicação com o SA e com o PA, respectivamente:

- **triCommunication:** Esta interface inclui todas as operações necessárias para implementar a comunicação entre um componente de teste TTCN-3 e o sistema a testar. Inclui operações para inicializar o Test System Interface (TSI) e estabelece conexão com o sistema a testar. Para além disso, inclui uma operação que é responsável pelo reset do System Adapter (SA).
- **triPlatform:** Esta interface inclui todas as operações necessárias para adaptar o executável TTCN-3 a um particular plataforma de execução. Inclui funcionalidades de start, stop, manipular temporizadores como consultar o seu estado e adicionar eventos de timeouts. Inclui também operações para chamar funções externas TTCN-3 e para o reset do Platform Adapter (PA).

Arquitectura do sistema e adaptação a um exemplo real

Como se pode constatar, a partir da figura seguinte e a partir do que já foi enunciado anteriormente, a arquitectura do sistema apresenta o seguinte formato.

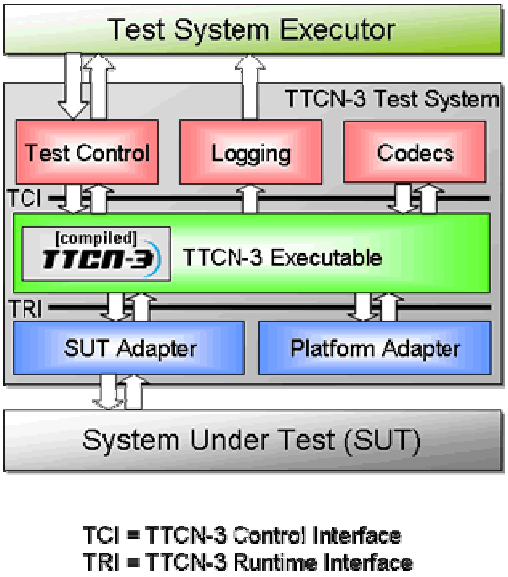


Figura 15 – Arquitectura de um sistema de teste

A construção de um sistema de teste TTCN-3 requer:

- Teste implementado em TTCN-3.
- Ferramenta TTCN-3, ou seja, um compilador TTCN-3 (ou interpretador).
- Implementações opcionais de controladores de teste, logging e codecs.
- Um adaptador do sistema a testar (SUT).
- Um adaptador de plataforma.

Para ilustrar a forma como será a arquitectura do sistema num caso de utilização específico, será apresentado um caso de teste IPv6.

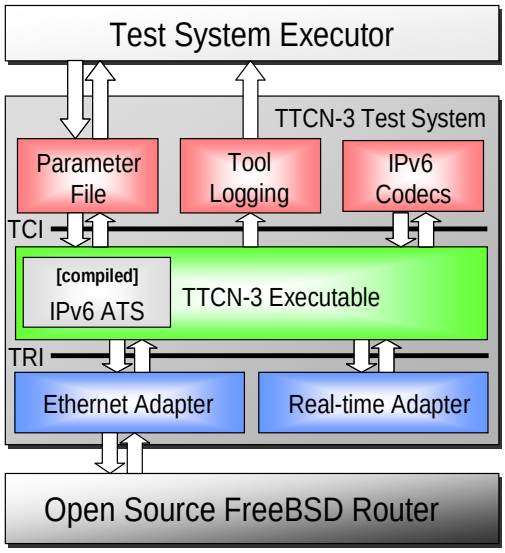


Figura 16 – Arquitectura de um sistema de teste IPv6

Como podemos constatar a partir da figura anterior, estamos perante um caso de teste IPv6 cujo sistema a testar é um FreeBSD Router. Assim sendo, cada entidade representada nesta figura apresenta especificações para este caso de utilização, desde os Codecs específicos IPv6 representando a entidade Codecs ao Adaptador Ethernet representando o SUT Adapter passando por todas as outras entidades (Parameter File representando o Test Control, Tool Logging representando o Logging e Real Time Adapter representando a Platform Adapter).

Tendo em conta este caso de utilização e as abordagens efectuadas ao longo deste capítulo, podemos concluir que para cada caso de utilização, apenas as entidades Test Control, Logging e Codecs apresentam a opção de não se encontrarem representadas, se o respectivo componente de teste assim não o exigir, sendo que as outras entidades têm de estar obrigatoriamente representadas e adaptadas. Só assim podemos afirmar que o componente de teste se encontra bem implementado.

Benefícios da tecnologia TTCN-3

- A linguagem é especialmente utilizada para testes.
- Apresenta um suporte de testes sistemáticos e automatizados.
- Implementável via interfaces normalizadas (TRI&TCI) nos sistemas existentes.
- Rápido acesso ao já normalizado conjunto de testes TTCN-3 (B-ISDN, UMTS, SIP, H.248).
- A síntaxe e operações semânticas de TTCN-3 são normalmente compreendidas e não dizem respeito a uma linguagem de programação particular.

- Os testes em TTCN-3 concentram-se no propósito dos testes e abstraem-se dos detalhes particulares do resto do sistema.
- A linguagem sofre uma constante manutenção.
- A manutenção dos testes é simples.
- Existência de editores livre de TTCN-3 no mercado.

3.2 TTworkbench

Apresenta-se numa fase inicial desta secção a empresa Testing Technologies [23], responsável pelo desenvolvimento da ferramenta de testes TTworkbench [24]. Com uma elevado nível de experiência em testes automatizados, Testing Technologies desenvolve e comercializa ferramentas de teste inovadoras no mercado, tendo em conta o custo optimizado e a elevada qualidade da tecnologia.

Como líder do mercado em tecnologias orientada a clientes, Testing Tecnologies dedica-se apenas à tecnologia TTCN-3. Esta poderosa linguagem de teste garante uma elevada flexibilidade na concepção e na manutenção de testes de software a um nível base.

Testing Technologies foi fundado a partir do Instituto Fraunhofer em 2000. Apresenta uma equipa de peritos que trabalha no desenvolvimento de novos produtos juntamente com o apoio a clientes e formação. Ao mesmo tempo, a divisão de investigação assegura uma rápida adaptação dos produtos tendo em vista as necessidades competitivas do mercado actual.

Testing Technologies coopera com uma forte componente de parceiros globais e locais, de modo a responder mais eficientemente às necessidades dos clientes. Até à data, apresentam 16 sociedades com companhias por toda a Europa (Austria, Estonia, Finland, France, Germany, Israel, UK), Usa (New Hampshire, Delaware) e Ásia (China, India, Korea, Japan, Taiwan), espalhando as ideias e princípios básicos sobre a tecnologia TTCN-3 um bocado por todo o mundo.

Sendo a Testing Technologies uma empresa que desenvolve novos produtos, podemos constatar, a partir da Tabela 1, alguns dos produtos desenvolvidos:

TTworkbench	Software de desenvolvimento de componentes de teste
TTplugin Concept	Adaptador de componentes de teste
TTsuite Series	Conjunto de aplicações dos componentes de teste
TTtwo2three	Software para migração automática de TTCN-2 para TTCN-3

Tabela 1 – Produtos Desenvolvidos pela Testing Technologies

Uma vez apresentada a empresa Testing Technologies, apresenta-se agora a ferramenta de testes TTworkbench. O TTworkbench é um software baseado na tecnologia de teste normalizada TTCN-3 combinada com a IDE framework Eclipse, que inclui uma interface gráfica onde os componentes de teste são implementados e executados. Proporciona uma gama completa de características no que se refere a especificação do teste, execução e análise. Para além do texto, na especificação do teste, baseado em TTCN-3, TTworkbench também proporciona a opção de definição de ambiente gráfico para os testes.

Após esses componentes de teste terem sido definidos em TTCN-3, TTworkbench faz a respectiva compilação, criando um executável para cada um deles. O completo ambiente de administração e de execução dos testes permite aos utilizadores administrar, executar e analisar os seus componentes de teste.

Suporta também métodos de automatização de testes que irá reduzir significativamente os custos assegurando a óptima qualidade dos mesmos, do início até ao fim de um completo ciclo de teste.

Encontra-se disponível em diferentes formatos ou versões, cada uma delas com especificações próprias consoante o que o cliente pretende:

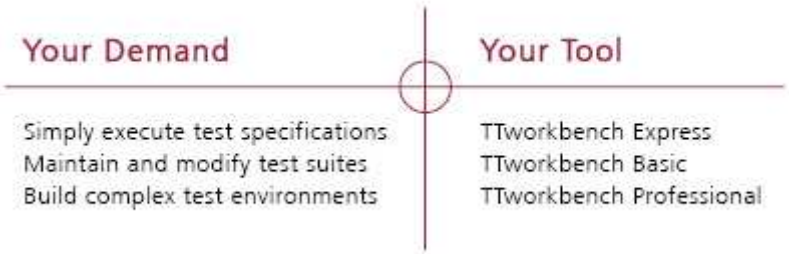


Figura 17 – Produtos TTworkbench

Todas as versões apresentam características para especificação, compilação e execução para casos de testes.

Para além disso, podem ser combinados com todas as versões TTworkbench, módulos de suporte de dados (plugins):

- **ASN.1 – Abstract Syntax Notation One**

ASN.1 é uma linguagem formal para descrever mensagens de uma forma abstracta. Apresenta um campo de aplicação envolvendo a Internet, redes inteligentes, telefones celulares, serviços multimédia, Voice Over IP. ASN.1 está associado a diversas regras de codificação normalizadas, como basic encoding rules (BER), ou mais recentemente packed encoding rules (PER).

- **IDL – Interface Descriptor Language**

IDL é uma linguagem utilizada para descrever interfaces de software. Escrita de uma maneira neutral, permite uma comunicação entre diferentes componentes de software, por exemplo, entre C++ e Java. A partir do plugin IDL proveniente do TTworkbench, o adaptador de teste e o codec encontram-se incorporados no produto. Apenas é necessário especificar o IDL e automaticamente é importado e mapeado a linguagem.

- **MOST FCat – Media Oriented Systems Transport Function Catalog**

MOST é um standard para multimédia e redes de informação, em indústrias automotores. A tecnologia foi desenvolvida para proporcionar uma eficiente e uma estrutura de custo efectivo para transmitir áudio, vídeo, dados e parâmetros entre dispositivos vizinhos. MOST Function Catalog define todas as funções e os seus parâmetros de um ou mais blocos em XML. Esses ficheiros XML são facilmente importados para a tecnologia TTCN-3 e durante essa operação de importação, o plugin proporciona a geração automática de um codec Fblock específico para a definição do tipo derivado.

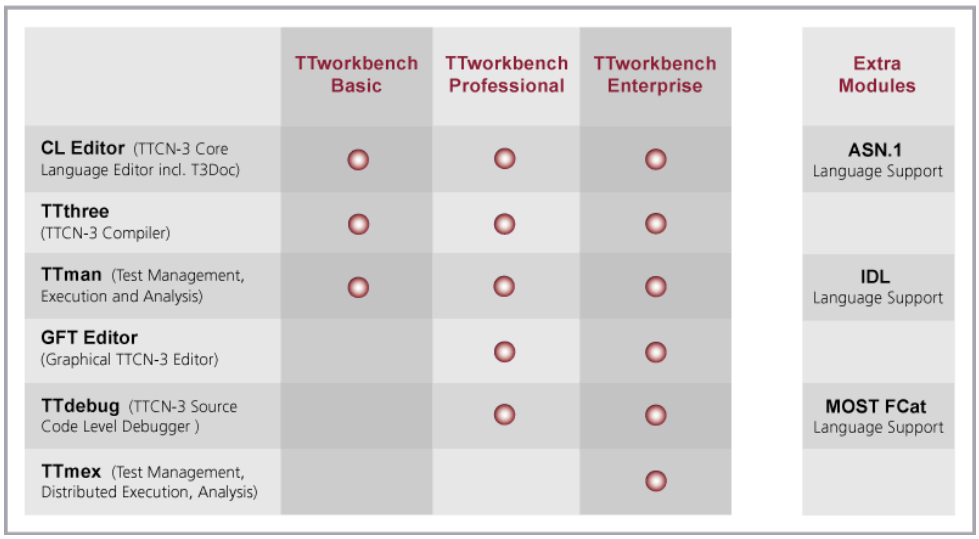


Figura 18 – Componentes e módulos integrantes das versões TTworkbench

A figura 18 apresenta os componentes contituíntes, para cada versão TTworkbench, bem como os seus módulos integrantes. Podemos constatar que o TTworkbench Enterprise é considerado a versão mais completa, uma vez que contém todos os componentes constituíntes das outras versões e ainda o componente TTmex, responsável pela administração, execução e análise do componente de teste.

O software TTworkbench pode ser utilizado no desenvolvimento de componentes de teste em diferentes sectores da indústria. Os sistemas candidatos a serem testados podem ser, tanto em hardware como em software. Apresenta-se de seguida os campos de aplicação deste software:

- **Comunicação:**
  - Comunicações Móveis (3G, IMS, WiMAX)
  - Middleware (CORBA, CCM, EJB, Webservices
  - Voz sobre IP (SIP, H.248)
  - Tecnologia de Internet (IPv6, SCTP, M3UA)
- **Transporte/Automotor:**
  - MOST, CAN, Powertrain
  - Controlador de Sistemas, Sistemas de Segurança
  - Sistemas de Computação
- **Militar e Defesa:**
  - Detecção de Sistemas

- Sistemas de Controlo
- Sistemas de Satélites
- Sistemas de Navegação

Após ter sido abordado neste ponto o software TTworkbench, nomeadamente a nível de versões de software existentes, integração de plugins e campos de aplicação, resta apenas enunciar as vantagens de utilização este software, em diferentes níveis:

- **Qualidade de Optimização:**
  - Analisador de testes reduz o erro em casos de teste.
  - Permite testes em estados primários do processo de testes devido à rápida especificação e execução.
  - Elevada reusibilidade e fácil extensão dos já definidos testes.
- **Produtividade:**
  - Apresenta uma ferramenta única para especificação, execução e análise de sistemas de teste.
  - Elevada transparência através da definição de testes textuais/ou gráficos.
- **Elevada Flexibilidade:**
  - Integração rápida dos sistemas de teste devido à fácil escrita em adaptadores de teste.
  - Tecnologia independente do design do sistema de teste utilizando a linguagem genérica de teste normalizada TTCN-3.

3.2.1 TTworkbench Basic

TTworkbench Basic é um software de desenvolvimento de componentes de teste baseado em especificações TTCN-3, compilação, execução de casos de teste e análise. Apresenta 3 diferentes componentes:

- CL Editor: Editor TTCN-3.
- TTthree: Compilador TTCN-3.
- TTman: Componente responsável pela manutenção, execução e análise de testes.

Abordaremos em seguida cada componente do TTworkbench Basic, apresentando as suas funcionalidades e características:

CL Editor: Editor TTCN-3

- Suporta a completa tecnologia TTCN-3.
- Permite a validação dos testes especificados.
- Suporta auto-edição.
- A exploração de TTCN-3 é vista como uma perspectiva de desenvolvimento.
- Apresenta um esboço da página segundo uma estrutura especificada.
- Apresenta uma tecnologia TTCN-3 específica.
- Suporta uma rápida solução de problemas.
- Exporta documento HTML T3Doc.

Na figura 19 é apresentada a interface CL Editor, associado a um caso de utilização. É constituída pelo TTCN-3 CL Editor, representado na janela mais ao centro, onde é editado o ficheiro TTCN-3. O Project View, representado na janela mais à esquerda, apresenta todos os ficheiros constituíntes do projecto. O Outline, representado na janela mais á direita, apresenta todos os componentes editados no CL Editor enquanto que nos Problems, representado na janela mais em baixo, é apresentado todos os erros e avisos, após a compilação do ficheiro TTCN-3. É de salientar a existência de três opções, nomeadamente o validador, compilador e recompilador, existentes nos botões de atalho, responsáveis pela correcta compilação de um componente de teste TTCN-3.

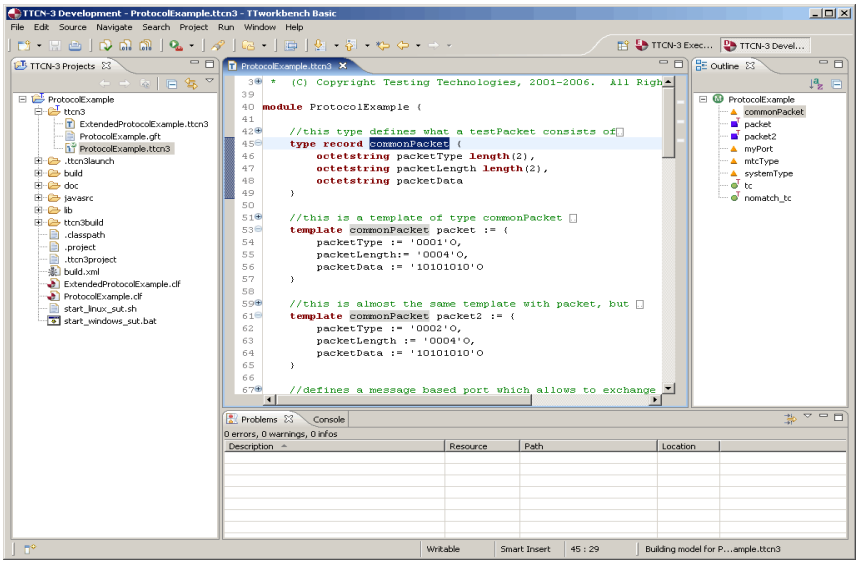


Figura 19 – Interface CL Editor

TTthree: Para compilar módulos TTCN-3 em testes executáveis:

- Apresenta diferentes plataformas de execução e compilação.
- Suporta a completa tecnologia TTCN-3.
- Apresenta uma compilação proveniente do CL Editor.
- Apresenta uma consola TTCN-3.
- Apresenta uma flexível adaptação para dispositivos de teste cuja linguagem normalizada é TTCN-3 Runtime Interface (TRI).
- Apresenta uma integração fácil de codecs externos via TTCN-3 Control Interfaces (TCI).
- Opção de linha de comandos.

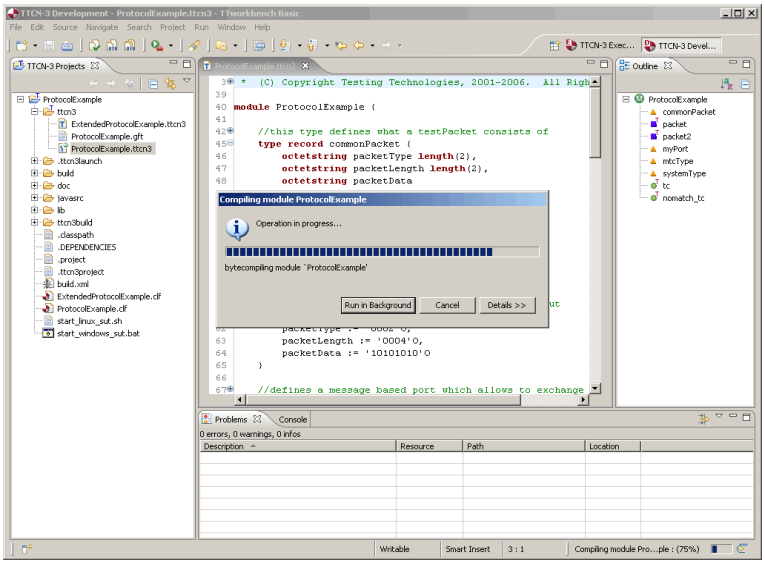


Figura 20 – Interface TTthree

TTman: Para administrar, executar e analisar módulos TTCN-3 compilados:

- Carrega e grava todos os resultados dos testes (registos, veredictos de teste).
- Apresenta um registo Textual e Gráfico com diferentes níveis.
- Filtragem de Registos.
- Apresenta uma componente de filtros utilizada para os registos.
- Apresenta registos Online e Offline.
- Visualiza as funções TTCN-3 da fonte.
- Selecção e Configuração do adaptador de teste e de plugins em tempo real (portas, codecs, funções externas).
- Apresenta a opção de Carregar/Salvar módulos.
- Geração de ficheiros reportando os testes (HTML, PDF, EXCEL).
- Opção de linha de comandos.

Na figura 21 é apresentada a interface TTman, associado a um caso de utilização. É constituída pelo Graphical Online Logging, representado na janela inferior da direita, responsável pela representação do formato gráfico do componente de teste. No Management View, representado na janela superior da esquerda podemos visualizar os casos de teste implementados naquele componente de teste TTCN-3 específico enquanto que no Test Parametrization, representado na janela inferior da esquerda, podemos visualizar os paramêtros existentes naquele componente de teste. Por fim, a janela superior da direita diz respeito ao Result Analyser, onde podemos visualizar a troca de informação entre as aplicações, passo a passo.

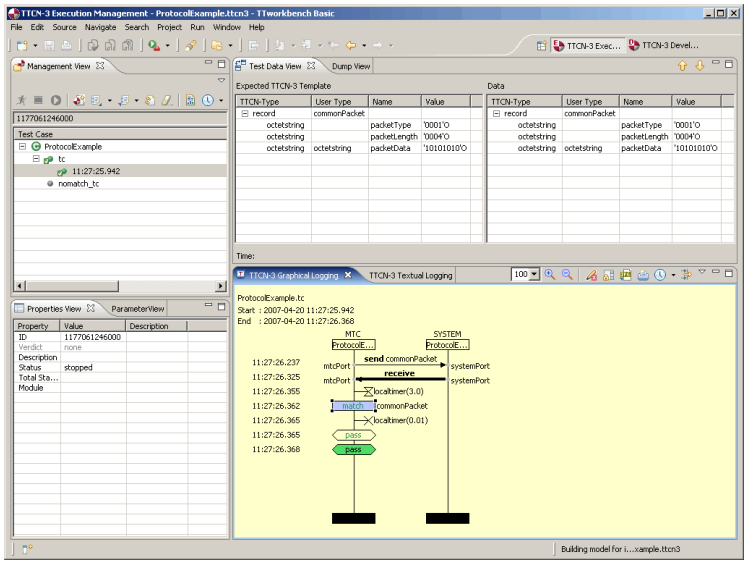


Figura 21 – Interface TTman

Após ser feito um levantamento das características e funcionalidades de cada componente do TTworkbench Basic, enunciaremos as vantagens de utilização do mesmo. No fundo, estas vantagens de utilização são comuns para as duas versões do TTWorkbench analisadas, a Basic e a Professional, independentemente das diferenças existentes de versão para versão a nível de funcionalidades disponíveis. A versão Professional é um software mais orientado à implementação de complexos componentes de teste, comparativamente à versão Basic. Assim sendo, as vantagens de utilização destas duas versões são as seguintes:

- Apresentam uma fácil e rápida definição de componentes de teste em formato textual.
- Apresentam um conceito de desenvolvimento de componentes de teste num patamar reduzido de tempo e custos.
- Apresentam uma tecnologia independente e testes de sistema programáveis em TTCN-3.
- Apresentam uma elevada reusabilidade e fácil execução de componentes de teste predefinidos.
- Combináveis com diferentes plugins.

NOTA: Inicialmente esta versão do TTworkbench (Basic) foi a adoptada para implementar os componentes de teste sobre o servidor. No entanto, e uma vez que a versão que estávamos a utilizar era uma versão de avaliação, apenas se encontrava disponível para utilização durante 14 dias. Se porventura fosse pedida outra licença, o prazo alargar-se-ia durante mais 14 dias e assim sucessivamente. Este facto revelou-se como um dos factores porque decidimos adoptar outra versão do TTworkbench, uma vez que de 14 em 14 dias, estávamos dependentes de enviar um novo pedido de licença, isto se pretendessemos efectuar um trabalho contínuo na implementação dos componentes de teste. Para além deste importante factor, o factor preponderante porque decidimos adoptar pela versão Professional está associada ao facto das funcionalidades da versão Basic se encontrarem limitadas, uma vez que não se encontrava disponíveis as funcionalidades de validação e compilação de componentes de teste em TTCN-3, funcionalidades essas essenciais para implementação de um simples componente de teste.

3.2.2 TTworkbench Professional

TTworkbench Professional é um software de desenvolvimento de testes baseado em especificações TTCN-3, compilação, execução de casos de teste e análise. Relativamente ao TTworkbench Basic, apresenta uma grande diferença nas especificações TTCN-3, pois apresenta um editor gráfico de TTCN-3, especificação essa que o TTworkbench Basic não apresenta.

TTworkbench Professional é constituído por:

- Todas os componentes do TTworkbench Basic (CL Editor: Editor TTCN-3, TTthree: Compilador TTCN-3, TTman: Manutenção, execução e análise de testes)
- GFT Editor: Editor Gráfico de TTCN-3.
- TTdebug: Debugger de Código fonte TTCN-3.

Abordaremos em seguida apenas as componentes GFT Editor e TTdebug do TTworkbench Professional, uma vez que as outras componentes constituíntes já foram abordadas no ponto anterior, apresentando as suas funcionalidades e características:

GFT Editor: Para especificações gráficas de teste e documentação:

- Apresenta um design gráfico e permite uma visualização de casos de teste como uma sequência de diagramas.
- Apresenta vários editores de frames que podem ser abertos para permitir um trabalho simultanêo.
- É possível gerar código TTCN-3 para cada parte das definições gráficas.
- Apresenta uma geração automática de GFT (gráficos) com ausência de código TTCN-3.

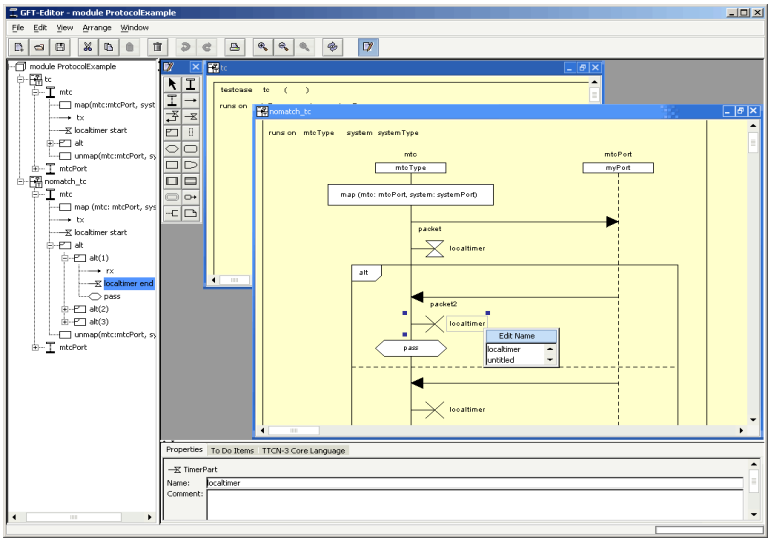


Figura 22 – Interface GFT Editor

TTdebug: Para debugging do código fonte TTCN-3:

- Apresenta um debugger de TTCN-3 e Java.
- Apresenta a opção manual de suspender/continuar um conjunto de testes.
- Apresenta um conjunto de funções “step into”, “step over” e “step return” responsáveis pela opção de executar um componente de teste passo a passo.
- Apresenta a opção de visualizar o estado de múltiplos componentes.
- Apresenta a opção de visualizar o estado dos timers e verifica se o tempo definido para cada operação, se for o caso, já terminou.

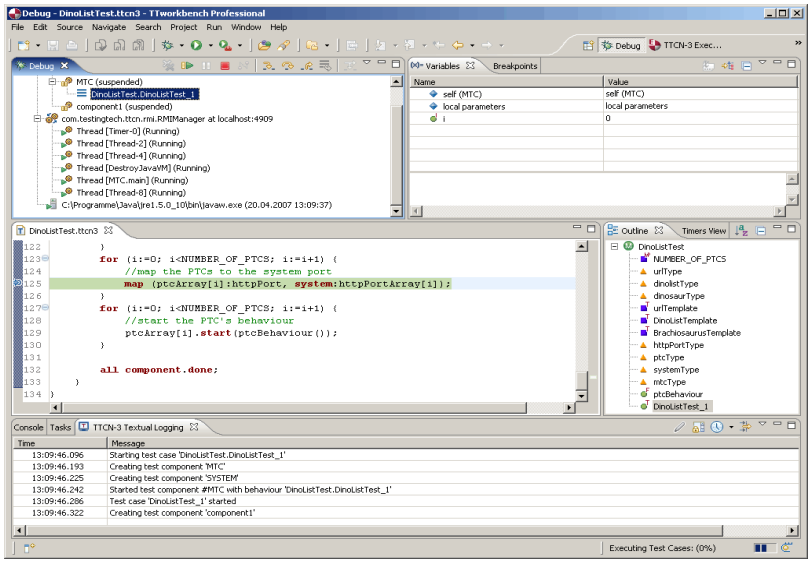


Figura 23 – Interface TTdebug

Capítulo 4

4. Metodologia de Teste

Neste capítulo apresenta-se a arquitectura adoptada bem como alguns dos componentes de teste realizados. Procurou-se nesta fase optar por implementar um conjunto de componentes de teste que valide o funcionamento dos elementos desenvolvidos no projecto “Serviço de VoD (Video on Demand) usando DRM (Digital Rights Management)”, que concilie a simplicidade com a eficiência, sem nunca esquecer a sua interactividade. Esses componentes de teste foram implementados na aplicação denominada ***TTworkbench Professional***, apresentada na secção 3.2.

Uma vez que o projecto anterior estava principalmente associado ao conceito de DRM, propôs-se para este trabalho a definição de uma arquitectura e de um conjunto de componentes de teste que permita uma avaliação das funcionalidades de DRM.

Uma vez analisado esse projecto, chegou-se à conclusão que era possível apenas implementar um componente de teste a nível de funcionalidades de DRM, que seria a validação de um ficheiro XML, resultante da criação de uma licença, abordada na secção 4 deste capítulo.

Os componentes de teste abordados no ponto 2 e 3 deste capítulo, basicamente eram testes que pretendem demonstrar a viabilidade de comunicação com um sistema real, podendo ter utilidade num projecto futuro.

Assim, nas secções seguintes apresenta-se a arquitectura definida para a realização desses componentes de teste bem como alguns desses componentes de teste.

4.1 Arquitectura de Teste

Com base no estudo efectuado no projecto anterior, chegou-se a conclusão que os componentes de teste desenvolvidos estão associados apenas a arquitectura do servidor, uma vez que o repositório de licenças se encontra no servidor, e como tal, o principal objectivo deste projecto é implementar componentes de teste que permita avaliar funcionalidades de DRM. A arquitectura do cliente, para este projecto, não teria nenhuma utilidade prática, uma vez que não será efectuado nenhum componente de teste sobre a aplicação cliente.

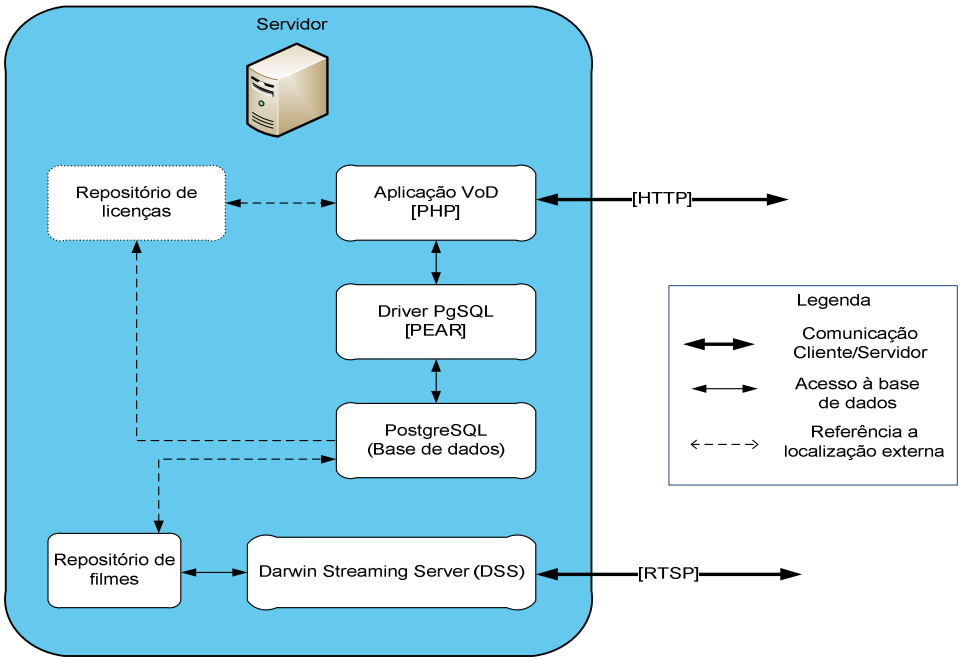


Figura 24 – Arquitectura do Servidor

Após a análise da arquitectura do servidor, e atendendo aos requisitos deste projecto, chegou-se a conclusão que os testes implementados estariam associados concretamente a nível da aplicação VoD e da base de dados. Para isso seria necessário encontrar uma solução tendo em vista a comunicação entre a aplicação adoptada e essas aplicações, implementando em seguida um conjunto de componentes de teste.

Assim sendo, a comunicação entre o software adoptado e a aplicação VoD é efectuada via HTTP, ou seja, o utilizador na implementação dos testes, especifica o Uniform Resource Locator (URL) da aplicação, composto pelo protocolo, host e ficheiro.

Já a comunicação entre o software adoptado e a base de dados é efectuada através de queries desenvolvidas em Java.

No contexto da arquitectura do servidor podemos definir vários módulos importantes para a criação de um sistema seguro e fiável. Como é possível ver na Figura 25, esses módulos são a aplicação TTworkbench Professional, a aplicação VoD, o driver PgSQL e a Base de Dados.

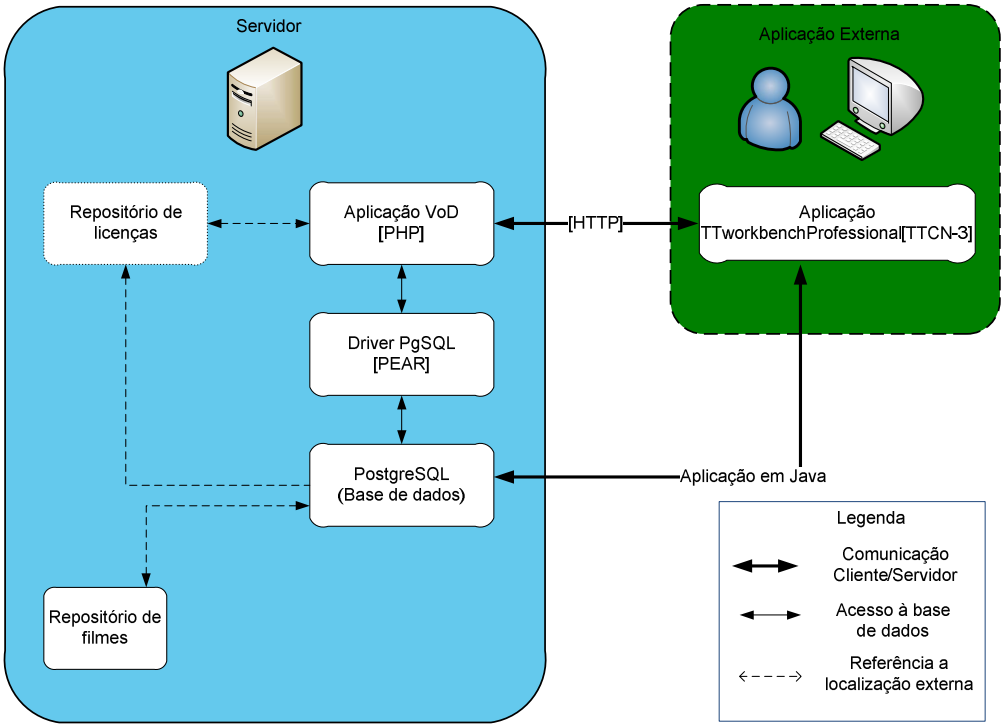


Figura 25 – Arquitectura do Servidor para Teste

Comparando a arquitectura do servidor, na figura 24, com a arquitectura do servidor para teste, na figura 25, podemos constatar que:

- A base de dados, que na arquitectura do servidor do projecto "Serviço de VoD (Video on Demand) usando DRM (Digital Rights Management)" comunicava com a aplicação web, desenvolvida em PHP, comunica agora também com uma aplicação TTworkbench Professional, cuja linguagem de programação é o TTCN-3, tecnologia essa abordada no capítulo 3.
- A ausência do servidor de streaming de código livre, pertencendo ao projecto Open Source Streaming Server da Apple (Darwin Streaming Server) na arquitectura do servidor para teste, uma vez que os componentes de teste implementados estão associados apenas a nível da aplicação VoD e da base de dados e não a nível de servidores de streaming.

Tendo em vista a implementação dos testes e tendo em conta os requisitos que essa implementação exigia, foram desenvolvidos, para cada teste, um ficheiro com extensão ttcn-3, associado às operações do Abstract Testsuite (ATS), responsável pela definição das sequências de mensagens que vão ser trocadas entre a aplicação e o sistema a testar. Para além desse ficheiro, foram também desenvolvidos numa linguagem de mais baixo nível, ficheiros com extensão Java, responsáveis pela contrução dessas mensagens e comunicação com o sistema a testar. Esses ficheiros estão inseridos nas operações do Test Adapter

(TA) e dos Codecs. Nos anexos encontra-se o código fonte dos componentes de teste definidos, não só os ficheiros com extensão TTCN-3 mas também os ficheiros com extensão Java. Apenas o componente de teste relativo à funcionalidade da base de dados não se encontra em anexo, uma vez que será apresentada na secção 3 deste capítulo.

4.2 Teste de ligação entre a Aplicação e a Aplicação VoD e troca de mensagens

O primeiro componente de teste desenvolvido está associado a um teste de comunicação entre a nossa aplicação e a aplicação VoD. Pretende-se com este teste demonstrar que é possível comunicar, através da nossa aplicação, com um sistema real. Para isso serão estabelecidos pedidos entre as duas aplicações e serão estabelecidos veredictos conforme a nossa aplicação receba ou não as respostas esperadas aos pedidos.

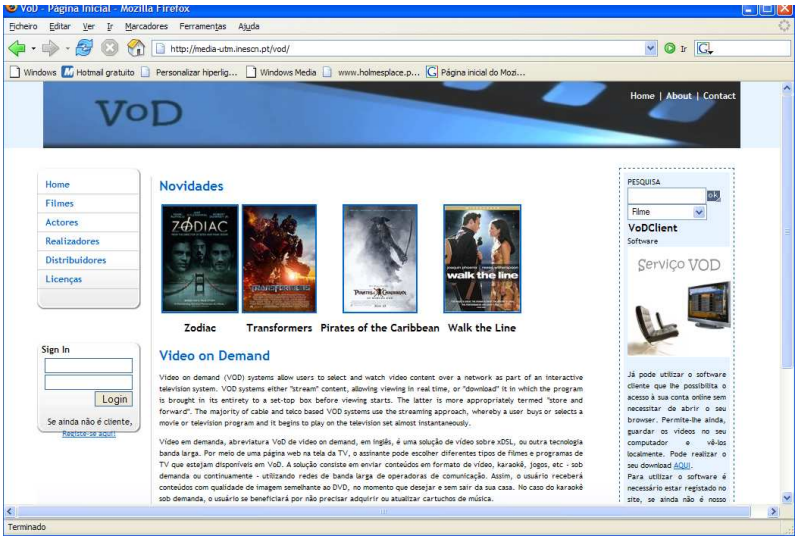


Figura 26 – Aparênica Gráfica da aplicação VoD

4.2.1 Ligação com a Aplicação VoD

Como podem constatar na Figura 27, este componente de teste está associado a um simples teste de ligação com a aplicação VoD. Para isso, é necessário especificar o URL da aplicação VoD no ficheiro com extensão ttcn-3, composto pelo protocolo, host e ficheiro.

```
template urlType urlTemplate := {
                protocol    := "http://",
                host        := "media-utm.inescn.pt",
                file        := "/vod/listaFilmes.php
                }
```

Após o pedido de ligação, a aplicação VoD envia uma mensagem de resposta. Se porventura a mensagem de resposta for proveniente do URL especificado inicialmente, surge o

veredicto “match”, que compara o URL recebido com o URL especificado inicialmente. A mensagem “A aplicação VoD encontra-se conectada”, bem como outras mensagens existentes ao longo deste e de outras componentes de teste, apenas apresentam um carácter informativo, não tendo nenhuma influência na execução do componente de teste.

O veredicto “pass” indica que a conexão foi efectuada com sucesso e iniciar-se-á a comunicação entre as duas aplicações.

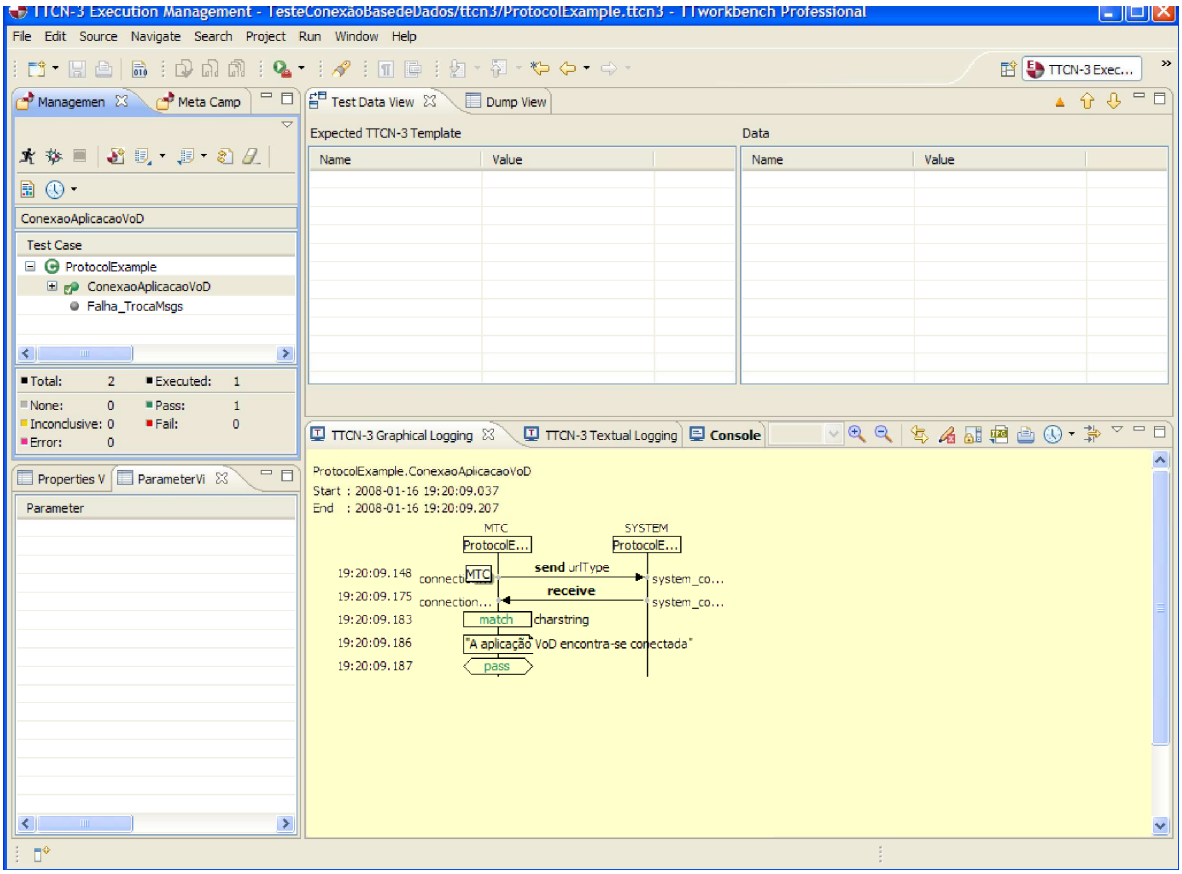


Figura 27 – Interface do Teste de Conexão com a Aplicação VoD

4.2.2 Troca de mensagens com a Aplicação VoD

O componente de teste seguinte, no fundo, é uma continuação do anterior e permite comparar, após ser feito um pedido à aplicação VoD, a resposta da mesma com a resposta esperada e estabelecer veredictos conforme essa resposta. Como se pode constatar na figura 28, inicia-se com a ligação à aplicação VoD. Após essa primeira fase de ligação, iniciar-se-à a comunicação entre as aplicações. Espera-se que a aplicação TTworkbench Professional receba uma resposta específica de listagem de filmes por parte da aplicação VoD, surgindo o veredicto “match” se a resposta recebida for igual à resposta esperada ou o veredicto “mismatch” se a resposta recebida for diferente da resposta esperada. Neste primeiro caso, a resposta enviada pela aplicação VoD foi a resposta esperada, ou seja, uma

listagem de filmes correcta, surgindo o veredicto “match” e em seguida a mensagem informativa “Troca de mensagens iniciada”. O veredicto “pass” indica que o componente de teste foi efectuado com sucesso. Após esta primeira fase de comunicação, é enviado o mesmo pedido de listagem de filmes, mas desta feita, a resposta enviada pela aplicação VoD não foi a resposta esperada, uma vez que esta tinha sido propositamente alterada, surgindo o veredicto “mismatch” e em seguida a mensagem informativa “Erro na troca de mensagens”. O veredicto “fail” indica que o componente de teste não foi efectuado com sucesso.

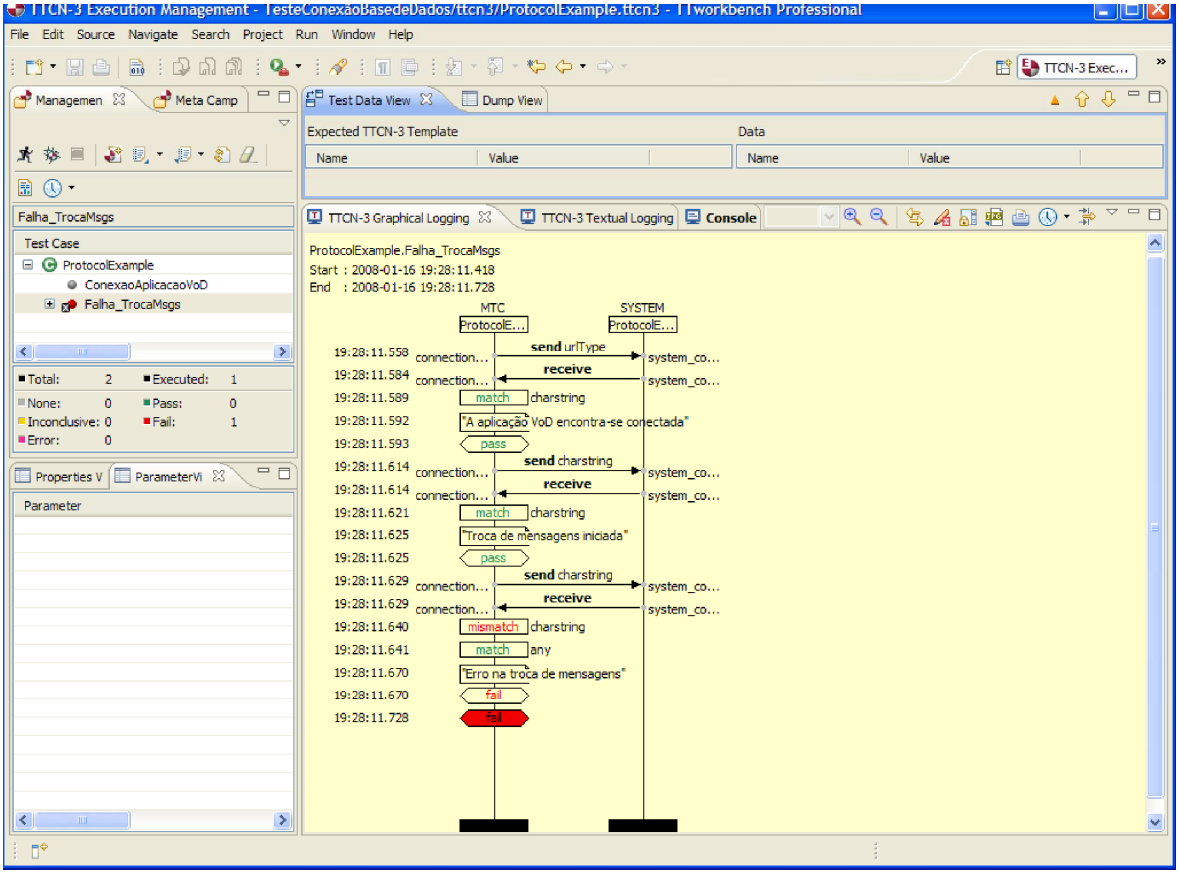


Figura 28 – Interface do Teste de troca de mensagens com a Aplicação VoD

4.3 Teste de Funcionalidade da Base de Dados

Este componente de teste desenvolvido tem como objectivo testar uma funcionalidade específica da Base de dados [25], nomeadamente a funcionalidade de listagem de filmes. Pretende-se com isto testar a comunicação com a base de dados. Para isso será enviado um pedido de listagem de filmes à base de dados. Se a resposta for a pretendida, significa que a comunicação entre as aplicações é efectuada com sucesso surgindo o veredicto “pass”, caso contrário, significa que a comunicação entre as partes do sistema não se encontra a comunicar satisfatóriamente surgindo o veredicto “fail”.

Como foi enunciado na secção anterior deste documento, a comunicação entre o TTwork-bench Professional e a aplicação VoD é efectuada via HTTP, ou seja, o utilizador na implementação dos testes, específica o URL da aplicação, composto pelo protocolo, host e ficheiro.

Já a comunicação entre o TTworkbench Professional e a Base de Dados é efectuada via queries, implementadas em Java, comunicando aí directamente. Pode ver-se esse código no excerto seguinte.

```
import java.sql.*;
public class SQLStatement {
public static void main(String args[]) {
String url = "jdbc:postgresql://media-utm.inescn.pt:5432/vod";
Connection con;
String query = "select title from movies.media order by title";
Statement stmt;
    try {
        Class.forName("org.postgresql.Driver");
        }
        catch(java.lang.ClassNotFoundException e) {
        System.err.print("ClassNotFoundException: ");
        System.err.println(e.getMessage());
        }
    try {
        con = DriverManager.getConnection(url,"vod", "drm");
        stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery(query);
        ResultSetMetaData rsmd = rs.getMetaData();
        int numberOfColumns = rsmd.getColumnCount();
        int rowCount = 1;
        System.out.println("Lista de Filmes: ");
        System.out.println("");
        while (rs.next()) {
        //System.out.println("Filme " + rowCount);
            for (int i = 1; i <= numberOfColumns; i++) {
                //System.out.print(" Title " + i + ": ");
                System.out.println(rs.getString(i));
                }
        rowCount++;
        }
stmt.close();
con.close();
} catch(SQLException ex) {
```

```
System.err.print("SQLException: ");
System.err.println(ex.getMessage());
}
}

}
```

O resultado da execução do código anterior é uma listagem de filmes, presentes na base de dados, listados por ordem alfabética, como se pode ver a seguir.

```
Lista de Filmes:

Amadeus
Barcelona
Crash
Fantastic Four
Harry Potter
K-Os Sunday Morning
O meu video
Pirates of the Caribbean
Queen at Wembley
Sweet Escape
Transformers
Volver
Walk the Line
Zodiac

BUILD SUCCESSFUL (total time: 1 second)
```

Como se pode constatar na figura seguinte, a comunicação entre o TTworkbench Profes-sional e a Base de Dados é efectuada com sucesso.

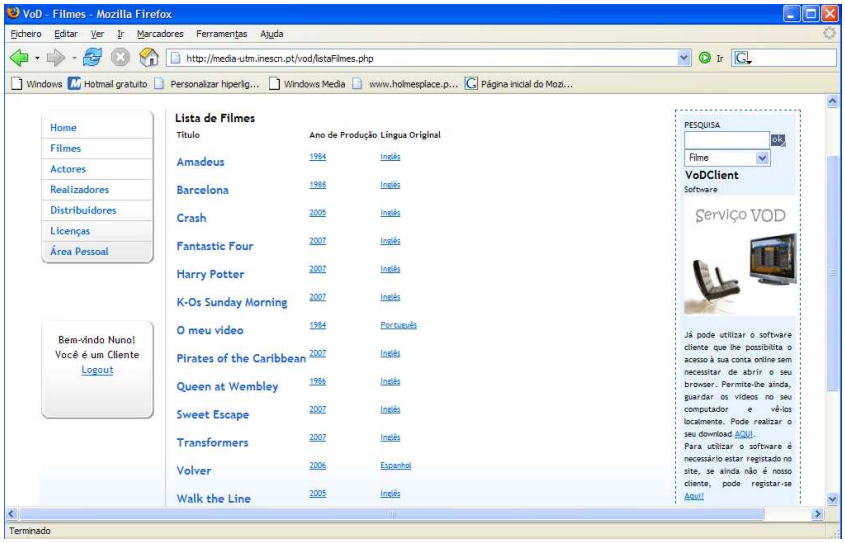


Figura 29 – Interface para a Listagem de Filmes

4.4 Teste de Validação do XML

O último componente de teste desenvolvido tem como objectivo testar uma funcionalidade específica do servidor, como a validação de um ficheiro XML, utilizando o TTworkbench Professional, resultante do preenchimento do seguinte formulário, como se pode constatar na figura 30. Este formulário está associado ao projecto "Serviço de VoD (Video on Demand) usando DRM (Digital Rights Management)" e tinha como objectivo criar uma licença para um filme, de um utilizador específico. Neste caso o utilizador específico é o utilizador Nuno. Dada a análise preliminar sobre o projecto anterior e sobre plataformas para criação de licenças, podemos constatar que esta interface foi implementada segundo as rotinas do LPD, do Chillout, evitando assim a integração de dois sistemas distintos.

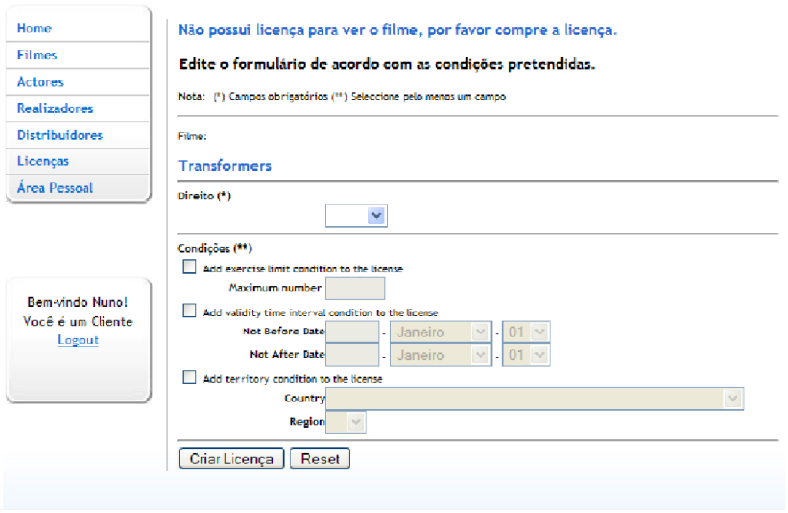


Figura 30 – Interface para a criação de licenças

A aplicação, através dos dados introduzidos pelo utilizador, cria uma licença no formato XML com o nome produzido através da associação entre o nome de utilizador e o filme para o qual a licença é gerada sendo ainda acrescentada uma marca temporal da altura em que é efectuado o download da licença. A figura 31 diz respeito ao conteúdo do ficheiro Nuno_Crash_ 1199808754.xml:

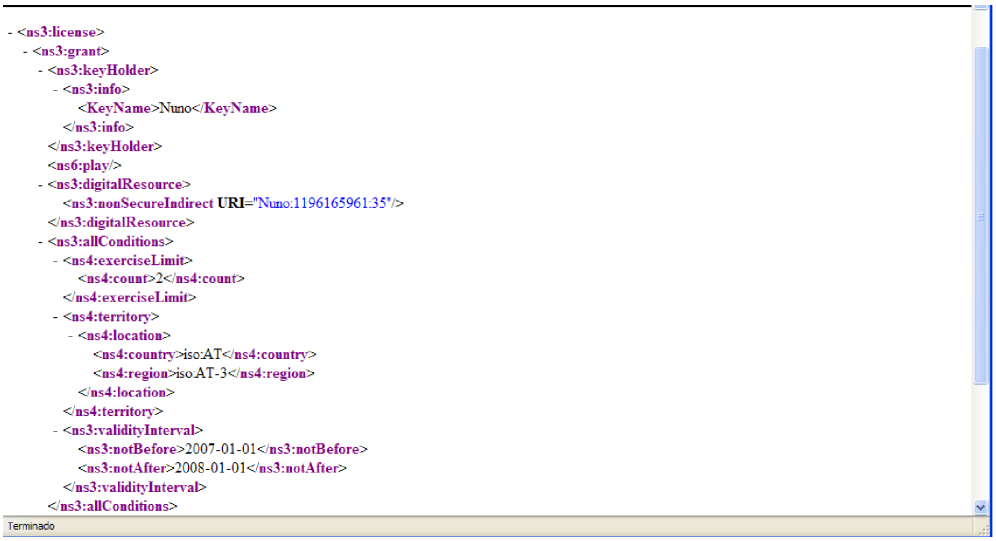


Figura 31 – Ficheiro XML resultante

Tal como é possível ver na figura anterior, o URL possui um valor único gerado pela aplicação. Esse valor é criado através da associação do nome do utilizador a uma marca temporal da altura em que a licença é gerada e também ao identificador do filme na base de dados.

De salientar que a estrutura do XML foi desenvolvida segundo a normal MPEG-21 REL. O MPEG-21 define uma framework normalizada para a disponibilização e entrega de multimédia baseada em dois conceitos essenciais: a definição de uma unidade fundamental para a distribuição e transacção (DI – Digital Item) e o conceito de utilizadores que interagem com os mesmos DIs. O DI é um objecto digital formado pelos conteúdos, por uma identificação única (URI) e a metadata.

O MPEG Rights Expression Language (REL), parte integrante do MPEG-21, é uma ferramenta desenhada para facilitar a criação de licenças fiáveis e seguras, requeridas pelos proprietários do conteúdo, sendo utilizadas ao longo da cadeia de utilizadores desde o criador até ao consumidor final do conteúdo.

O elemento básico para a definição do REL é a enunciação dos direitos da licença. Aqui se define qual ou quais as permissões que um utilizador ou consumidor tem perante esse conteúdo protegido, podendo existir vários níveis de complexidade ao nível das permissões e condições necessárias. Esta expressão de direitos pode ser criada por qualquer pessoa autorizada a disponibilizar permissões para conteúdo protegido, sendo normalmente um criador e/ou revendedor de licenças, devendo estar assegurada a assinatura digital para que a sua autenticidade possa ser comprovada.

São várias as plataformas que estabelecem os seus princípios básicos através da criação de licenças do tipo MPEG-21 REL, nomeadamente o VISNET II e o TIRAMISU.

Assim sendo a validação do XML foi efectuada com sucesso, como se pode constatar na figura 32. Este componente de teste inicia-se com a ligação ao servidor, cujo pedido de ligação é representado pelo URL onde se encontra alojado o XML, composto pelo protocolo, host e ficheiro. Após o servidor receber esse pedido, deverá responder eviando o ficheiro XML correspondente. Esse ficheiro é comparado com o ficheiro XML esperado, desenvolvido em TTCN-3, surgindo o veredicto "match" se a estrutura do XML recebido for semelhante à estrutura do XML esperado. O veredicto "pass" indica que o XML encontra-se bem validado.

Ao contrário dos outros componentes de teste implementados, foram utilizados dois tipos de componentes, Main Test Component (MTC) e Parallel Test Component (PTC). O componente PTC, neste teste representada por component1, tem como objectivo enviar o pedido de ligação ao servidor e validar a mensagem recebida. Já o componente MTC é criada automáticamente pelo sistema de teste no início de cada execução do teste e tem como objectivo indicar o veredicto final que foi estabelecido pelo componente PTC. Neste caso, apenas foi iniciado um componente PTC, caso contrário, o componente MTC só iria terminar a execução após todos os componentes PTC´s terminarem.

Foi também utilizado um temporizador local (localtimer) que foi inicializado na componente PTC, após o pedido de ligação ao servidor, que contabiliza o tempo de execução do pedido ao sistema. Assim sendo, podemos constatar que o ficheiro XML foi enviado num tempo inferior a um segundo (0,775 segundos).

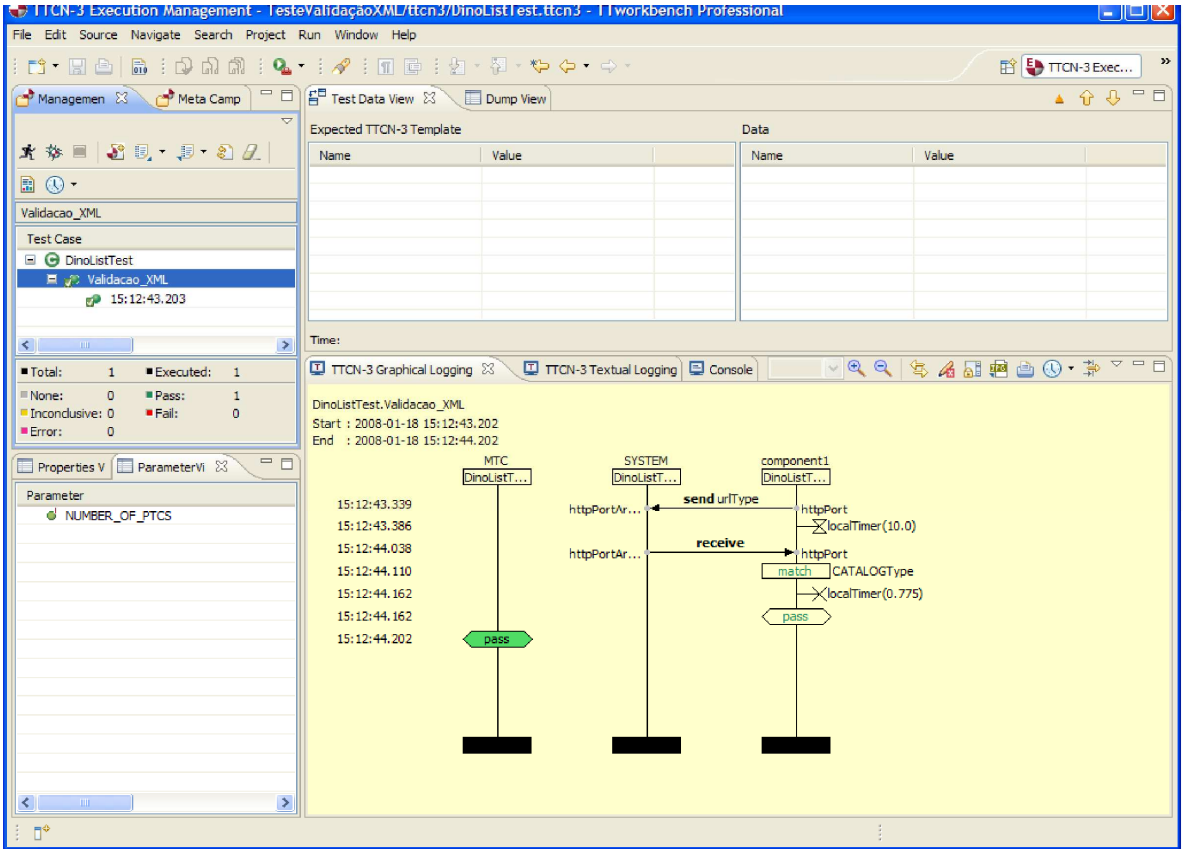


Figura 32 – Interface do Teste de Validação do XML

Capítulo 5

5. Perspectivas de Desenvolvimento VISNET II

Uma vez abordada a arquitectura DRM defendida pelo VISNET II, no capítulo 2, chegou-se a conclusão que é possível implementar componentes de teste, utilizando o software TTworkbench Professional, sobre os módulos enunciados anteriormente. Essa possibilidade de implementação está associada ao facto de existir uma constante troca de mensagens entre os módulos, durante o processo da reprodução do conteúdo protegido pela aplicação cliente, como se pode constatar na figura 5, e como tal, é possível controlar essa troca de mensagens. No fundo, estes componentes de teste visam simular o comportamento da arquitectura, forçando os módulos a enviar determinadas mensagens, como se se tratasse de um pedido de reprodução de conteúdo protegido por parte da aplicação cliente, bastando para isso conhecer apenas a estrutura das mensagens.

Essas mensagens são enviadas através de pedidos HTTP, para o servidor, definidas através de strings XML. Por sua vez, os módulos receptores, descodificam os pedidos efectuados pelos módulos emissores através da opção de parsing, ou seja, através da divisão da string XML. Desta forma é mais fácil identificar o pedido uma vez que a informação proveniente do XML encontra-se separada e não seguida, como seria de esperar se não se tivesse efectuado a divisão.

Assim sendo, um possível componente de teste estaria associado ao controlo de pedidos/respostas por parte de cada módulo, numa determinada zona da arquitectura, mais propriamente entre o módulo Intermediary e os três módulos com que comunica (Supervision Server, Governance Server e Content Server). Para isso, bastava adicionar à arquitectura DMAG-MIPAMS, representada na figura 2, quatro componentes PTCs, um para controlar as mensagens de entrada do módulo Intermediary, enviados pelo módulo Trusted Client, e cada um dos outros componentes para controlar as mensagens de saída enviada pelo módulo Intermediary para cada um dos módulos com quem estabelece comunicação.

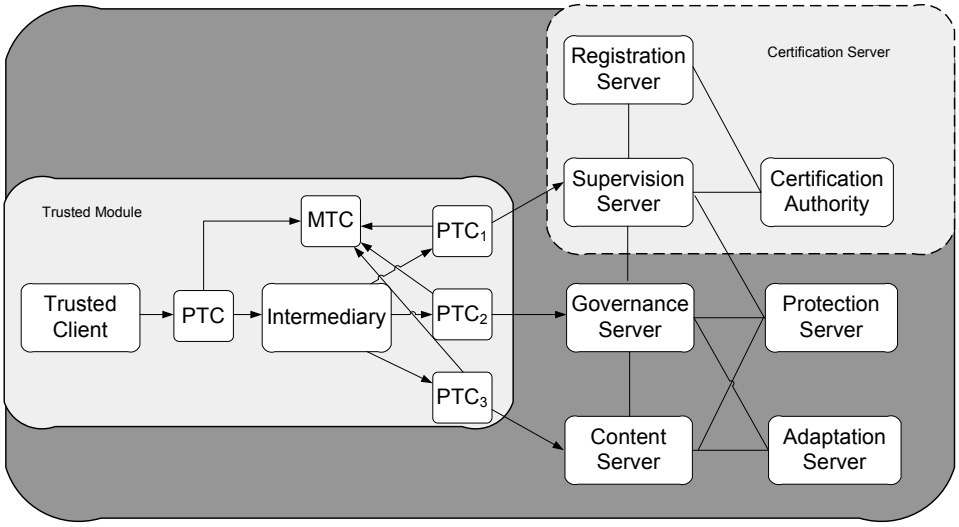


Figura 33 – Arquitectura de teste no projecto VISNET II

Como se pode constatar na figura anterior, existem quatro componentes PTCs que controlam as mensagens enviadas, entre dois módulos:

- PTC – Componente que controla as mensagens enviadas pelo módulo Trusted Client para o módulo Intermediary.
- PTC_1 – Componente que controla as mensagens enviadas pelo módulo Intermediary para o módulo Supervision Server.
- PTC_2 – Componente que controla as mensagens enviadas pelo módulo Intermediary para o módulo Governance Server.
- PTC_3 – Componente que controla as mensagens enviadas pelo módulo Intermediary para o módulo Content Server.

O primeiro passo na implementação do componente de teste seria interligar o componente PTC ao módulo Trusted Client e efectuar um pedido de envio das mensagens com a mesma estrutura das mensagens enviadas, durante o processo para a reprodução de conteúdo protegido pela aplicação cliente. Para isso, é necessário definir num ficheiro com extensão java, codec.java, a estrutura dessas mensagens, efectuando de seguida o pedido de envio.

De seguida, em cada componente PTC (PTC_1, PTC_2, PTC_3), é efectuada a comparação entre a estrutura das mensagens de saída do módulo Intermediary, com cada uma das mensagens de entrada do mesmo. Essa comparação indica-nos se essa mensagem foi enviada pelo módulo Trusted Client, no caso de se registar a mesma estrutura ou se essa mensagem foi enviada pelo módulo Intermediary, no caso de não se registar a mesma estrutura. Desta forma, seria possível saber que mensagens serão enviadas pelo módulo Trusted Client, que mensagens serão enviadas pelo módulo Intermediary e para que módulos serão enviados, respectivamente. O componente MTC, ao qual todos os componentes PTCs se encontram ligados, teria como único objectivo indicar o veredicto final que foi estabeleci-

do por cada componente PTC, individualmente. Esse veredicto final está associado ao sucesso do componente de teste, caso as condições testadas correspondam às esperadas ou o seu insucesso, caso não correspondam às esperadas. Uma vez que estamos perante vários componentes PTCs, o componente MTC só irá terminar a execução após todos os componentes PTCs terminarem a sua execução.

A nível da tecnologia Java, e uma vez que o que foi abordado no parágrafo anterior está associado a nível da tecnologia TTCN-3, essa comparação entre as estruturas das mensagens, de entrada e saída do módulo Intermediary, só é possível se se definir no mesmo ficheiro com extensão java a estrutura das mensagens de saída, uma vez que as mensagens de entrada já se encontram definidas. Ainda neste ficheiro seria necessário implementar funções relativas à codificação e à descodificação das mensagens.

Implementada a entidade Codecs, representada através do ficheiro com extensão java implementado, é necessário implementar dois outros ficheiros com extensão java, um associado ao modo como se processa a comunicação entre os módulos e outro associado às definições de envio das mensagens e ao mapeamento das portas entre os componentes de teste e as portas do sistema. Este último ficheiro está associado ao TTCN-3 Runtime Interface (TRI). Já o primeiro ficheiro está associado ao uso de sockets TCP/UDP. O uso deste tipo de sockets é um procedimento comum neste tipo de componentes de teste, uma vez que estamos a lidar com troca de mensagens.

Abordada esta perspectiva de desenvolvimento, podemos dizer que não se encontra limitada apenas aos módulos visados, podendo ser expandida. Uma possível expansão, muito ambiciosa por sinal, passaria por simular, utilizando o TTworkbench Professional, todas as mensagens trocadas entre os módulos, a nível de toda a arquitectura DMAG-MIPAMS e não apenas em relação a uma parte específica da arquitectura, como foi abordado anteriormente. Para isso seria necessária adicionar um componente PTC, para cada dois módulos interligados, e um componente MTC que teria o objectivo de estabelecer os veredictos, para cada componente PTC adicionado.

NOTA: Uma vez que estas perspectivas de desenvolvimento baseiam-se, numa dada quota-parte, nos componentes de teste implementados no capítulo anterior, o seu desenvolvimento futuro assenta num princípio de funcionalidade. No entanto, dado a complexidade da arquitectura DMAG-MIPAMS, arquitectura essa com uma complexidade incomparável relativamente a arquitectura do projecto “Serviço de VoD usando DRM”, diversos aspectos aqui enunciados podem ter de ser alterados, uma vez que esta perspectiva de desenvolvimento foi implementada, numa dada outra quota-parte, segundo aspectos puramente teóricos.

Capítulo 6

6. Conclusões

A realização deste projecto culminou com a produção de um conjunto de componentes de teste utilizados para validar o funcionamento dos elementos desenvolvidos no projecto “Serviço de VoD (Video on Demand) usando DRM (Digital Rights Management)”. Estes componentes de teste foram desenvolvidos numa aplicação denominada TTworkbench Professional, cujas suas funcionalidades foram reconhecidas como as pretendidas para este projecto, permitindo a sua integração futura no projecto VISNET II como ferramenta responsável pela comunicação entre os diferentes módulos do sistema, numa arquitectura DRM.

O TTworkbench Professional, desenvolvido pela empresa Testing Technologies, é uma aplicação interactiva. Apresenta uma interface gráfica, na interface TTman, que traduz o componente de teste implementado. Esta funcionalidade é uma mais valia uma vez que todas as interacções fisícas, passam agora a ser representadas gráficamente nessa interface, proporcionando uma melhor inteligíbilidade.

Ao longo deste documento percorreram-se as diversas fases que levaram à criação deste conjunto de componentes de teste. O arranque do projecto foi caracterizado pela análise do projecto anterior, tendo em vista uma melhor compreensão dos requisitos deste novo projecto. Diversas propostas de componentes de teste foram equacionadas tendo vindo a ser aprovadas ou rejeitadas em função da sua aplicabilidade no sistema a testar. Assim, após uma antevisão teórica, estes componentes de teste foram colocados em prática obtendo-se resultados imediatos sobre a sua performance.

Concluída a implementação dos componentes de teste, é possível admitir que os objectivos propostos foram atingidos na sua globalidade, havendo ainda disponibilidade para se atingir resultados para além dos objectivos propostos, deixando no ar várias perspectivas de desenvolvimento. Essas perspectivas de desenvolvimento estão associadas à arquitectura DRM do projecto VISNET II, pois trata-se de uma arquitectura mais complexa a todos os níveis, nomeadamente a nível de funcionalidades relacionadas com DRM, relativamente à analisada neste documento, tendo por isso um interesse acrescido na sua análise.

A nível pessoal destaque-se a evidente mais valia que este projecto produziu ao nível da aquisição de conhecimento na área de gestão dos direitos digitais através da criação de licenças, sobre as potencialidades do software TTworkbench Professional e da tecnologia TTCN-3, áreas essas desconhecidas até ao momento. Os objectivos foram cumpridos na íntegra dentro do prazo estabelecido devido a uma capacidade de organização e uma boa estruturação de tarefas entre as partes envolvidas no projecto.

Destaque-se ainda o enriquecimento profissional pela integração num ambiente jovem e extremamente dinâmico, que é predicado no INESC Porto.

Referências Bibliográficas

[1] The Apache HTTP Project, 2007 [Em Linha]. Disponível em http://httpd.apache.org [Consultado em 21/10/2007]

[2] PHP: Hypertext Processor, 2007. [Em linha]. Disponível em http://www.php.net/. [Consultado em 22/10/2007]

[3] PostgreSQL, 2007 [Em Linha]. Disponível em http://www.postgresql.org [Consultado em 23/10/2007]

[4] phpPgAdmin :: Web Based PostgreSQL Administration Tool, 2007 [Em Linha]. Disponível em http://phppgadmin.sourceforge.net [Consultado em 24/10/2007]

[5] Apache Tomcat, 2007 [Em Linha]. Disponível em http://tomcat.apache.org [Consultado em 25/10/2007]

[6] WebServices – Axis. [Em linha]. Disponível em http://ws.apache.org/axis/. [Consultado em 26/10/2007]

[7] VISNET II Networked Audiovisual Media Technologies, 2007 [Em Linha]. Disponível em http://www.visnet-noe.org/index.html [Consultado em 01/11/2007]

[8] VISNET II Networked Audiovisual Media Technologies, D3.1.2 - Intermediate Progress Report on DRM-based Content Protection Initial Integration, pp. 18-34, Setembro 2005 [Consultado em 08/11/2007]

[9] International Organization For Standardization, Introducing MPEG-21 REL - an Overview, Poznan, Poland, Julho 2005 [Consultado em 09/11/2007]

[10] TTCN, 2007 [Em Linha]. Disponível em http://en.wikipedia.org/wiki/TTCN [Consultado em 10/11/2007]

[11] The Evolution of TTCN, 2007 [Em Linha]. Disponível em http://www.itu.int/ITU-T/studygroups/com07/ttcn.html [Consultado em 10/11/2007]

[12] TTCN in a Nutshell, 2005 [Em Linha]. Disponível em http://www.site.uottawa.ca/~bernard/ttcn3_in_a_nutshell.html [Consultado em 13/11/2007]

[13] TTCN, 2004 [Em Linha]. Disponível em http://www.enel.ucalgary.ca/People/Smith/embeddedTDD/AgileInEmbbeded/TTCN3.html [Consultado em 13/11/2007]

[14] OpenTTCN [Em Linha]. Disponível em http://www.openttcn.com/Sections/Products/ [Consultado em 14/11/2007]

[15] The new standard TTCN: TTCN-3, 2006 [Em Linha]. Disponível em http://www.site.uottawa.ca/~bernard/ttcn.html [Consultado em 15/11/2007]

[16] ETSI´s official TTCN-3 home page, 2007 [Em Linha]. Disponível em http://www.ttcn-3.org/home.htm [Consultado em 20/11/2007]

[17] TTCN-3 Testing & Test Control Notation, 2007 [Em Linha]. Disponível em http://www.ttcn-3.org/oldsite/TTCN3about.htm [Consultado em 21/11/2007]

[18] Importing XMLSchema Datatypes into TTCN-3, 2007 [Em Linha]. Disponível https://syst.eui.upm.es/conference/sv04/papers/ImportXML_slides%20axel.pdf [Consultado em 22/11/2007]

[19] Models with TTCN-3, 2007 [Em Linha]. Disponível em http://www.ifi.uio.no/ecoop2004/docs/Ericsson.pdf [Consultado em 23/11/2007]

[20] Integration of TTCN-3, 2005 [Em Linha]. Disponível em http://www.site.uottawa.ca/~bernard/FOKUS_TTCN-3_Integration.pdf [Consultado em 24/11/2007]

[21] ETSI World Class Standards, 2007 [Em Linha]. Disponível em http://www.etsi.org/WebSite/Technologies/ttcn3.aspx [Consultado em 25/11/2007]

[22] International Institute for Software Technology, 2002 [Em Linha]. Disponível em http://www.iist.unu.edu/newrh/III/1/docs/techreports/report339.pdf [Consultado em 26/11/2007]

[23] Testing Technology, 2007 [Em Linha]. Disponível em http://www.testingtech.de/company/about.php [Consultado em 27/11/2007]

[24] TTworkbench, 2007 [Em Linha]. Disponível em http://www.testingtech.de/download/TTworkbench.pdf [Consultado em 27/11/2007]

[25] Banco de Dados: Migração, Integração, PostgreSQL e Oracle, 2007 [Em Linha]. Disponível em http://postmasterceara.blogspot.com/2007/10/conexo-do-postgresql-com-o-java.html [Consultado em 10/12/2007]

[26] Software Testing Process Automation Based on UTP – A Case Study, 2005[Em Linha]. http://www.cnsqa.com/cnsqa/jsp/html/spw/ppt/25104/Software%20Testing%20Process%20Automation%20Based%20on%20UTP%20A%20Case%20Study.pdf [Consultado em 10/12/2007]

[27] Viva o Linux, 2007 [Em Linha]. Disponível em http://www.vivaolinux.com.br/comunidades/verTopico.php?codigo=72&codtopico=12253 [Consultado em 15/01/2008]

[28] Conexão HTTP via POST para PHP, 2007 [Em Linha]. Disponível em http://www.javafree.org/javabb/viewtopic.jbb?t=855362 [Consultado em 16/01/2008]

[29] World Class Standards, 2008 [Em Linha]. Disponível em http://www.etsi.org/WebSite/Standards/Standard.aspx [Consultado em 17/01/2008]

[30] Internacional Telecommunication Union, 2007 [Em Linha]. Disponível em http://www.itu.int/ITU-T/ [Consultado em 19/01/2008]

Apêndice A- Conexão Aplicação VoD

Nesta secção é apresentado o código fonte dos ficheiros desenvolvidos, quer os ficheiros com extensão ttcn-3 como os com extensão java (ficheiro TTCN-3, SimpleCodec.java, PingPongAdapter.java e UDPTestAdapter.java) em TTworkbench Professional do primeiro componente de teste efectuado, referente à ligação e troca de mensagens entre a nossa aplicação e a aplicação VoD, com alguns comentários relevantes para a compreensão do código.

A.1 Ficheiro .TTCN3

```
module ProtocolExample {

//Definição dos campos do URL

type record urlType {
                  charstring  protocol,
                  charstring  host,
                  charstring  file
                  }

//Especificação do URL alvo de testes

template urlType urlTemplate := {
                  protocol    := "http://",
                  host        := "media-utm.inescn.pt",
                  file        := "/vod/listaFilmes.php
                  }

//Definição da informação que é enviada e recebida pelas portas

type port PingPongPortType message {

                  out charstring;
                  out   urlType;
                  in charstring;
            }

//Definição das portas da Main Test Component

type component MTCType{


            port PingPongPortType connection_port;
            //timer localtimer := 10.0;

            }

//Definição das portas do Sistema

type component SystemType{

            port PingPongPortType system_connection_port;
            }
```

//Teste de Conexão à aplicação VoD
//A aplicação envia o URL, se a aplicação VoD receber o URL, a aplicação envia uma mensagem de sucesso e inicia-se a troca de mensagens

```
testcase ConexaoAplicaçaoVoD() runs on MTCType system SystemType{

//Mapeamento das Portas de modo a estabelecer a comunicação

      map (mtc: connection_port,system: system_connection_port);
      connection_port.send(urlTemplate);
      alt {
            [] connection_port.receive("http://media-
utm.inescn.pt/vod/listaFilmes.php
                  log ("A aplicação VoD encontra-se conectada");
                  //localtimer.stop;
                  setverdict(pass)
                  }
                  []    connection_port.receive{
                  log ("A aplicação VoD não se encontra conectada");
                  //localtimer.stop;
                  setverdict(fail)
                  }
            }

}

//Desmapeamento das portas

            unmap(mtc:connection_port, system:system_connection_port);
            stop
}
```

// Teste de troca de mensagens em que a aplicação espera por uma mensagem específica da aplicação VoD e em caso de não a receber, retorna o veredicto "fail"

```
testcase Falha_TrocaMsgs() runs on MTCType system SystemType
{

//Mapeamento das Portas de modo a estabelecer a comunicação

      map (mtc: connection_port,system: system_connection_port);
      connection_port.send(urlTemplate);
      alt {
            [] connection_port.receive("http://media-
utm.inescn.pt/vod/listaFilmes.php"){
            log ("A aplicação VoD encontra-se conectada");
            //localtimer.stop;
            setverdict(pass)
            }
            [] connection_port.receive{
            log ("A aplicação VoD não se encontra conectada");
```

```
            //localtimer.stop;
            setverdict(fail)
            }
      }
      connection_port.send(CommonListagem)
      alt   {
            [] connection_port.receive(commonListagem_1)
            {
            log ("Troca de mensagens iniciada");
            setverdict(pass)
            }
      }
      connection_port.send(CommonListagem);
      alt {
            [] connection_port.receive(commonListagem_2){
            log ("Erro na troca de mensagens");
            setverdict(fail)
            }

            []    connection_port.receive{
            log ("Erro na troca de mensagens");
            setverdict(fail)
            }
      }

//Desmapeamento das portas

            unmap(mtc:connection_port, system:system_connection_port);
}
}
```

A.2 Ficheiro SimpleCodec.java

```
package com.testingtech.ttcn.tci.codec;

// Realiza a importação das bibliotecas necessárias para a implementação
das rotinas necessárias
import java.sql.*;
import java.util.*;
import org.etsi.ttcn.tci.RecordValue;
import org.etsi.ttcn.tci.TciCDProvided;
import org.etsi.ttcn.tci.TciTypeClass;
import org.etsi.ttcn.tci.Type;
import org.etsi.ttcn.tci.Value;
import org.etsi.ttcn.tri.TriMessage;
import com.testingtech.ttcn.tci.codec.base.AbstractBaseCodec;
import de.tu_berlin.cs.uebb.muttcn.runtime.RB;

// Este ficheiro é responsável pela codificação e descodificação das mensa-
gens trocadas a nível do ficheiro ttcn-3
public class SimpleCodec extends AbstractBaseCodec implements TciCDProvided
{
      private byte[] encodedMsg = null;

// Esta rotina cria um novo objecto Simple Codec rb utilizando um constru-
tor
public SimpleCodec(final RB rb) {
            super(rb);
      }

// Esta rotina codifica o template, de acordo com as regras de codificação
guardando na variável encodedMessage
      public TriMessage encode(final Value template) {

            final TriMessage encodedMessage = super.encode(template);
            return encodedMessage;
      }
// Esta rotina descodifica a mensagem de acordo com as regras de descodifi-
cação
      public Value decode(final TriMessage rcvdMessage,
                  final Type decodingHypothesis) {
            // Recebe o vector de bytes
            encodedMsg = rcvdMessage.getEncodedMessage();
            //Coloca a 0 para saber a actual posição do vector
            bitpos = 0;
            if (encodedMsg.length == 0) {
            return null;
            }
            // Verifica que class o decodingHypothesis pertence
            switch (decodingHypothesis.getTypeClass())
{
```

```
            case TciTypeClass.RECORD:
      if (decodingHypothesis.getName().equals("commonListagem")) {
                  final RecordValue commonListagem = (RecordValue) decodin-
gHypothesis
                  .newInstance();
                  return CommonListagem(commonListagem, encodedMsg);
                  }
      else {
                  return null;
                  }
                  default:
                  return super.decode(rcvdMessage, decodingHypothesis);
            }
      }
private RecordValue CommonListagem(final RecordValue commonListagem, com-
monListagem_1, commonListagem_2, final byte[] encodedMsg) {
      Connection con;
      String h1= "select h1 from movies.media";
      Statement stmt;
      con = DriverManager.getConnection(url);
      stmt = con.createStatement();
      ResultSet listagem = stmt.executeQuery(h1);
      String s1="Amadeus";
      String s2="Crash";
      setField(commonListagem,listagem.getString(i));
switch(op){
case 1:
            if(listagem.getString(1).equals(s1))
                {
                System.out.println("A resposta recebida corresponde à res-
posta esperada");
                 }
            else
                 {
                 System.out.println("A resposta recebida não corresponde à
resposta esperada");
                 }
            return commonListagem_1;
            break;
case 2:
             if(listagem.getString(1).equals(s2))
                 {
                  System.out.println("A resposta recebida não corresponde à
resposta esperada");
                 }
             else
                 {
                     System.out.println("A resposta recebida corresponde à
resposta esperada");
                 }
            return commonListagem_2;
            break;
      stmt.close();
      con.close();
       }
```

A.3 Ficheiro PingPongAdapter.java

```
package com.testingtech.ttcn.tri;

// Realiza a importação das bibliotecas necessárias para a implementação
das rotinas necessárias

import org.etsi.ttcn.tci.TciCDProvided;
import org.etsi.ttcn.tri.TriAddress;
import org.etsi.ttcn.tri.TriComponentId;
import org.etsi.ttcn.tri.TriMessage;
import org.etsi.ttcn.tri.TriPortId;
import org.etsi.ttcn.tri.TriStatus;

import com.testingtech.ttcn.logging.RTLoggingConstants;
import com.testingtech.ttcn.tci.codec.SimpleCodec;

// Este é um simples teste adaptador. Envia as mensagens para o seu desti-
no, especificado no ficheiro .ttcn3 e recebe respostas pela mesma porta por
onde enviou. A rotina triSend é a rotina responsável pelo envio e recepção
de mensagens.

public class PingPongAdapter extends TestAdapter {


// Esta rotina cria de um novo objecto adaptador

public PingPongAdapter() {
      super();
      }

// Esta rotina é utilizada para enviar uma mensagem TRI pela mesma porta
por onde foi recebida uma mensagem de pedido. Retorna o estado desta opera-
ção

public TriStatus triSend(final TriComponentId componentId,
                  final TriPortId tsiPortId, final TriAddress sutAddress,
                  final TriMessage message) {

            Cte.triEnqueueMsg(tsiPortId, new TriAddressImpl(new byte[] {}),
                        componentId, message);
            return new TriStatusImpl();
      }

//Rotina que retorna a implementação do codec base de um teste

public TciCDProvided getCodec(String encodingName) {
            if ((encodingName == null) || encodingName.equals("")) {
                  encodingName = "SimpleCodec";
            }
```

```
            TciCDProvided codec = super.getCodec(encodingName);
            if (codec != null) {
                  return codec;
            }

            if (encodingName.equals("SimpleCodec")) {
                  codec = new SimpleCodec(RB);
                  codecs.put(encodingName, codec);
            } else {

                  RB.getTciTLProvided().tliRT("", Sys-
tem.currentTimeMillis(), "", -1,
                  null, RTLoggingConstants.RT_LOG_ERROR,"Unknown decoding "
            + encodingName);
                  RB.tciTMProvided.tciError("Unknown decoding " + encoding-
Name);
            }
            return codec;
      }
}
```

A.4 Ficheiro UDPTestAdapter.java

```
package com.testingtech.ttcn.tri;

// Realiza a importação das bibliotecas necessárias para a implementação
das rotinas necessárias

import java.io.IOException;
import java.io.InterruptedIOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;

import org.etsi.ttcn.tci.CharstringValue;
import org.etsi.ttcn.tci.IntegerValue;
import org.etsi.ttcn.tci.TciCDProvided;
import org.etsi.ttcn.tri.TriAddress;
import org.etsi.ttcn.tri.TriComponentId;
import org.etsi.ttcn.tri.TriMessage;
import org.etsi.ttcn.tri.TriPortId;
import org.etsi.ttcn.tri.TriPortIdList;
import org.etsi.ttcn.tri.TriStatus;
import org.etsi.ttcn.tri.TriTestCaseId;

import com.testingtech.ttcn.logging.RTLoggingConstants;
import com.testingtech.ttcn.tci.TciModuleParameterIdImpl;
import com.testingtech.ttcn.tci.codec.SimpleCodec;


// Este é um teste adaptador UDP. Envia a mensagem por uma porta UDP para o
sistema a testar e recebe a mensagem que é enviada do sistema a testar pela
mesma porta UDP.


public class UDPTestAdapter extends TestAdapter {


      //Tamanho por default
      private final int MSG_BUF_SIZE = 1600;

      // Socket de recepção utilizada para comunicação assíncrona
      private DatagramSocket rxSocket;

      // Socket de transmissão utilizada para comunicação assíncrona
      private DatagramSocket txSocket;
      private final Object threadlock = new Object();
      private Thread udpReceiverThread = null;
      private boolean runThread = false;
```

```
	private String remoteIPAddress = ""
	private int remotePortNumber = 0;
	private int localPortNumber = 0;

	// Cria um novo objecto adaptador
	public UDPTestAdapter() {
		super();
	}


	// Rotina que retorna a implementação de um codec base de um teste

	public TciCDProvided getCodec(String encodingName) {
		if ((encodingName == null) || encodingName.equals("")) {
			encodingName = "SimpleCodec";
		}

		TciCDProvided codec = super.getCodec(encodingName);

		if (codec != null) {
			return codec;
		}

		if (encodingName.equals("SimpleCodec")) {
			codec = new SimpleCodec(RB);
			codecs.put(encodingName, codec);
		} else {

			RB.getTciTLProvided().tliRT("", Sys-
tem.currentTimeMillis(), "", -1,
			null, RTLoggingConstants.RT_LOG_ERROR,
			"Unknown decoding " + encodingName);
			RB.tciTMProvided.tciError("Unknown decoding " + encoding-
Name);
		}

		return codec;
	}
	// Esta operação é chamada pelo executável do teste imediatamente
antes da execução de algum caso de teste. O caso de teste que será executa-
do está indicado pelo testCaseId. A tsiPortList contém todas as portas que
foram declaradas na definição da componente do sistema para o caso de teste
(TSI ports).
	//Esta rotina apresenta como argumentos o testcase, identificador do
caso de teste que será executado e o tsiList, que contém a lista de portas
definidas para o sistema de teste.

	public TriStatus triExecuteTestcase(final TriTestCaseId testcase,
			final TriPortIdList tsiList) {
		// Recebe o IP remoto
		remoteIPAddress = ((CharstringValue) RB.getTciTMProvided()
		.tciGetModulePar(TciModuleParame-
terIdImpl.valueOf("REMOTE_IP_ADDRESS", null)))
		.getString();
```

```
            if (remoteIPAddress.equals("")) {
                  return new TriStatusImpl(
                              "TCPTestAdapter.triExecuteTestcase");
            }

            remotePortNumber = ((IntegerValue) RB.getTciTMProvided()
            .tciGetModulePar(TciModuleParame-
terIdImpl.valueOf("REMOTE_PORT_NUMBER", null)))
            .getInt();

            if (remotePortNumber == 0) {
                  return new TriStatusImpl(
                              "TCPTestAdapter.triExecuteTestcase:");
            }

            localPortNumber = ((IntegerValue) RB.getTciTMProvided()
            .tciGetModulePar(TciModuleParame-
terIdImpl.valueOf("LOCAL_PORT_NUMBER", null)))
            .getInt();

            if (localPortNumber == 0) {
                  return new TriStatusImpl(
                              "TCPTestAdapter.triExecuteTestcase");
            }

            rxSocket = null;
            txSocket = null;

            return new TriStatusImpl();
      }

      //Esta rotina mapea as portas dos componentes de teste de maneira a
estarem prontas a comunicar. Retorna TRI_OK se a operação for executada
correctamente e TRI_ERROR no caso contrário. Apresenta como argumentos o
compPortID como referência às portas dos componentes de teste e tsiPortId
como referência às portas do sistema.

      public TriStatus triMap(final TriPortId compPortId,
            final TriPortId tsiPortId) {
            final TriStatus mapStatus = CsaDef.triMap(compPortId, tsiPor-
tId);

            if (mapStatus.getStatus() != TriStatus.TRI_OK) {
                  return mapStatus;
            }

            //Prepara para comunicar
            try {
                  rxSocket = new DatagramSocket(localPortNumber);
                  txSocket = new DatagramSocket();
            } catch (final SocketException sex) {
                  return new TriStatusImpl("Unable to open socket for TSI
Port: "
                              + tsiPortId.getPortName());
            }
```

```
            //Encontra-se à escuta na socket de recepção
            runThread = true;
            udpReceiverThread = new Thread() {

                  public void run() {
                        boolean mylock = runThread;

                        try {rxSocket.setSoTimeout(100);

                              while (mylock) {
                              final byte[] msg = new byte[MSG_BUF_SIZE];
                                    final DatagramPacket commonListagem =
new DatagramPacket(msg,msg.length);

                        try {rxSocket.receive(commonListagem);
                        final byte[] trimmedMsg = new byte[commonListagem
                        .getLength()];
                        System.arraycopy(msg, 0, trimmedMsg, 0, com-
monListagem
                        .getLength());
                        System.out.println("\nMyTestAdapter: Received ( "
                                    + trimmedMsg.length
                                    + " bytes)\n----------------\n"
                                    + hexString(trimmedMsg));


                        final TriMessage rcvMessage = new TriMessageImpl(
                                                trimmedMsg);

                              synchronized (threadlock) {
                              if (runThread) {
                              enqueueMsg(tsiPortId, new TriAddressImpl(
                  new byte[] {}), compPortId.getComponent(), rcvMessage);
                                    }
                                    }
                  }     catch (final InterruptedIOException iioex) {
                  }     catch (final IOException ioex) {
                  System.out.println("MyTestAdapter: IOException "
                              + ioex + " - closing the receiver socket");

                                    if (rxSocket != null) {
                                    rxSocket.close();
                                    rxSocket = null;
                                    }

                                    return;
                                    }

                                    synchronized (threadlock) {
                                          mylock = runThread;
                                    }
                              }

                              if (rxSocket != null) {
```

```
                                    rxSocket.close();
                                    rxSocket = null;
                              }
                        } catch (final SocketException se) {
                              System.out.println("MyTestAdapter: SocketEx-
ception " + se);
                        }
                  }
            };
            udpReceiverThread.start();

            return new TriStatusImpl();
      }



      private void enqueueMsg(final TriPortId tsiPort,
                  final TriAddress sutAddress, final TriComponentId re-
ceiverComp,
                  final TriMessage rcvMessage) {
            Cte.triEnqueueMsg(tsiPort, new TriAddressImpl(new byte[] {}),
                        receiverComp, rcvMessage);
      }
//Rotina que desmapea as portas. Retorna TRI_ERROR se o fecho da conexão
não for efecutada correctamente e TRI_OK caso contrário.
      public TriStatus triUnmap(final TriPortId compPortId,
      final TriPortId tsiPortId) {
      final TriStatus mapStatus = CsaDef.triUnmap(compPortId, tsiPortId);

      if (mapStatus.getStatus() != TriStatus.TRI_OK) {
                  return mapStatus;
            }


      synchronized (threadlock) {
                  if (!runThread) {
                        return new TriStatusImpl();
                  }

                  runThread = false;
            }

            while (udpReceiverThread.isAlive()) {
                  try {
                        Thread.sleep(500);
                  } catch (final InterruptedException ie) {
                  }
            }

            //Fecha a socket de envio
            if (txSocket != null) {
                  txSocket.close();
                  txSocket = null;
            }
```

```
            return new TriStatusImpl();
      }
// Esta rotina é chamada pelo Executavel do teste quando é efectuado uma
operação de envio TTCN-3, mapeada como uma porta TSI. Esta operação é cha-
mada pelo Executável do teste para todas as operações de envio TTCN-3, se o
componente de sistema não for especificado nos casos de teste. A codifica-
ção do sendMessage é efectuada no Executável do teste antes desta operação
a nível do TRI.

      public TriStatus triSend(final TriComponentId componentId,
                  final TriPortId tsiPortId, final TriAddress address,
                  final TriMessage sendMessage) {
            try {
                  final byte[] mesg = sendMessage.getEncodedMessage();
                  final InetAddress addr = InetAd-
dress.getByName(remoteIPAddress);
                  final DatagramPacket commonListagem = new Datagram-
Packet(mesg, mesg.length,
                  addr, remotePortNumber);
                  System.out.println("\nTCPTestAdapter: Sending (to: " +
addr
                  + ")\n-------\n" + hexString(mesg));
                  txSocket.send(commonListagem);

                  return new TriStatusImpl();
            } catch (final IOException ioex) {
                  return new TriStatusImpl(ioex.getMessage());
            }
      }

}
```

Apêndice B – Validação XML

Nesta secção é apresentado o código fonte dos ficheiros desenvolvidos, quer os ficheiros com extensão ttcn-3 como os com extensão java (ficheiro TTCN3, Codec.java e XMLTestAdapter.java) em TTworkbench Professional do último componente de teste efectuado, referente á validação do XML, associado à criação de uma licença por parte de um utilizador, com alguns comentários relevantes para a compreensão do código.

B.1 Ficheiro .TTCN-3

```
module DinoListTest {

modulepar integer NUMBER_OF_PTCS := 1
            with {
             extension (NUMBER_OF_PTCS) "Numero de PTCS" ;
    }

//Definição dos campos do URL

type record urlType {
            charstring  protocol,
            charstring  host,
            charstring  file
    }

//Contrução do XML
type set of ns3:issuerType CATALOGType;

type record ns3:infoType {
            charstring  KeyName
            }

type record ns3:digitalResourceType {
            charstring  ns3:nonSecureIndirect
                }

type record ns4:exerciseLimitType {
            charstring  ns4:count
                  }

type record ns4:locationType {
            charstring  ns4:country,
            charstring  ns4:region
                  }

type record ns3:validityIntervalType {
            charstring  ns3:notBefore,
            charstring  ns3:notAfter
                  }

type record ns3:issuerType {
            charstring  ns3:keyHolder
                  }

type record ns3:keyHolderType {
            ns3infoType ns:3info
                  }
```

```
type record ns4:territoryType {
            ns4:locationType ns4:location
                  }

type record ns3:allConditionsType {
                  ns4:exerciseLimitType ns4:exerciseLimit,
                  ns4:territoryType ns4:territory,
                  ns3:validityIntervalType ns3:validityInterval
                                    }

type record ns6:playType {}

type record ns3:grantType {
            ns3:keyHolderType keyHolder,
            ns6:playType ns:6play,
            ns3:digitalResourceType ns3:digitalResource,
            ns3:allConditionsType ns3:allConditions
                                    }

type record ns3:licenseType {
            ns3:grantType ns3:grant,
            ns3:issuerType ns3:issuer
                                          }
//Especificação do URL alvo de testes

template urlType urlTemplate := {
            protocol    := "http://",
            host        := "media-utm.inescn.pt",
            file        := "/vod/tmp/Nuno:Crash:1199808754.xml"
    }

// Templates responsáveis pela validação do XML, onde se espera que o XML
apresente um campo KeyName cujo nome é Nuno.

template ns3:licenseType ValidatorTemplate := {XMLTemplate, *};

template ns3:infoType XMLTemplate := {
             KeyName          := "Nuno"

    }
//Definição do que é enviado e recebido pela porta httpPort

type port httpPortType message {
            out   urlType;
            in    ns3:licenseType;
    }

//Definição das portas responsáveis pela comunicação entre PTC e o System

type component ptcType {

            port httpPortType httpPort;
            timer localTimer := 10.0;
    }
```

```
type component systemType {
            port httpPortType   httpPortArray[NUMBER_OF_PTCS];
    }

type component mtcType {}

//Função responsável pelo envio do URL e recepção do XML. Retorna veredic-
tos “pass” ou “fail” em caso de sucesso ou insucesso na procura do campo
KeyName igual a Nuno.

function ptcBehaviour() runs on ptcType {
          httpPort.send(urlTemplate);
          localTimer.start;
            alt {
                  [] httpPort.receive(ValidatorTemplate) {
                  localTimer.stop;
                  setverdict(pass);
                  }
                  [] httpPort.receive {
                  localTimer.stop;
                  setverdict(fail);
                  }
                  [] localTimer.timeout {
                     setverdict(fail);
                  }
            }
      }


testcase Validacao_XML() runs on mtcType system systemType {

          var ptcType ptcArray[NUMBER_OF_PTCS];
          var integer i := 0;
          for (i:=0; i<NUMBER_OF_PTCS; i:=i+1)
{
       //Cria as PTCS

            ptcArray[i] := ptcType.create;
          }
          for (i:=0; i<NUMBER_OF_PTCS; i:=i+1) {

       //Mapea as PTCS com as portas do sistema

              map (ptcArray[i]:httpPort, system:httpPortArray[i]);
          }
          for (i:=0; i<NUMBER_OF_PTCS; i:=i+1)
{
       //Inicia o ptcBehaviour, função essa definida anteriormente
            ptcArray[i].start(ptcBehaviour());
          }
            all component.done;
    }
}
```

B.2 Ficheiro Codec.java

```
package com.testingtech.ttcn.tci.codec;

// Realiza a importação das bibliotecas necessárias para a implementação
das rotinas necessárias

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.Date;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.etsi.ttcn.tci.CharstringValue;
import org.etsi.ttcn.tci.RecordOfValue;
import org.etsi.ttcn.tci.RecordValue;
import org.etsi.ttcn.tci.TciCDProvided;
import org.etsi.ttcn.tci.TciTypeClass;
import org.etsi.ttcn.tci.Type;
import org.etsi.ttcn.tci.Value;
import org.etsi.ttcn.tri.TriMessage;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

import com.testingtech.ttcn.tci.ExtendedTciTypeClass;
import com.testingtech.ttcn.tci.codec.base.AbstractBaseCodec;

import de.tu_berlin.cs.uebb.muttcn.runtime.RB;
import de.tu_berlin.cs.uebb.muttcn.runtime.ValueImpl;

// Esta rotina implementa a interface TciCDProvided
// Codifica um valor TTCN-3 numa representação de Mensagem Tri

public class Codec extends AbstractBaseCodec implements TciCDProvided {

//Esta rotina cria um novo codec

public Codec(final RB rb) {

    super(rb);
  }
```

```
// Codifica um valor TTCN-3 (URL) numa representação de Mensagem Tri. Esse
valor contém uma representação do URL

  public TriMessage encode(final Value value) {

    final TriMessage encodedMessage = super.encode(value);

    return encodedMessage;
  }

//Rotina de descodificação do URL. Copia para uma String XML o XML codifi-
cadoe efectuada de seguida a divisão, ou seja, o parsing

  public Value decode(final TriMessage message, final Type type) {

    final ByteArrayInputStream bais = new ByteArrayInputStream(message
        .getEncodedMessage());

    return decodeNodes(type.newInstance(), parse(bais));
  }

// A rotina DecodeNodes descodifica um nó XML numa representação TTCN-3.
Este método é chamado recursivamente para descodificar dados estruturados
de um XML

  private Value decodeNodes(final Value value2feed, final Node node) {

    final Type type = value2feed.getType();

    switch (type.getTypeClass()) {
      case TciTypeClass.RECORD:
      case TciTypeClass.SET:

        if (node != null && node.getNodeType() == Node.ELEMENT_NODE) {
          final RecordValue record = (RecordValue) type.newInstance();
          final String[] ttcnFieldNames = record.getFieldNames();
          final NodeList nodeChildren = node.getChildNodes();

          for (final String element : ttcnFieldNames) {
            final Value fieldValue = record.getField(element);

            for (int i = 0; i < nodeChildren.getLength(); i++) {
              final Node child = nodeChildren.item(i);

              switch (fieldValue.getType().getTypeClass()) {
                case TciTypeClass.RECORD:
                case TciTypeClass.RECORD_OF:
                case TciTypeClass.SET:
                case TciTypeClass.SET_OF:
                case TciTypeClass.CHARSTRING:

                  if (child.getNodeType() == Node.ELEMENT_NODE
                      && child.getNodeName().equals(element)) {
```

```
                    record.setField(element, decodeNodes(fieldValue,
child));
                  }

                  break;

                default:
                  RB.getLogging().logError(new Date().getTime(), this,
                      "Unsuppported Type " + field-
Value.getType().getName());
                  RB.getTciTMProvided().tciError(
                      "Unsuppported Type " + field-
Value.getType().getName());

                  return null;
              }
            }
          }

          return record;
        }

        break;

      case TciTypeClass.RECORD_OF:
      case TciTypeClass.SET_OF:

        if (node != null && node.getNodeType() == Node.ELEMENT_NODE) {
          final RecordOfValue recordOf = (RecordOfValue)
type.newInstance();
          final Value elementValue = recor-
dOf.getElementType().newInstance();

          recordOf.setLength(0);

          final NodeList nodeChildren = node.getChildNodes();

          for (int i = 0; i < nodeChildren.getLength(); i++) {
            final Node child = nodeChildren.item(i);

            switch (elementValue.getType().getTypeClass()) {
              case TciTypeClass.RECORD:
              case TciTypeClass.RECORD_OF:
              case TciTypeClass.SET:
              case TciTypeClass.SET_OF:
              case TciTypeClass.CHARSTRING:

                if (child.getNodeType() == Node.ELEMENT_NODE) {
                  recordOf.appendField(decodeNodes(elementValue, child));
                }

                break;

              default:
                RB.getLogging().logError(new Date().getTime(), this,
```

```
                    "Unsuppported Type " + element-
Value.getType().getName());
                RB.getTciTMProvided().tciError(
                    "Unsuppported Type " + element-
Value.getType().getName());

                return null;
              }
            }

            return recordOf;
          }

          break;

        case TciTypeClass.CHARSTRING:

          final NodeList nodeChildren = node.getChildNodes();

          for (int i = 0; i < nodeChildren.getLength(); i++) {
            final Node child = nodeChildren.item(i);

            if (child != null && child.getNodeType() == Node.TEXT_NODE) {
              final CharstringValue charstring = (CharstringValue) type
                  .newInstance();
              charstring.setString(child.getNodeValue());

              return charstring;
            }
          }

          break;

        case ExtendedTciTypeClass.ANY:

          final ValueImpl anytype = (ValueImpl) type.newInstance();

          if (node != null) {
            anytype.setPresent();
          }

          return anytype;

        default:
          RB.getLogging().logError(new Date().getTime(), this,
              "Unsuppported Type " + type.getName());
          RB.getTciTMProvided().tciError("Unsuppported Type " +
type.getName());
      }

      return null;
    }
```

```
// Rotina responsável pelo parsing da string XML. Retorna o ficheiro XML ou
a excepção TCIException se ocorrer um erro durante a operação de parsing.

private Element parse(final InputStream input)
  {

    if (RB.debug) {
      RB.getLogging().logDecode(new Date().getTime(), this,
          "Codec: parse(InputStream input) entered");
    }
    try {
      final DocumentBuilderFactory docBuilderFactory = DocumentBuilderFac-
tory
          .newInstance();
      docBuilderFactory.setValidating(true);

      final DocumentBuilder docBuilder = docBuilderFac-
tory.newDocumentBuilder();
      final Document document = docBuilder.parse(input);
      input.close();

      return document.getDocumentElement();
    }
    catch (final Exception e) {
      RB.getLogging().logError(new Date().getTime(), this, e.getMessage());
      RB.getTciTMProvided().tciError(e.getMessage());
    }

    return null;
  }
}
```

B.3 Ficheiro .XMLTestAdapter.java

```
package com.testingtech.ttcn.tri;

// Realiza a importação das bibliotecas necessárias para a implementação
das rotinas necessárias

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;

import org.etsi.ttcn.tci.TciCDProvided;
import org.etsi.ttcn.tri.TriAddress;
import org.etsi.ttcn.tri.TriCommunicationSA;
import org.etsi.ttcn.tri.TriComponentId;
import org.etsi.ttcn.tri.TriMessage;
import org.etsi.ttcn.tri.TriPlatformPA;
import org.etsi.ttcn.tri.TriPortId;
import org.etsi.ttcn.tri.TriPortIdList;
import org.etsi.ttcn.tri.TriStatus;
import org.etsi.ttcn.tri.TriTestCaseId;

import com.testingtech.ttcn.logging.RTLoggingConstants;
import com.testingtech.ttcn.tci.TciEncoding;
import com.testingtech.ttcn.tci.codec.Codec;

public class XMLTestAdapter extends TestAdapter implements TriCommunica-
tionSA,
    TriPlatformPA, TciEncoding {

private static final long serialVersionUID = 1952169012424376970L;

// Contrutor do objecto XMLTestAdapter

  public XMLTestAdapter() {

    super();
  }
```

```
// Esta rotina retorna um codec capaz de codificar segundo o EncodingName

  public TciCDProvided getCodec(final String anEncodingName) {

    String encodingName = anEncodingName;
    if (encodingName == null || encodingName.equals("")) {
      encodingName = "XMLCodec";
    }

    TciCDProvided codec = super.getCodec(encodingName);

    if (codec != null) {
      return codec;
    }

    if (encodingName.equals("XMLCodec")) {
      codec = new Codec(RB);
      codecs.put(encodingName, codec);
    }
    else {
            RB.getTciTLProvided().tliRT("", System.currentTimeMillis(), "",
-1, null,
          RTLoggingConstants.RT_LOG_INFO, "Unknown codec " + encodingName);
      RB.getTciTMProvided().tciError("Unknown codec " + encodingName);
    }

    return codec;
  }

// Esta operação é chamada pelo executável de teste imediatamente antes da
execução de casos de teste. O caso de teste que vai ser executado é indica-
do pelo testCaseID.


  public TriStatus triExecuteTestcase(final TriTestCaseId testcase,
      final TriPortIdList tsiList) {

    return new TriStatusImpl();
  }

//Esta rotina mapea as portas de componentes de teste numa porta do siste-
ma. O triMap retorna TRI_OK em caso desse mapeamento ser feita com sucesso
e TRI_Error no caso oposto.

  public TriStatus triMap(final TriPortId compPortId, final TriPortId tsi-
PortId) {

    final TriStatus mapStatus = CsaDef.triMap(compPortId, tsiPortId);

    RB.getTciTLProvided().tliRT("", System.currentTimeMillis(), "", -1,
null,
        RTLoggingConstants.RT_LOG_INFO,
        "tsiPortId.getPortIndex() = " + tsiPortId.getPortIndex());
```

```
    if (mapStatus.getStatus() != TriStatus.TRI_OK) {
      return mapStatus;
    }

    return new TriStatusImpl();
  }

//Esta rotina é chamada pelo Executável do teste quando executa uma opera-
ção de envio TTCN-3 por uma porta que foi mapeada. Esta rotina é chamada
por todas as operações de envio TTCN-3 se nenhum componente do sistema for
especificado nos casos de teste, ou seja, só um componente MTC é criado
para o caso de teste. A codificação da mensagem enviada tem de ser efectua-
do no Executável do teste antes de ser chamada esta rotina. Retorna TRI_OK
no caso de sucesso ou TRI_Error no caso de falha.

  public TriStatus triSend(final TriComponentId componentId,
      final TriPortId tsiPortId, final TriAddress address,
      final TriMessage sendMessage) {

    RB.getTciTLProvided().tliRT("", System.currentTimeMillis(), "", -1,
null,
        RTLoggingConstants.RT_LOG_INFO,
        "triSend: sendingComponent: " + componentId.getComponentId());
    RB.getTciTLProvided().tliRT("", System.currentTimeMillis(), "", -1,
null,
        RTLoggingConstants.RT_LOG_INFO,
        "triSend: destination tsiPortId: " + tsiPortId);

    try {
      final byte[] mesg = sendMessage.getEncodedMessage();
      final URL url = new URL(new String(mesg));
      final HttpURLConnection httpConn = (HttpURLConnection) url
          .openConnection();
      httpConn.setRequestMethod("GET");
      httpConn.setUseCaches(false);

      // Cria uma thread de recepção

      final Thread receiverThread = new Thread() {

        public void run() {

          try {
            if (httpConn.getResponseCode() != HttpURLConnection.HTTP_OK) {
              RB.getTciTLProvided().tliRT(
                  "",
                  System.currentTimeMillis(),
                  "",
                  -1,
                  null,
                  RTLoggingConstants.RT_LOG_INFO,
                  "triSend: ResponseCode is not HTTP_OK, ResponseMessage is
"
                      + httpConn.getResponseMessage());
            }
```

```
            else {
              StringBuffer sb = new StringBuffer();
              InputStream is = httpConn.getInputStream();
              BufferedReader br = new BufferedReader(new InputStream-
Reader(is));
              String line = br.readLine();

              while (line != null) {
                sb.append(line);
                line = br.readLine();
              }

              TriMessage rcvMessage = new TriMessageImpl(sb.toString()
                  .getBytes("UTF-8"));

              enqueueMsg(tsiPortId, new TriAddressImpl(new byte[]{}),
                  componentId, rcvMessage);
              is.close();

            // Fecha a conexão

              httpConn.disconnect();
            }
          }
          catch (IOException ex) {
            RB.getTciTLProvided().tliRT("", System.currentTimeMillis(), "",
-1,
                null, RTLoggingConstants.RT_LOG_ERROR,
                "XMLTestAdapter: triSend: " + ex.toString());
            RB.getTciTMProvided().tciError(
                "XMLTestAdapter: triSend: " + ex.toString());
          }
        }
      };

      receiverThread.start();
    }
    catch (final MalformedURLException ex) {
      return new TriStatusImpl(ex.toString());
    }
    catch (final IOException ex) {
      return new TriStatusImpl(ex.toString());
    }

    return new TriStatusImpl();
  }

  private void enqueueMsg(final TriPortId tsiPort, final TriAddress sutAd-
dress,
      final TriComponentId receiverComp, final TriMessage rcvMessage) {

    Cte.triEnqueueMsg(tsiPort, new TriAddressImpl(new byte[]{}), receiver-
Comp,
        rcvMessage
```