Interoperability tests for components based architectures.

22 May 2018
Agenda

- **Test Strategy**
  - Objectives
  - Overview
  - Test Design Process
  - Compliance checkpoints definition
  - Modeling
  - Test generation

- **Test Bench modularity & automation**
  - Objectives
  - Overview
  - Test execution software capabilities.
  - Real implementation on JTRS Audio Device

- **Test Bench Dataflow**
  - Example on startTone() function test.
Generic approach for components based architectures by interfaces testing.

Test design based on the behavior of the system under Test (Model Based Testing).

Tests should be exported into several formats and programming languages.
Test Strategy
Overview

- Traceability
- SDR Standard Requirements (JTRS, SCA 2.2.2., etc..)
- Compliance checkpoints definition
- Static Tests
- Functional Behavior Tests
- Non-functional Tests (Performance)
- Static Analysis
- Model-Based Testing
- Custom test design
- Traceability
- SDR Standard compliance Test repository
Test Strategy
Functional Test design Process

SDR Standard

Test Design
MBT Model

Test Management
Abstract Tests
Requirements
Compliance checkpoints

Test Execution
Tests scripts (language dependant)
Adaptation layer Specification

Adaptation Layer

Compliance Checkpoints definition
Requirements Extraction

Test Architect

Test
Architect

Compliance

Requirements

SDR Standard

Ingenior

Copyright © 2018 5

Test Strategy
Functional Test design Process
Test Strategy

Compliance checkpoints definition

- Compliance checkpoint defines the test objectives
- Success case(s) or Error case(s) definition
- Definition of test success criteria.
- Definition of the applicability of the test.

Sample on the startTone function of Audio Device

<table>
<thead>
<tr>
<th>Requirement Identifier</th>
<th>Requirement Text</th>
<th>PDC Identifier</th>
<th>PDC Applicab</th>
<th>Component</th>
<th>PDC Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JTRS_AD_PROVIDE_START_TONE</td>
<td>The startTone operation provides the user the ability to start the generation of a previously created tone/beep to the device user.</td>
<td>JTRS_AD_PROVIDE_START_TONE_SUCCESS_001</td>
<td>Platform</td>
<td>GPP</td>
<td>* Success case * the tone or beep identification number is valid * Check the tone is started</td>
</tr>
<tr>
<td>JTRS_AD_PROVIDE_START_TONE</td>
<td>InvalidToneId (see A.5.3.2) A CORBA exception is raised when the tone/beep identification number is invalid.</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>JTRS_AD_PROVIDE_START_TONE_EXCEPTION_INVALIDTONEID</td>
<td>JTRS_AD_PROVIDE_START_TONE_EXCEPTION_INVALIDTONEID_001</td>
<td>Platform</td>
<td>GPP</td>
<td>* Check an exception: InvalidToneId is raised * Not existing Tone Id</td>
<td></td>
</tr>
</tbody>
</table>
Test Strategy
Modeling

UML Class diagrams for abstract test implementation

Functions to call on set up before the test body

Test body

Functions to call on Tear down to return to initial state

Test edition

| api_AudioPortDeviceConfig.api_setAudioPortParams(Enum_Audio_Params_Valid_1) |
| api_AudibleAlertsAndAlarms.api_createTone(Enum_Audio_Channel_2, Enum_Tone_Profile_Multi_Tone_Valid_withOneTone_1) |
| api_AudibleAlertsAndAlarms.prepare_StartRecording() |
| api_ChannelAudioConfig.api_getOutputGain(Enum_Audio_Channel_2) |
| api_ChannelAudioConfig.api_setOutputGain(Enum_Audio_Channel_2, Enum_Output_Gain_Valid_1) |
| common_body() |
| api_AudibleAlertsAndAlarms.api_startTone(Enum_Audio_Channel_2, Enum_Tone_id_1) |
| api_ChannelAudioConfig.api_setOutputGain(Enum_Audio_Channel_2, Enum_Output_Gain_Default) |
| api_AudibleAlertsAndAlarms.api_destroyTone(Enum_Audio_Channel_2, Enum_Tone_id_1) |
| api_AudibleAlertsAndAlarms.api_stopAllTones(Enum_Audio_Channel_2) |
Test Strategy

Test generation: Abstract Tests

- Example of C++ test with the startTone function of JTRS Audio Device
  - Each generated function is a single test step.
  - Each test will be an assembly of single steps.

```cpp
bool JTRS_AD_PROVIDE_START_TONE_1::setUp()
{
    current_result = m_adapter->api_setAudioPortParams(<params>);
    current_result = m_adapter->api_getOutputGain(<params>);
    current_result = m_adapter->check_getOutputGain_Record_DefaultOutputGain(<params>);
    current_result = m_adapter->prepare_StartRecording(<params>);
    current_result = m_adapter->api_createTone(<params>);
    current_result = m_adapter->check_createTone_Record_ReturnedId(<params>);
    return current_result;
}

bool JTRS_AD_PROVIDE_START_TONE_1::test()
{
    current_result = m_adapter->api_startTone(<params>);
    current_result = m_adapter->check_ToneStatusForChannel(<params>);
    return current_result;
}

bool JTRS_AD_PROVIDE_START_TONE_1::tearDown()
{
    current_result = m_adapter->api_setOutputGain(<params>);
    current_result = m_adapter->api_destroyTone(<params>);
    current_result = m_adapter->api_stopAllTones(<params>);
    current_result = m_adapter->bench_tearDown();
    return current_result;
}
```
**Agenda**

- Test Strategy
  - Objectives
  - Overview
  - Test Design Process
  - Compliance checkpoints definition
  - Modeling
  - Test generation

- Test Bench modularity & automation
  - Objectives
  - Overview
  - Test execution software capabilities.
  - Real implementation on JTRS Audio Device

- Test Bench Dataflow
  - Example on startTone() function test.
Test Bench modularity & Automation

Objectives

- Maximum automation of test campaigns
- In the same spirit automatic management of measurement tools
- No dependency between Test management software and test execution software.
- Portability of test execution software to address several systems under test.
**Test Bench modularity & Automation**

**Overview**

- **Test Management Software**
  - HMI
  - Command Test scripts
  - Test Campaigns Management
  - Reports generation
  - Parameters Management
  - Test Data Base

- **Test Execution Software**
  - Test control Software
  - Deployment
  - Specific Test harness Software
  - Measurement Tools
  - System under Test
  - Component Under test

Custom Protocol On TCP/IP Interoperable Between OS
Test Bench modularity & Automation
Test execution software capabilities.

- Case n°1: Interfaces of the component under test are reachable from outside (CORBA bus available for example)
Test Bench modularity & Automation
Test execution software capabilities.

- Case n°2: Interfaces of the component under test are not reachable from outside.
Test Bench modularity & Automation
Real implementation on JTRS Audio Device

Deployment phase by using SCA 2.2.2.
Test Bench modularity & Automation
Real implementation on JTRS Audio Device

- Connection phase using SCA 2.2.2. capabilities

Diagram:
- Test Execution software
- Domain Manager
  - Application
  - Audio Device
    - Tests
    - Harness
  - Device Manager
  - GPP Device
  - Command & Control
- Test control Software
- System under Test
  - Naming Service
  - Domain Manager
  - Device Manager
  - Audio Device
  - Audio board
Agenda

- Test Strategy
  - Objectives
  - Overview
  - Test Design Process
  - Compliance checkpoints definition
  - Modeling
  - Test generation

- Test Bench modularity & automation
  - Objectives
  - Overview
  - Test execution software capabilities.
  - Real implementation on JTRS Audio Device

- Test Bench Dataflow
  - Example on startTone() function test.
Test Bench Data flow

Example on startTone() function test.

- Test request from Test management Software

![Diagram showing Test Bench Data flow](image)
Test Bench Data flow

Example on startTone() function test.

- Test Request on Audio Device Harness

```
runTest (StartTone_001)
```

Example on startTone() function test.
Test Bench Data flow

Example on startTone() function test.

- Test Set Up

```java
setUp() :
    SetAudioParams()
    CreatTone()
    Etc..
```

Prepare Capture
Test Bench Data flow

Example on startTone() function test.

- Test main initial phase
Test Bench Data flow
Example on startTone() function test.

- Test main capture phase
Test Bench Data flow

Example on startTone() function test.

- **Test Tear Down**

![Diagram of Test Bench Data flow]

- **Test Execution software**
  - **Domain Manager**
    - **Application**
    - **Audio Device**
      - Tests
      - Harness
    - **GPP Device**
    - **Device Manager**
      - **Test control Software**

- **System under Test**
  - **Naming Service**
  - **Domain Manager**
  - **Device Manager**
    - **Audio Device**

- **Capture**
  - **Audio board**

```plaintext
TearDown() : 
  DestroyTone() Etc ..
```

Test Management Software
Test Bench Data flow

Example on startTone() function test.

- Result Return from Harness

Test Execution software

Domain Manager

Application

Audio Device
Tests
Harness

Return Result of
StartTone_001

Device Manager

GPP Device

Test control
Software

Capture

Audio board

Test Management
Software

System under Test

Naming
Service

Domain Manager

Device Manager

Audio Device

Copyright © 2018
Test Bench Data flow

Example on startTone() function test.

- Result return to Test Management Software

![Diagram showing the data flow process with various components including Test Management Software, Domain Manager, System under Test, and Audio Board. The diagram illustrates the flow from Test Execution software capturing results and logs of the StartTone_001 function test.]
Questions