CERTIF: Conformity tests on software defined radio platforms

15 May 2019

Agenda

- SDR conformance assessment: the needs

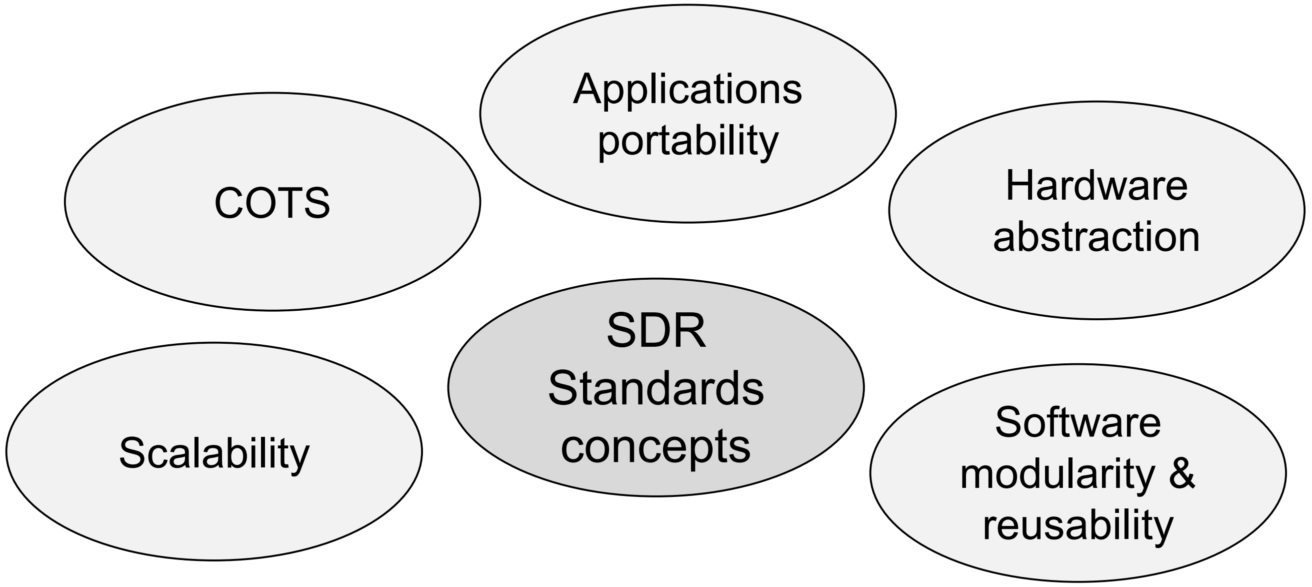
- Testing methodology
  - Test design process
  - From the SDR requirements to the tests
  - Compliance checkpoints definition
  - Modeling
  - Testing generation

- Non conformity detection
  - Not Implemented Interface
  - Wrong interface
  - Non conform behavior
  - Non conform data processing
  - Test of boundaries values

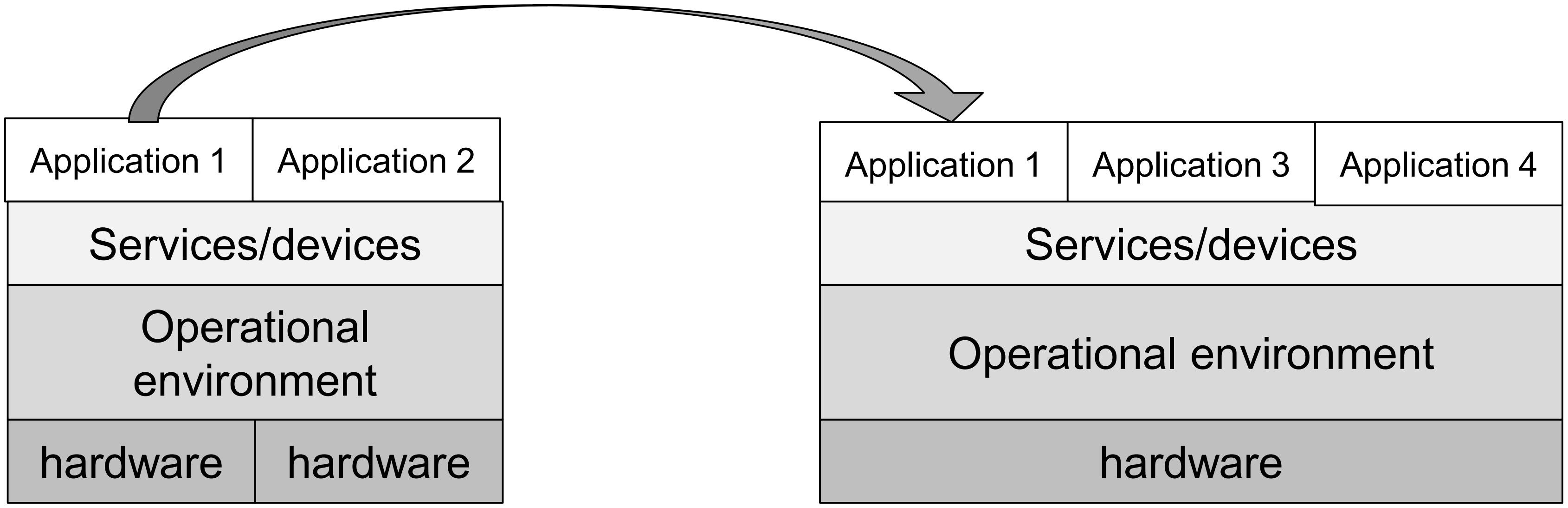
- Conclusion / Q&A

SDR Compliance Assessment

The needs



Needs to assess the compliance to these SDR Standards concepts



SDR Compliance Assessment

Assumptions

- Assumptions on the nature of the systems under test
  - The Software radio platforms

    The system under test is an SDR platform with GPP, DSP and FPGA processing resources running a compliant ESSOR Architecture operating Environment

  - The Application (Waveforms)

    The system under test is a set of source code files that compiles including IDL, C/C++,VHDL and XML

- Assumptions on the compliance check method
  - The Software radio platforms

    The compliance analysis is performed through dynamic tests by calling platform interfaces

  - The Applications

    The compliance analysis is performed through source code static analysis. A porting stage is not needed.

Agenda

- SDR conformance assessment: the needs

- Testing methodology
  - Test design process
  - From the SDR requirements to the tests
  - Compliance checkpoints definition
  - Modeling
  - Testing generation

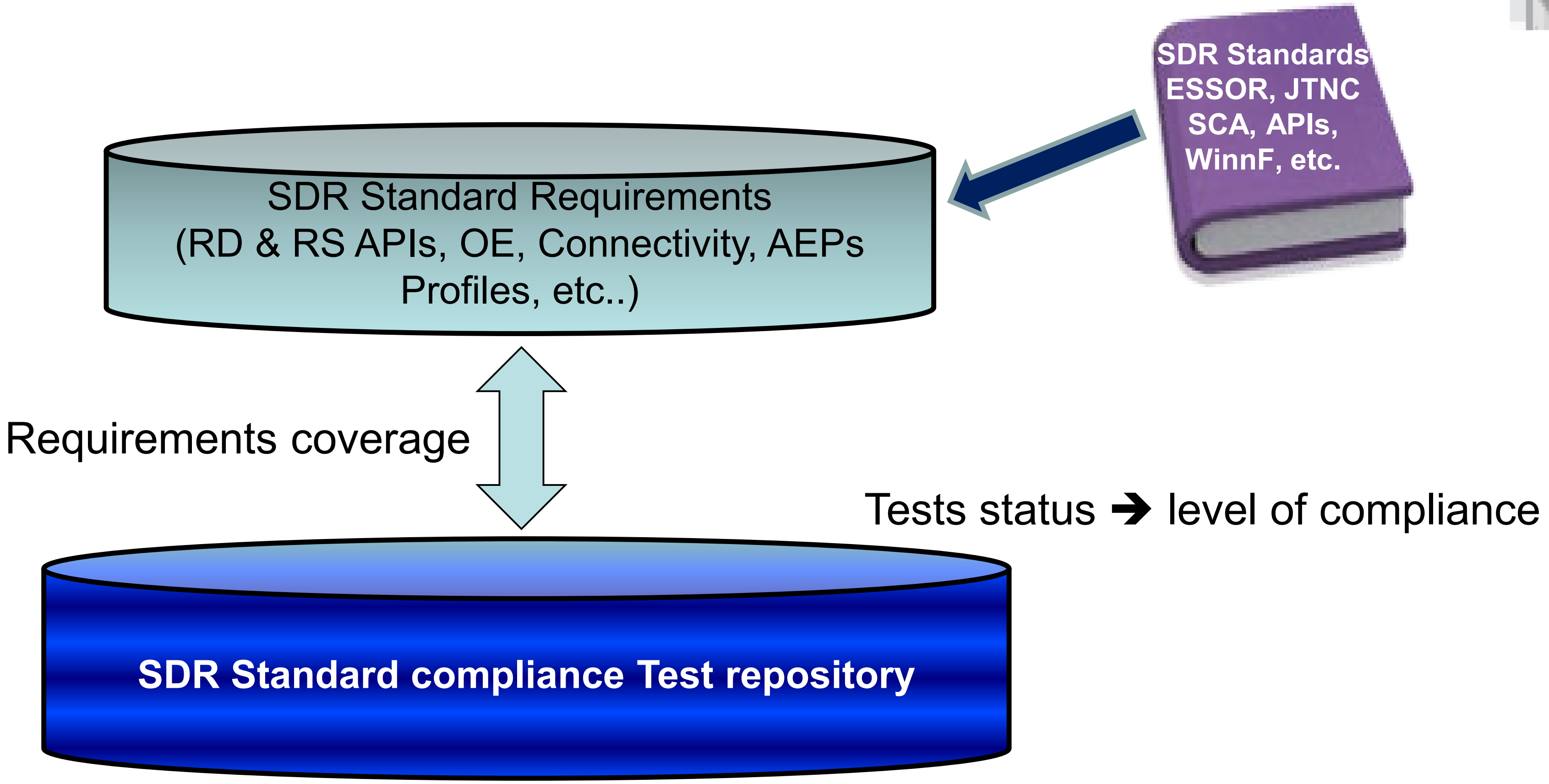
- Non conformity detection
  - Not Implemented Interface
  - Wrong interface
  - Non conform behavior
  - Non conform data processing
  - Test of boundaries values

- Conclusion / Q&A

Testing methodology

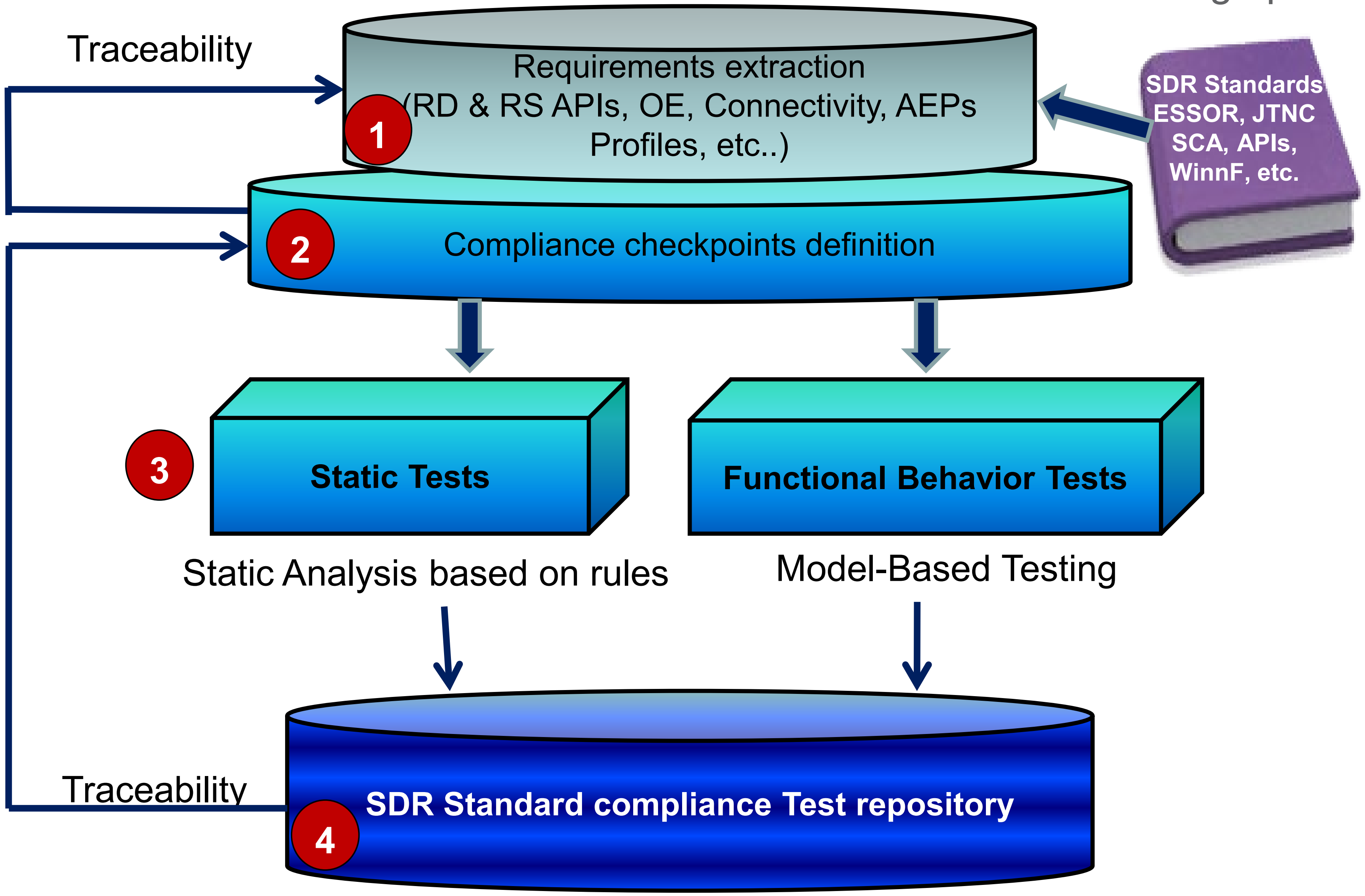
From the SDR requirements to the tests

- Why going from requirements to tests?


SDR Standards
ESSOR, JTNC
SCA, APIs,
WinnF, etc.
SDR Standard Requirements
(RD & RS APIs, OE, Connectivity, AEPs
Profiles, etc..)
Requirements coverage
Tests status ➔ level of compliance
SDR Standard compliance Test repository

Testing methodology

Test design process

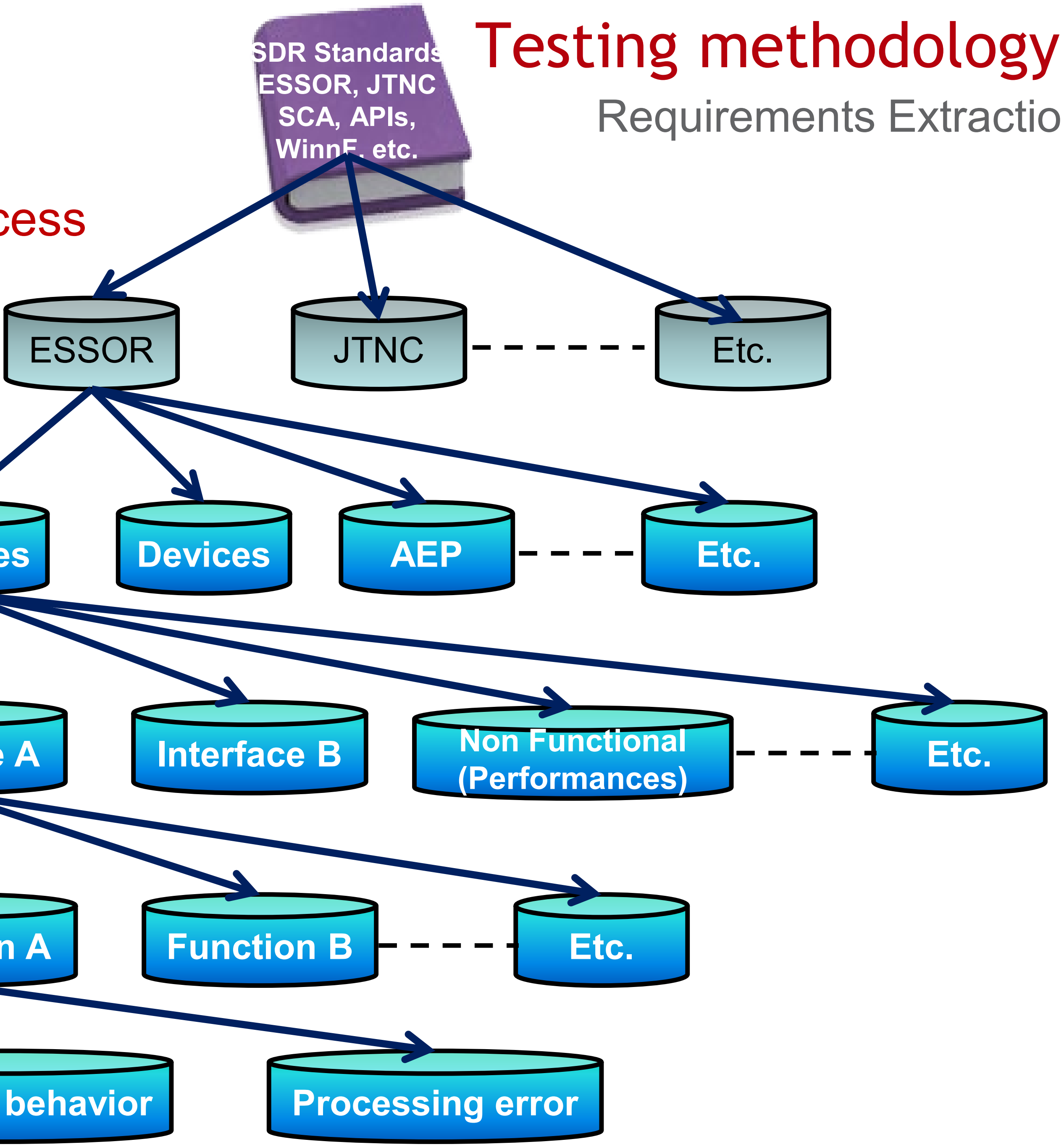

Traceability
Requirements extraction
(RD & RS APIs, OE, Connectivity, AEPs Profiles, etc..)
1
SDR Standards
ESSOR, JTNC
SCA, APIs,
WinnF, etc.
2
Compliance checkpoints definition
3
Static Tests
Functional Behavior Tests
Static Analysis based on rules
Model-Based Testing
Traceability
4
SDR Standard compliance Test repository

1

Testing methodology

Requirements Extraction

- The extraction process follows the good practices promoted by IREB

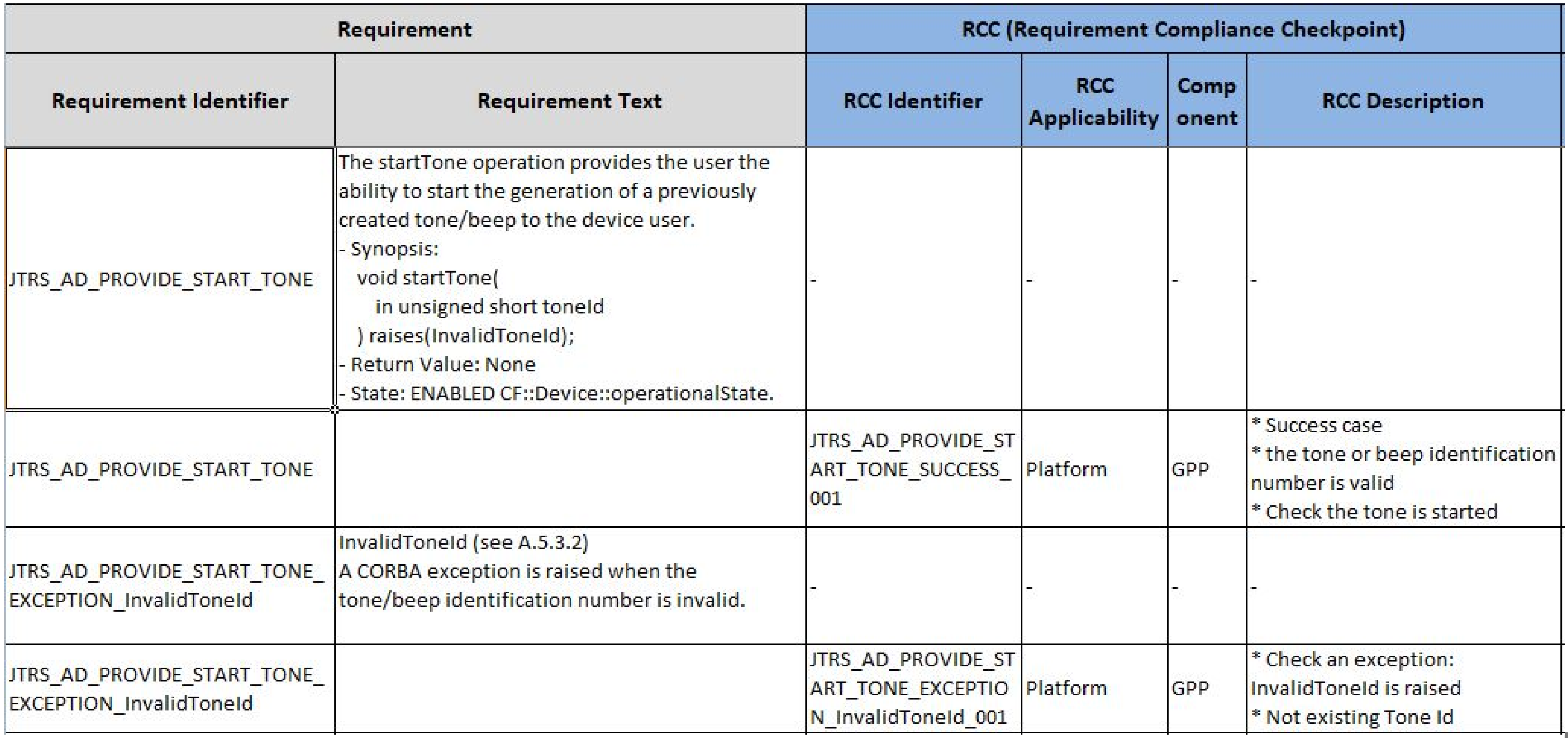


Testing methodology

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



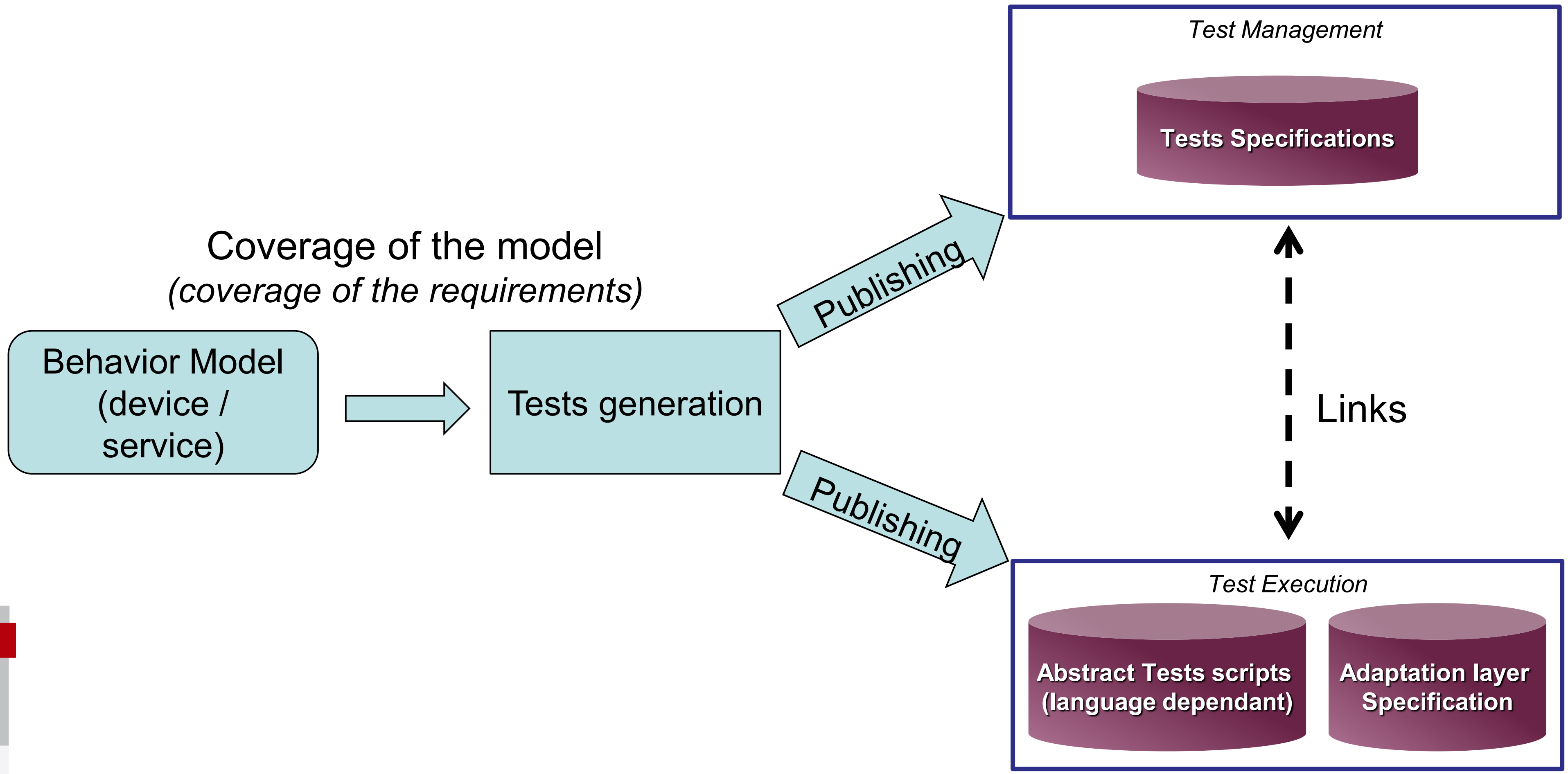
Requirement		RCC (Requirement Compliance Checkpoint)			
Requirement Identifier	Requirement Text	RCC Identifier	RCC Applicability	Component	RCC Description
JTRS_AD_PROVIDE_START_TONE	The startTone operation provides the user the ability to start the generation of a previously created tone/beep to the device user. - Synopsis: void startTone(in unsigned short toneId) raises(InvalidToneId); - Return Value: None - State: ENABLED CF::Device::operationalState.	-	-	-	-
JTRS_AD_PROVIDE_START_TONE		JTRS_AD_PROVIDE_START_TONE_SUCCESS_001	Platform	GPP	* Success case * the tone or beep identification number is valid * Check the tone is started
JTRS_AD_PROVIDE_START_TONE_EXCEPTION_InvalidToneId	InvalidToneId (see A.5.3.2) A CORBA exception is raised when the tone/beep identification number is invalid.	-	-	-	-
JTRS_AD_PROVIDE_START_TONE_EXCEPTION_InvalidToneId		JTRS_AD_PROVIDE_START_TONE_EXCEPTION_InvalidToneId_001	Platform	GPP	* Check an exception: InvalidToneId is raised * Not existing Tone Id

3

Testing methodology

Modeling

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

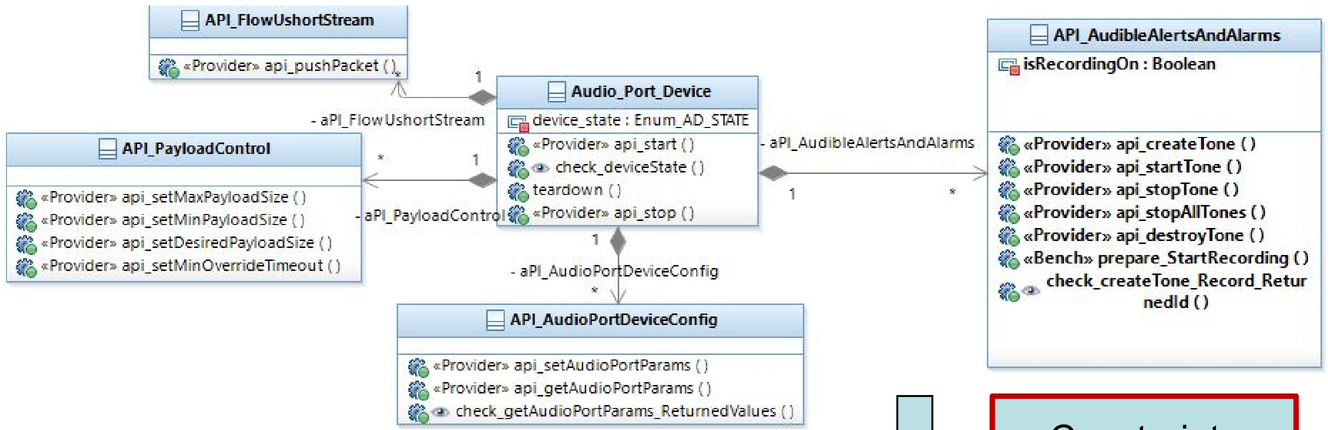

Coverage of the model
(coverage of the requirements)
Behavior Model (device / service)
Tests generation
Publishing
Publishing
Test Management
Tests Specifications
Links
Test Execution
Abstract Tests scripts (language dependant)
Adaptation layer Specification

Testing methodology

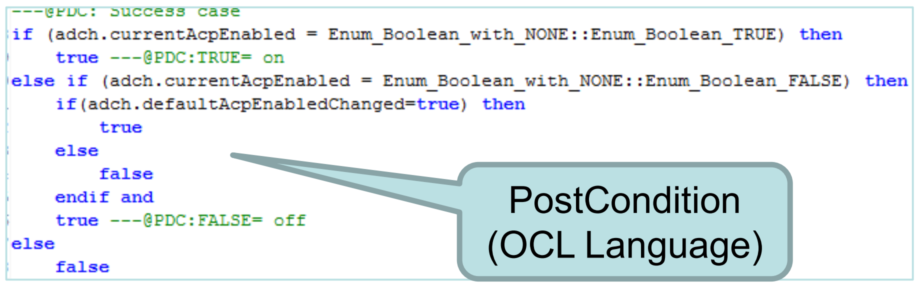
Modeling

3

UML Class diagrams design for abstract test implementation

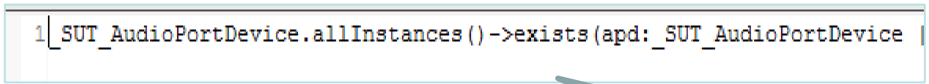

API_FlowUshortStream
«Provider» api_pushPacket ()
Audio_Port_Device
device_state : Enum_AD_STATE
«Provider» api_start ()
check_deviceState ()
teardown ()
«Provider» api_stop ()
- aPI_FlowUshortStream
API_PayloadControl
«Provider» api_setMaxPayloadSize ()
«Provider» api_setMinPayloadSize ()
«Provider» api_setDesiredPayloadSize ()
«Provider» api_setMinOverrideTimeout ()
- aPI_PayloadControl
- aPI_AudioPortDeviceConfig
API_AudioPortDeviceConfig
«Provider» api_setAudioPortParams ()
«Provider» api_getAudioPortParams ()
check_getAudioPortParams_ReturnedValues ()
- aPI_AudibleAlertsAndAlarms
API_AudibleAlertsAndAlarms
isRecordingOn : Boolean
«Provider» api_createTone ()
«Provider» api_startTone ()
«Provider» api_stopTone ()
«Provider» api_stopAllTones ()
«Provider» api_destroyTone ()
«Bench» prepare_StartRecording ()
check_createTone_Record_ReturnedId ()
1
*

Constraints expression



```
---@PDC: Success case
if (adch.currentAcpEnabled = Enum_Boolean_with_NONE::Enum_Boolean_TRUE) then
    true ---@PDC:TRUE= on
else if (adch.currentAcpEnabled = Enum_Boolean_with_NONE::Enum_Boolean_FALSE) then
    if(adch.defaultAcpEnabledChanged=true) then
        true
    else
        false
    endif and
    true ---@PDC:FALSE= off
else
    false
```

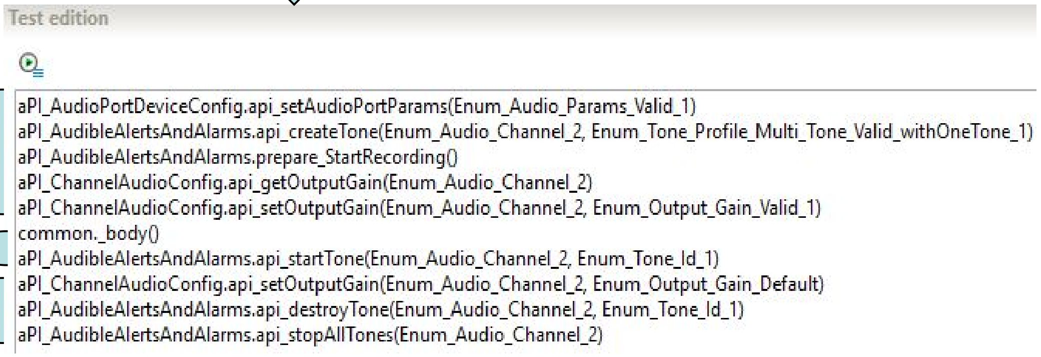
PostCondition (OCL Language)



```
SUT_AudioPortDevice.allInstances()->exists(apd:_SUT_AudioPortDevice
```

Precondition (OCL Language)

Automatic Tests Generation



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)
```

Functions to call on set up before the test body

Test body

Functions to call on Tear down (return to initial state)

4

Testing methodology

Test generation: Abstract Tests

- Example of C++ test with a start function of a radio Device
  - Each generated function is a single test step
  - Each test is an assembly of single steps

```
bool JTRS_AD_PROVIDE_API_START_1::setUp()
{
current_result = m_adapter->api_set_API_Params(<params>);
current_result = m_adapter->api_get_API_Params(<params>);
current_result = m_adapter->check_API_Params(<params>);
current_result = m_adapter->prepare_StartRecording(<params>);
current_result = m_adapter->api_create(<params>);
current_result = m_adapter->check_create_Record_ReturnedId(<params>);
return current_result;
}

bool JTRS_AD_PROVIDE_API_START_1::test()
{
current_result = m_adapter->api_start(<params>);
current_result = m_adapter->check_StatusForStarted(<params>);
return current_result;
}

bool JTRS_AD_PROVIDE_API_START_1::tearDown()
{
current_result = m_adapter->api_stopAll(<params>);
current_result = m_adapter->api_destroy(<params>);
current_result = m_adapter->api_set_API_Params(<default_params>);
current_result = m_adapter->api_get_API_Params(<default_params>);
current_result = m_adapter->check_API_Params(<default_params>);
current_result = m_adapter->bench_tearDown();
return current_result;
}
```

API — Call SUT interface

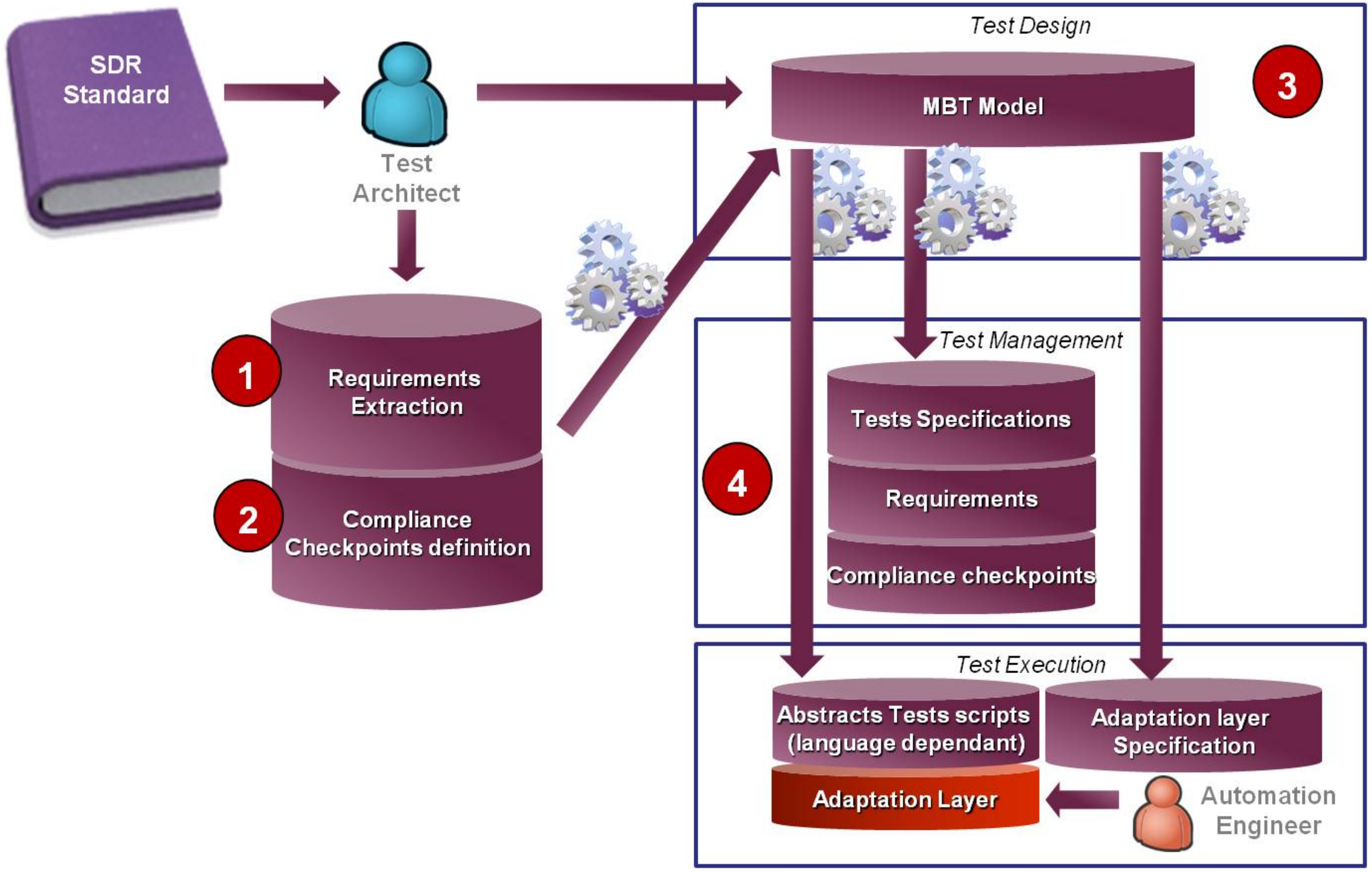
prepare — Prepare measurement tools

Check — Compare received value with expected value

Bench — Specifications on Test Bench

Testing methodology

Test design process summary



Testing methodology

Advantages

- Behavior modeling strategy provides us
  - Independence of the model from the target.
  - A complete coverage of the behavior.
  - An easier maintenance and easier rework.
  - A Definition of conformance criteria independently from the test definition itself.
- Abstract Tests could be exported into different programming languages
  - C/C++, JAVA, python, etc ...
- All needed Tests artifact could be exported into different formats
  - Database export (test management software), XML files, Excel files, etc.

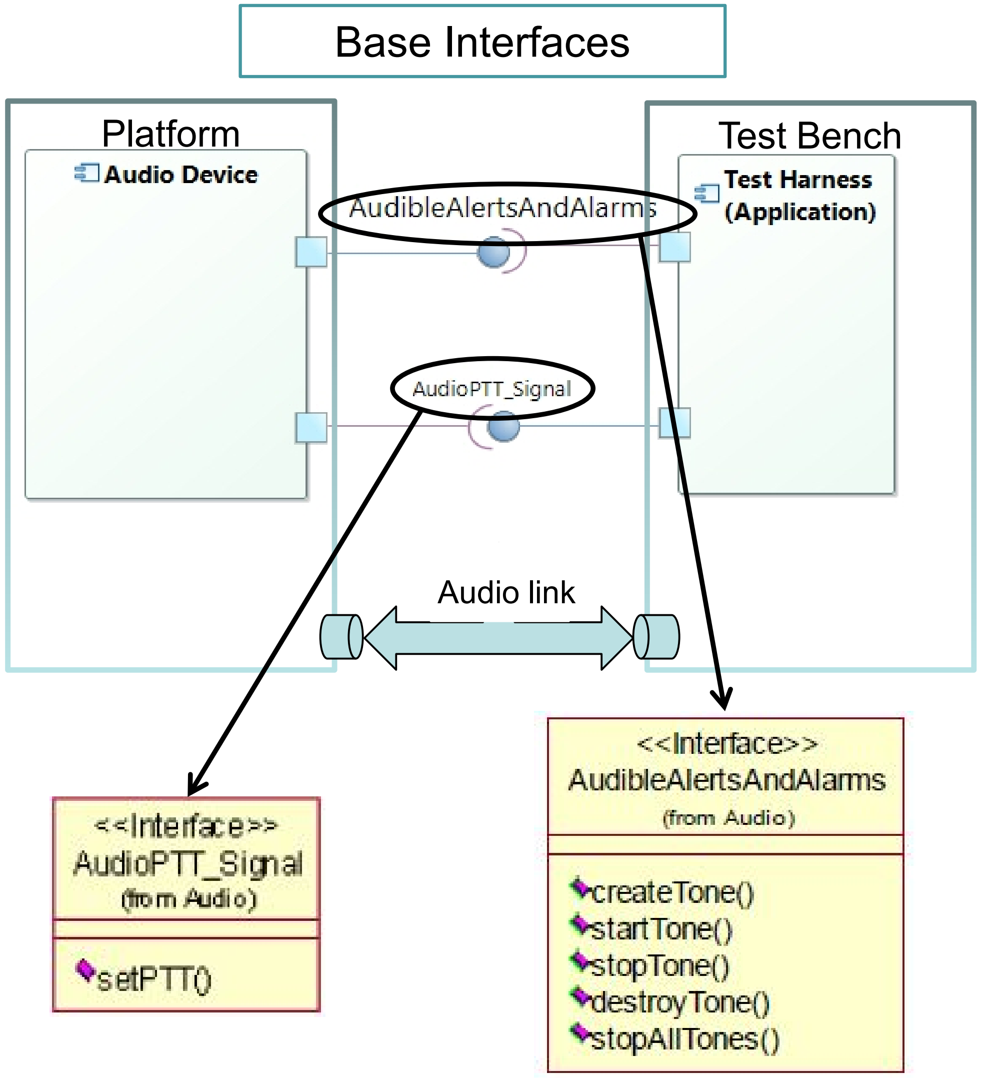
Agenda

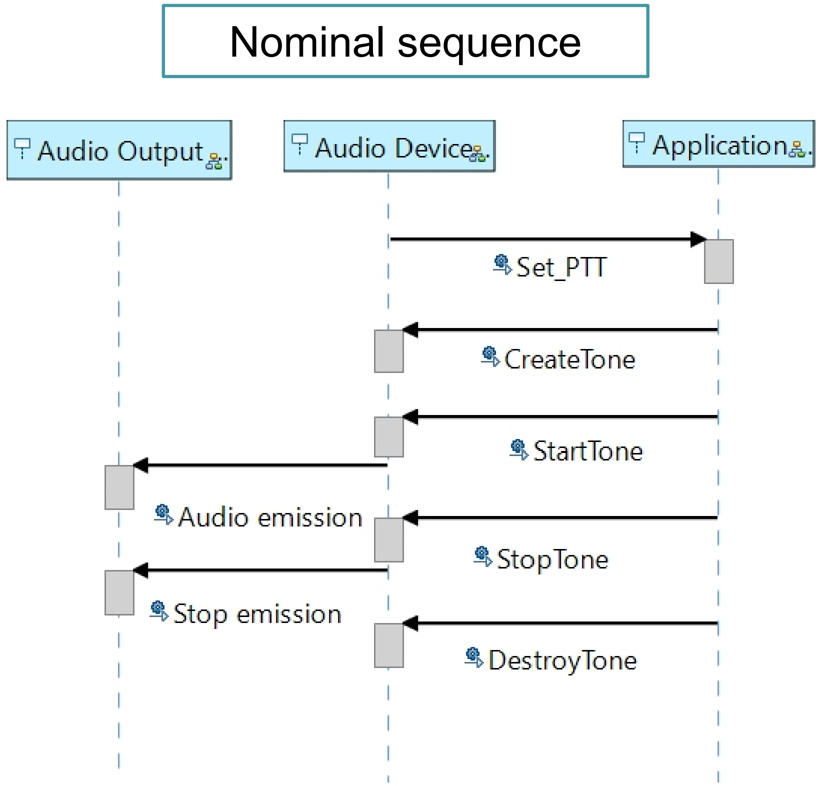
- SDR conformance assessment: the needs
- Testing methodology
  - Test design process
  - From the SDR requirements to the tests
  - Compliance checkpoints definition
  - Modeling
  - Testing generation
- Non conformity detection
  - Not Implemented Interface
  - Wrong interface
  - Non conform behavior
  - Non conform data processing
  - Test of boundaries values
- Conclusion / Q&A

Non conformity detection

Audio Device description

Audio Device Example



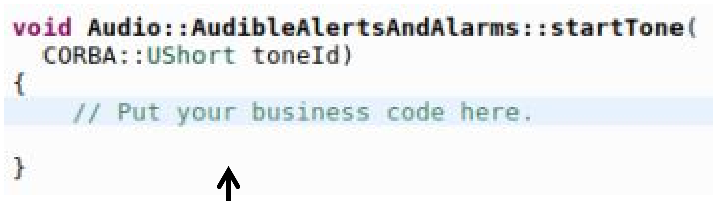


Non conformity detection

Not Implemented Interface

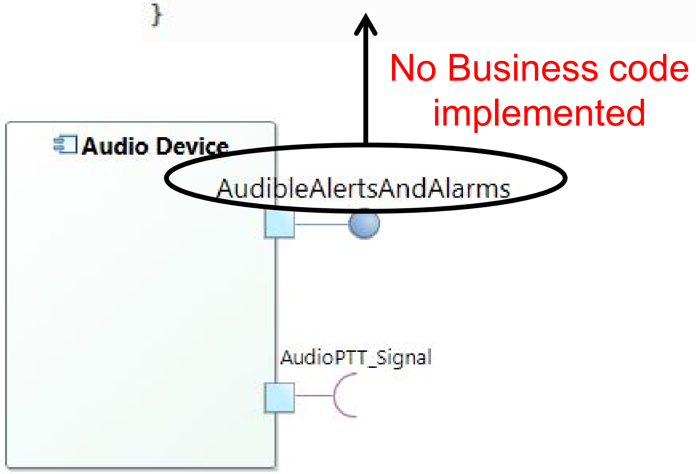
Non conformity description

- The Port is functional but the behavior required is not implemented



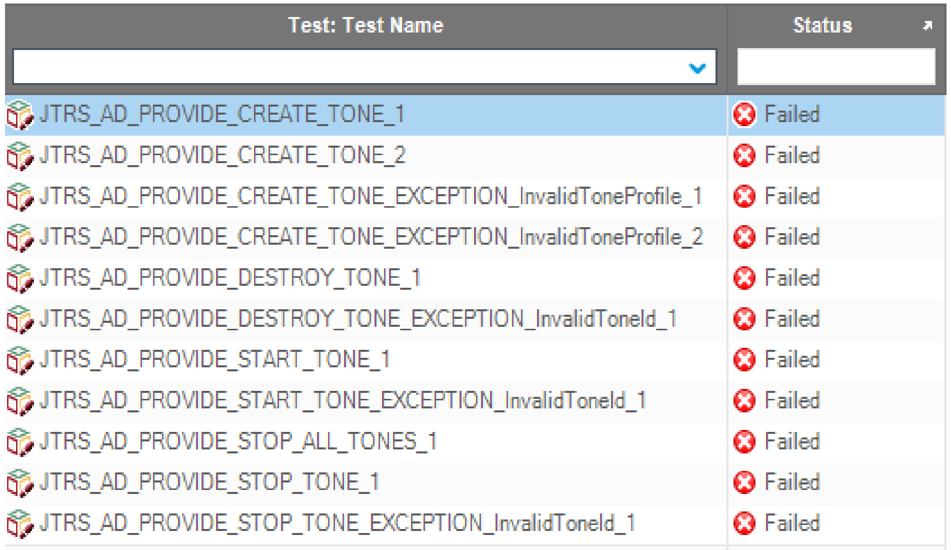
```
void Audio::AudibleAlertsAndAlarms::startTone(
  CORBA::UShort toneId)
{
    // Put your business code here.

}
```



Results

- All tests failed on the interface



Test: Test Name	Status
JTRS_AD_PROVIDE_CREATE_TONE_1	Failed
JTRS_AD_PROVIDE_CREATE_TONE_2	Failed
JTRS_AD_PROVIDE_CREATE_TONE_EXCEPTION_InvalidToneProfile_1	Failed
JTRS_AD_PROVIDE_CREATE_TONE_EXCEPTION_InvalidToneProfile_2	Failed
JTRS_AD_PROVIDE_DESTROY_TONE_1	Failed
JTRS_AD_PROVIDE_DESTROY_TONE_EXCEPTION_InvalidToneId_1	Failed
JTRS_AD_PROVIDE_START_TONE_1	Failed
JTRS_AD_PROVIDE_START_TONE_EXCEPTION_InvalidToneId_1	Failed
JTRS_AD_PROVIDE_STOP_ALL_TONES_1	Failed
JTRS_AD_PROVIDE_STOP_TONE_1	Failed
JTRS_AD_PROVIDE_STOP_TONE_EXCEPTION_InvalidToneId_1	Failed

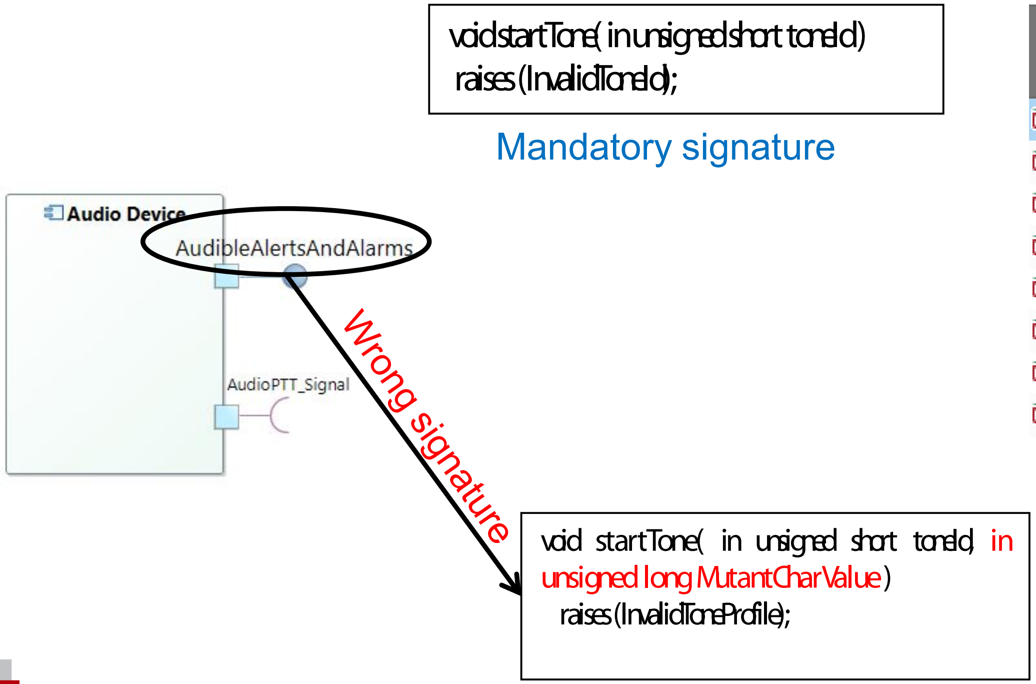
Lesson learnt

- The bench is able to detect empty implementation.

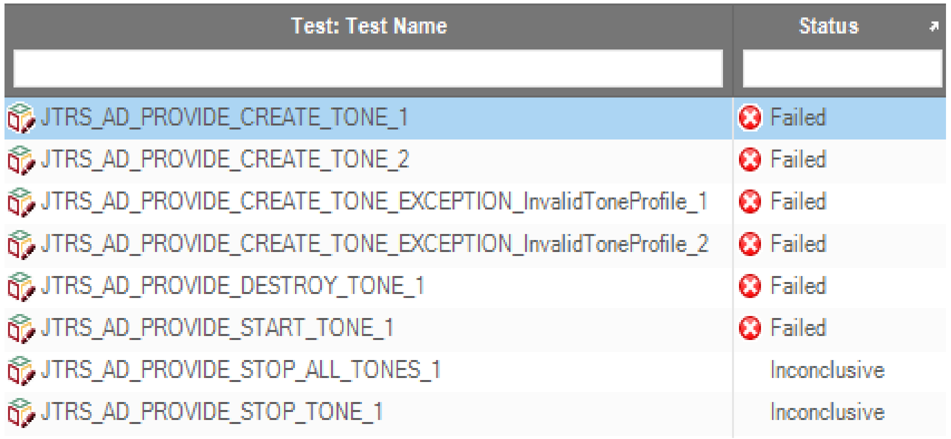
Non conformity detection

Wrong interface

- Non conformity description
  - The function *destroyTone* is not available
  - The signature of the function *startTone* is wrong
  - The signature of the exception *InvalidToneProfile* is wrong


void startTone(in unsigned short toneId) raises (InvalidToneId);
Mandatory signature
Audio Device
AudibleAlertsAndAlarms
AudioPTT_Signal
Wrong signature
void startTone(in unsigned short toneId, in unsigned long MutantCharValue) raises (InvalidToneProfile);

- Results
  - All tests failed or are inconclusive on the interface



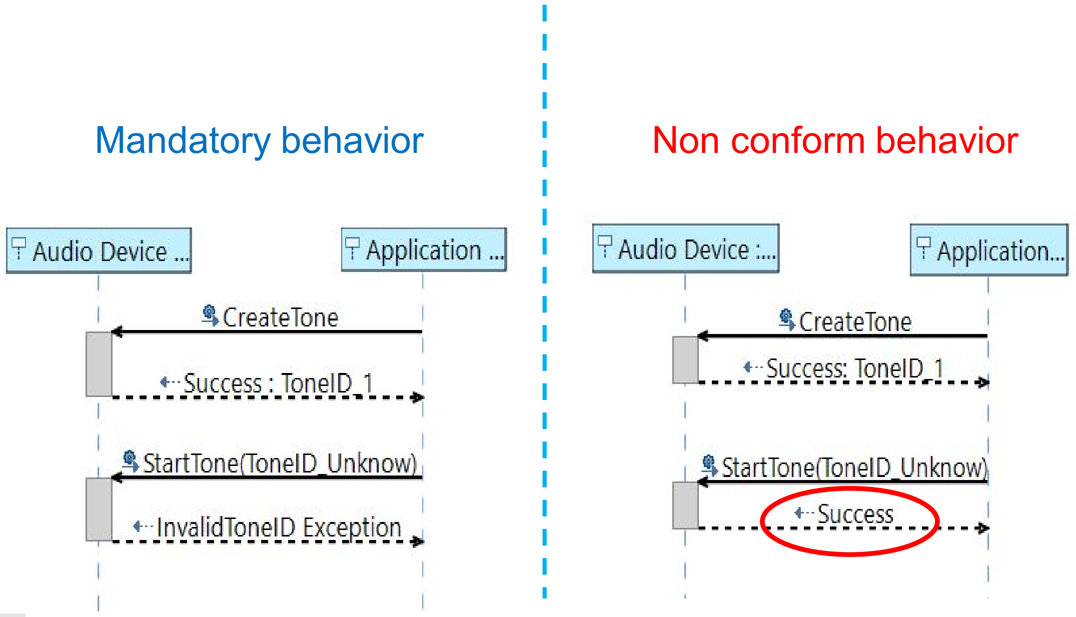
Test: Test Name	Status
JTRS_AD_PROVIDE_CREATE_TONE_1	Failed
JTRS_AD_PROVIDE_CREATE_TONE_2	Failed
JTRS_AD_PROVIDE_CREATE_TONE_EXCEPTION_InvalidToneProfile_1	Failed
JTRS_AD_PROVIDE_CREATE_TONE_EXCEPTION_InvalidToneProfile_2	Failed
JTRS_AD_PROVIDE_DESTROY_TONE_1	Failed
JTRS_AD_PROVIDE_START_TONE_1	Failed
JTRS_AD_PROVIDE_STOP_ALL_TONES_1	Inconclusive
JTRS_AD_PROVIDE_STOP_TONE_1	Inconclusive

- Lesson learnt
  - This example highlights the capacity of the bench to detect bad implementation of the interfaces defined in the standard.

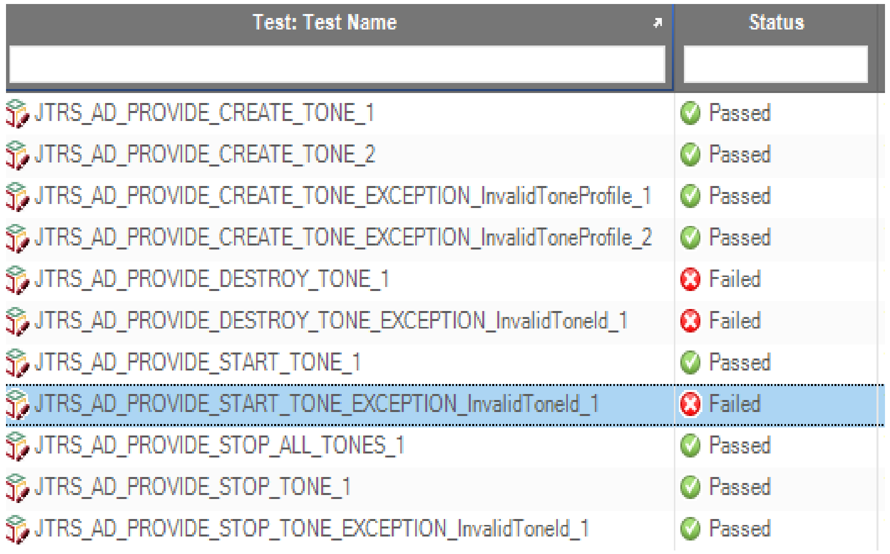
Non conformity detection

Non conform behavior

- Non conformity description
  - The function *DestroyTone* do not delete the tone designated by the tone ID
  - The function *startTone* do not raised an exception on unknown tone ID



- Results
  - Three tests on the interface failed



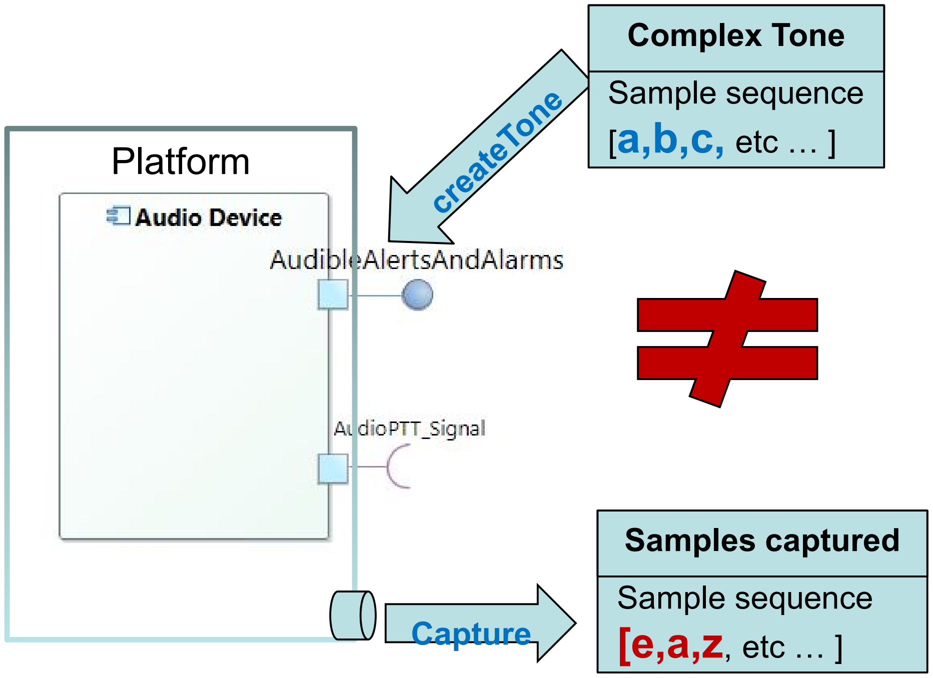
Test: Test Name	Status
JTRS_AD_PROVIDE_CREATE_TONE_1	Passed
JTRS_AD_PROVIDE_CREATE_TONE_2	Passed
JTRS_AD_PROVIDE_CREATE_TONE_EXCEPTION_InvalidToneProfile_1	Passed
JTRS_AD_PROVIDE_CREATE_TONE_EXCEPTION_InvalidToneProfile_2	Passed
JTRS_AD_PROVIDE_DESTROY_TONE_1	Failed
JTRS_AD_PROVIDE_DESTROY_TONE_EXCEPTION_InvalidToneId_1	Failed
JTRS_AD_PROVIDE_START_TONE_1	Passed
JTRS_AD_PROVIDE_START_TONE_EXCEPTION_InvalidToneId_1	Failed
JTRS_AD_PROVIDE_STOP_ALL_TONES_1	Passed
JTRS_AD_PROVIDE_STOP_TONE_1	Passed
JTRS_AD_PROVIDE_STOP_TONE_EXCEPTION_InvalidToneId_1	Passed

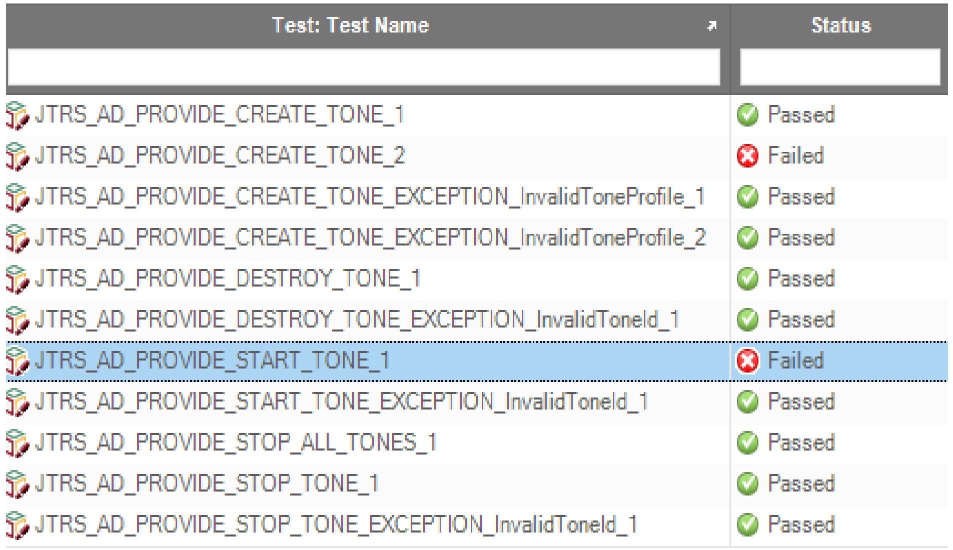
- Lesson learnt
  - This example shows the advantage of the behavior modeling applied to the tests to detect tricky defects

Non conformity detection

Non conform data processing

- Non conformity description
  - The tone emitted by the audio device do not match the values sent in the tone sample sequence.
- Results
  - One test on the StartTone function et another on the createTone function failed





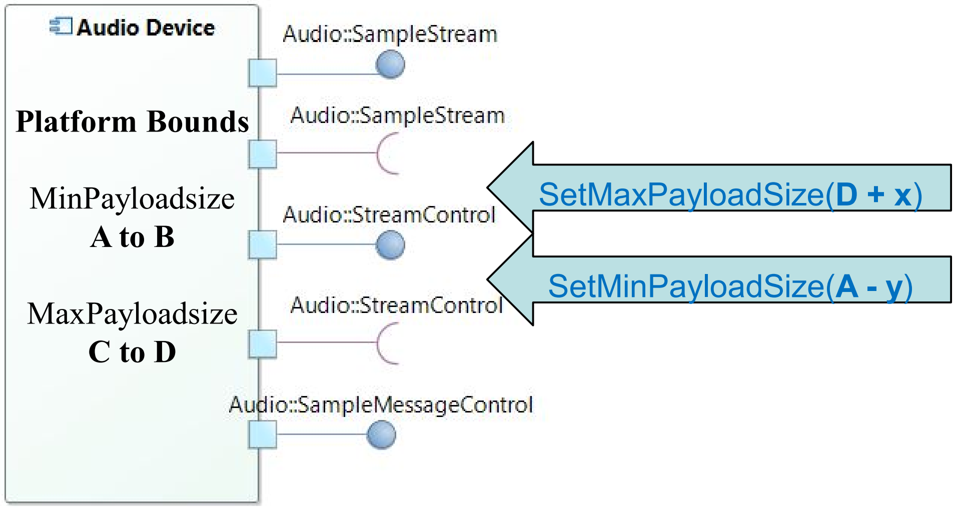
Test: Test Name	Status
JTRS_AD_PROVIDE_CREATE_TONE_1	Passed
JTRS_AD_PROVIDE_CREATE_TONE_2	Failed
JTRS_AD_PROVIDE_CREATE_TONE_EXCEPTION_InvalidToneProfile_1	Passed
JTRS_AD_PROVIDE_CREATE_TONE_EXCEPTION_InvalidToneProfile_2	Passed
JTRS_AD_PROVIDE_DESTROY_TONE_1	Passed
JTRS_AD_PROVIDE_DESTROY_TONE_EXCEPTION_InvalidToneId_1	Passed
JTRS_AD_PROVIDE_START_TONE_1	Failed
JTRS_AD_PROVIDE_START_TONE_EXCEPTION_InvalidToneId_1	Passed
JTRS_AD_PROVIDE_STOP_ALL_TONES_1	Passed
JTRS_AD_PROVIDE_STOP_TONE_1	Passed
JTRS_AD_PROVIDE_STOP_TONE_EXCEPTION_InvalidToneId_1	Passed

- Lesson learnt
  - This kind of non-conformity is more difficult to detect and could be interpreted as a performance test instead of functional test.
  - However the use of measurement tools and the check of data processing is clearly a good way for detecting functional defects.

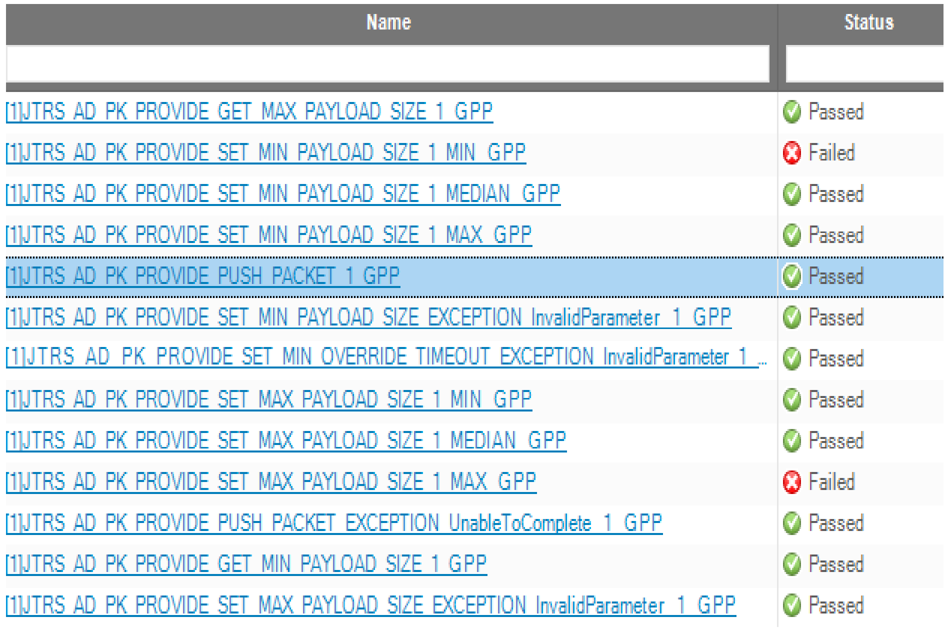
Non conformity detection

Test of boundaries values

- **Non conformity description**
  - The bounds of the Payload size defined in the Audio Sample Stream Extension are not compliant either in the SDR specification or in the Datasheet of the platform



- **Results**
  - The tests of lower bound of Set_Min_Payload_size and the upper bound of Set_Max_Payload_size failed



Name	Status
[1]JTRS_AD_PK_PROVIDE_GET_MAX_PAYLOAD_SIZE_1_GPP	✓ Passed
[1]JTRS_AD_PK_PROVIDE_SET_MIN_PAYLOAD_SIZE_1_MIN_GPP	✗ Failed
[1]JTRS_AD_PK_PROVIDE_SET_MIN_PAYLOAD_SIZE_1_MEDIAN_GPP	✓ Passed
[1]JTRS_AD_PK_PROVIDE_SET_MIN_PAYLOAD_SIZE_1_MAX_GPP	✓ Passed
[1]JTRS_AD_PK_PROVIDE_PUSH_PACKET_1_GPP	✓ Passed
[1]JTRS_AD_PK_PROVIDE_SET_MIN_PAYLOAD_SIZE_EXCEPTION_InvalidParameter_1_GPP	✓ Passed
[1]JTRS_AD_PK_PROVIDE_SET_MIN_OVERRIDE_TIMEOUT_EXCEPTION_InvalidParameter_1_...	✓ Passed
[1]JTRS_AD_PK_PROVIDE_SET_MAX_PAYLOAD_SIZE_1_MIN_GPP	✓ Passed
[1]JTRS_AD_PK_PROVIDE_SET_MAX_PAYLOAD_SIZE_1_MEDIAN_GPP	✓ Passed
[1]JTRS_AD_PK_PROVIDE_SET_MAX_PAYLOAD_SIZE_1_MAX_GPP	✗ Failed
[1]JTRS_AD_PK_PROVIDE_PUSH_PACKET_EXCEPTION_UnableToComplete_1_GPP	✓ Passed
[1]JTRS_AD_PK_PROVIDE_GET_MIN_PAYLOAD_SIZE_1_GPP	✓ Passed
[1]JTRS_AD_PK_PROVIDE_SET_MAX_PAYLOAD_SIZE_EXCEPTION_InvalidParameter_1_GPP	✓ Passed

- **Lesson learnt**
  - The boundaries values tests ensure:
    - The conformity to the SDR standard
    - The validity of the values provided by the manufacturer.

Agenda

- SDR conformance assessment: the needs
- Testing methodology
  - Test design process
  - From the SDR requirements to the tests
  - Compliance checkpoints definition
  - Modeling
  - Testing generation
- Non conformity detection
  - Not Implemented Interface
  - Wrong interface
  - Non conform behavior
  - Non conform data processing
  - Test of boundaries values
- Conclusion / Q&A

Conclusion

Results & Perspectives

- Results
  - Wide coverage of non conformities.
    - Not Implemented Interface
    - Wrong interface
    - Non conform behavior
    - Non conform data processing
    - Test of boundaries values
  - 86 % of the requirements extracted from the ESSOR Architecture covered
    - Remaining 14 % related to internal behaviors.
  - 96 % of the tests are fully automated.
  - Tests results and logs available in centralized database
- Perspectives
  - Performance tests under study
  - Continuous improvement due to capitalization on test execution
  - Evolution of the SDR standard

Questions