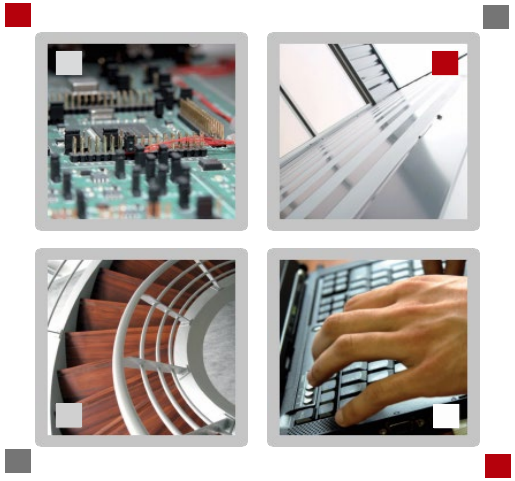
CERTIF: *a testing methodology and a test bench to assess the Compliance to software defined radio standards*



15 November 2018

Agenda

- SDR compliance assessment: the needs

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

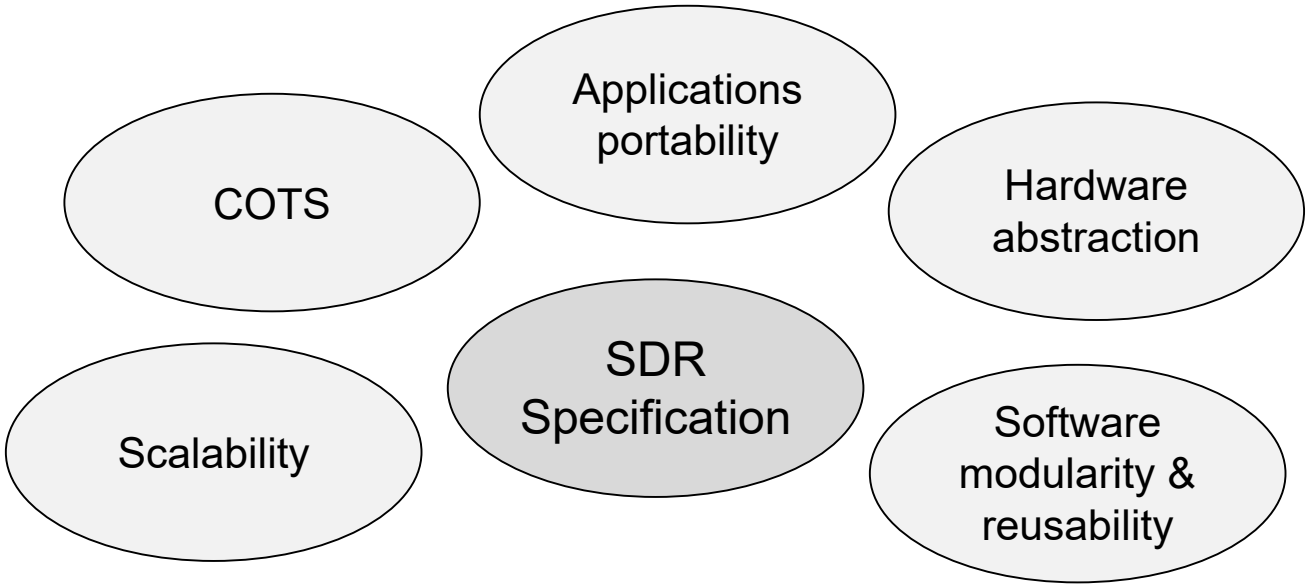
- The test bench

- Test of SDR components: an example

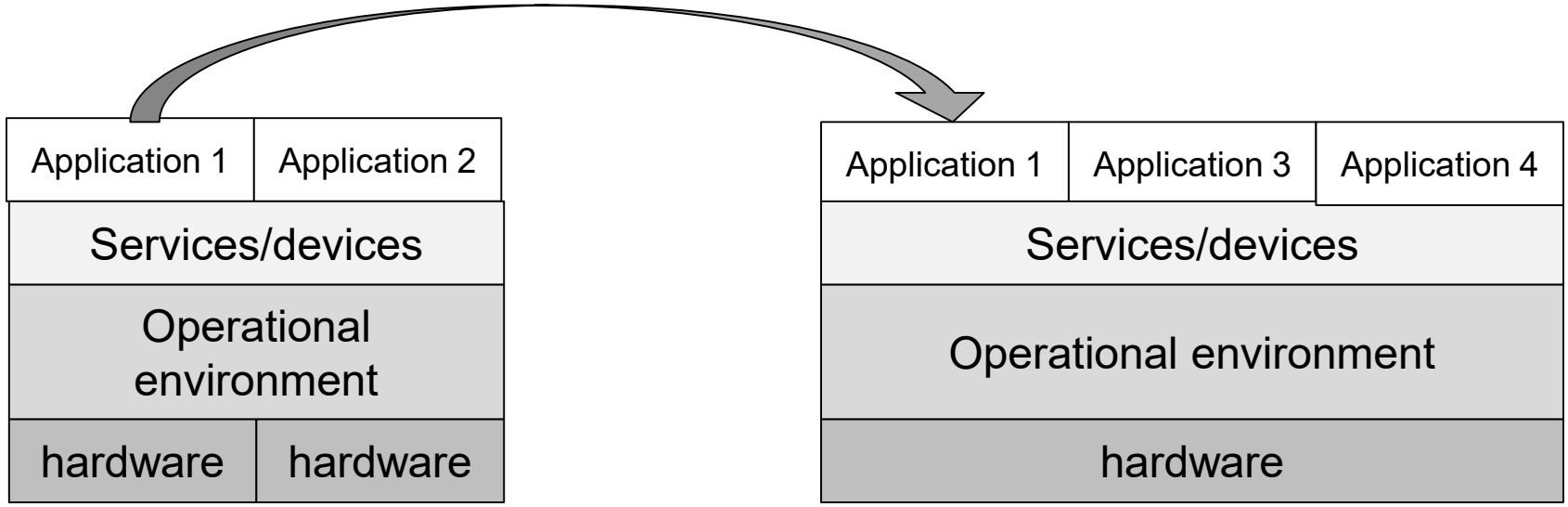
- Conclusion / Q&A

SDR Compliance Assessment

The needs



Needs to assess the compliance to these SDR standards



SDR Compliance Assessment

Assumptions

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

    The system under test is an equipment running under an Operational Environment including GPP and/or DSP and FPGA
  - The Applications

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

- 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.

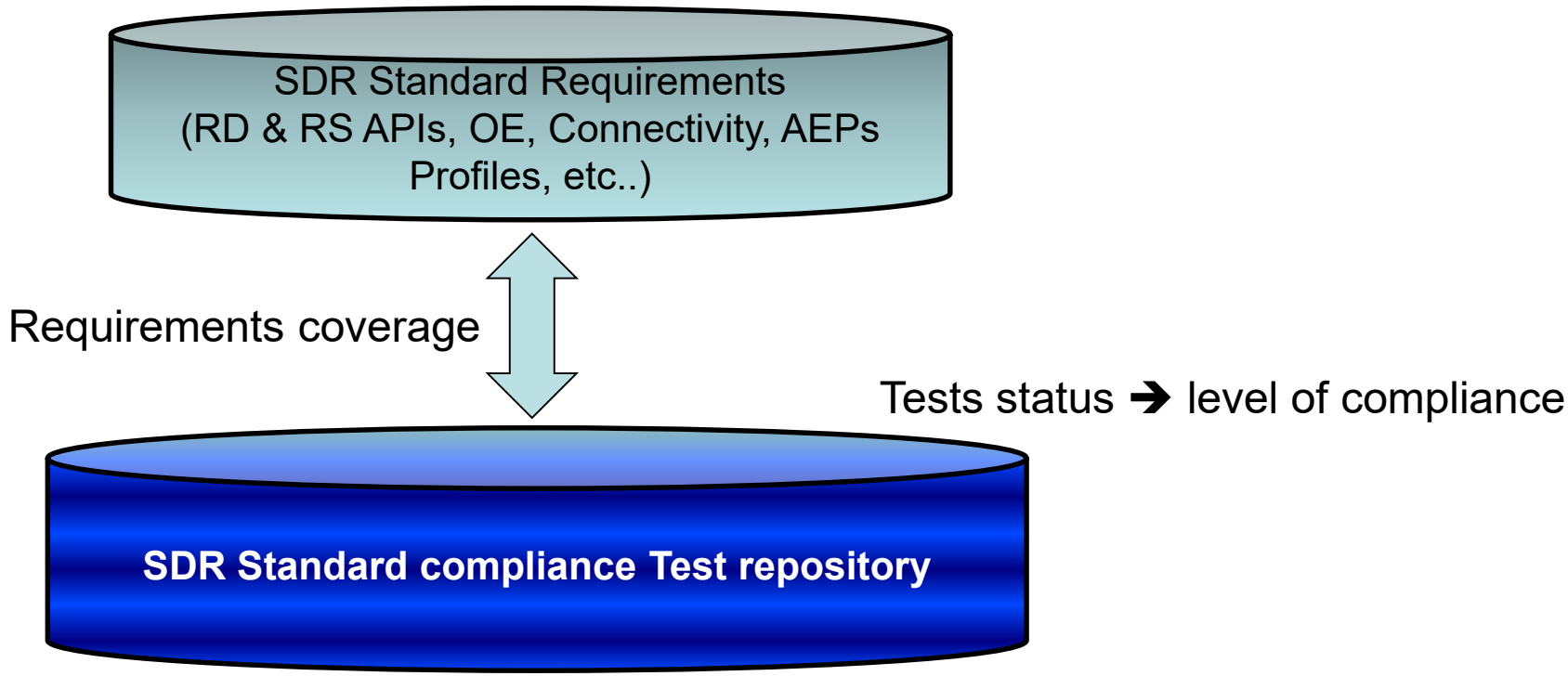
Agenda

- SDR conformance assessment: the needs
- Testing methodology
  - From the SDR requirements to the tests
  - Test design process
  - Compliance checkpoints definition
  - Modeling
  - Testing generation
- The test bench
- Test of SDR components: an example
- Conclusion / Q&A

Testing methodology

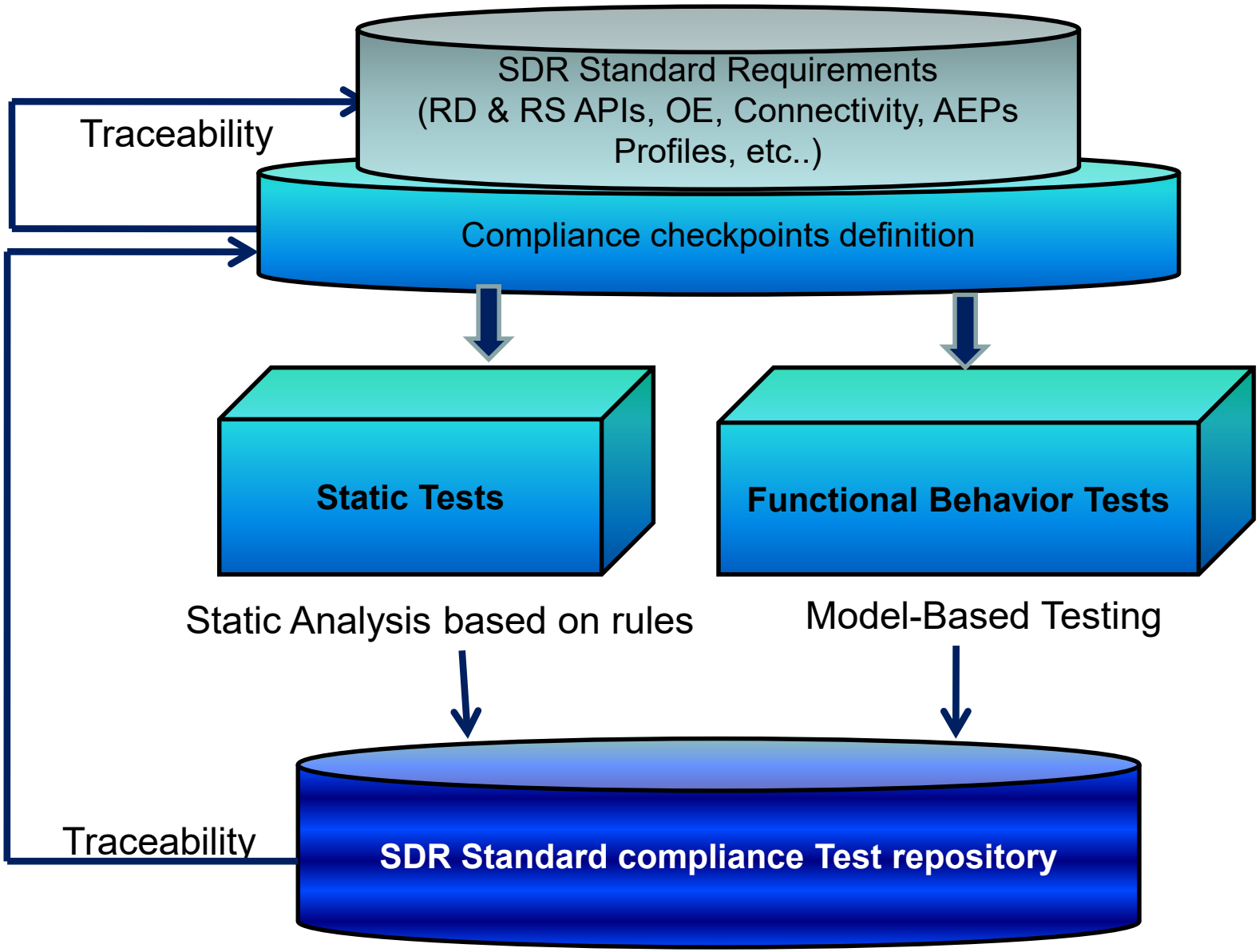
From the SDR requirements to the tests

- Why going from requirements to tests?



Test Strategy

Overview



Test Strategy

Static analysis design Process

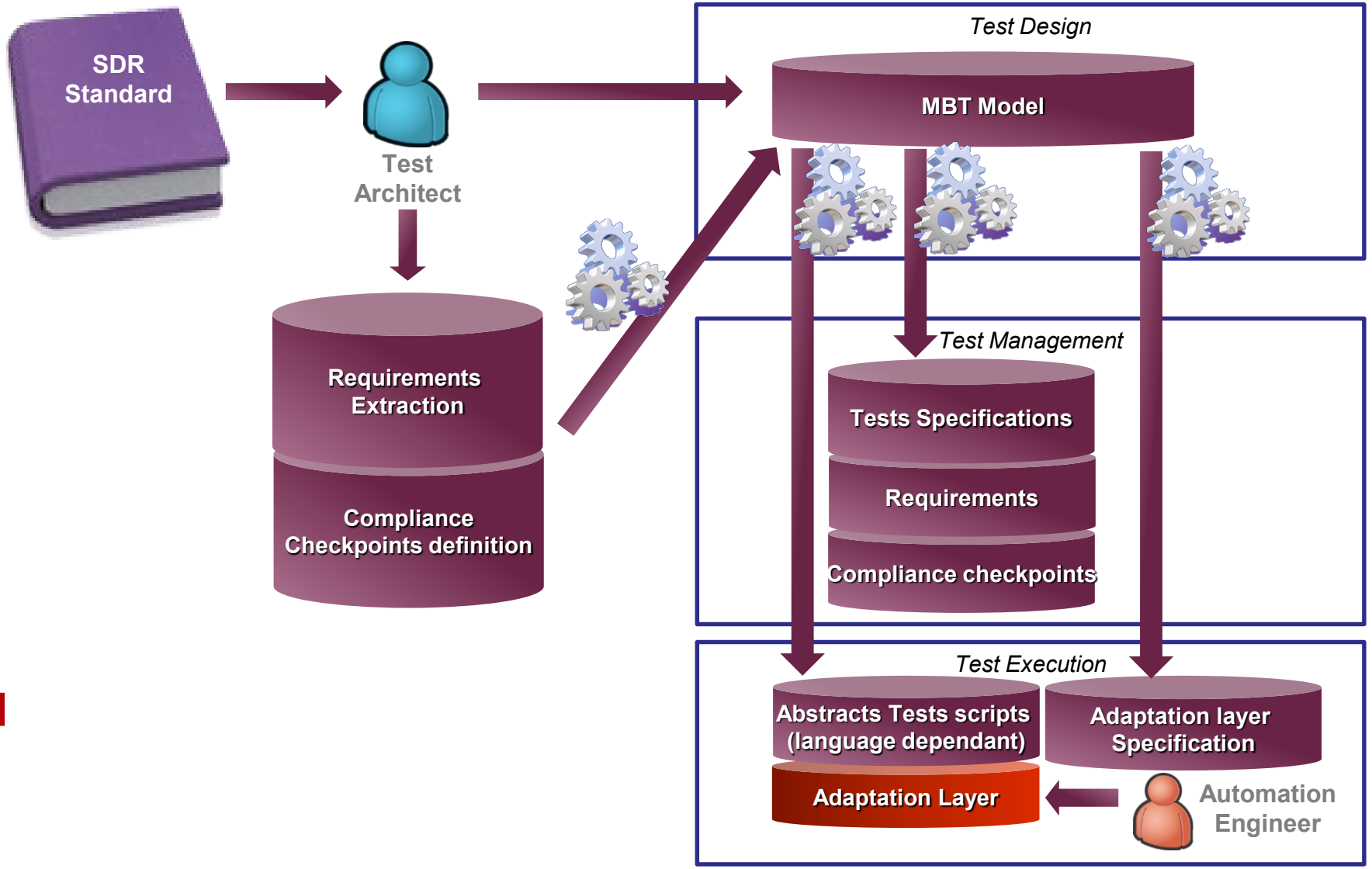
- Static analysis rules are based on source code language and provide/use functions

This test strategy is applied to the check of the compliance of the applications to the interfaces defined by the software radio standard

- Provide: The aim is to find an implementation of each function of the interface.
  - CORBA into C/C++
  - C/C++
  - VHDL
- Use: The aim is to find a call of each function of the interface.
  - CORBA into C/C++
  - C/C++
  - VHDL

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

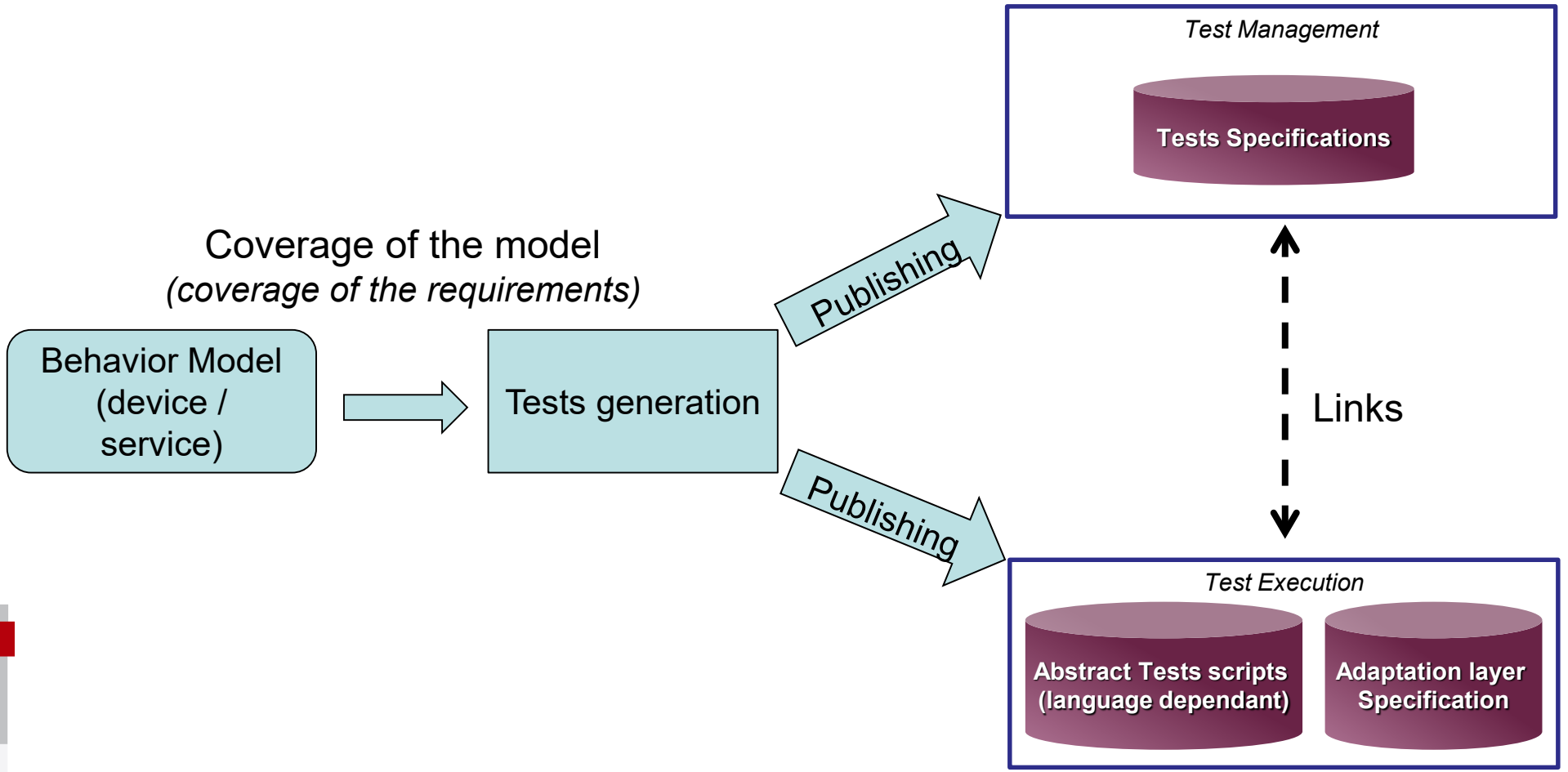
Sample on the startTone() function of Audio Device

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

Testing methodology

Modeling

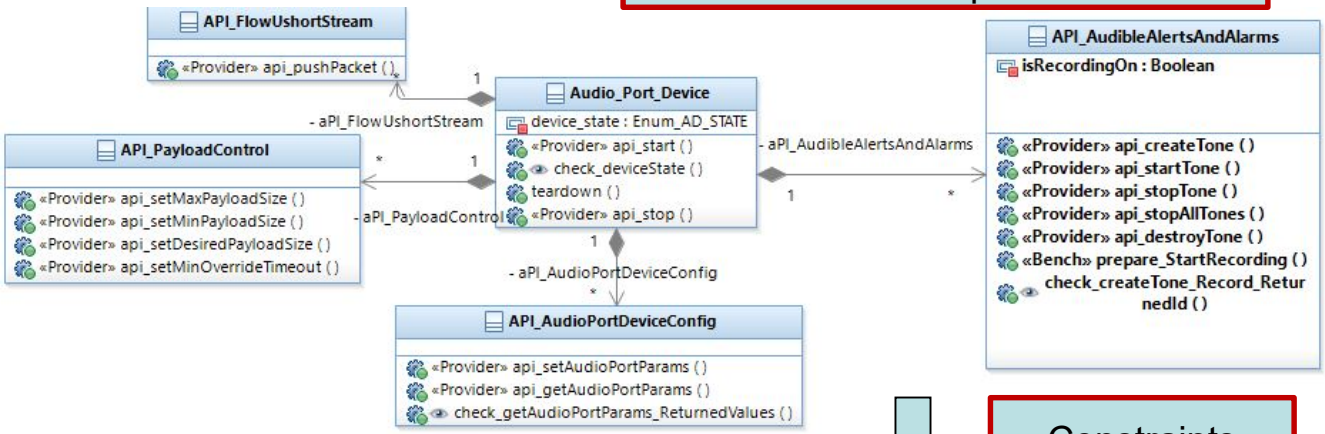
- Test design based on the behavior of the system under Test (Model Based Testing)



Test Strategy

Modeling

UML Class diagrams design for abstract test implementation



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)

```
1 _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)

Testing methodology

Test generation

- Constraints design in OCL language allows test generation based on component behavior
- Abstract Tests could be exported into different programming languages
  - C/C++, JAVA, etc ...
- Tests specification could be exported into different formats
  - Database export, XML files, etc ...

Test Strategy

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

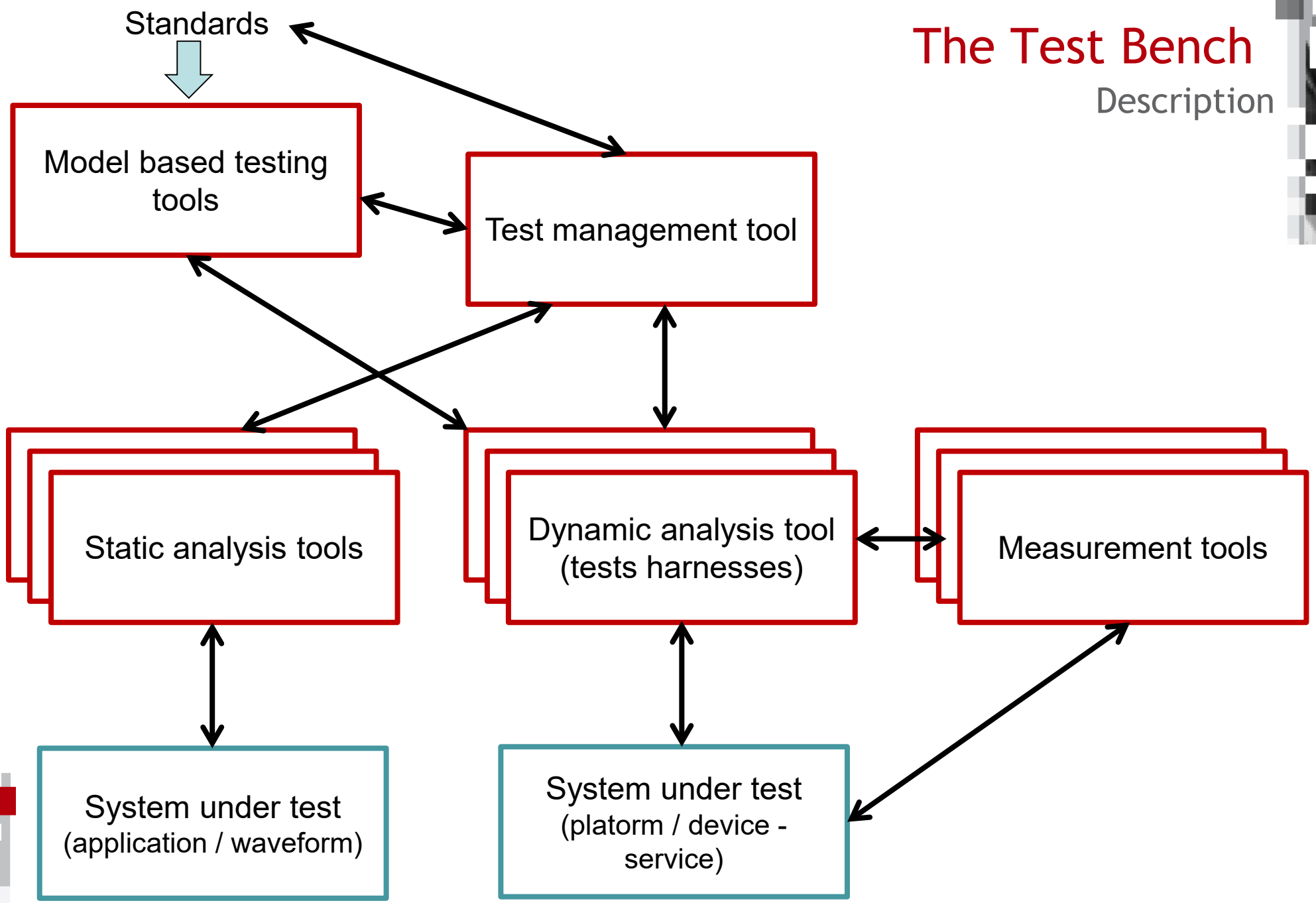
Bench — Specific actions on Test Bench

Agenda

- SDR compliance assessment: the needs
- Testing methodology
  - From the SDR requirements to the tests
  - Test design process
  - Compliance checkpoints definition
  - Modeling
  - Testing generation
- **The test bench**
- Test of SDR components: an example
- Conclusion / Q&A

The Test Bench

Description



The Test Bench

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 different systems under test

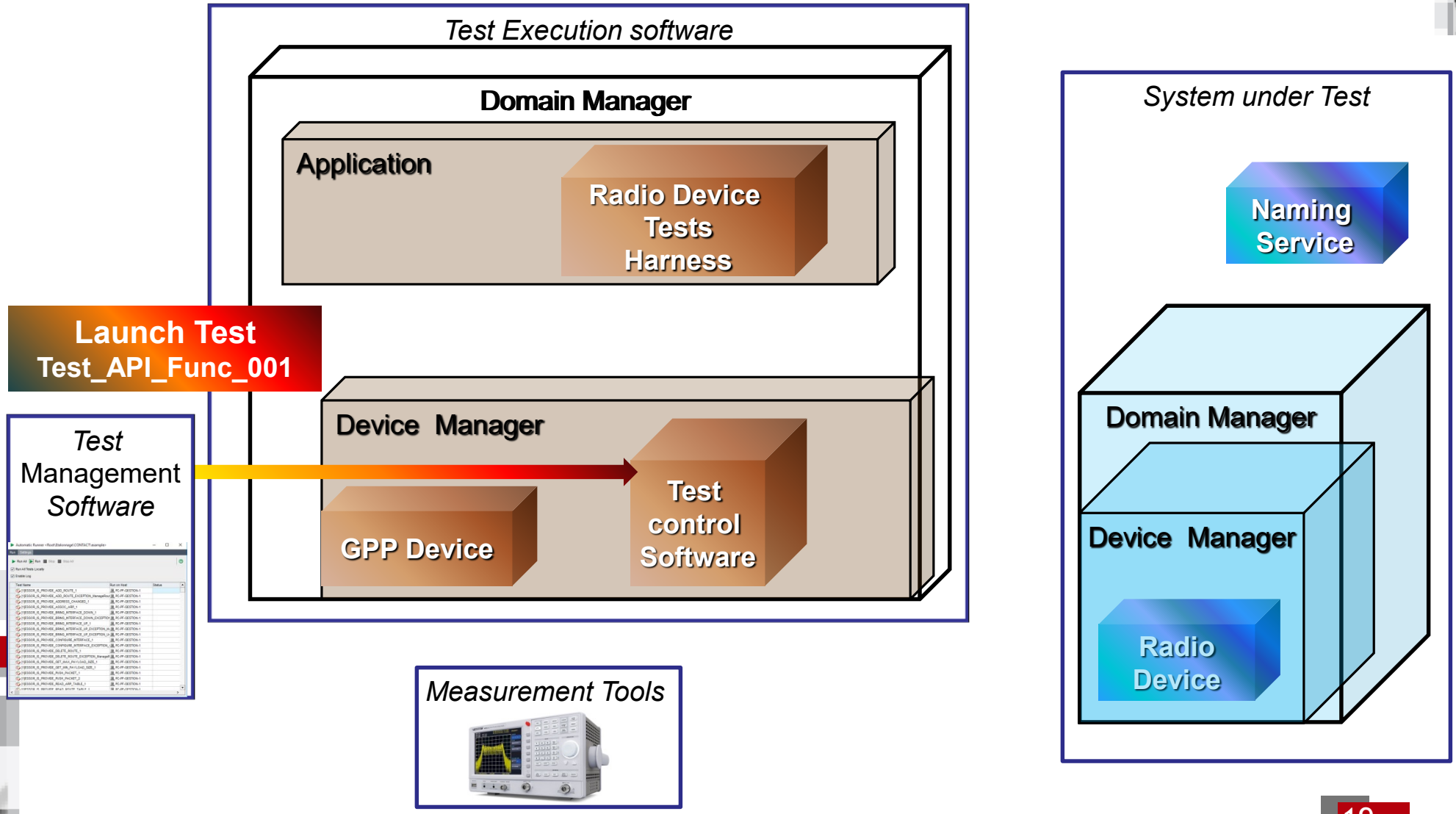
Agenda

- SDR compliance assessment: the needs
- Testing methodology
  - From the SDR requirements to the tests
  - Test design process
  - Compliance checkpoints definition
  - Modeling
  - Testing generation
- The test bench
- **Test of SDR components: an example**
- Conclusion / Q&A

Test of SDR components

Example of Radio Device API Platform testing

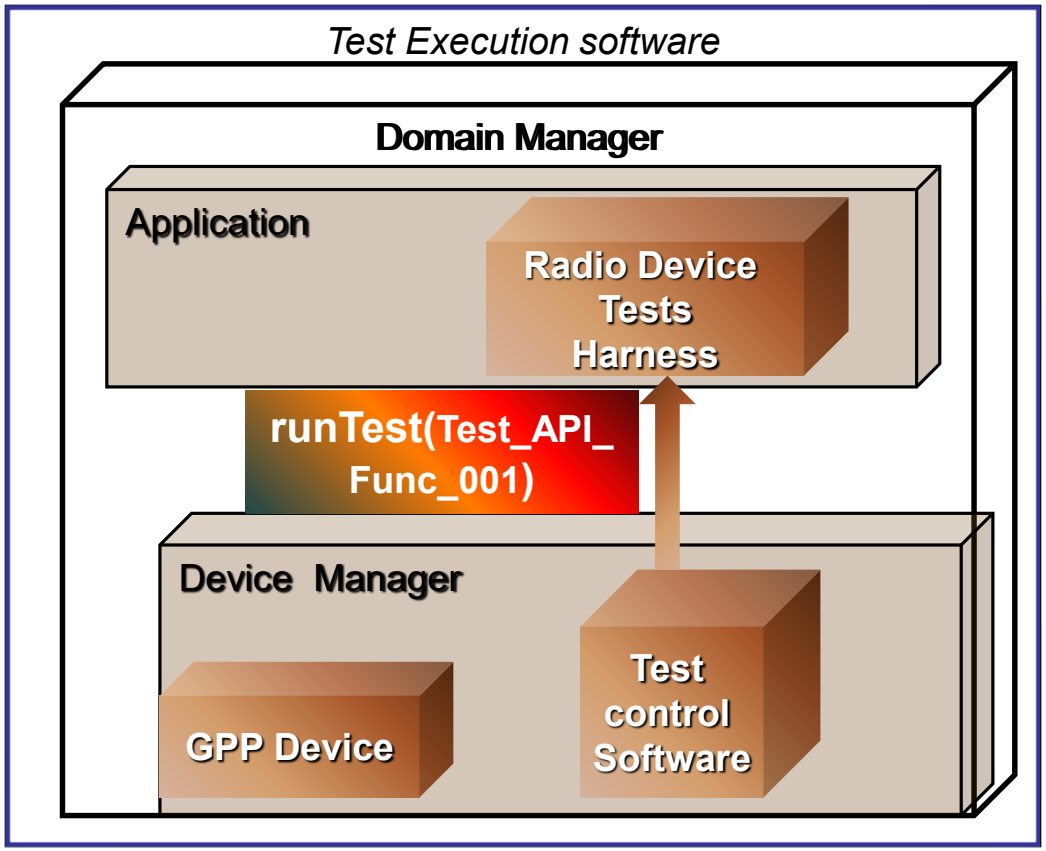
- Test request from Test management Software

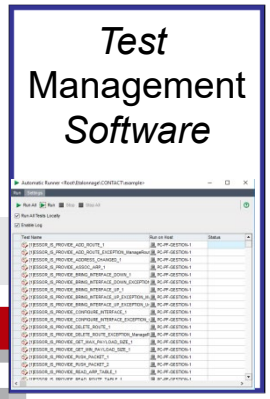


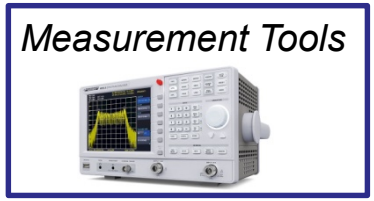
Test of SDR components

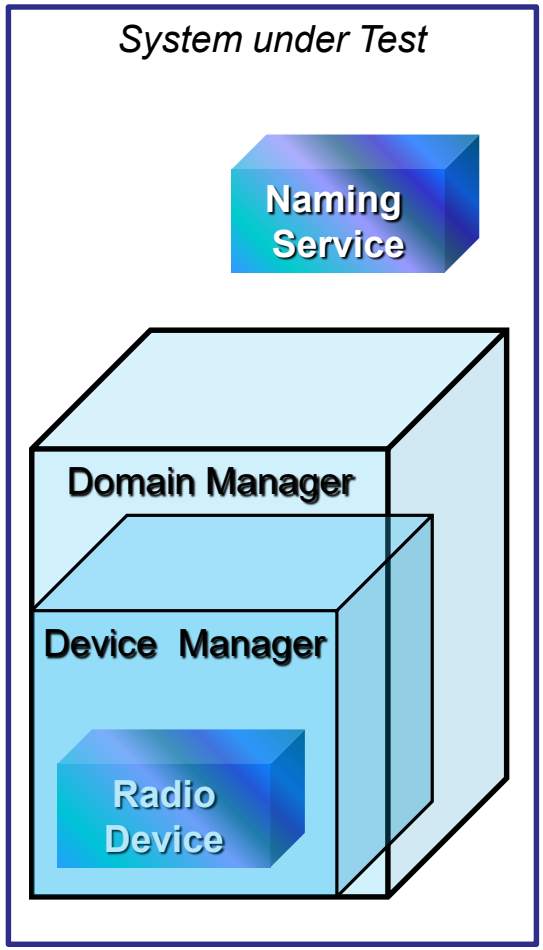
Example of Radio Device API Platform testing

- Test request on Radio Device Harness





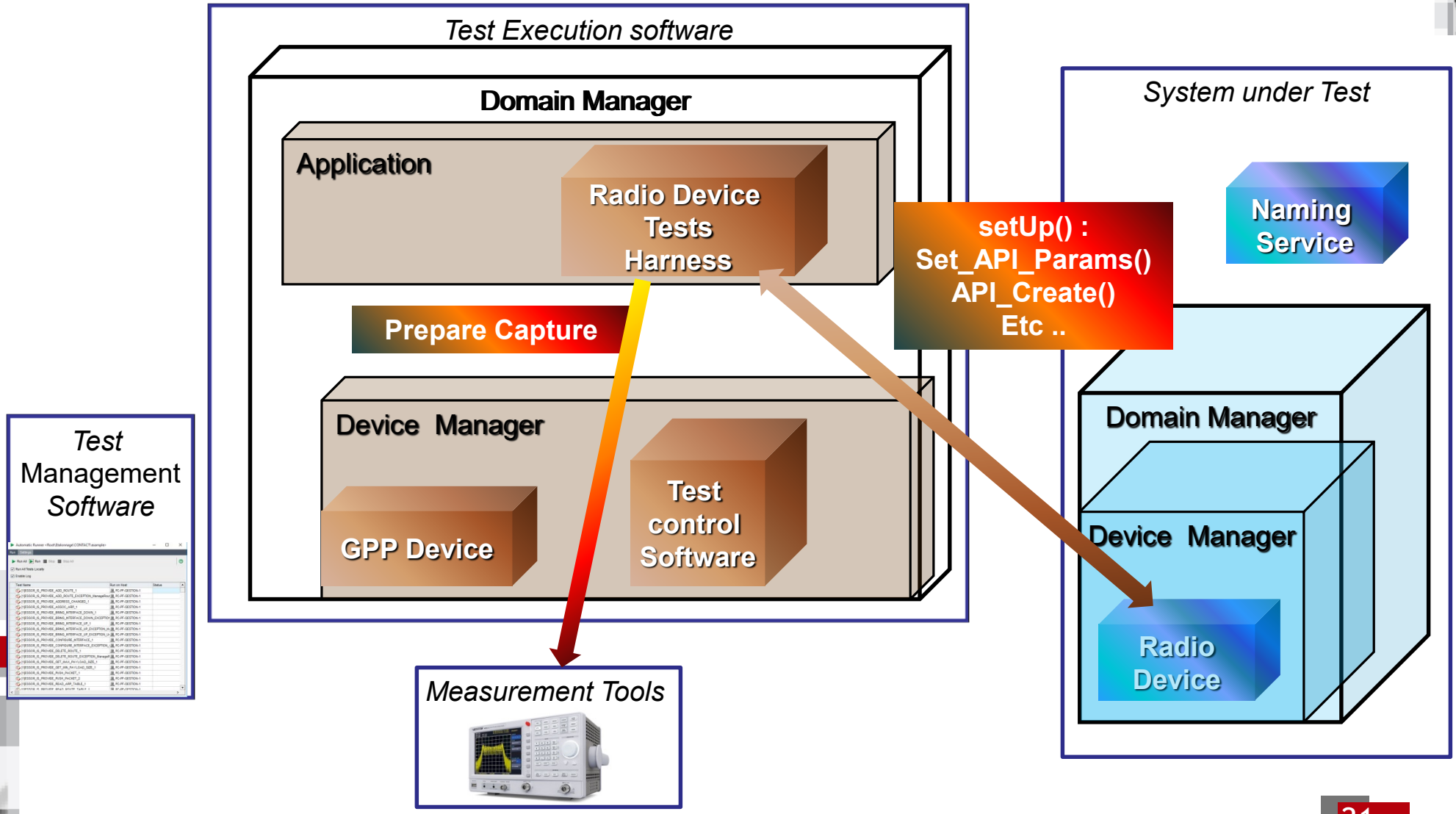




Test of SDR components

Example of Radio Device API Platform testing

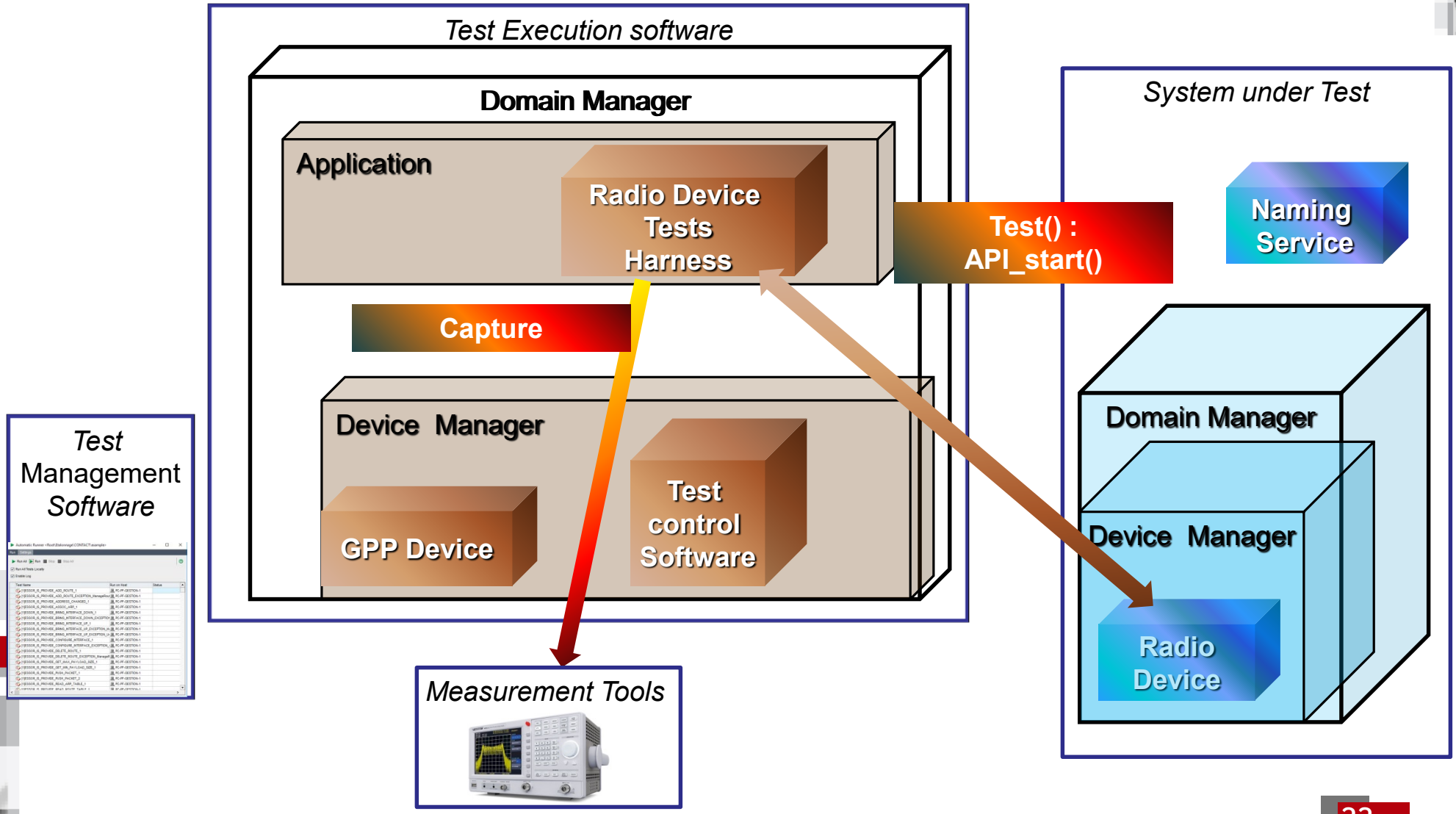
- Test Set Up



Test of SDR components

Example of Radio Device API Platform testing

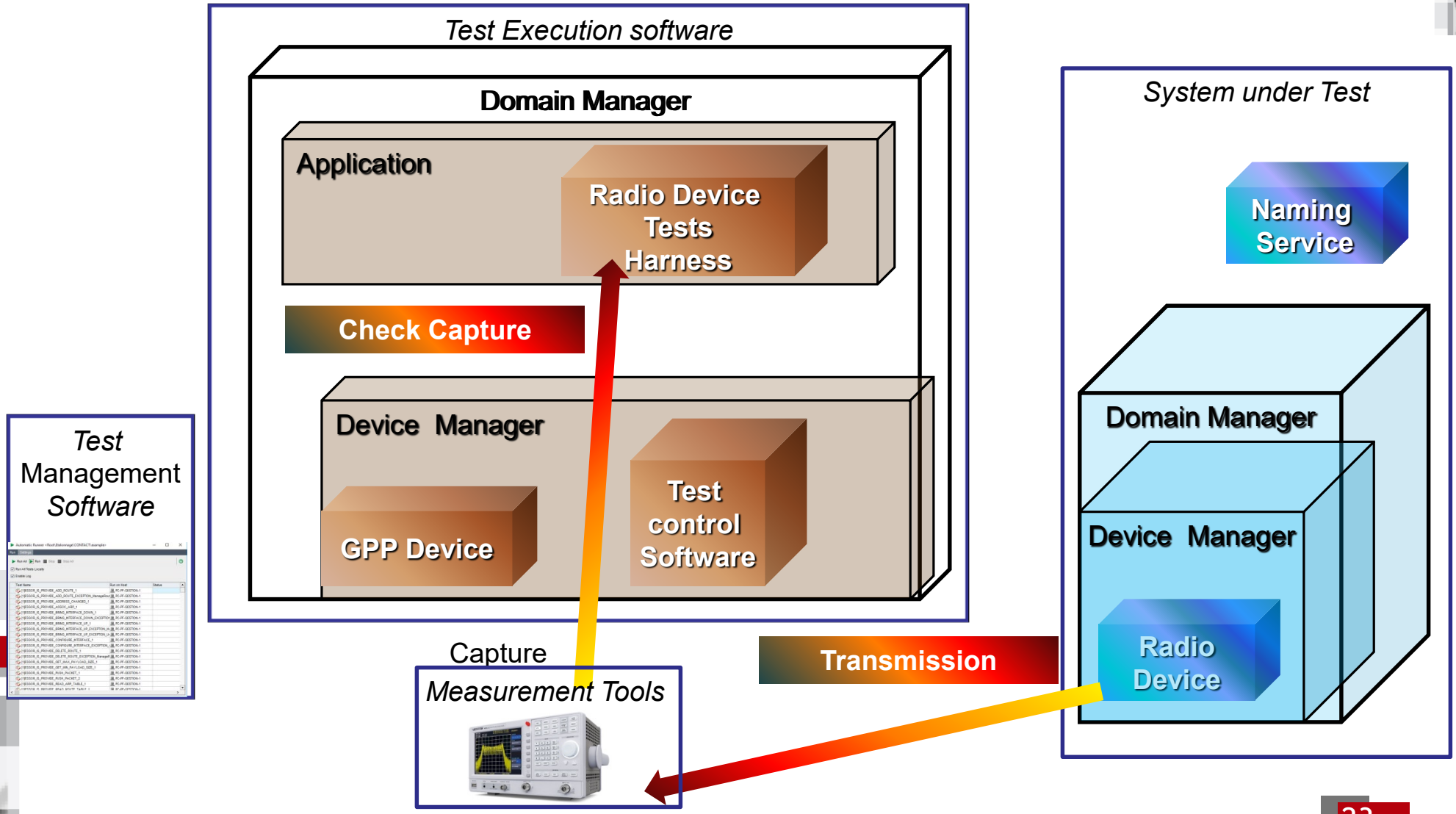
- Test main initial phase



Test of SDR components

Example of Radio Device API Platform testing

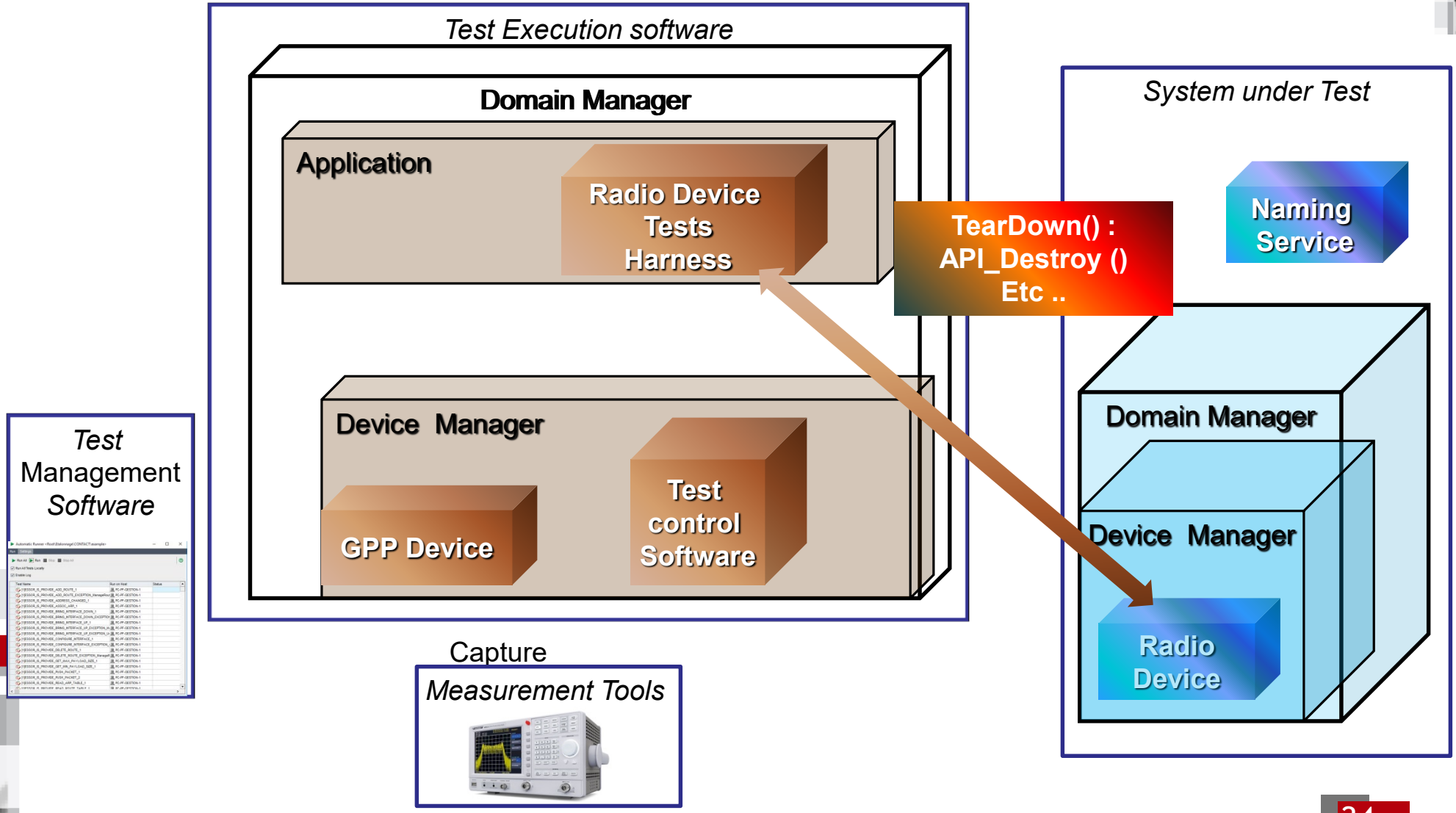
- Test main capture phase



Test of SDR components

Example of Radio Device API Platform testing

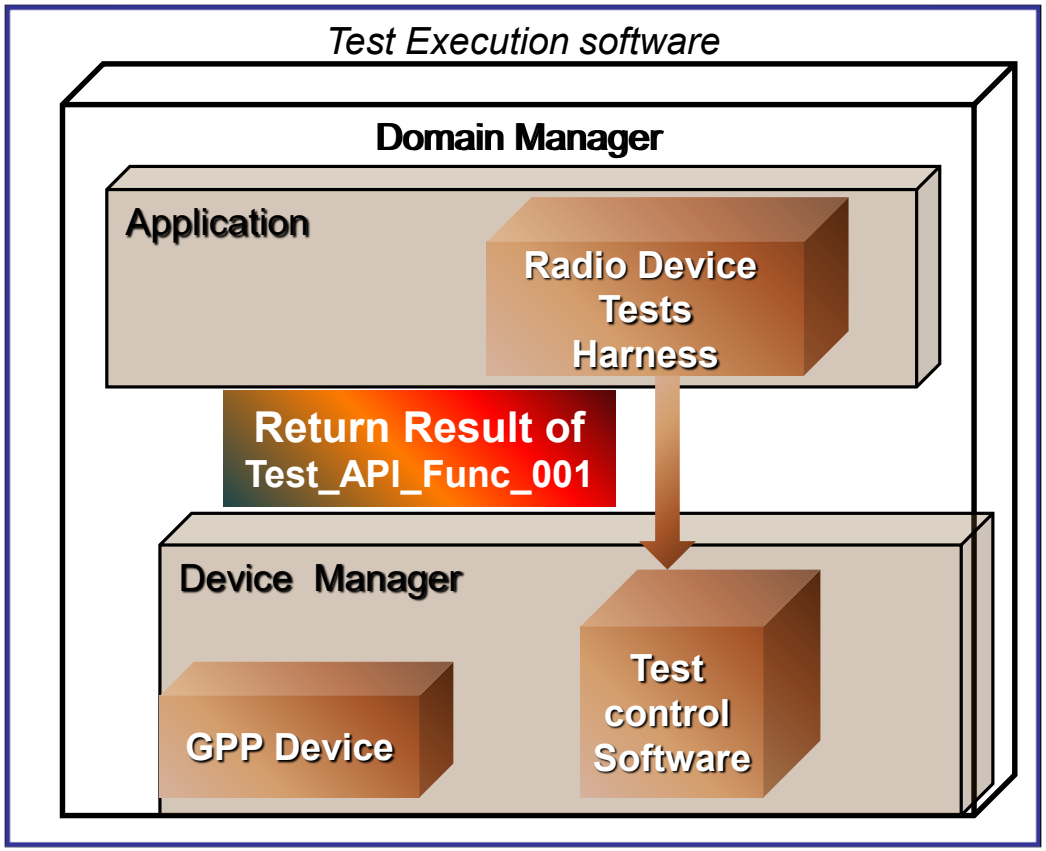
- Test Tear Down

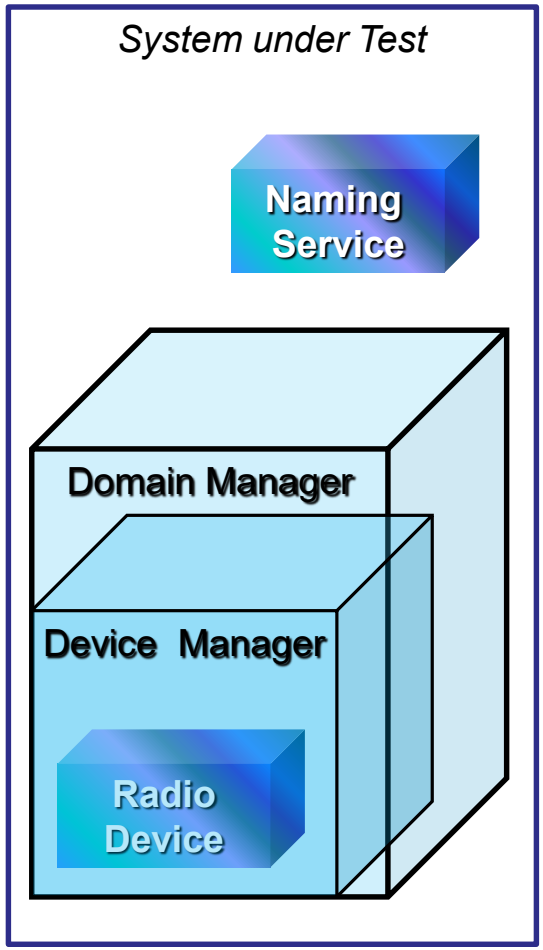


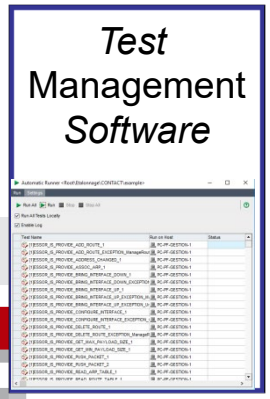
Test of SDR components

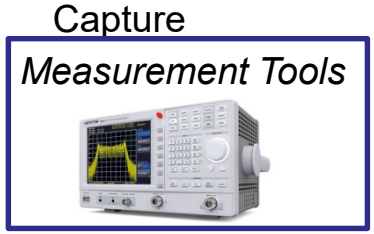
Example of Radio Device API Platform testing

- Result Return from Harness





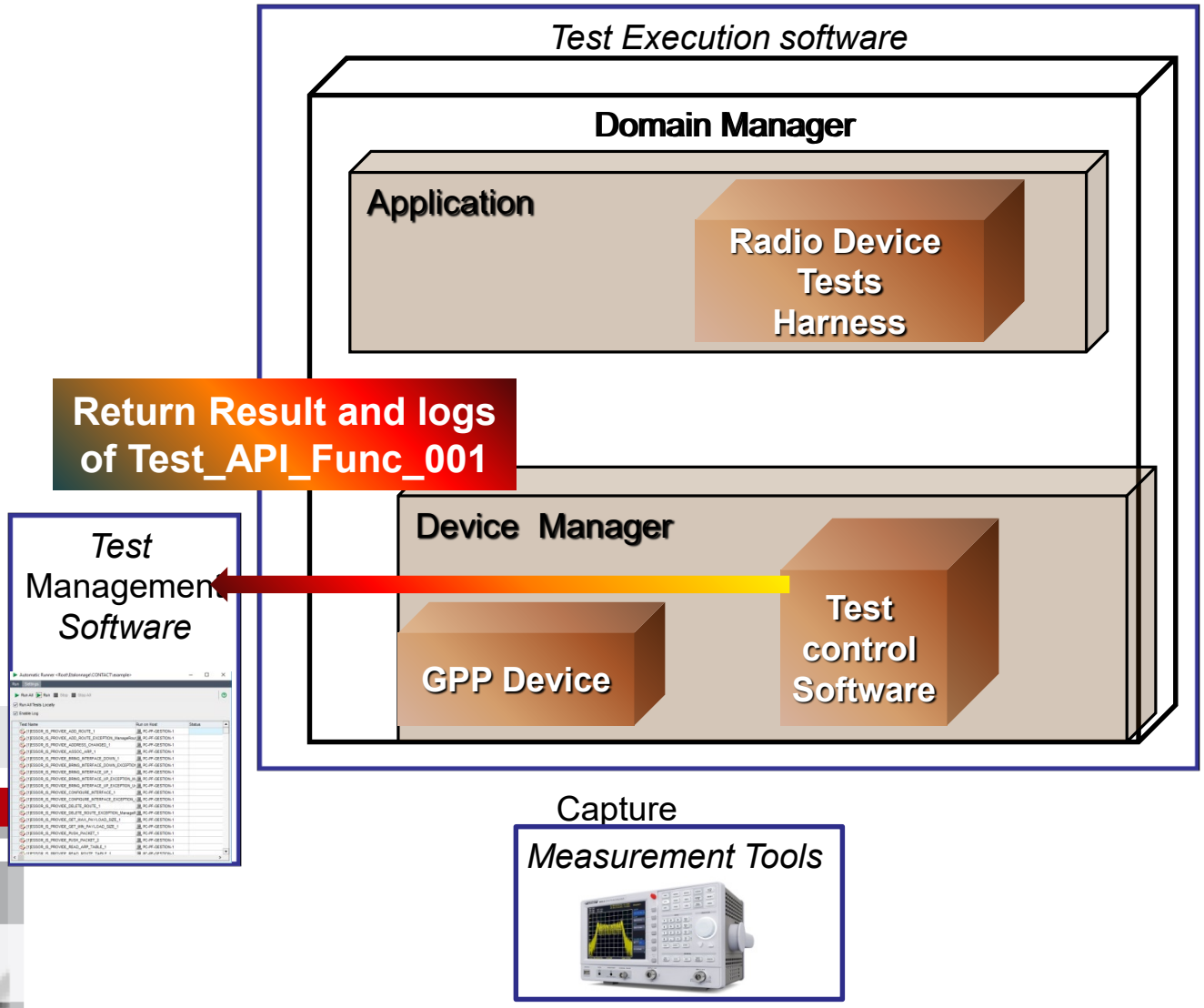


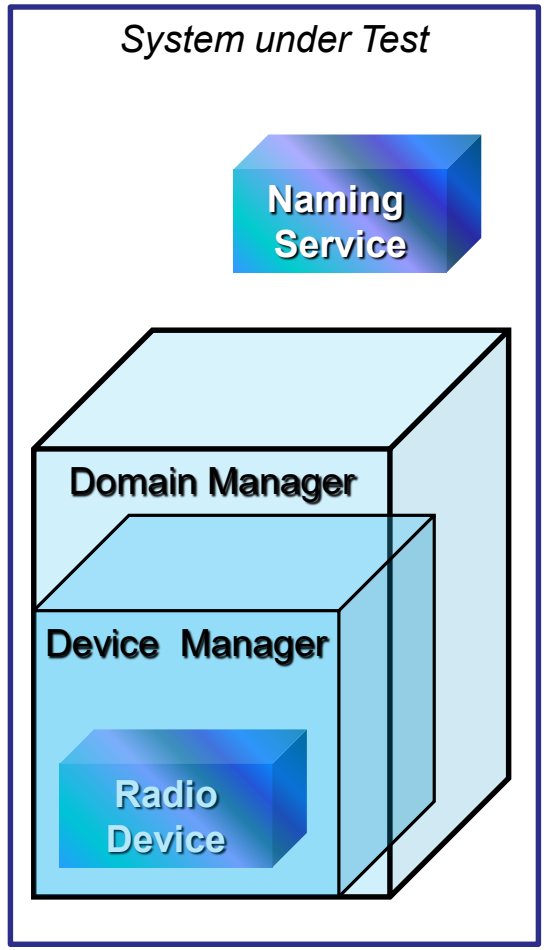


Test of SDR components

Example of Radio Device API Platform testing

- Result return to Test Management Software





Agenda

- SDR compliance assessment: the needs
- Testing methodology
  - From the SDR requirements to the tests
  - Test design process
  - Compliance checkpoints definition
  - Modeling
  - Testing generation
- The test bench
- Test of SDR components: an example
- Conclusion / Q&A

Conclusion

Results & Perspectives

- Results
  - 96 % of automated tests among more than 500 tests
- Perspectives
  - Extensibility of the test bench
    - Evolution of the standard
    - New application or waveform
    - New service or device
    - Middleware adaptation
  - Testing methodology and some testing tools reusable for other SDR standards

Questions