Implementing IEEE 1641 – Compilation Techniques

Matt Cornish *Principal Consultant* EADS TES (UK)

Summary

A Process from *Signal* Requirement *to* Platform Native *Driver* Code

- In contrast to **previous run-time** implementations
- **Proved** through a recent **IEEE 1641 study**, sponsored by the **UK MoD**
  - Example tests; **Gain** and **1 dB Compression** Point, for a mobile communications device
  - Tests written as 1641 in a **C# carrier** language
- IEEE **ATML Capability description** of test resources used to determine suitable test resources
- An XML schema is described to describe the translation from IEEE 1641 Signals to test resources' IVI driver calls
- Tests translated to IVI driver functions, then incorporated into a C# program

Effectively, 'compiling' the IEEE 1641 Signals into IVI Driver code

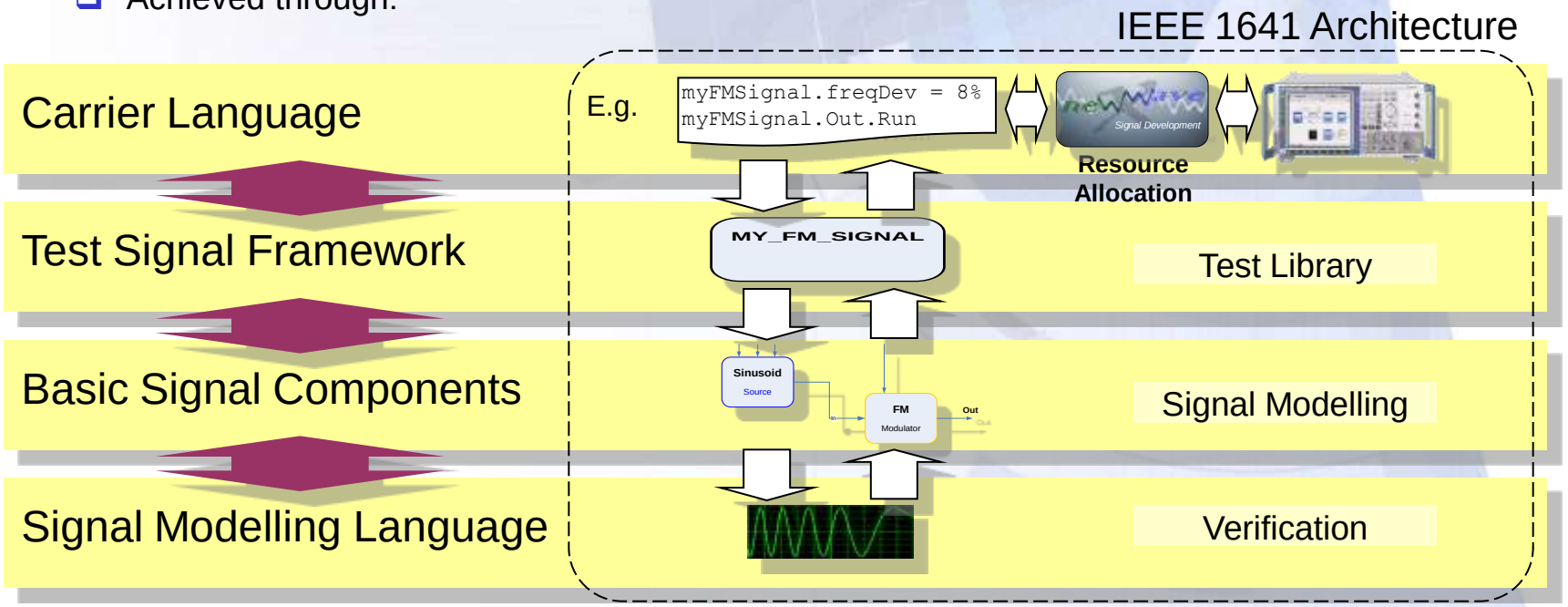
EADS DEFENCE & SECURITY

Implementing IEEE 1641 – Compilation Techniques

Introduction

IEEE 1641 – Signal & Test Definition

- Major **objective** has always been '**openness**'
  - IEEE 1641 a key element of the UK MoD's Open System Architecture
    - Choice of TPS development tools, through interoperability
    - TPSs remain optimally portable and resistant to obsolescence
- Achieved through:



IEEE 1641 – Carrier Language Interfaces

- **1641 defines interfaces for:**
  - Test program: IDL & TPL
  - Test description: TPL & XML (e.g. ATML Test Description)
- **Primary methods are:**
  - ***Require*** (<SignalDescriptor>, [UniqueID])
    - SignalDescriptor:
      - BSC (Basic Signal Component)
      - TSF (Test Signal Framework – signal library)
      - XML signal definition (i.e. an 'anonymous TSF')
    - UniqueID, allowing specific resource information to be specified
  - ***Run***(timeOut)
  - ***Change***(timeOut)
  - ***Stop***(timeOut)
  - **.properties**

E.g. Resource Manager with COM interface, IDL

```
// Using my TSF library
AM_SIGNAL myAMSig
myAMSig = Require(AM_SIGNAL)
myAMSig.modIndex = 8%
myAMSig.Out.Run
```

IEEE 1641 – Carrier Language Interfaces

- **The 1641 methods map to ATML Test Description *Operations***
  - E.g. IEEE ATML Test Description

```
<!-- ATML Test Description -->
<Operation xsi:type="OperationSetup" … >
  <Sensor>
    <LocalSensorSignalReference localSignalID="ls1"/>
    <std:Signal>
      <myTSFLib:AM_SIGNAL name="myAMSig" modIdx="8 %"/>
    </std:Signal>
  </Sensor>
  …

<Operation xsi:type="OperationConnect" ... >
  <Signal>
    <LocalSignalReference localSignalID="ls1"/>
  </Signal>
  …
```

EADS DEFENCE & SECURITY

Implementing IEEE 1641 – Compilation Techniques

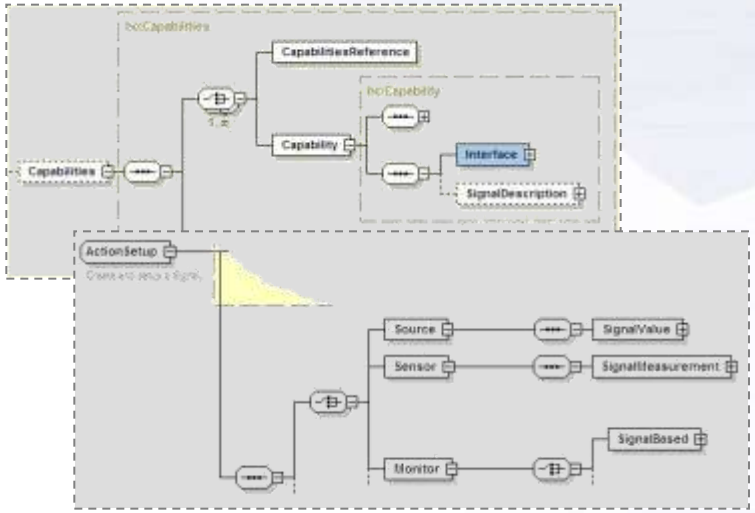
ATML Capability Description

ATML Capability Description

- **Uses IEEE 1641 Signal Models**

ATML

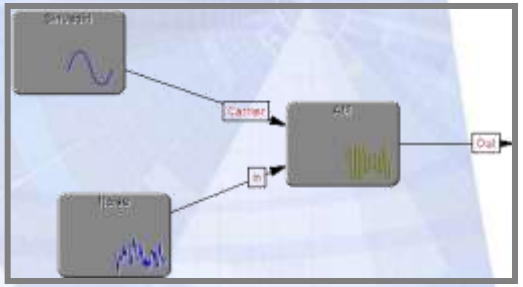
Test Station & Instrument Description

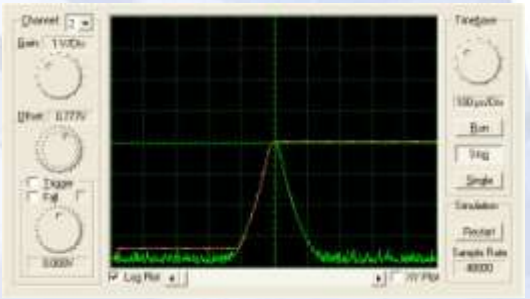


<ATML/>

Automatic Test Mark-up Language

IEEE 1641 Signal Models





Analogue

Digital

Stimulus

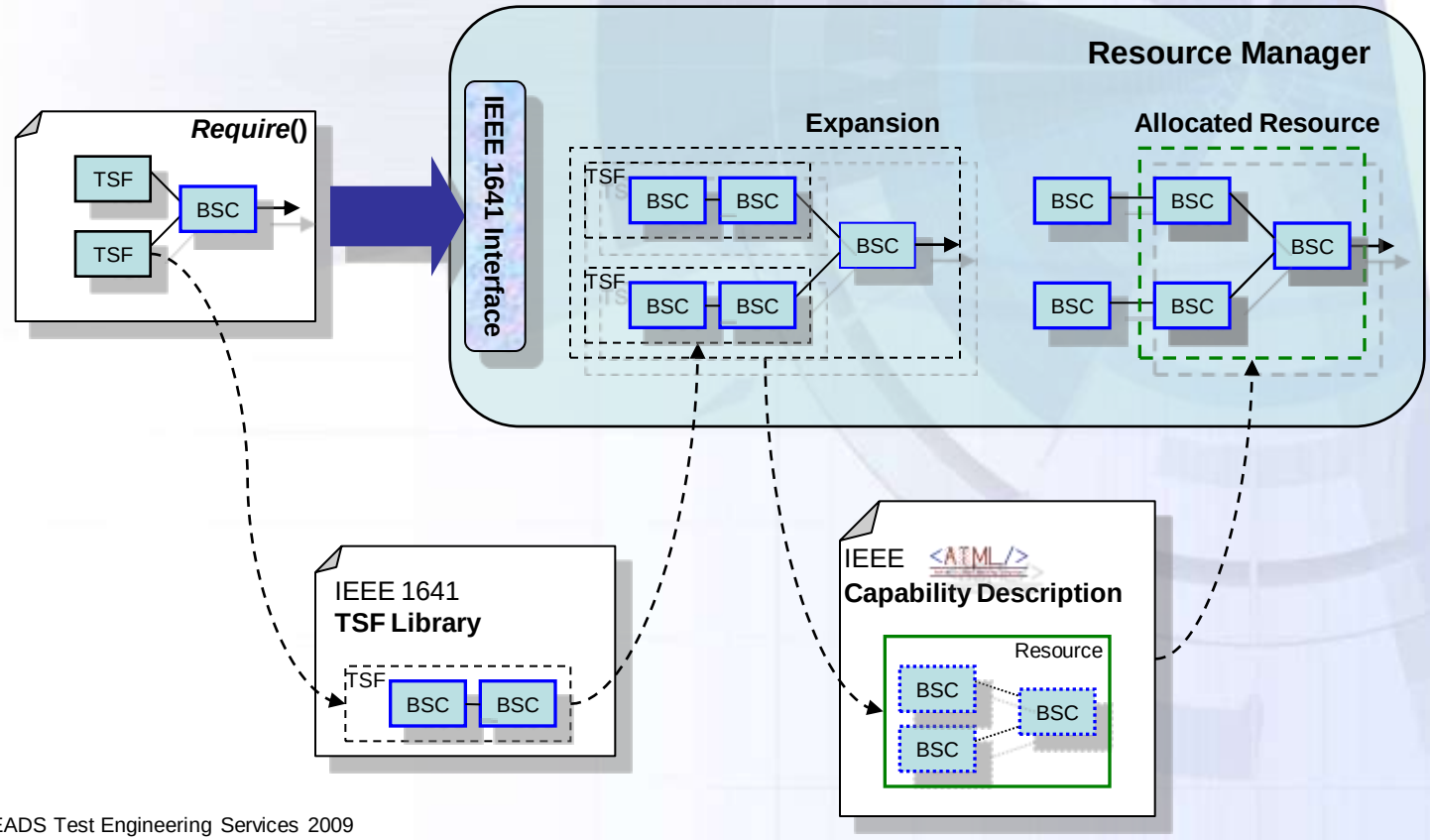
Measurement

ATML Capability Description

- **Signal-based *requirements [optionally expanded]***

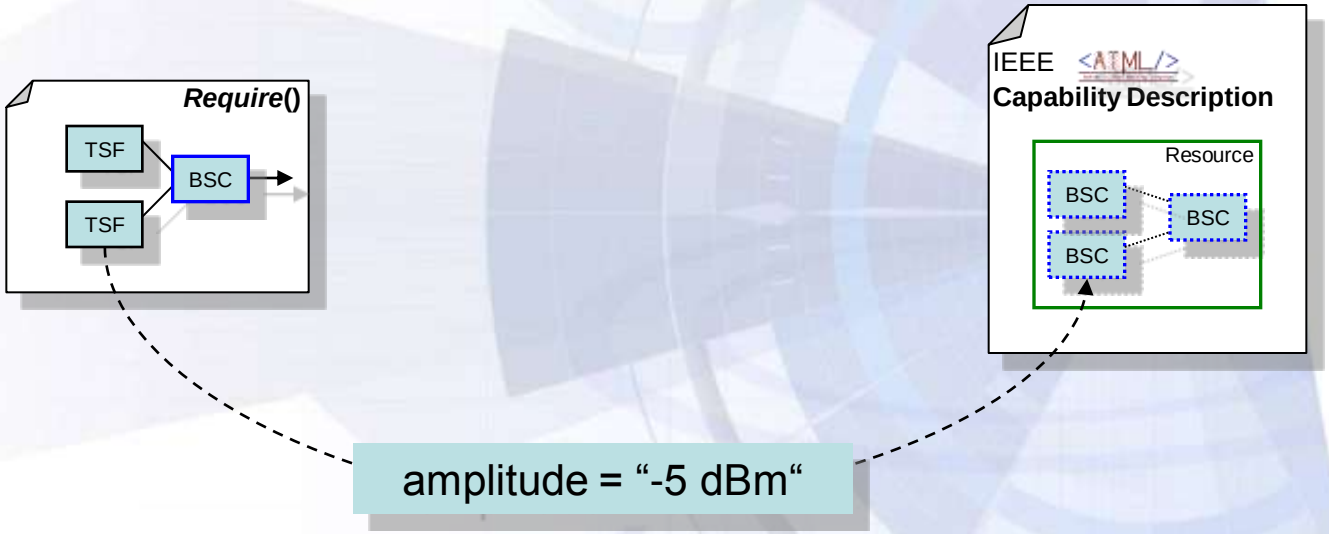
  Compared and matched to

- **Signal-based capabilities**



ATML Capability Description

- **Signal attributes are similarly transferred to the capability**

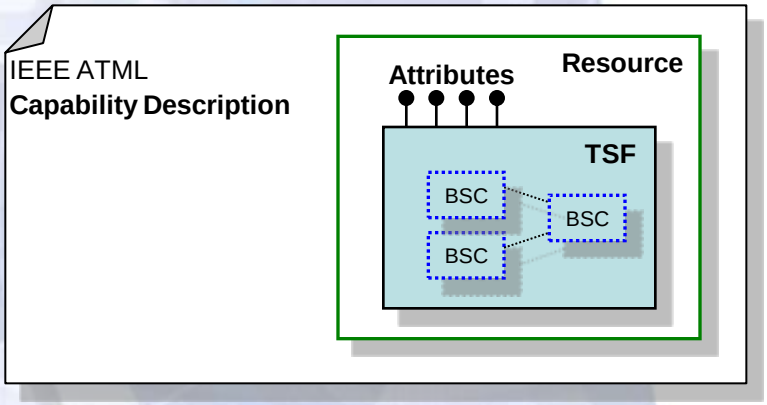


- **How can these values be transferred to the parameters of the resource?**
  - Since, each controllable parameter of a resource maybe bound to more than one signal aspect
  - E.g. Three Phase Synchro

    Three sinusodal phase attributes, each bound to a single attribute of rotor angle

ATML Capability Description

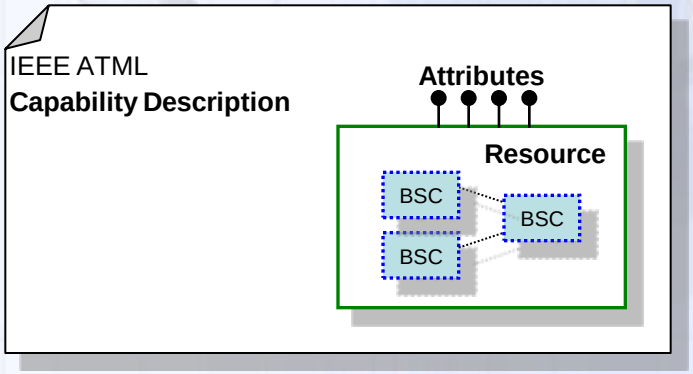
- **Resources may be described in terms of TSFs**



- **However:**
  - TSF attribute formulae propagate towards the signal
  - Resource descriptions require signal properties to propagate towards the controllable resource attributes

∴

Capability descriptions require attributes



EADS DEFENCE & SECURITY

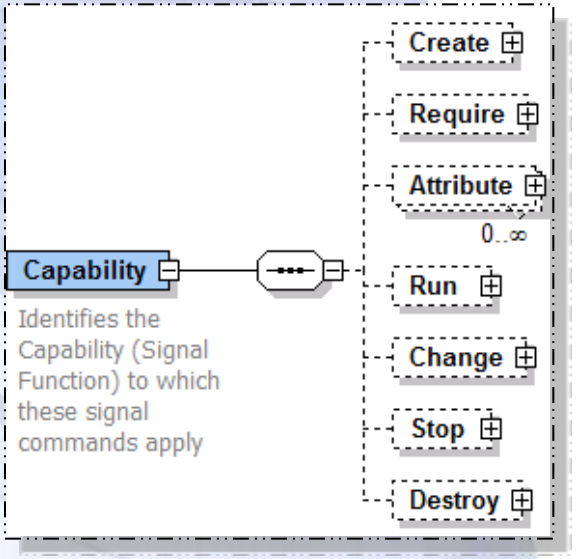
Implementing IEEE 1641 – Compilation Techniques

Resource Driver Description

Resource Driver Description

- Each of the 1641 interface elements is used to build up a dictionary of driver functions to implement each capability:
  - ***Require*** (<SignalDescriptor>)
  - ***Run***(<timeOut>)
  - ***Change***(<timeOut>)
  - ***Stop***(<timeOut>)
- Additionally:
  - ***.attribute***
  - ***Create*** (to control object lifetimes)
  - ***Destroy*** (to control object lifetimes)

E.g. Capability Driver Categories

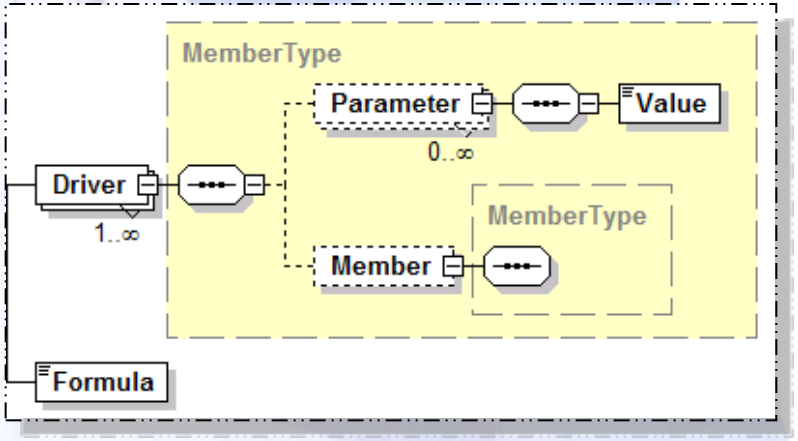


Resource Driver Description

Functions

- Each driver function may, typically, require members, parameters and values
- Additionally, IEEE 1641 formulae support complex mapping and mapping of multiple signal attributes to a given driver parameter
  - E.g. '{{MeasOutputPower.measurement.withUnit("dBm").magnitude} - {input_pwr.withUnit("dBm").magnitude}}'

E.g. Capability Driver Members

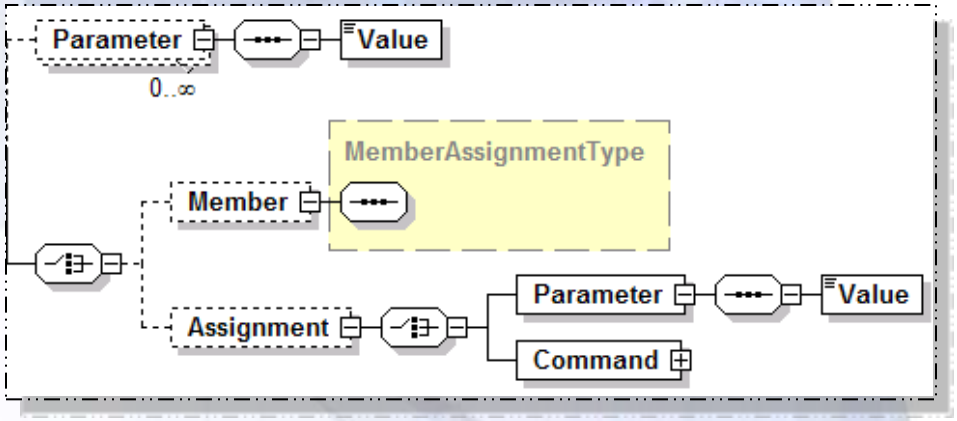


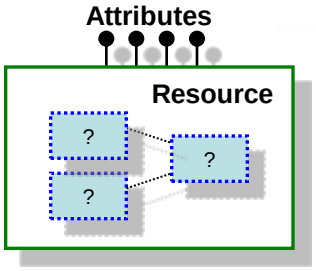
Resource Driver Description

Assignment

- Finally, assignment may, typically, be provided via a parameter value or value obtained through a further driver command

E.g. Capability Driver Assignment





Note: This, additionally, provides capability attributes, absent from ATML Capabilities

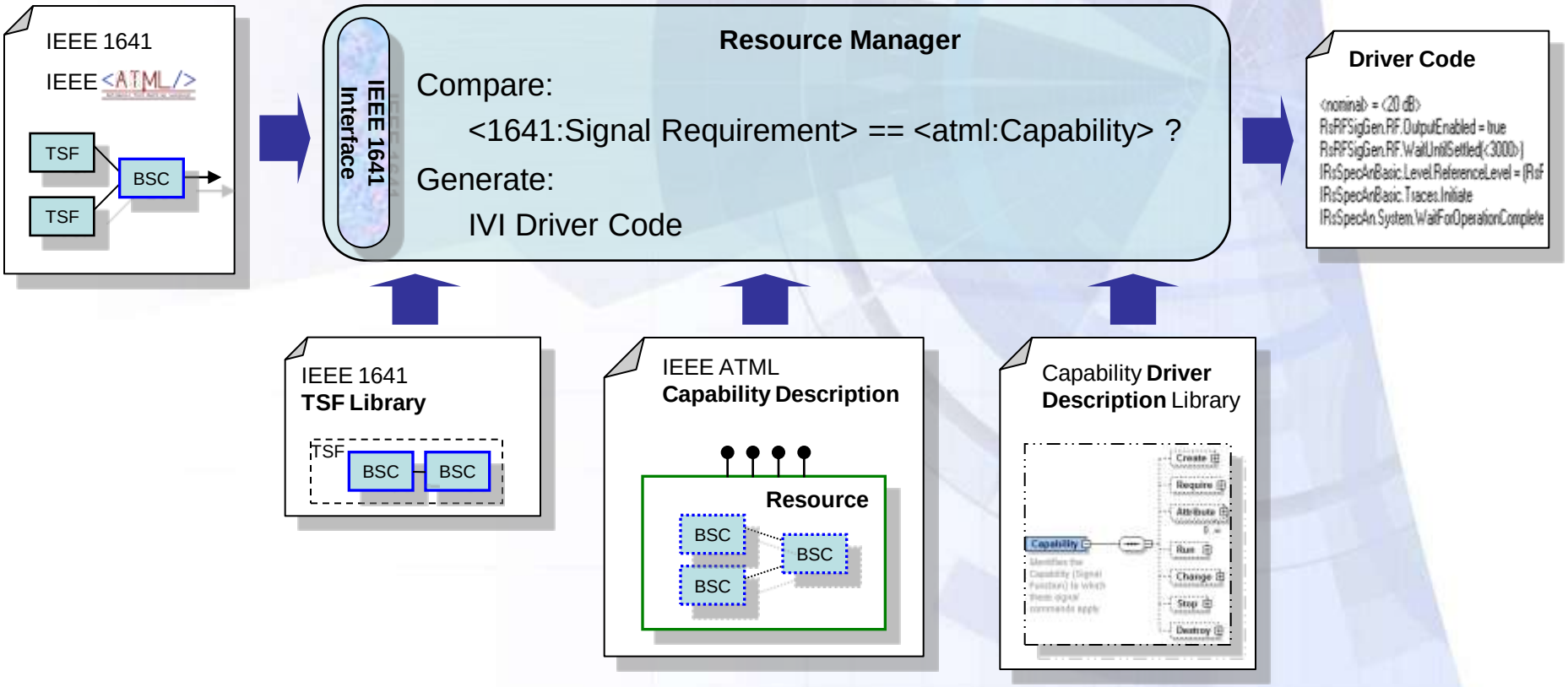
EADS DEFENCE & SECURITY

Implementing IEEE 1641 – Compilation Techniques

Compilation

Compilation

- **Combining the elements described thus far gives the following structure:**



Compilation

- **1641 Demo II.V** and **ATML Demo** have trialled this process:
  - **Required** 1641 **Signal** is **matched to** an ATML **Capability** of a given resource
  - Identified **resource** is **located in** Resource **Driver Description**
  - Each **1641** interface action **yields** a set of driver (IVI) **commands or assignments.**
  - Process **cannot determine** whether the assignment value (e.g. '-20'), is a **literal or** a **variable**. <> denote a possible variable**.**
  - Doesn't manage **loops** or **conditional** statements
  - Output is generated in a **syntactically independent** form, leaving it open to incorporation into a choice of programming languages

E.g. IVI Driver Commands & Assignments

nWXResource

```
RsRFSigGen.new
RsSpecAn.new

RsSpecAn.Initialize("FSG", true, true, "")
IRsSpecAnBasic = RsSpecAn.Personality.Select("Basic")
RsRFSigGen.Initialize("SMJ", true, true, "")
IRsSpecAnBasic.Frequency.Span = 0
RsSpecAn.System.IO.WriteScpi("CALC:MARK:FUNC:SUM:RMS ON")
RsSpecAnBasic.Acquisition.DetectorType = RsSpecAnBasicDetectorTypeEnum.Rs
RsSpecAnBasic.SweepCoupling.SweepTime = 0.2
RsSpecAnBasic.Acquisition.SweepModeContinuous = false

RsRFSigGen.RF.Level = <-20>

RsRFSigGen.RF.Frequency = <1500000000>
IRsSpecAnBasic.Frequency.Center = <1500000000>

<nominal> = <20 dB>
RsRFSigGen.RF.OutputEnabled = true
RsRFSigGen.RF.WaitUntilSettled(<3000>)
IRsSpecAnBasic.Level.ReferenceLevel = (RsRFSigGen.RF.Level + <nominal> + 2)
IRsSpecAnBasic.Traces.Initiate
IRsSpecAn.System.WaitForOperationComplete(<3000>)

<measurement_complete> = true

RsRFSigGen.RF.OutputEnabled = false

<measurement> = RsSpecAn.System.IO.QueryScpi(20, "CALC:MARK:FUNC:SUMM
```

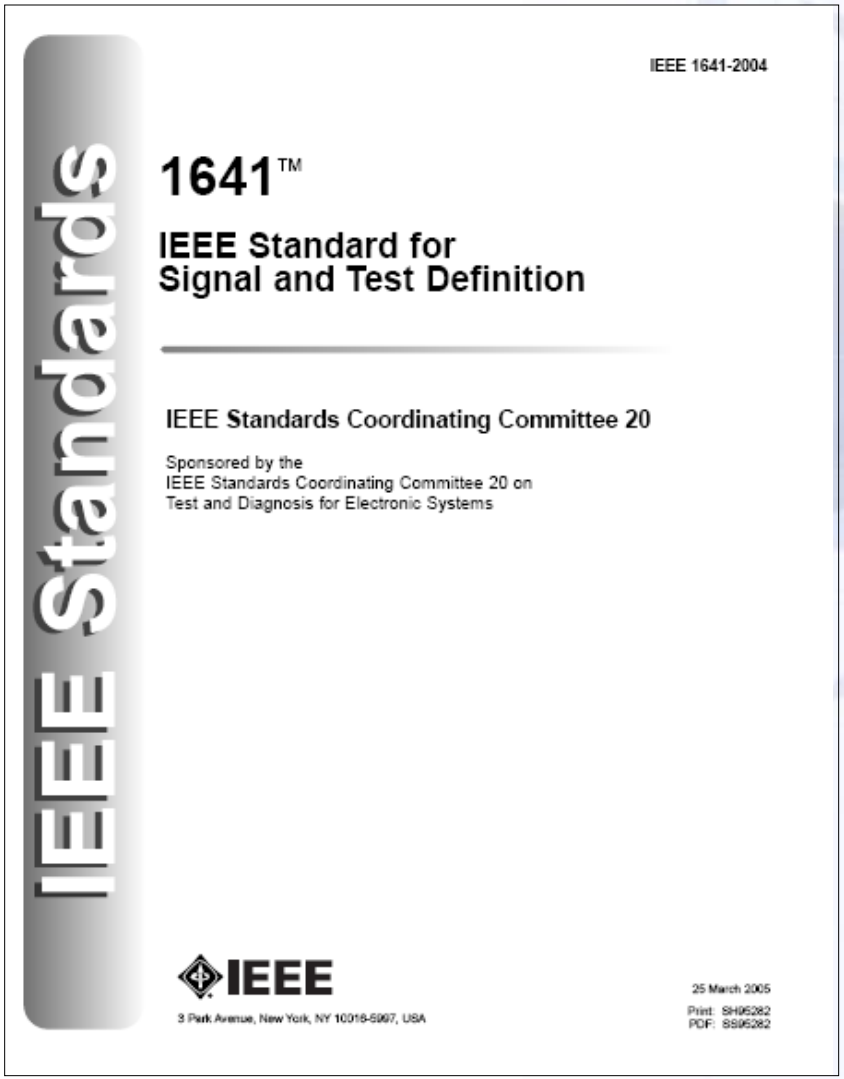
Implementing IEEE 1641 – Compilation Techniques

Conclusions

Conclusions

- Architecture described to compile **signals** into native **driver code**
  - **IEEE 1641**
  - **ATML** Test Description
- **Open Architecture,** through the 1641 interfaces, IDL, XML & TPL
  - **Test environments** & **test languages**
  - Meets **UK MoD's 'Open System Architecture'** for **portability** and **interoperability**
- **ATML Capability Description** using 1641 signals to **automate resource allocation**
- **XML Resource Driver** document (supplementing ATML Capability), to describe a dictionary of **actions** and **attributes**
- A **compile-time** approach, in **contrast** to previous implementations, which have adopted a **run-time** approach
- Retains the **automatic validation** benefits of the run-time approach, since source documents are XML (including 1641 & ATML)
- In conjunction with **1641 signal modelling** tools, this approach **supports verification** of **resource** performance.

IEEE 1641 – Signal & Test Definition



- **Available from IEEE**
  in paper or PDF format
- **Paper –**
  Product No: SH95282
  ISBN 0-7381-4500-9
  Price $105 or $85 (IEEE members)
- **PDF –**
  Product No: SS95282
  ISBN 0-7381-4501-7
  Price $90 or $70 (IEEE members)
- **Website**
  http://shop.ieee.org/ieeestore
  Search for **1641**; type **Standards**

- **Also now available:**
  IEEE Std 1641.1™-2006
  IEEE Guide for the Use of IEEE Std 1641, Standard for Signal and Test Definition