

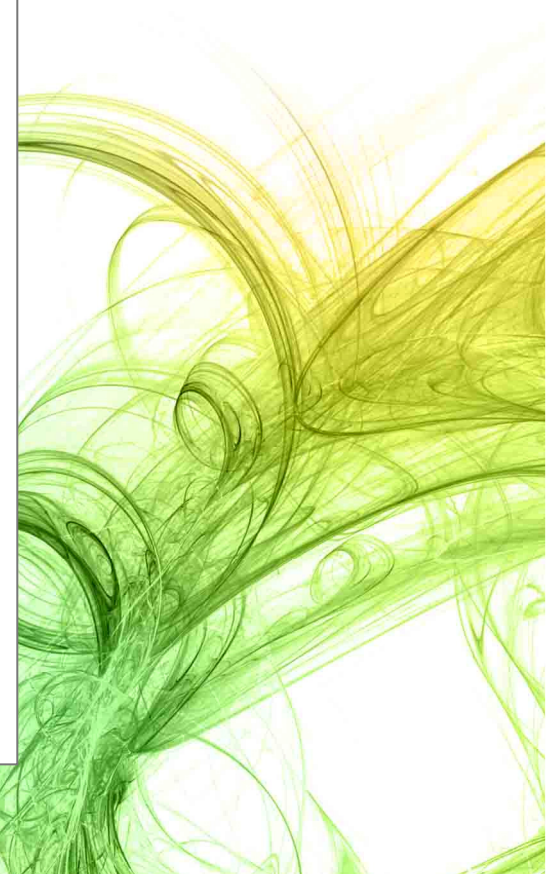
**indra**

Eurofighter & A400M Programmes, Automatic Test Equipments

# TESTBRICKS

Software framework for IEEE Standard 1641 test programs.

FERNANDO MUÑOZ MANRIQUE – Engineering Manager  
SLGAT Meeting 2012-2 / MBDA Stevenage, UK / November 27, 2012



# INDEX

01 Introduction

02 Carrier language

03 Writing the test program

04 Execution environment

05 Finding errors

06 Exchanging ATS and test information

07 Conclusions

# SLGAT MEETING – BATH SEPTEMBER 2007



Presentation of the case study:

## Developing a Complete Test Program Using IEEE 1641

# SLGAT MEETING – BATH SEPTEMBER 2007

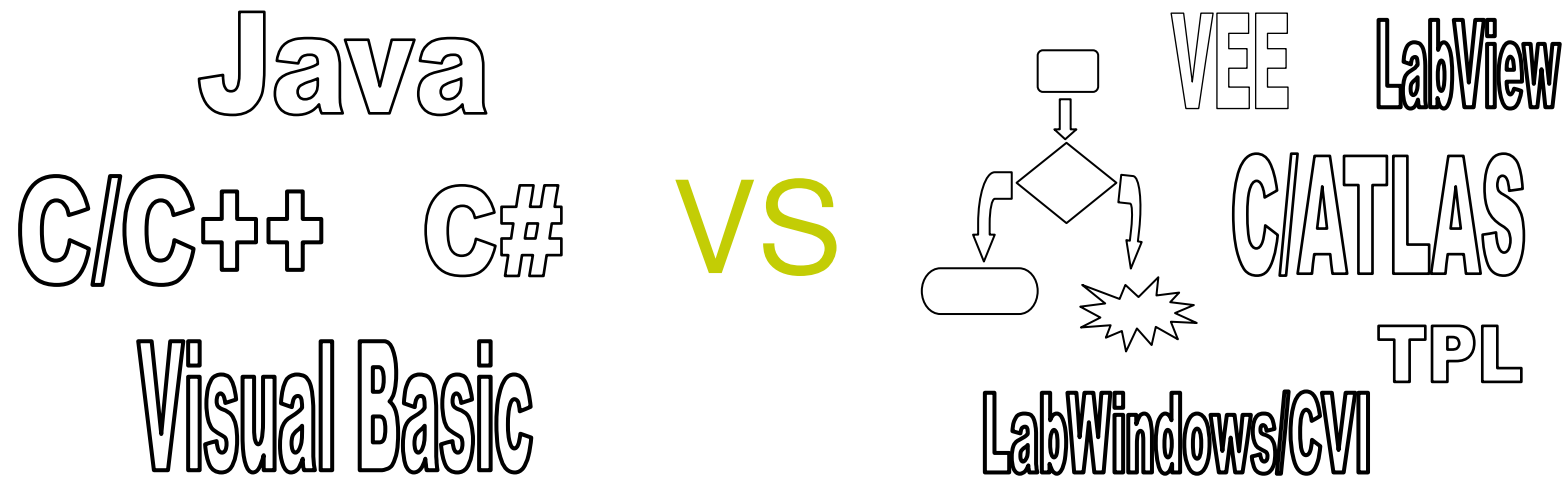
## Main conclusions (2007)

- The overall conclusion is that developing a test program based on the IEEE 1641 elements and philosophy is a **valid, working and interesting approach**, and a good alternative to traditional methods based on test actions
- Probably the **learning curve** for an electrical/test engineer is less arduous than using other techniques (ATLAS, general-purpose languages)
- The **portability** of the products obtained is granted by the use of an IEEE standard
- The **most outstanding problem is the lack of a true programming language and/or environment** to create complete test programs including BSC's/TSF's as native elements

# PROGRAMMING LANGUAGE OPTIONS

General Purpose

Domain Specific



“Development with general purpose languages is **more complex and less productive** than using specific languages for test and measurement”

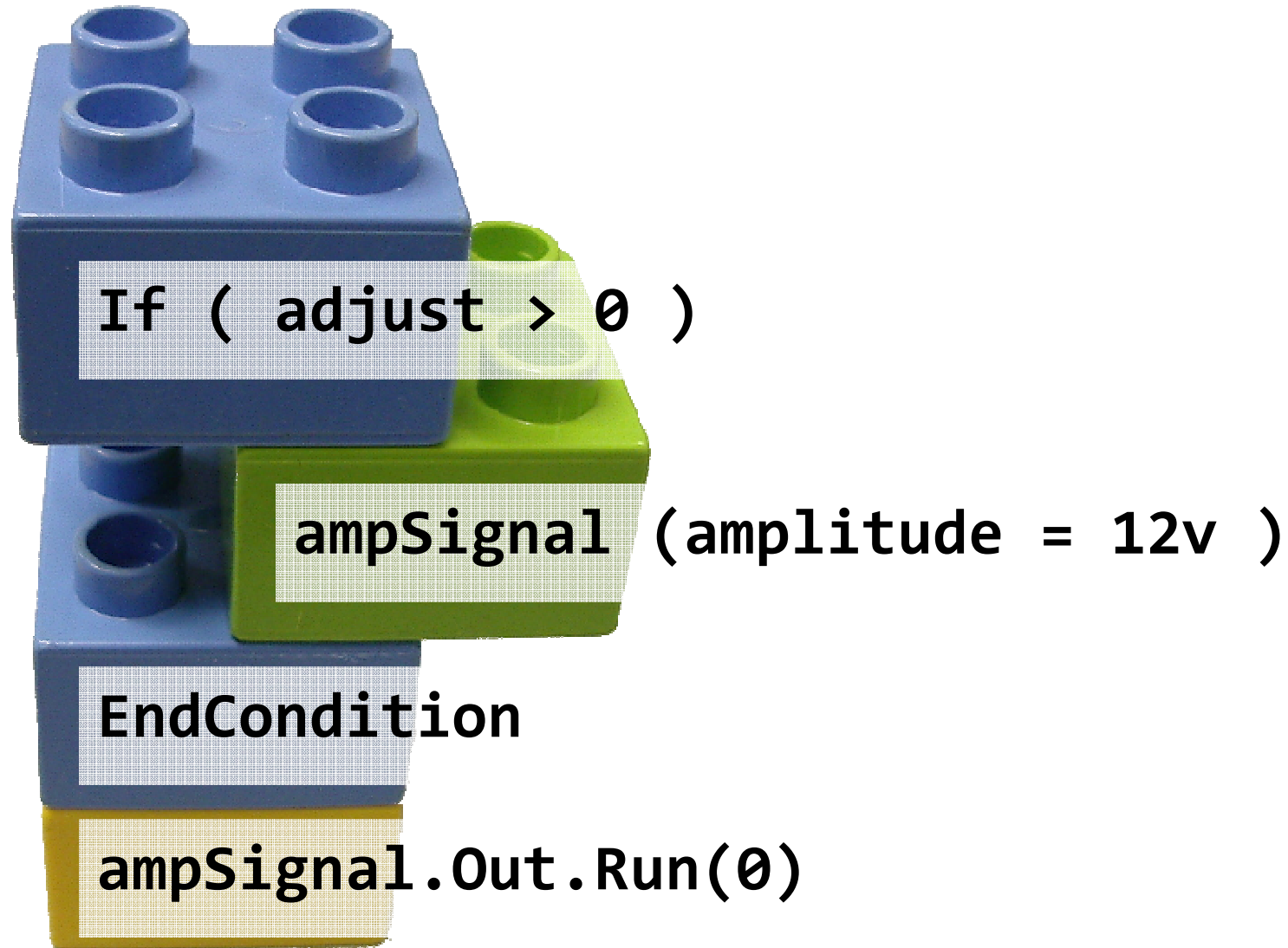
# REQUIREMENTS

For truly portable test requirements, **carrier language** ideally should be:

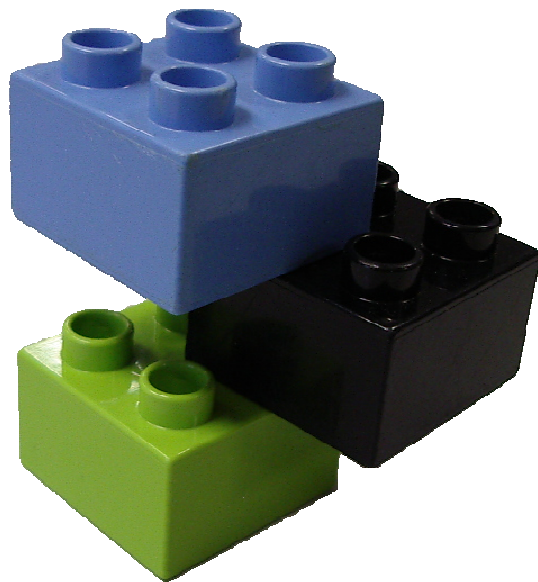
- Platform independent
- Signals natively supported
- Supported by development tools

**Development  
process should  
be easy and  
straightforward**

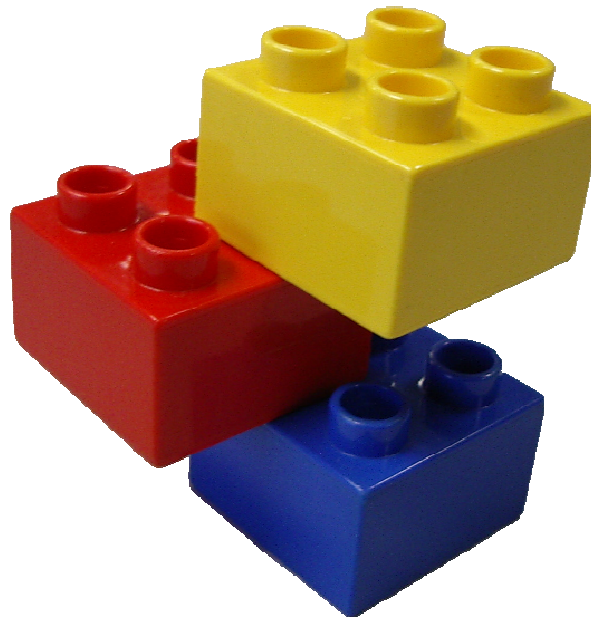
# PROGRAMMING IN PSEUDOCODE



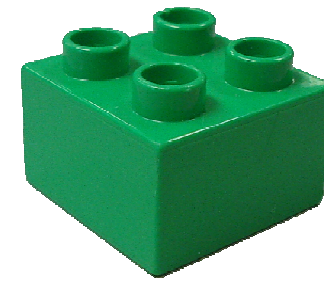
# BUILDING BLOCKS



**Variables**



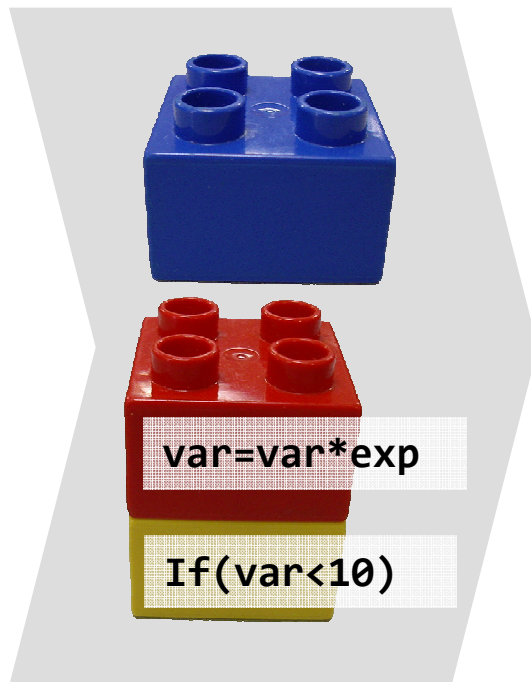
**Functions**



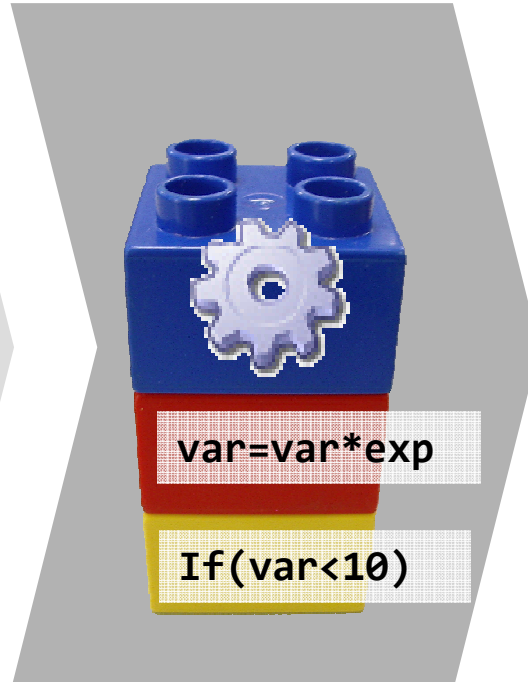
**Add new  
blocks**

# CREATING A SEQUENCE

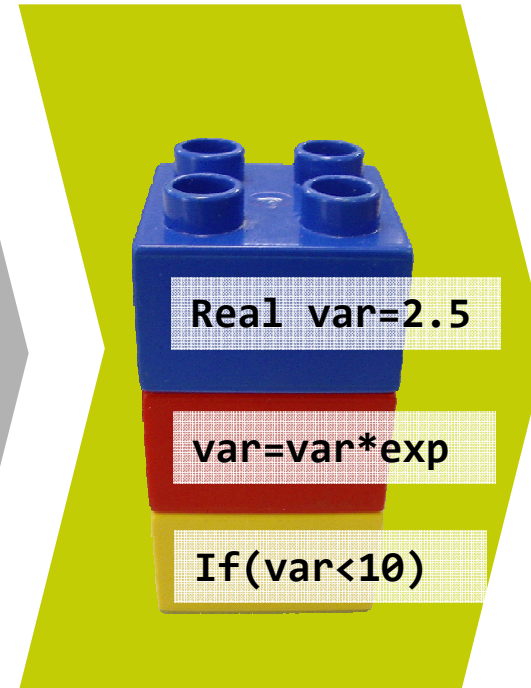
1 Place blocks in a stack



2 Configure parameters



3 Read pseudocode from top to bottom

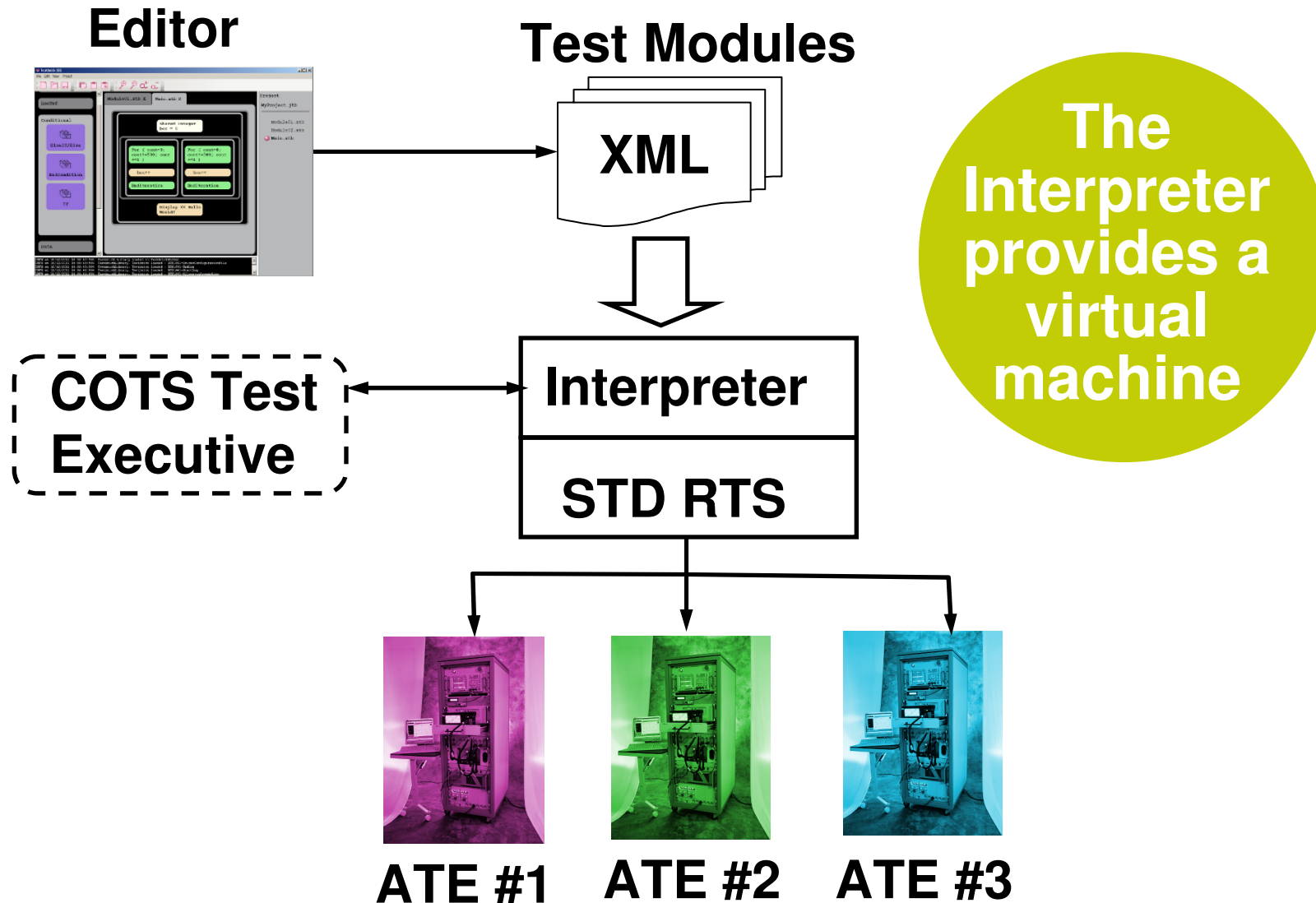


# EDITOR

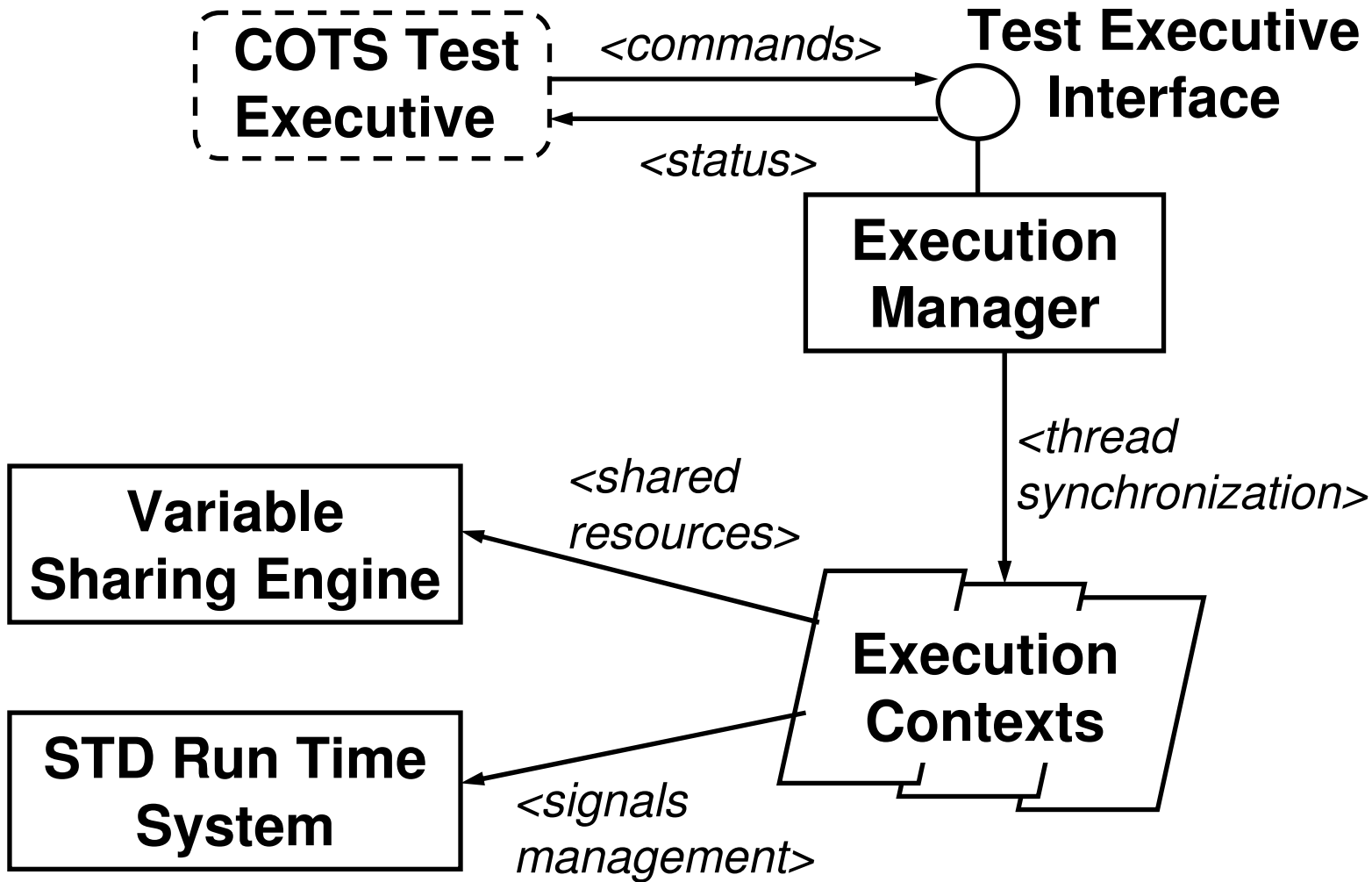
The screenshot shows the Testbrick IDE interface. The main workspace displays a test program flowchart for 'Example2.xtb'. The flowchart starts with two shared variables: 'Shared Integer Alarm2 = 0' and 'Shared Boolean Alarm3 = true'. The main logic is contained within a 'While ( loop )' block. Inside this loop, the following steps are executed: 'Integer tmp = 0', 'Measure Name=ChannelA nominal=2 v condition=LT', 'Measure Name=ChannelB', 'ChannelA.Out.Run(0)', 'Alarm1 = ChannelA.NOGO', 'ChannelB.Out.Run(0)', 'tmp = ChannelB.measurement', 'If ( (tmp & 0x3F) > 0 )', 'Alarm2 = 1', 'EndCondition', and 'EndIteration'. A nested 'While ( loop )' block is also present, containing 'If ( (Alarm1+Alarm2>0) && Alarm3 )', 'Call EmergencyShutdown()', and 'EndCondition'. The left sidebar shows a 'BscTsf' library with components like Clock, Constant, Counter, Decode, and Diff. The right sidebar shows the project structure with 'Example2.jtb' and its sub-files. The bottom status bar displays system logs.

```
INFO at 11/07/2012 11:44:10:382 Testbrick Library loaded in TestbrickPicker
INFO at 11/07/2012 11:44:10:776 TestbrickLibrary. Testbrick loaded : RTS1641-DriverConfigurationFile
INFO at 11/07/2012 11:44:10:776 TestbrickLibrary. Testbrick loaded : RTS1641-AllocatorAssemblies
INFO at 11/07/2012 11:44:10:776 TestbrickLibrary. Testbrick loaded : RTS1641-DriverAssemblies
INFO at 11/07/2012 11:44:10:776 TestbrickLibrary. Testbrick loaded : RTS1641-StartLog
INFO at 11/07/2012 11:44:10:776 TestbrickLibrary. Testbrick loaded : RTS1641-BscConfigurationFile
```

# THE VIRTUAL MACHINE

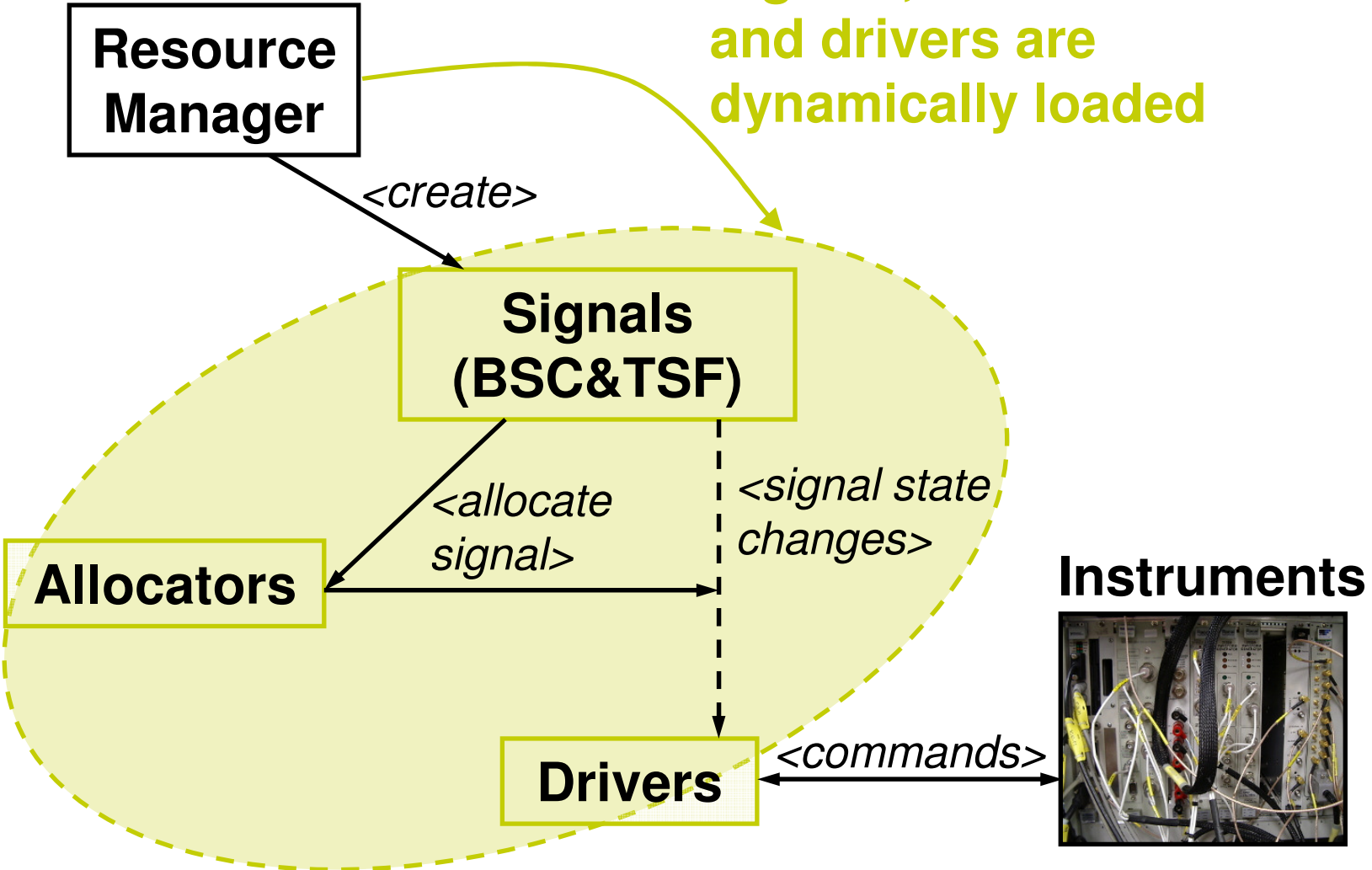


# INTERPRETER



# STD RUN TIME SYSTEM

Signals, allocators and drivers are dynamically loaded



## FINDING ERRORS

# DEBUGGER



The screenshot shows the Testbricks Debugger interface. The main window is titled "Testbricks Debugger" and has a menu bar with "File", "View", "Interpreter", and "Debugger". Below the menu bar is a toolbar with various icons for file operations, search, and execution control. The interface is divided into several panels:

- RTS STATUS:** Shows "PAUSED" and "Main Module" set to "D:\fmmanrique\Mis documentos\Testbric".
- Contexts in execution:** A list of execution contexts with their respective line numbers:
  - 1: S0 / Init2.xtb Line: 0
  - 2: S1 / Init2.xtb Line: 6
  - 3: S0 / Example2.xtb Line: 0
  - 4: S1 / Example2.xtb Line: 4
  - 5: S2 / Example2.xtb Line: 9
  - 6: S3 / Example2.xtb Line: 3
- Execution Context 5:** A detailed view of the current execution context, showing a list of code lines:
  - 0: Integer tmp = 0
  - 1: Measure Name=ChannelA nominal=2 v condition=LT
  - 2: Measure Name=ChannelB
  - 3: While ( loop )
  - 4: ChannelA.Out.Run(0)
  - 5: Alarm1 = ChannelA.NOGO
  - 6: ChannelB.Out.Run(0)
  - 7: tmp = ChannelB.measurement
  - 8: If ( (tmp & 0x3F) > 0 )
  - 9: Alarm2 = 1
  - 10: EndCondition
  - 11: EndIteration
- Project:** A list of project files:
  - Example2.jtb
  - Example2.xtb
  - EmergencyShutdown.xtb
  - Init2.xtb
- Log:** A bottom panel showing system logs with timestamps and context IDs.

# ATML SUPPORT

ATML Components	TESTBRICKS FRAMEWORK UPGRADES
Test Description	Editor function to convert Test Description to pseudocode and to generate Test Description from pseudocode
Test Configuration	RTS Resource Manager to load dynamic assemblies depending on Test Configuration file
Test Station Instrument Description Test Adapter UUT Description	Allocator assemblies to allocate signals to drivers based on data from Test Station, Instrument Description, Test Adapter and UUT Description files
Test Results Diagnostics	Responsibility of the COTS Test Executive (Not part of the framework)

Testbricks: Software framework for IEEE Standard 1641 test programs

## CONCLUSIONS

- Create **STD compliant** test programs following an **easy and straightforward** process
- Benefit from less errors and more quality, more productivity and less cost
- Write test modules independent of the ATS in **pseudocode**
- Minimize the cost of instruments obsolescence and TPS rehosting with the interpreter **virtual machine**



# indra

## **Fernando Muñoz Manrique**

Eurofighter & A400M Programmes, Automatic Test Equipments

@ [fmmanrique@indra.es](mailto:fmmanrique@indra.es) / [fmmanrique@ieee.org](mailto:fmmanrique@ieee.org)

 [@fmmanrique](https://twitter.com/fmmanrique)

 <http://www.linkedin.com/in/fernandomunozmanrique/en>

C\ Mar Egeo, 4 Pol. Ind. N°1  
28830 San Fernando de Henares,  
Madrid España

T +34 91 627 31 21

M +34 686 521 981

[www.indra.es](http://www.indra.es)

