ESUG 2009
Brest

Thales RWS Brest

Reference - date

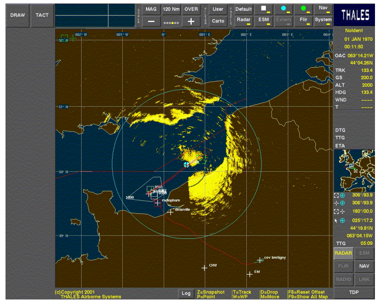
Aéronautique

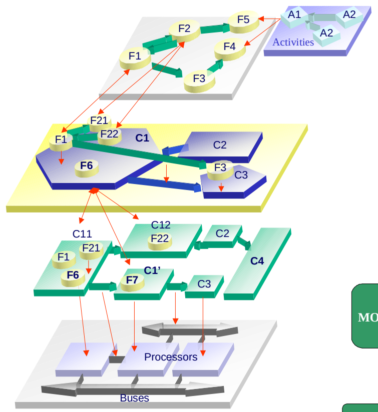
Agenda

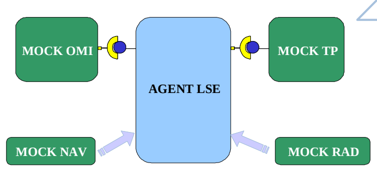
Smalltalk in Thales Brest

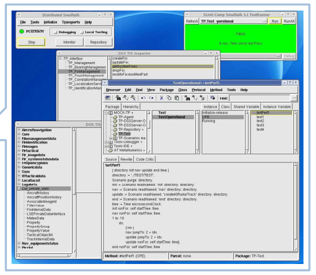
A story of twenty years:

- 1989-1997: Sensor software development
  - MMI development on real time operating system (VRTX), C (TNI) code generator
  - MMI development for embedded workstation, Static typing, C++ (Thales) code generator
  - Automatic test workbench (IEEE488, VxWorks)
- 1996-2000: Training centers
- 1996-2003: MMI workstation for Maritime patrol aircraft demonstrator
- 2002-2009:
  - System Modeling and Mockup
  - Component testing









Why/ Which Smalltalk

Why Smalltalk

- MMI RAD
- Efficient solution proved by TNI feedbacks

Which Smalltalk

- Parc Place version & Cincom versions
  - The only multiplatform version available in the early 80's.
  - From Smalltalk 2.5 up to VisualWorks 7i

Packages:

- Smalltalk Compiler / Refactoring browser → Code generation
- UI → MMI building with dedicated look & feel
- DLLCC → External access
- DST, Opentalk → OMG Corba
- Internal package:
  - SCM → Component Model CCM Like
  - DOS → Distributed Object System
  - DSS → DST tools and addOn

Agenda

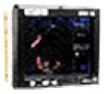
Domain : Patmar, Surmar and EWC2

- **Civilian Missions (SURMAR)**
  - Assistance and support to police & customs
  - Detection & control of pollution
  - Assistance to scientific observations
  - Surveillance of maritime traffic and offshore oil fields
  - Search & Rescue medical evacuation
  - Control & monitoring of fisheries
  - Antipollution
- **Military Missions (PATMAR)**
  - Anti-Submarine Warfare (ASW)
  - Anti-Surface Unit Warfare (ASUW)
  - EEZ Surveillance(Exclusive Economic Zone)
  - Maritime Traffic Control
  - Surveillance of off shore oil fields
  - Search and Rescue
  - Fishing Surveillance
- **Electronic Warfare C2 Functions + RESM**
  - EW Picture elaboration
  - EW engagement management
  - On-board Combat Management System interface

THALES

TCS Product

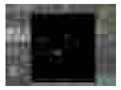
Cockpit Display



MPA UE

MSA LE

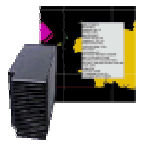
Laptop Computer (training)



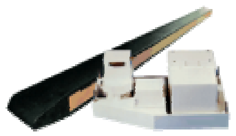
Ocean Master Radar

FLIR

AIS

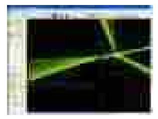


Anti-pollution Sensors

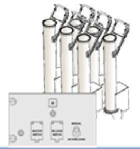


Self protection

COMINT (option)

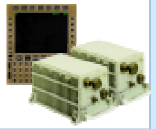


Sonobuoy & marker launchers

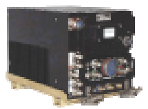


Mission Data Recorder

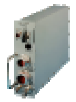
Navigation System



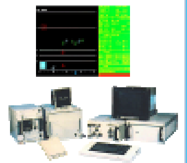
IFF



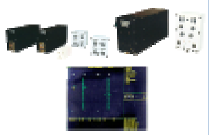
Comms Data Links Satcom



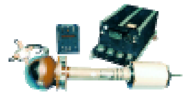
ESM



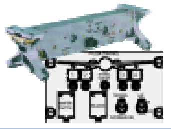
Acoustic (TMS 2000)



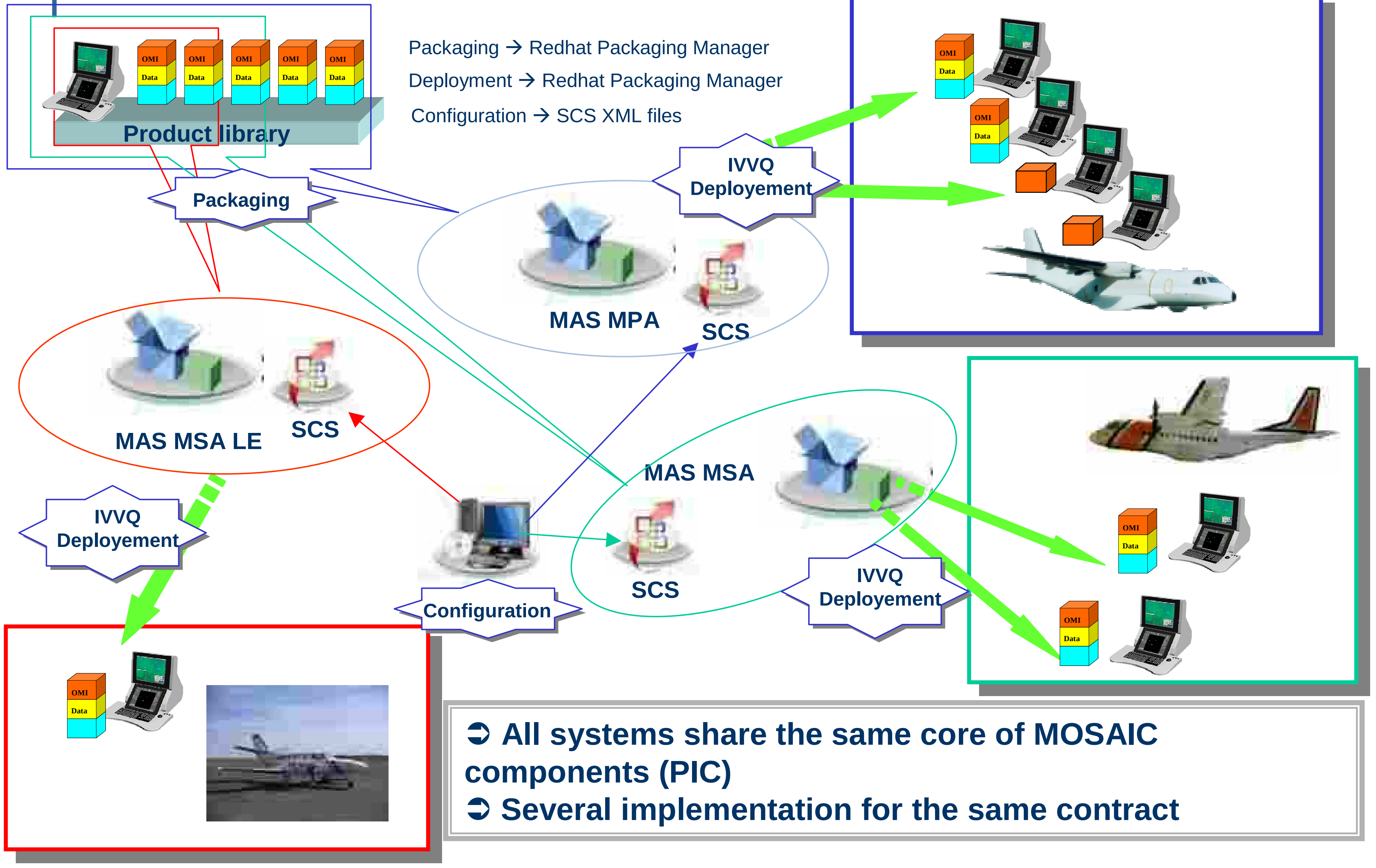
MAD



Torpedoes & Depth charges



Product line : a configurable system



Packaging → Redhat Packaging Manager

Deployment → Redhat Packaging Manager

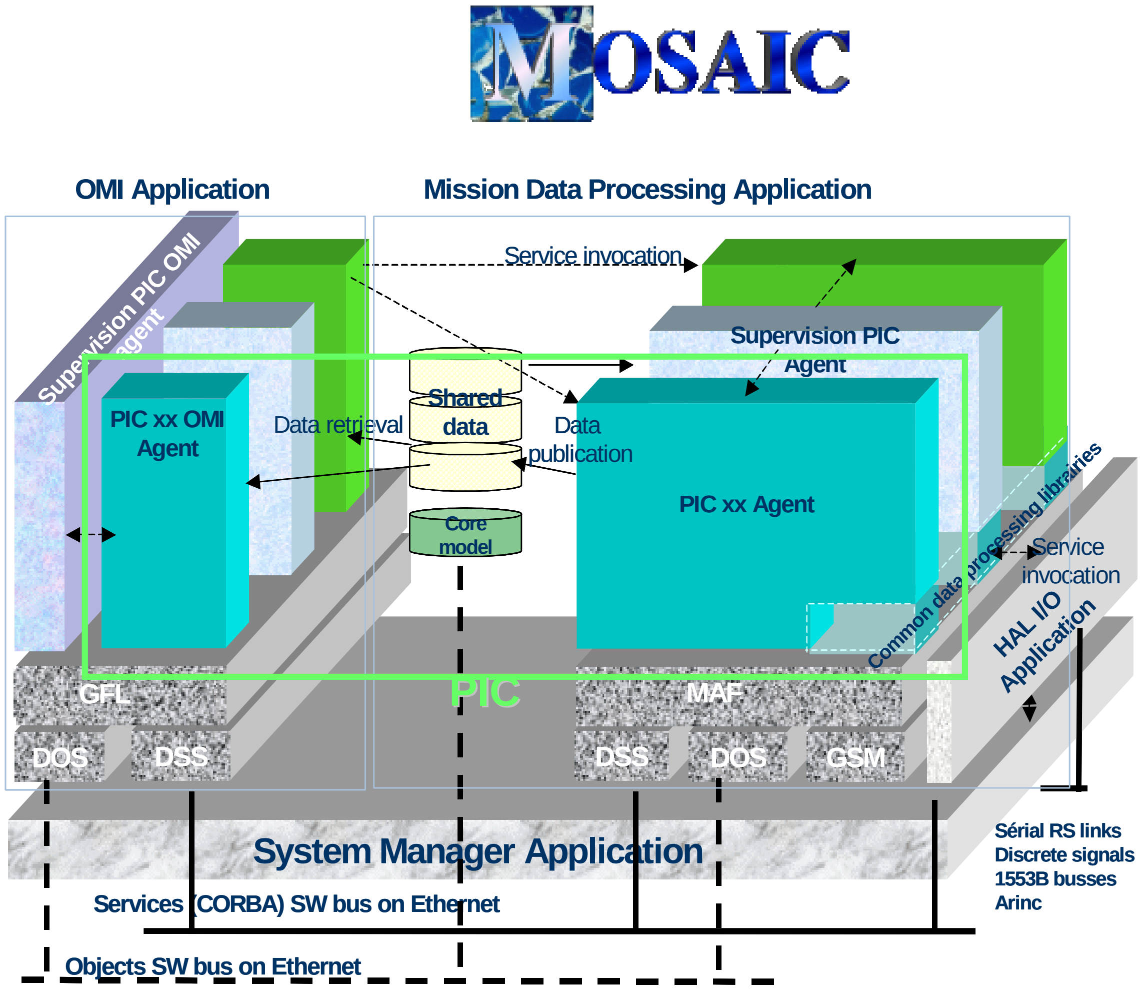
Configuration → SCS XML files

➲ **All systems share the same core of MOSAIC components (PIC)**

➲ **Several implementation for the same contract**

THALES

MOSAIC Architecture



MOSAIC
« Data Centric » Architecture

- To provide mechanism to share the Tactical Situation
- To provide Command & Control functions
- To minimize integration Risks
- To allow System Product Evolution & Adaptation
- To give a Common Framework to multi-company projects
- Scalability & performances (through data base caching, data and agents distribution…)
- Reliability (when an agent crashes, others can survive)
- Interoperability through Corba standard compliant services
- Extensibility (ability to add / modify agents, equipment, with no / few modification)
- High software and system integration productivity
- Openness (to various languages, hardware platforms, COTS software products integration…)
- Long term availability (through standard based, COTS independence and market driven COTS / technologies choices)

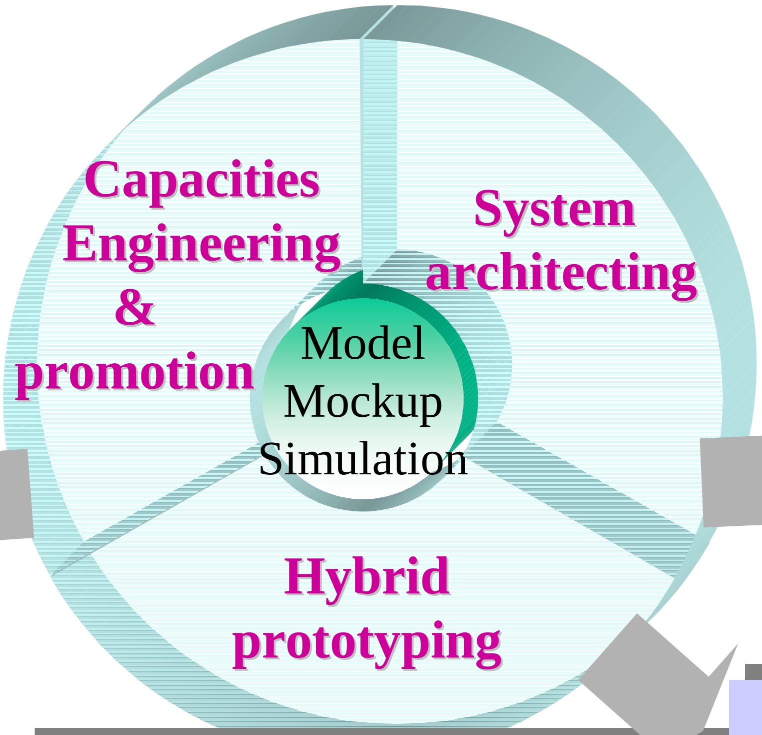
THALES

Modeling, Mockup and Simulation

New concepts engineering

- Operational concept mining
- Usage concept mining
- Human factors
- Value analysis
- Demonstrator for show room

Innovate
Define the right offer
& promote it



System solutions engineering

- Technical system parameters identification
- System sizing
- Performance engineering
- TRL assessment
- Design solution assessment
- Impact analysis (new function/ new technology)

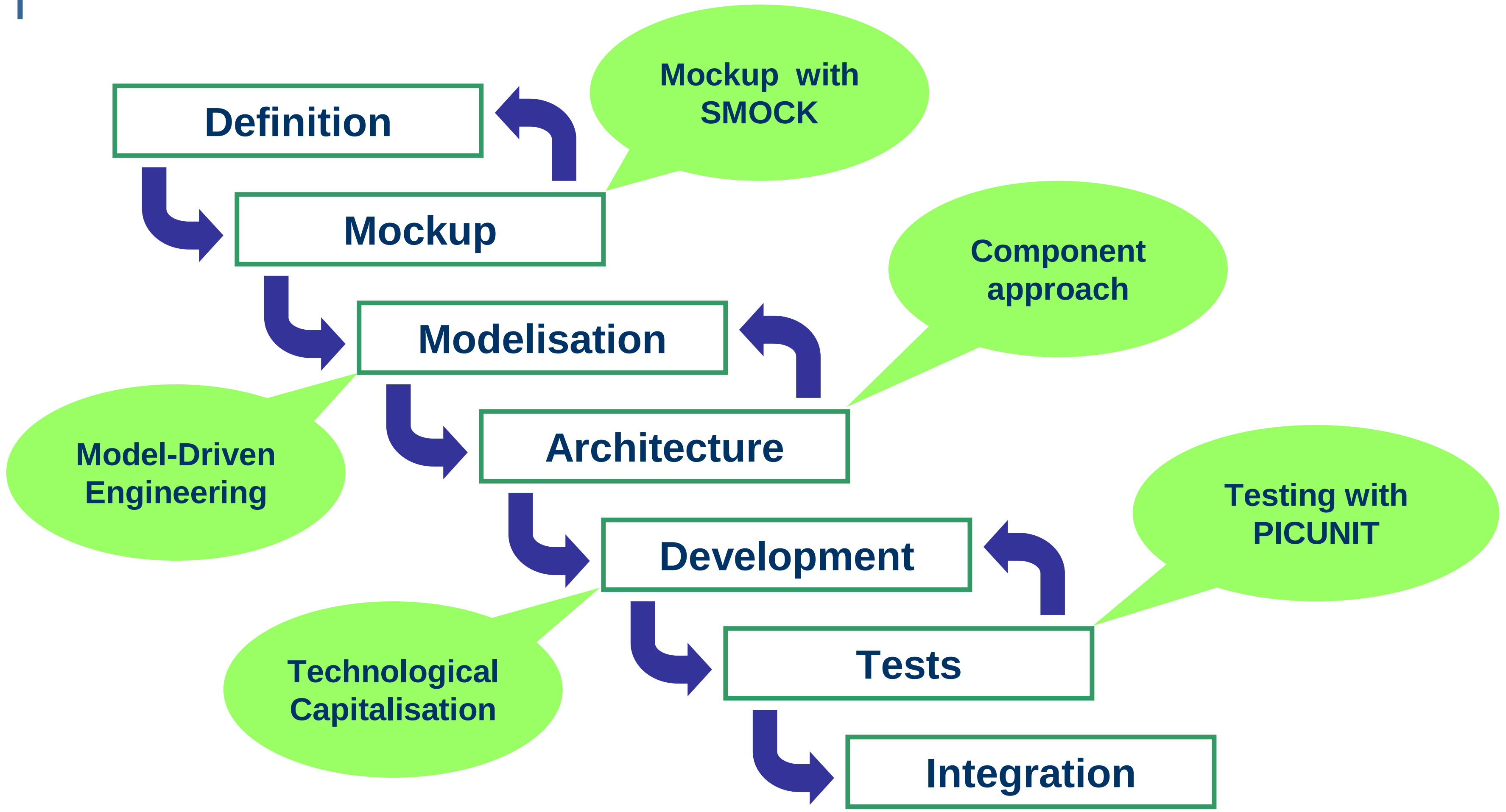
Define the right architecture

Secure the development

Technical solutions engineering

- Prototyping
- Pilot development
- **Development support**:
  - Specifications validation
  - Pre integration & verification (host)

Software production process

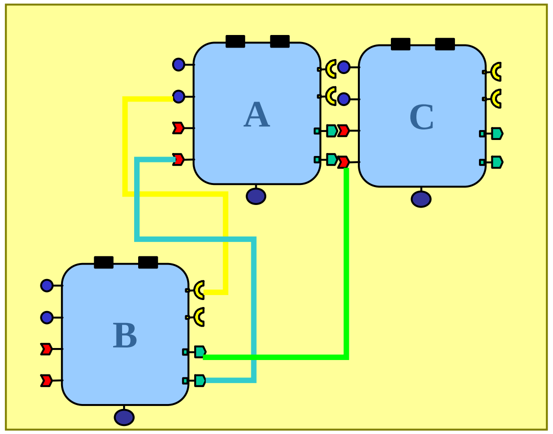


THALES

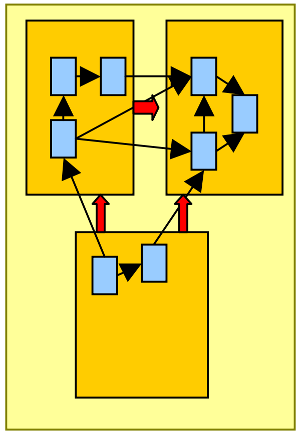
Agenda

Component: Aim Architecture

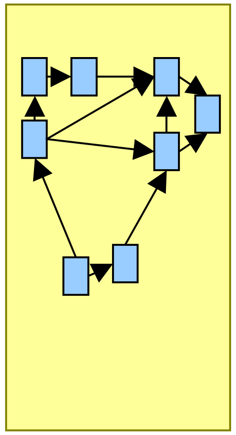
Component approach structures the system and the development



- Component
- Architecture
- Packaged
- Contract



- Object
- Architecture appear
- Packaged
- No contract



- Object
- No architecture
- No contract
- flat

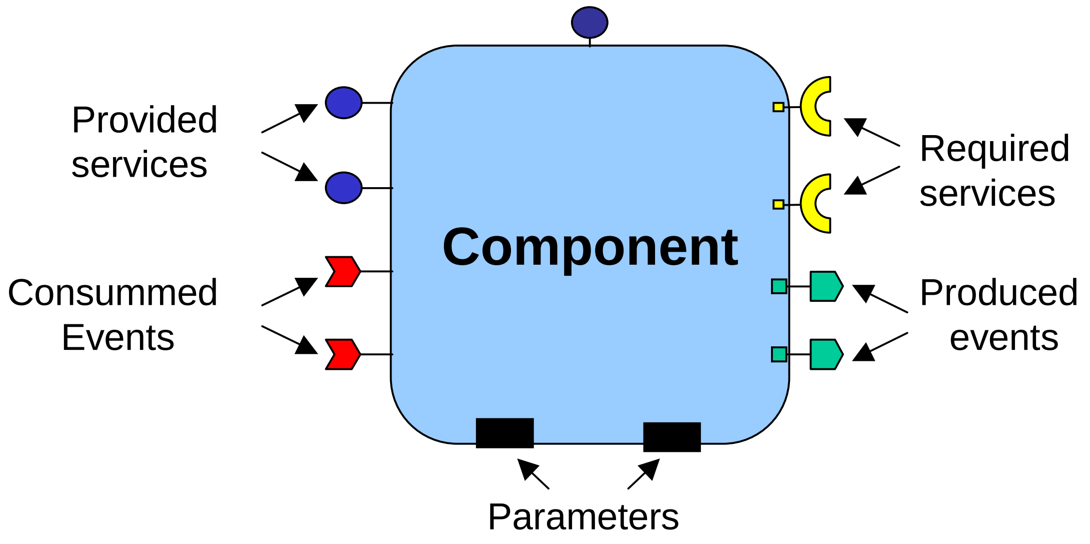
Component approach : SCM

Smalltalk Component Model

- CCM Like Component approach inherited from Jaguar (Thales/DCNS Java Framework)

Introduction of interface in Smalltalk

Notion of component manager, deployment services home services, locator services, …



THALES

Example

Interface definition

```
MyNameSpace
        defineInterface: #MyServices
        super: nil
        with: #(serviceOne: serviceTwo )
```

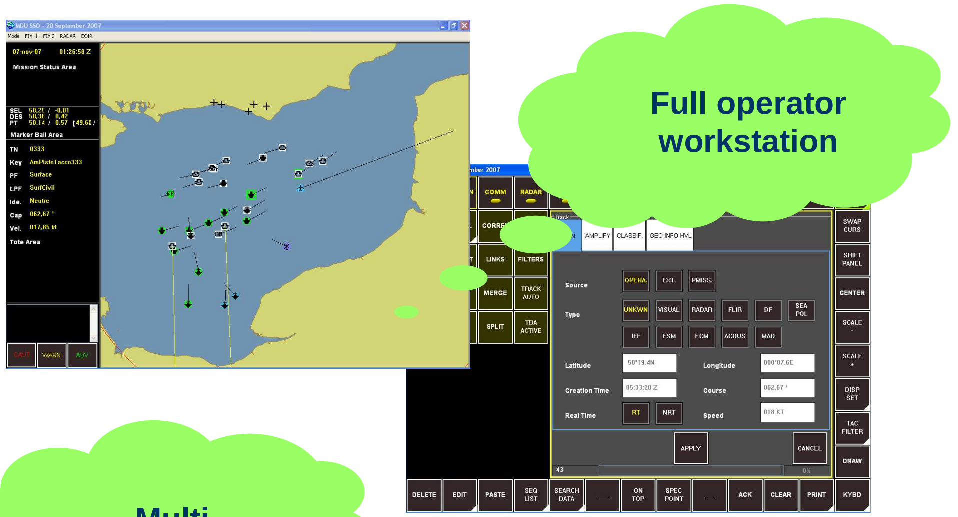
Component definition

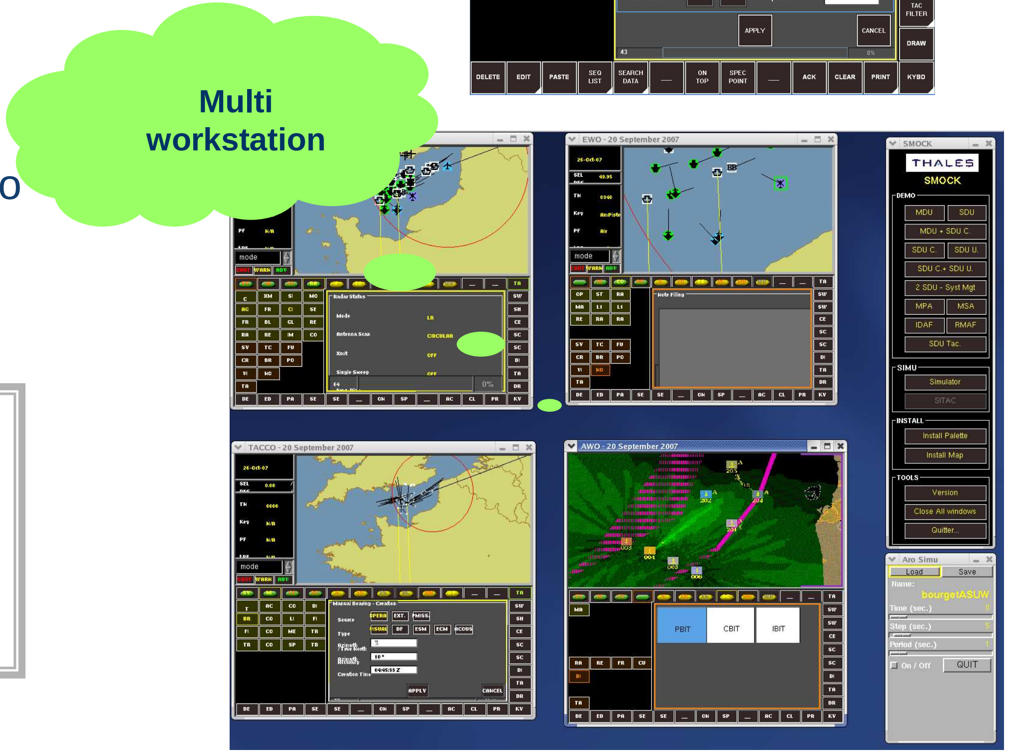
```
defineComponent
        self
                providedServices: #(#{MyNameSpace.MyServices})
                requiredServices: #()
                consumedEvents: #()
                producedEvents: #(#{MyNameSpace.MyEvents})
```

Mockup environment : SMOCK

Agile development environment:

- Full operator workstation mockup
- Component based architecture modeling
- Interaction behaviour simulation
- Multi workstation on a simple laptop
- Easy sharing between all actors including customer:
  - easy packaging
  - easy deployment on a simple laptop
- Early evaluation of relevant scenario
- Integrated development cost model





➲ **Few days to one month for a new mockup**
➲ **Significant evolution in real time**
➲ **Collaborative work**

On the future

Architecture modeling

- From system architecture to executable model in Smalltalk

Link from Mockup to System and Software tools

- Models and Requirements definition
- Product line management
- Configuration management
- Code generation

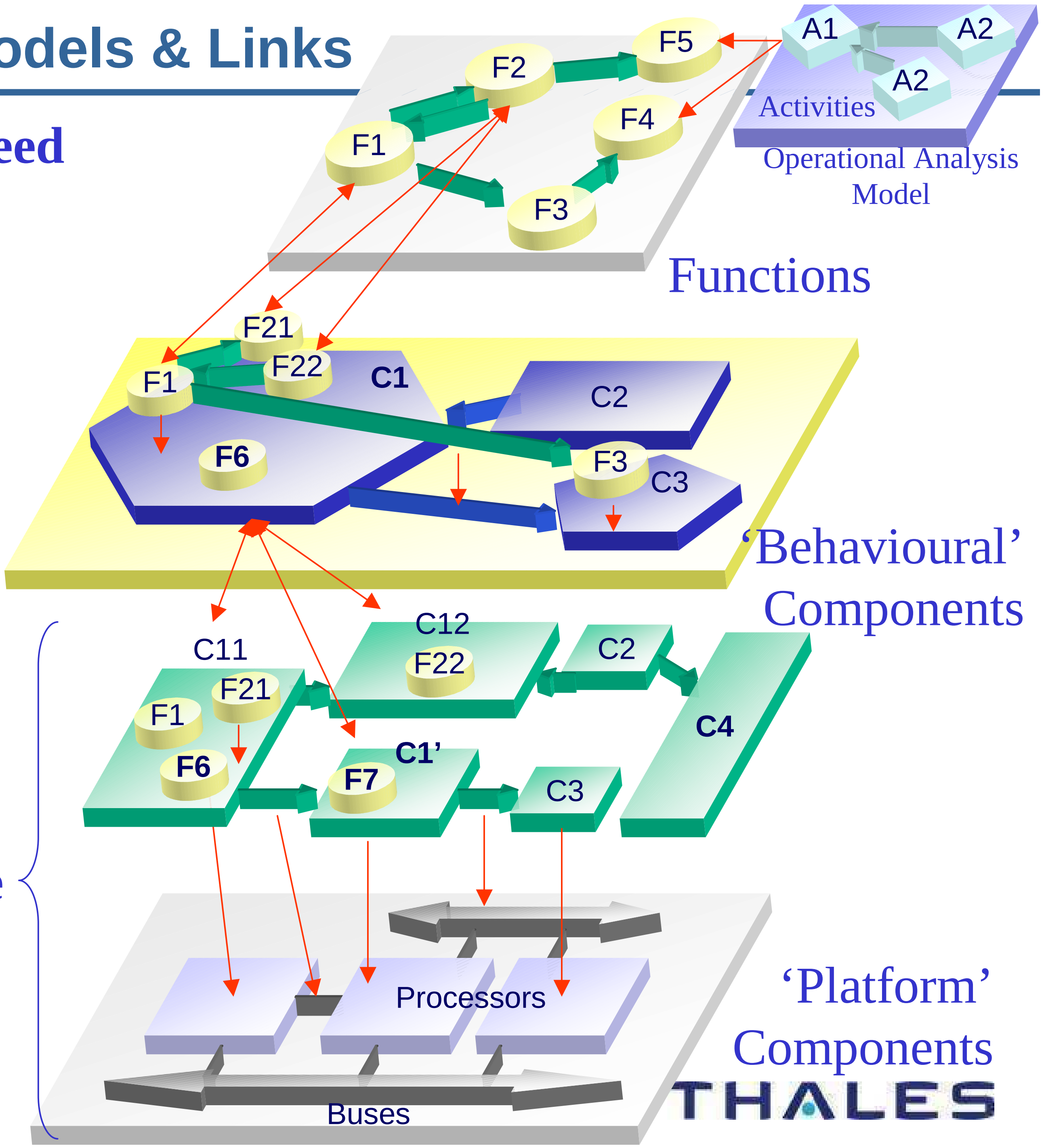
THALES

Global View of Models & Links

System Functional Need Model

Logical Architecture Model

Physical Architecture Model

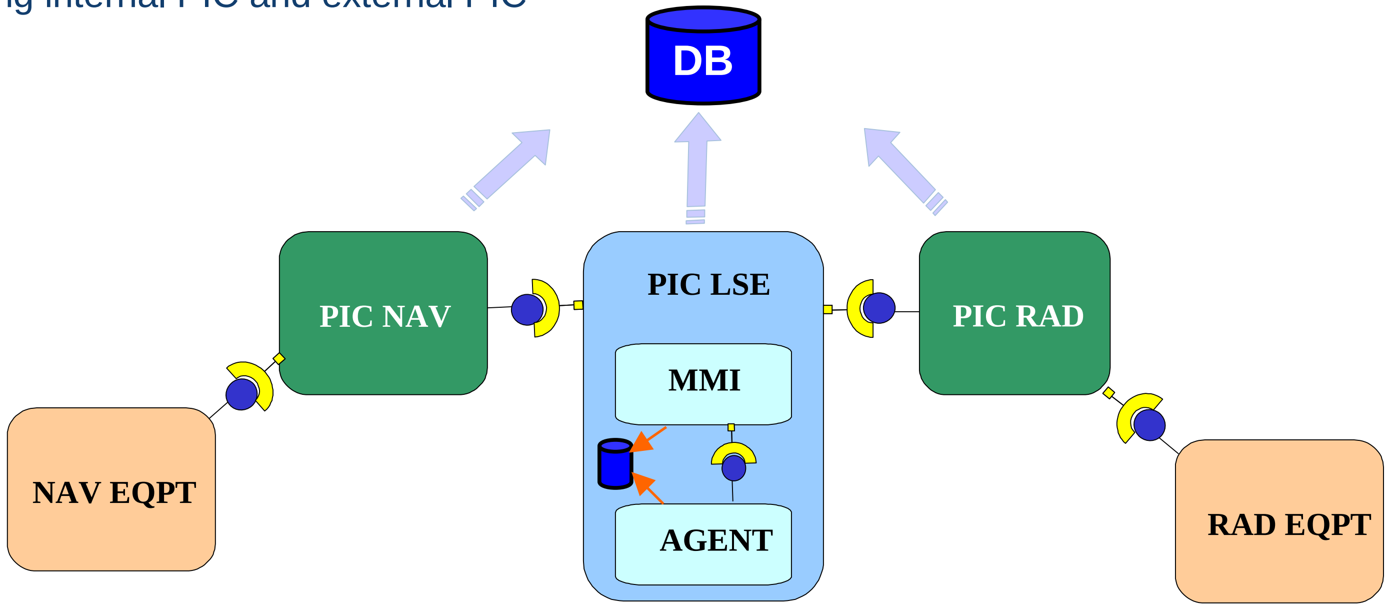


Agenda

PicUnit tool in which context?

Context :

- Component architecture with few levels
- Service interfaces : Corba
- Datamodel interface : Database
- Dev context : other components not yet available when developping a component
- Testing internal PIC and external PIC



Validate component itself and also communication with other components around

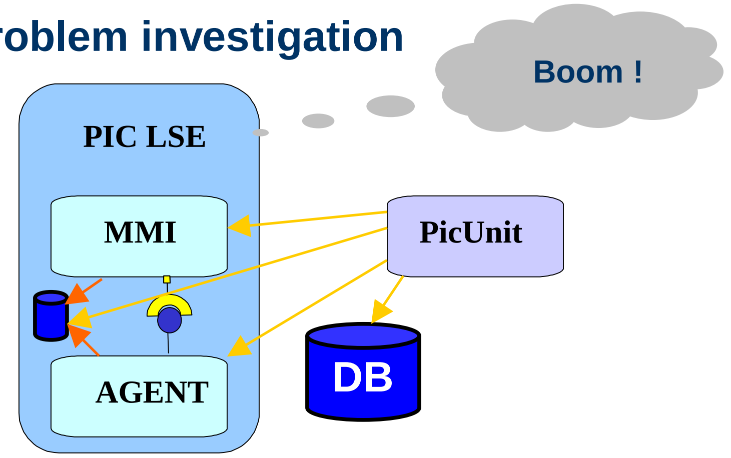
THALES

PicUnit tool for which usages?

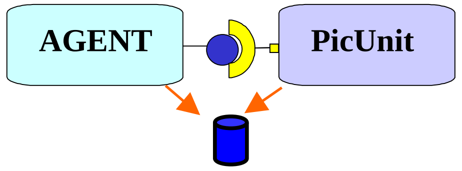
Usages :

- Unit test for component
- Simulate middleware interfaces
- Simulate other components
- Swiss knife for problem investigation
- Prepare a defined state
- Simulators

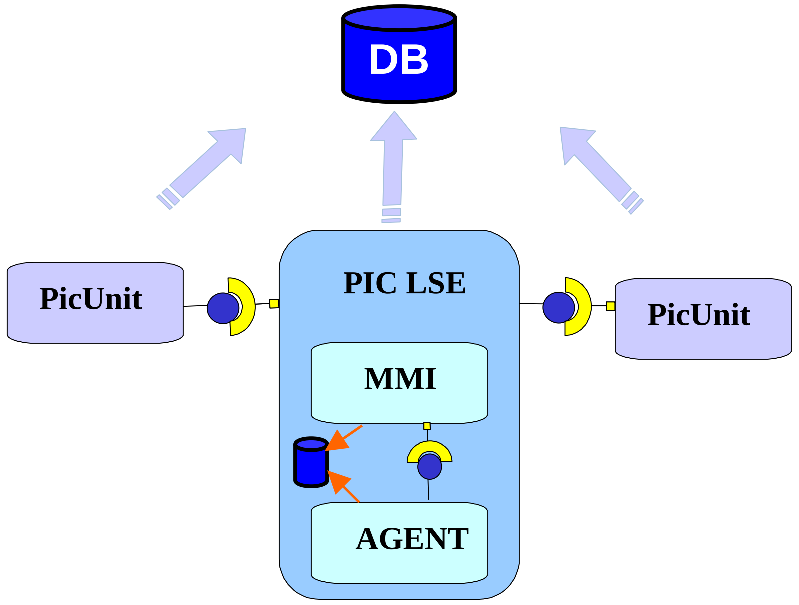
Problem investigation



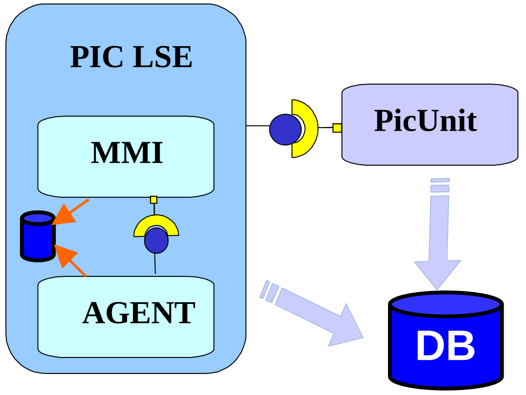
Testing a standalone component



Simulate other components

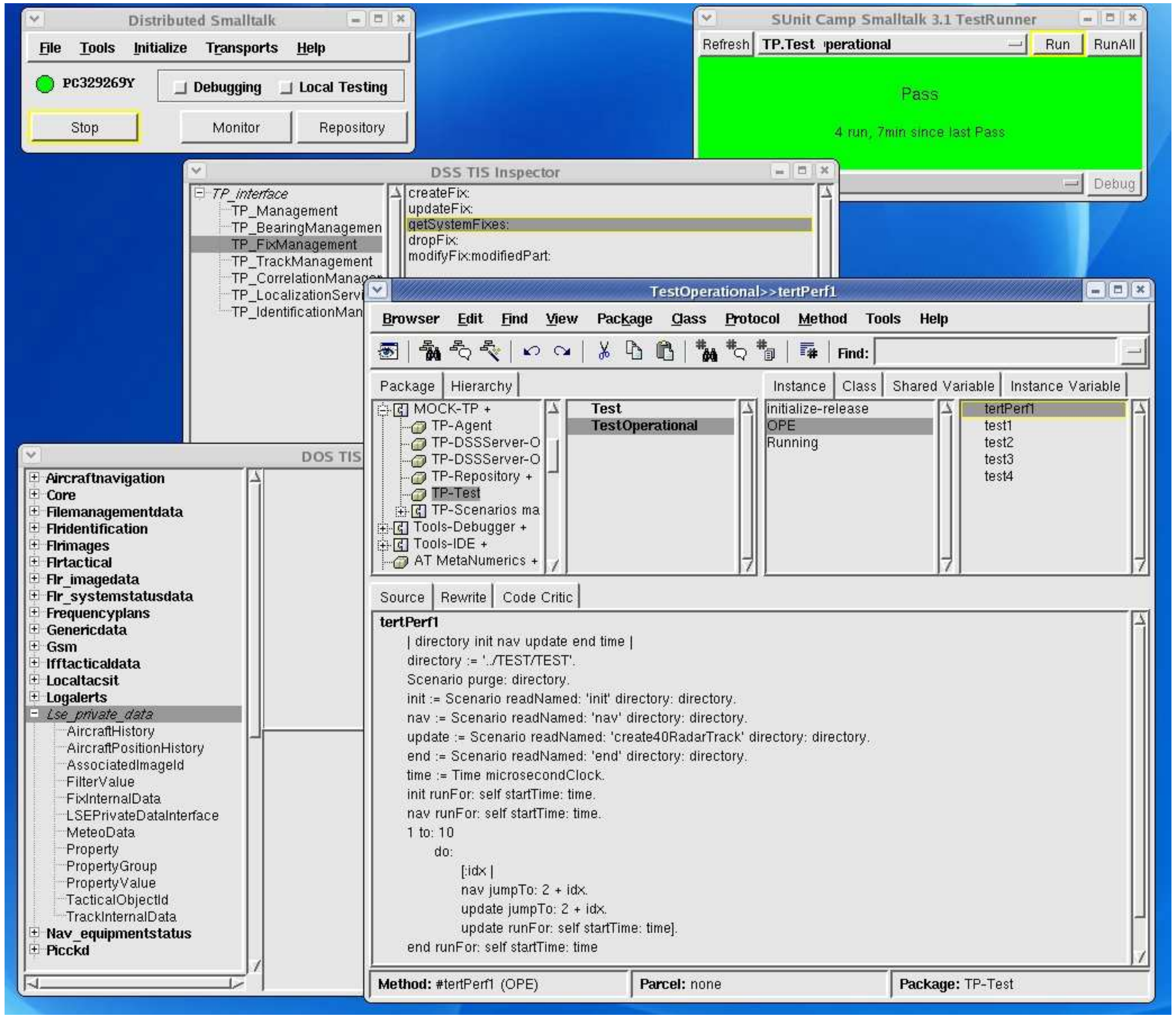


Testing a Standalone PIC



THALES

PicUnit Screenshot



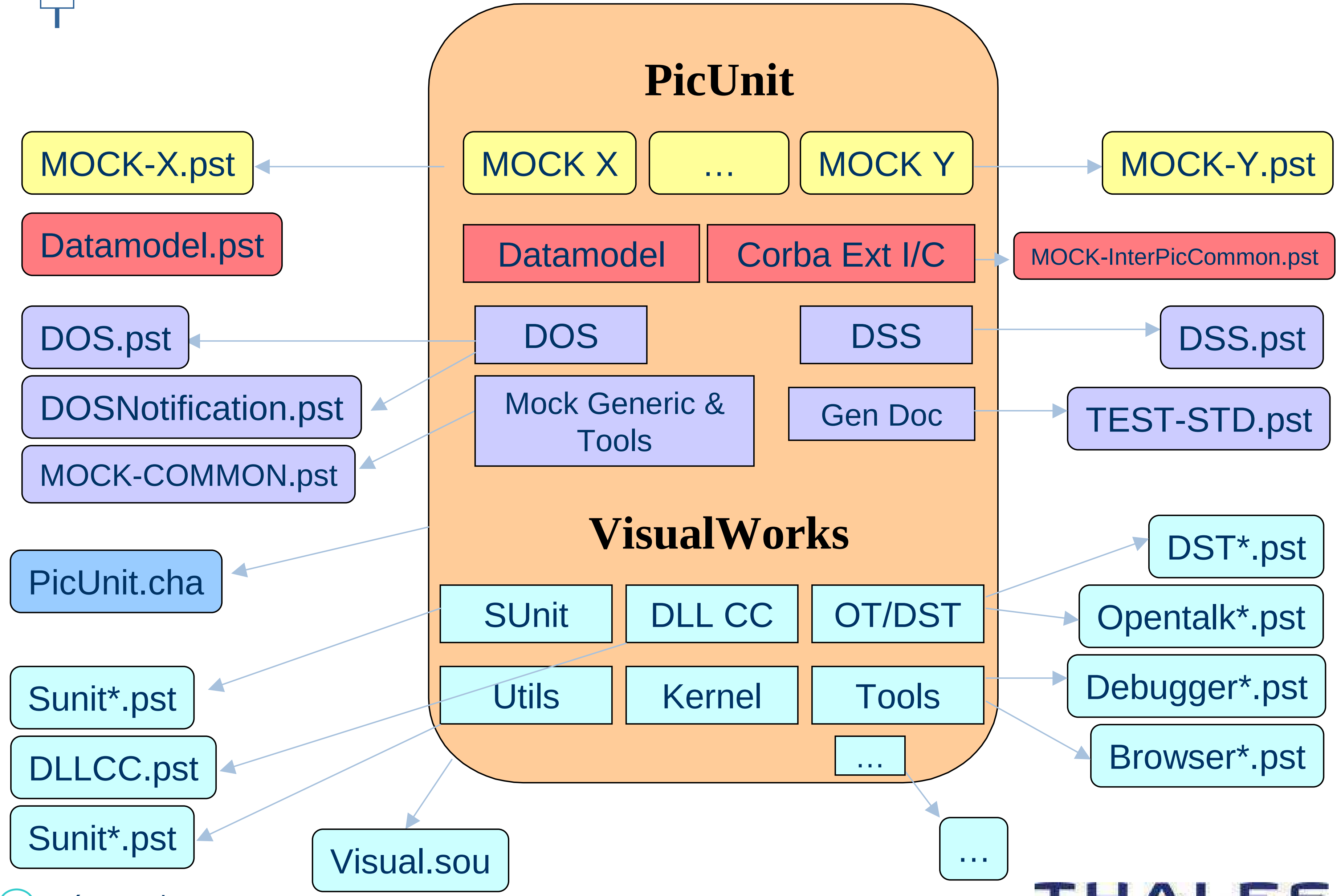
```
tertPerf1
    | directory init nav update end time |
    directory := '../TEST/TEST'.
    Scenario purge: directory.
    init := Scenario readNamed: 'init' directory: directory.
    nav := Scenario readNamed: 'nav' directory: directory.
    update := Scenario readNamed: 'create40RadarTrack' directory: directory.
    end := Scenario readNamed: 'end' directory: directory.
    time := Time microsecondClock.
    init runFor: self startTime: time.
    nav runFor: self startTime: time.
    1 to: 10
        do:
            [:idx |
            nav jumpTo: 2 + idx.
            update jumpTo: 2 + idx.
            update runFor: self startTime: time].
    end runFor: self startTime: time
```

Method: #tertPerf1 (OPE) **Parcel:** none **Package:** TP-Test

Reference - date

THALES

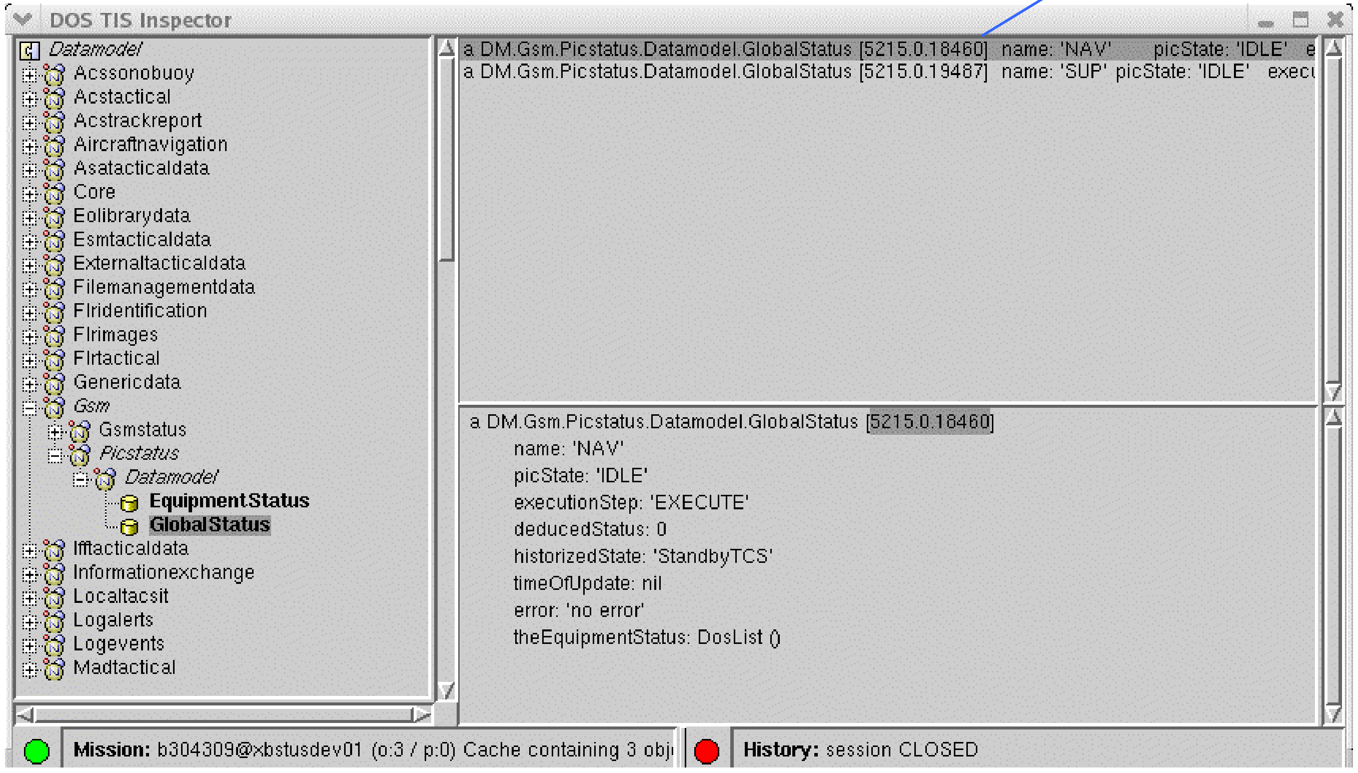
PicUnit Map


PicUnit
MOCK X
…
MOCK Y
MOCK-X.pst
MOCK-Y.pst
Datamodel.pst
Datamodel
Corba Ext I/C
MOCK-InterPicCommon.pst
DOS.pst
DOS
DSS
DSS.pst
DOSNotification.pst
Mock Generic & Tools
Gen Doc
TEST-STD.pst
MOCK-COMMON.pst
VisualWorks
PicUnit.cha
DST*.pst
SUnit
DLL CC
OT/DST
Opentalk*.pst
Utils
Kernel
Tools
Debugger*.pst
Sunit*.pst
DLLCC.pst
…
Browser*.pst
Sunit*.pst
Visual.sou
…

DOSTool

DB Inspector

- Browse the database objects
- Generate Datamodel smalltalk classes
- Create / Delete / Modify object
- Same features available in the Tests via an API

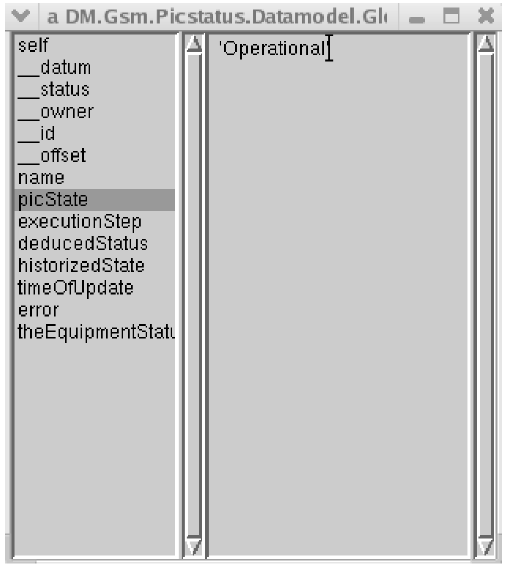


```
a DM.Gsm.Picstatus.Datamodel.GlobalStatus [5215.0.18460]  name: 'NAV'     picState: 'IDLE'   e
a DM.Gsm.Picstatus.Datamodel.GlobalStatus [5215.0.19487]  name: 'SUP' picState: 'IDLE'   execu
```

```
a DM.Gsm.Picstatus.Datamodel.GlobalStatus [5215.0.18460]
      name: 'NAV'
      picState: 'IDLE'
      executionStep: 'EXECUTE'
      deducedStatus: 0
      historizedState: 'StandbyTCS'
      timeOfUpdate: nil
      error: 'no error'
      theEquipmentStatus: DosList ()
```

Mission: b304309@xbstusdev01 (o:3 / p:0) Cache containing 3 obj **History:** session CLOSED

Inspect it



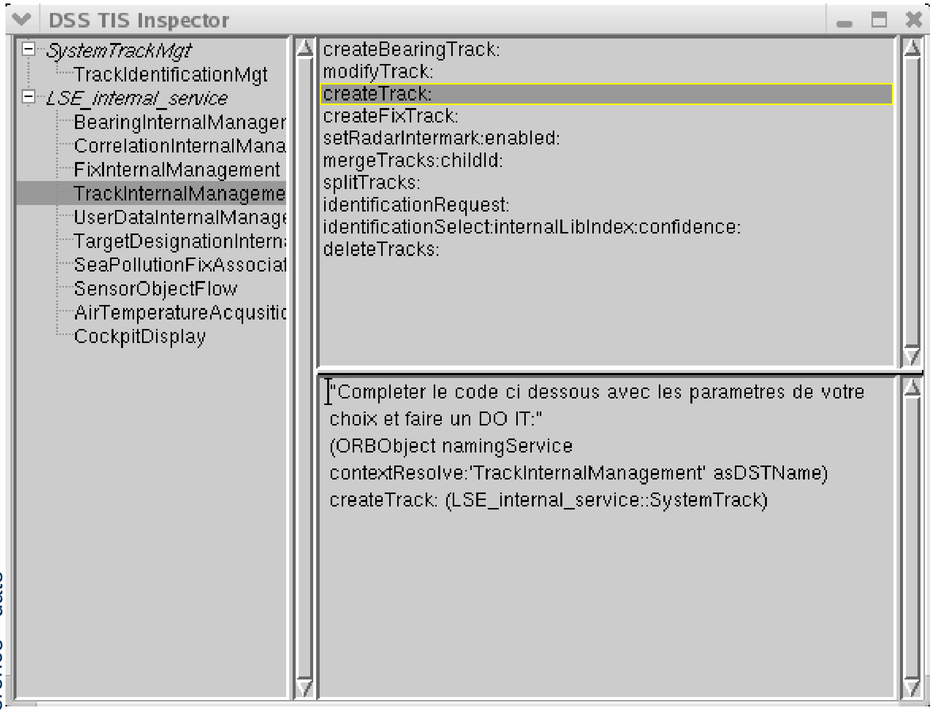
```
'Operational'
```

Modify object and Accept

THALES

DSS Tool

- Load an IDL file or an alreaded recorded module in Repository
- Corba modules tree view : Module > Interfaces > services
- Generate services (a Module = a Package + a namespace)
- Simple usage : register I/C in NS, call service by DoIt, unregister I/C
- Service argument editor

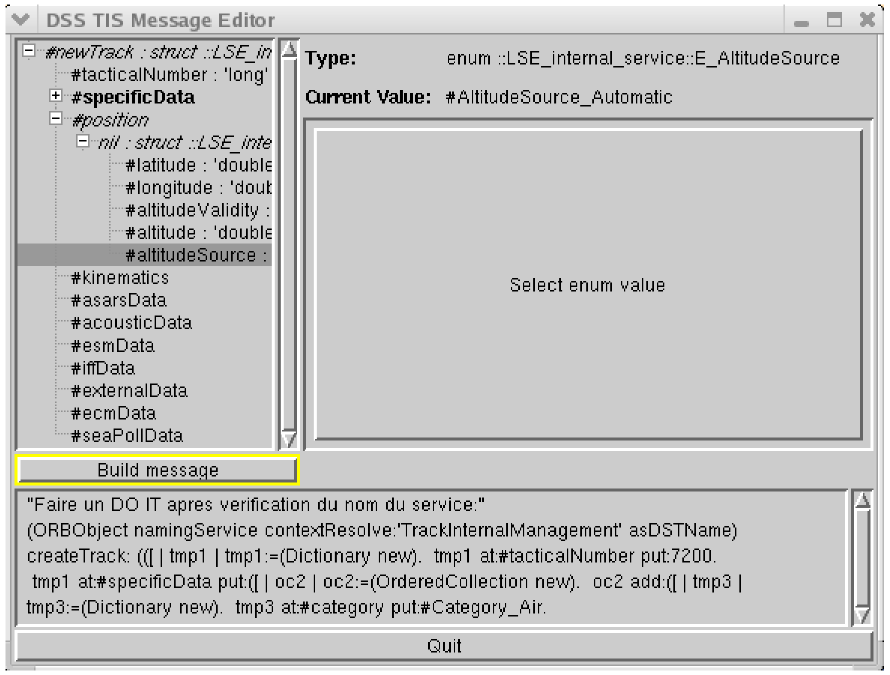


SystemTrackMgt
TrackIdentificationMgt
LSE_internal_service
BearingInternalManager
CorrelationInternalMana
FixInternalManagement
TrackInternalManageme
UserDataInternalManage
TargetDesignationIntern
SeaPollutionFixAssociat
SensorObjectFlow
AirTemperatureAcquisitio
CockpitDisplay

createBearingTrack:
modifyTrack:
createTrack:
createFixTrack:
setRadarIntermark:enabled:
mergeTracks:childId:
splitTracks:
identificationRequest:
identificationSelect:internalLibIndex:confidence:
deleteTracks:

```
"Completer le code ci dessous avec les parametres de votre
choix et faire un DO IT:"
(ORBObject namingService
contextResolve:'TrackInternalManagement' asDSTName)
createTrack: (LSE_internal_service::SystemTrack)
```

Browse modules / (un)register in NS / services call



#newTrack : struct ::LSE_in
#tacticalNumber : 'long'
#specificData
#position
nil : struct ::LSE_inte
#latitude : 'double
#longitude : 'doub
#altitudeValidity :
#altitude : 'double
#altitudeSource :
#kinematics
#asarsData
#acousticData
#esmData
#iffData
#externalData
#ecmData
#seaPollData

Type: enum ::LSE_internal_service::E_AltitudeSource

Current Value: #AltitudeSource_Automatic

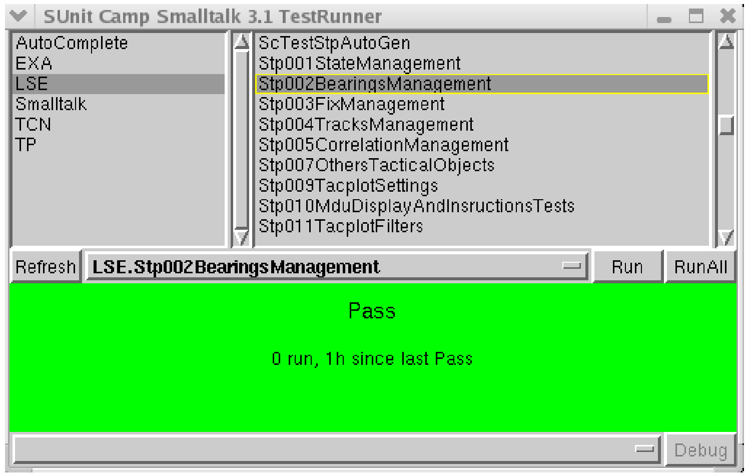
Select enum value

```
"Faire un DO IT apres verification du nom du service:"
(ORBObject namingService contextResolve:'TrackInternalManagement' asDSTName)
createTrack: (([ | tmp1 | tmp1:=(Dictionary new).  tmp1 at:#tacticalNumber put:7200.
 tmp1 at:#specificData put:([ | oc2 | oc2:=(OrderedCollection new).  oc2 add:([ | tmp3 |
tmp3:=(Dictionary new).  tmp3 at:#category put:#Category_Air.
```

Edit complex arguments / service call

SUnit

- Famous xUnit framework
- Dev context : launch a test and debug it in the browser
- Qualification context: launch all tests in TestRunner
- Generate Test Description document (STD)



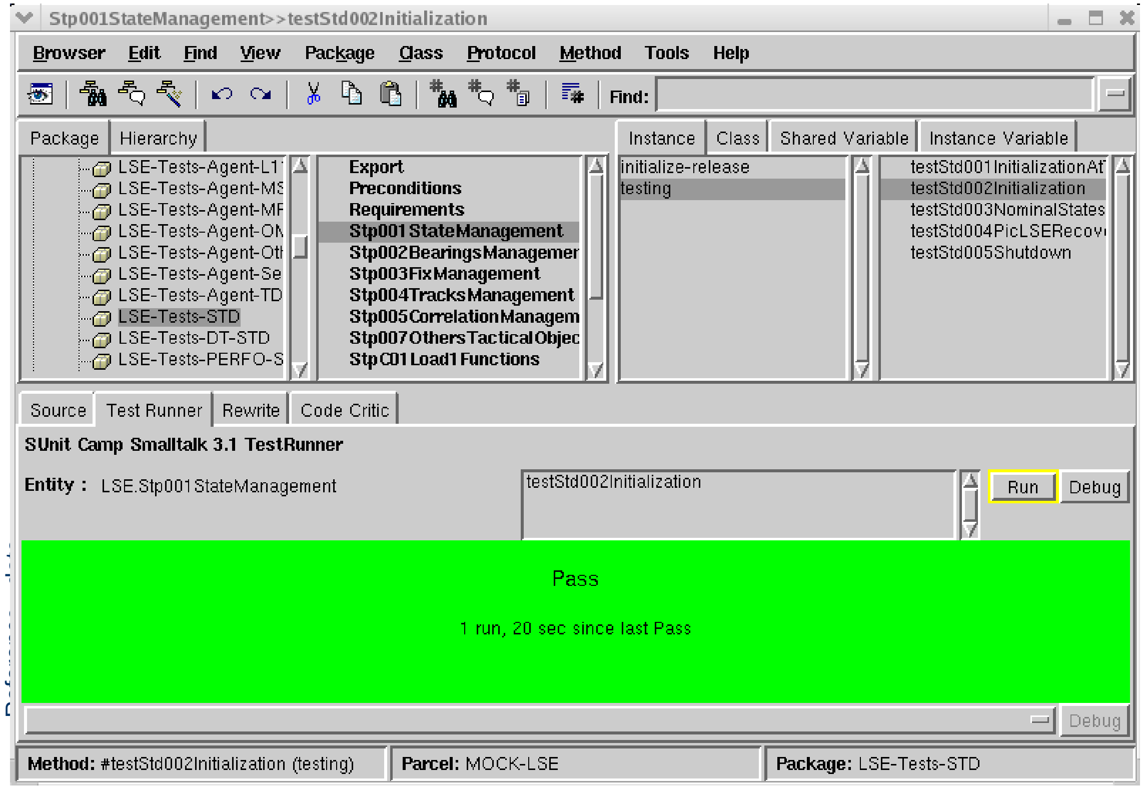
AutoComplete
EXA
LSE
Smalltalk
TCN
TP

ScTestStpAutoGen
Stp001StateManagement
Stp002BearingsManagement
Stp003FixManagement
Stp004TracksManagement
Stp005CorrelationManagement
Stp007OthersTacticalObjects
Stp009TacplotSettings
Stp010MduDisplayAndInsructionsTests
Stp011TacplotFilters

LSE.Stp002BearingsManagement

Pass

0 run, 1h since last Pass



LSE-Tests-Agent-L1
LSE-Tests-Agent-MS
LSE-Tests-Agent-MF
LSE-Tests-Agent-ON
LSE-Tests-Agent-Ot
LSE-Tests-Agent-Se
LSE-Tests-Agent-TD
LSE-Tests-STD
LSE-Tests-DT-STD
LSE-Tests-PERFO-S

Export
Preconditions
Requirements
Stp001StateManagement
Stp002BearingsManagemer
Stp003FixManagement
Stp004TracksManagement
Stp005CorrelationManagem
Stp007OthersTacticalObjec
StpC01Load1Functions

initialize-release
testing

testStd001InitializationAt
testStd002Initialization
testStd003NominalStates
testStd004PicLSERecov
testStd005Shutdown

SUnit Camp Smalltalk 3.1 TestRunner

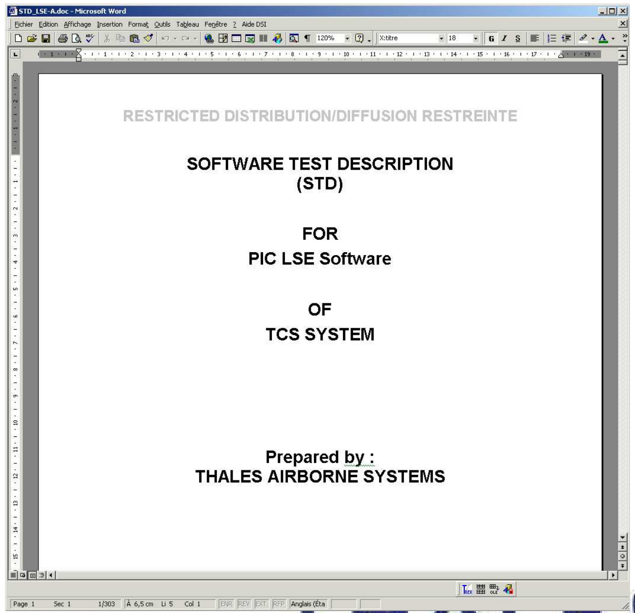
Entity : LSE.Stp001StateManagement

testStd002Initialization

Pass

1 run, 20 sec since last Pass

Method: #testStd002Initialization (testing) **Parcel:** MOCK-LSE **Package:** LSE-Tests-STD



RESTRICTED DISTRIBUTION/DIFFUSION RESTREINTE

SOFTWARE TEST DESCRIPTION
(STD)

FOR
PIC LSE Software

OF
TCS SYSTEM

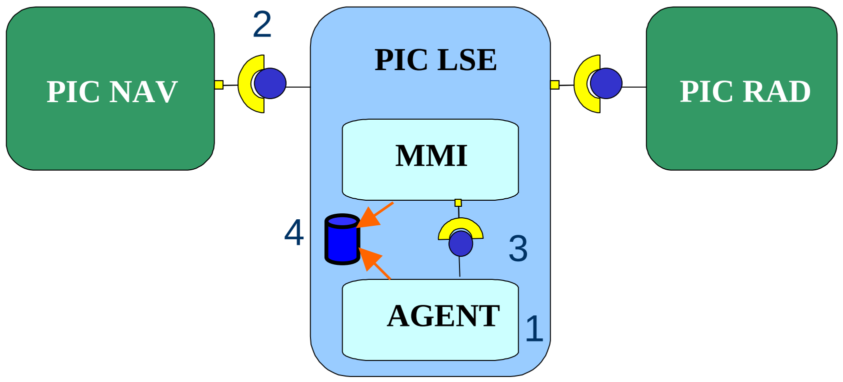
Prepared by :
THALES AIRBORNE SYSTEMS

a PIC = a Mock

a Mock = a smalltalk bundle than represents a PIC

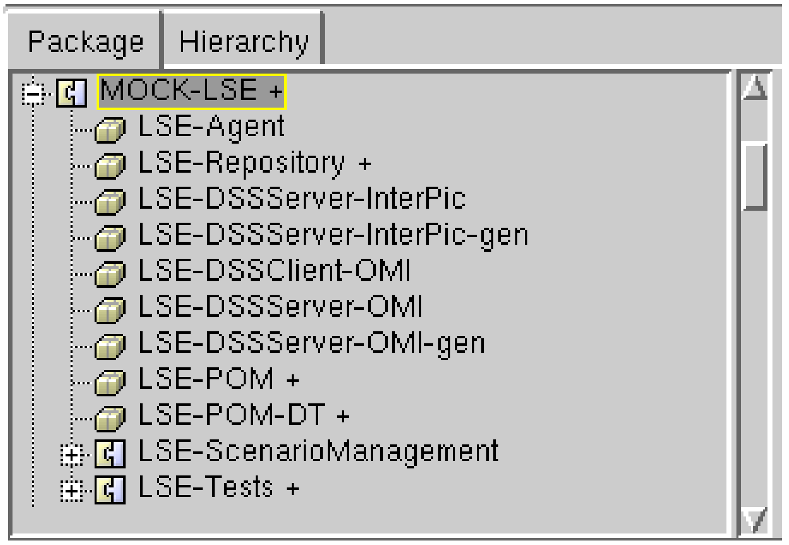
- Same interfaces
- Replace the PIC and/or control the PIC
- DSS server (generated + implementation)
- DSS client
- PIC Datamodel (generated + customization)
- Tests

The LSE PIC



The LSE Mock

1
2
3
4



- MOCK-LSE +
  - LSE-Agent
  - LSE-Repository +
  - LSE-DSSServer-InterPic
  - LSE-DSSServer-InterPic-gen
  - LSE-DSSClient-OMI
  - LSE-DSSServer-OMI
  - LSE-DSSServer-OMI-gen
  - LSE-POM +
  - LSE-POM-DT +
  - LSE-ScenarioManagement
  - LSE-Tests +

THALES

What next?

Link between PicUnit and Smock :

- Define Model in Smock and Generate it in PicUnit
- Generate source code for PIC Developement
- Generate tests for PIC

Integration with Continuous Integration Platform :

- Output TestResults in XMLUnit format
- Automaticaly test launching for each PIC changes (each commit in SCM)
- Instant Map of the project

Agenda

History : Smalltalk in Thales Brest

Domain Context

Modelling and mockup with Smock

Component Testing with PicUnit

Conclusions

Conclusion

Why we love it :

- RAD with regular and high quality language
- Efficient IDE
- Model oriented
- Easy code generation
- Very improved Debugger
- Swiss knife

but :

- Image concept in training context
- Important psychological first step
- Not widespread language

→ Is it the first model centric workshop ?

Questions ?

Engineering phase : the TFC

Technical Foundation Classes

=

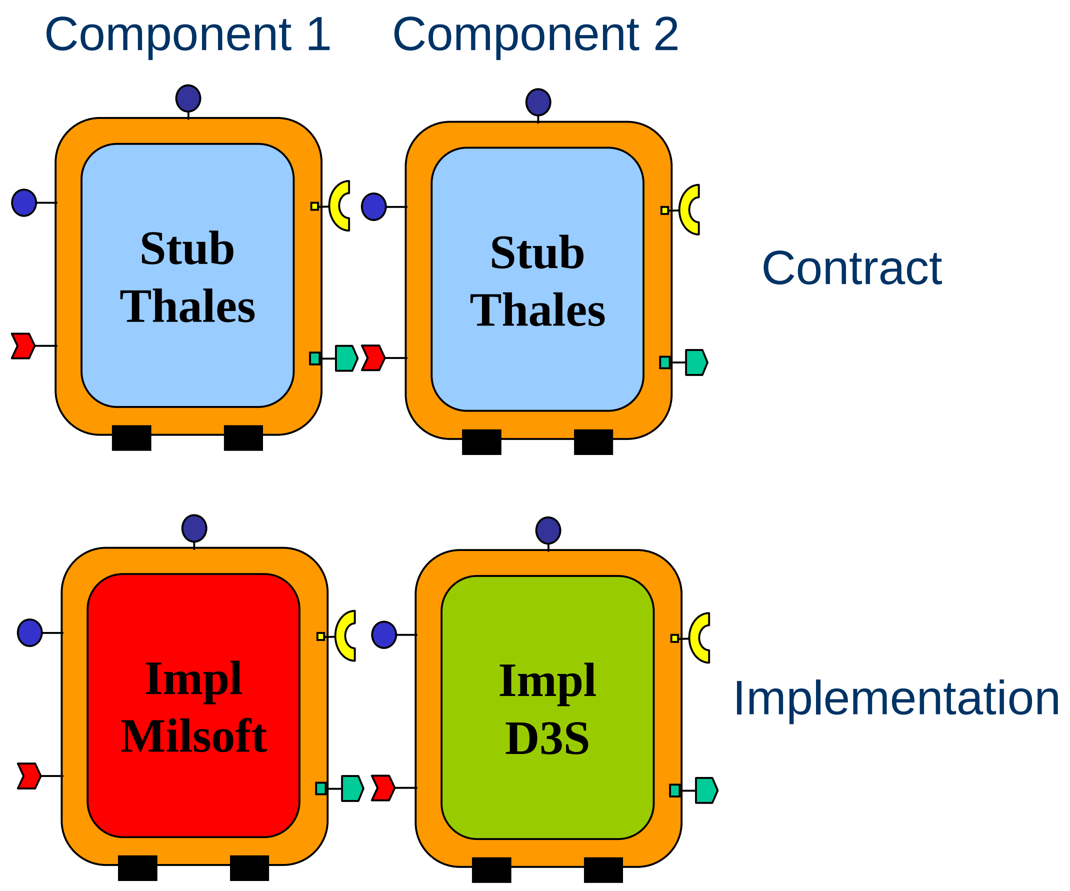
components interfaces

+

components stubs

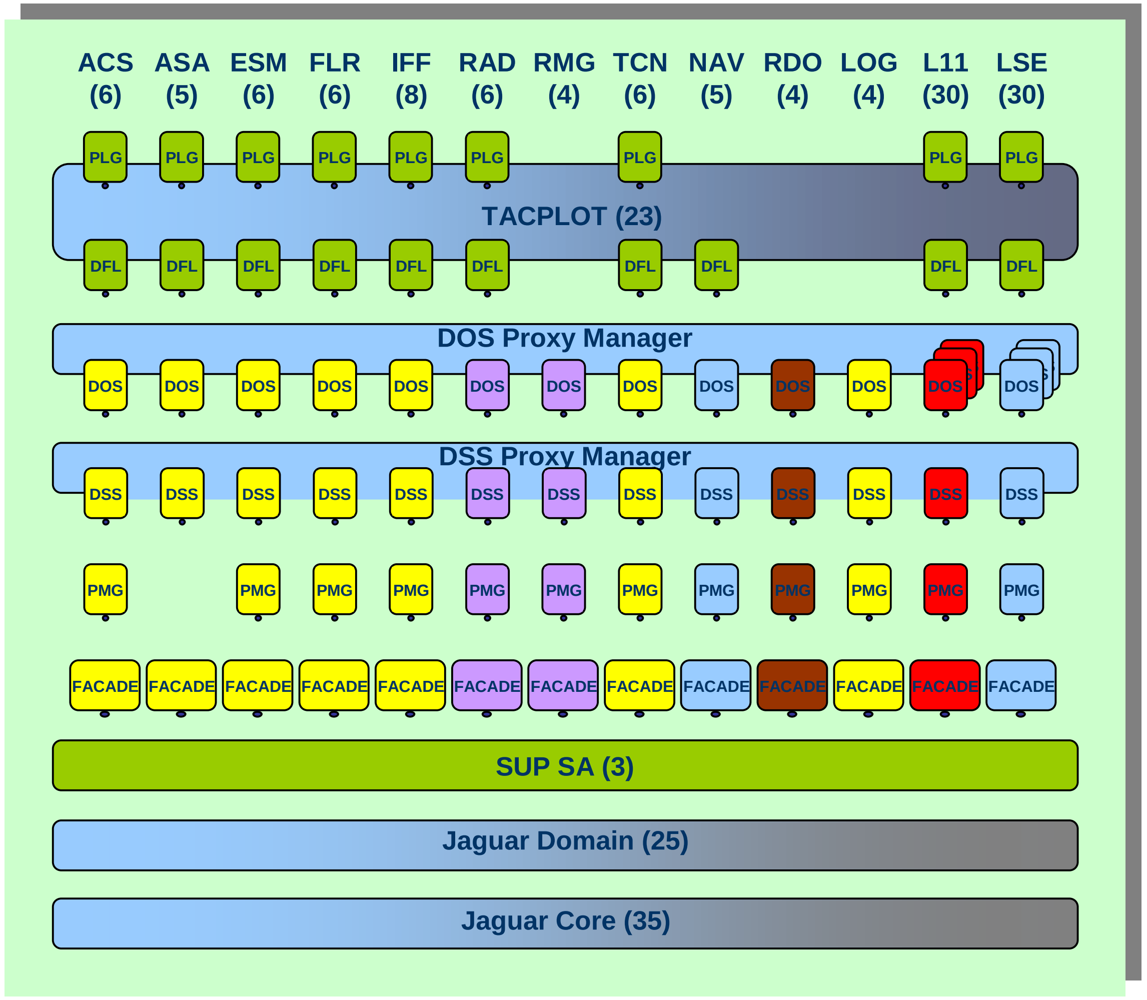
+

components description



➲ TFC provides a full contract (model, interface, stub, test)

Component cartography (partial)



PLG: Action On Selection Plugging

DFL: Data Flow

DOS: DOS Proxy

DOS: DSS Proxy

PMG: Panels Manager

Facade: SUP Facade

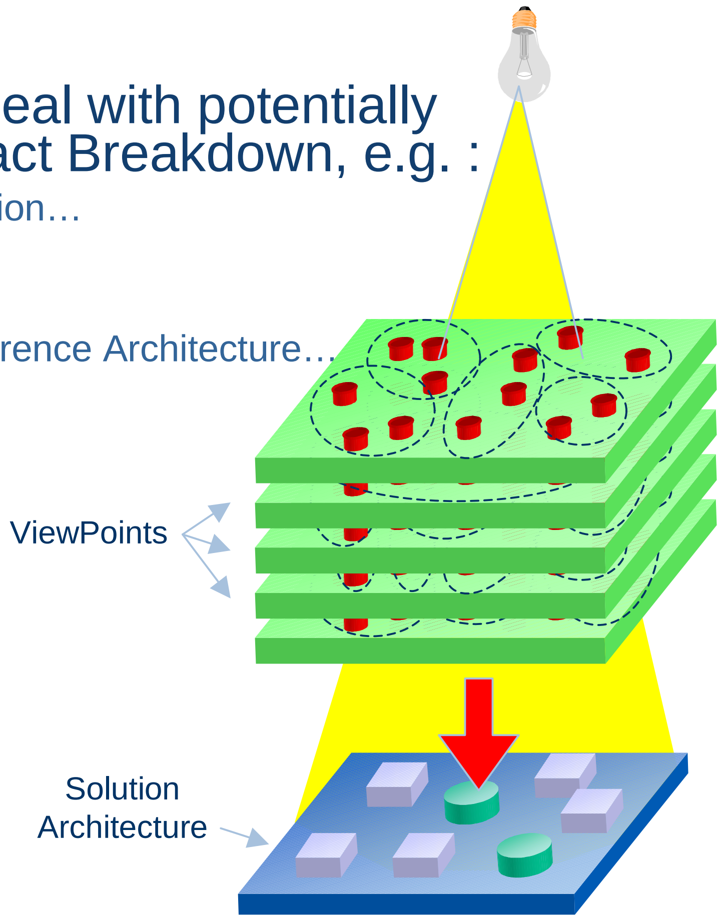
🟦	**Thales**
🟨	**Havelsan**
🟥	**Milsoft**
🟫	**Silicomp**
🟪	**Coframi**
🟩	**D3S**

➲ Total of more than 25 PICs and 300 Components

THALES

Architecture Design: the Art of Compromise

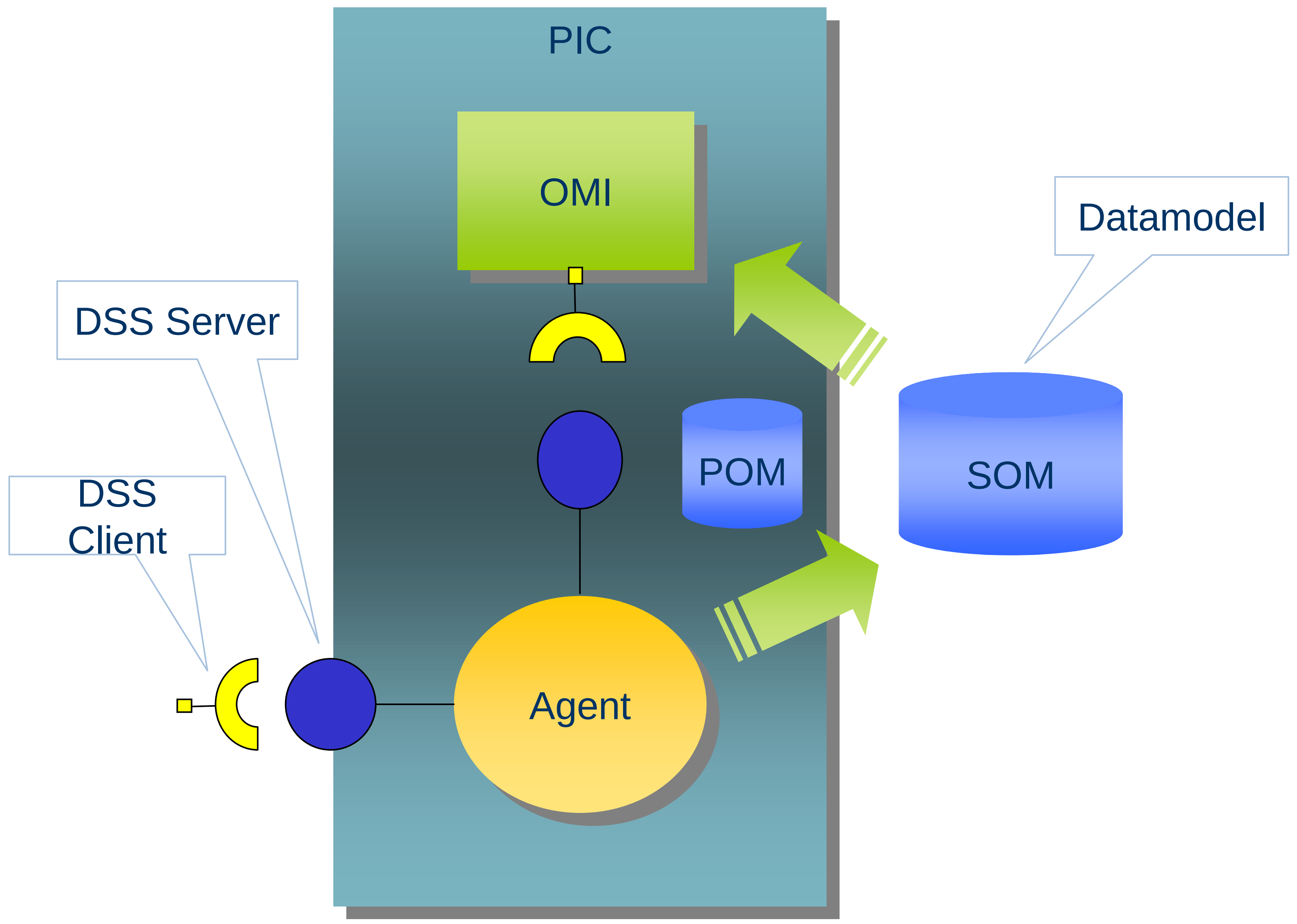
- The final Product Architecture must deal with potentially contradictory Constraints, which impact Breakdown, e.g. :
  - Safety, Dependability / Fault Tolerance, Certification…
  - Time-critical functional Paths
  - Performances & required [hardware] Resources
  - Mapping on [existing] hardware, middleware, reference Architecture…
  - Functional grouping Consistency
  - Dynamic Behaviour
  - System Modes & States
  - Complexity of internal interfaces
  - Human Factors
  - Dependency in System Integration
  - Security
  - Ease of sub-contracting
  - Reuse, existing Legacy, Product Line Policy
  - Modularity, Ability to evolve
  - Available technologies, COTS…



⇨**Building an appropriate Architecture means finding the most acceptable Compromise between these *Viewpoints***

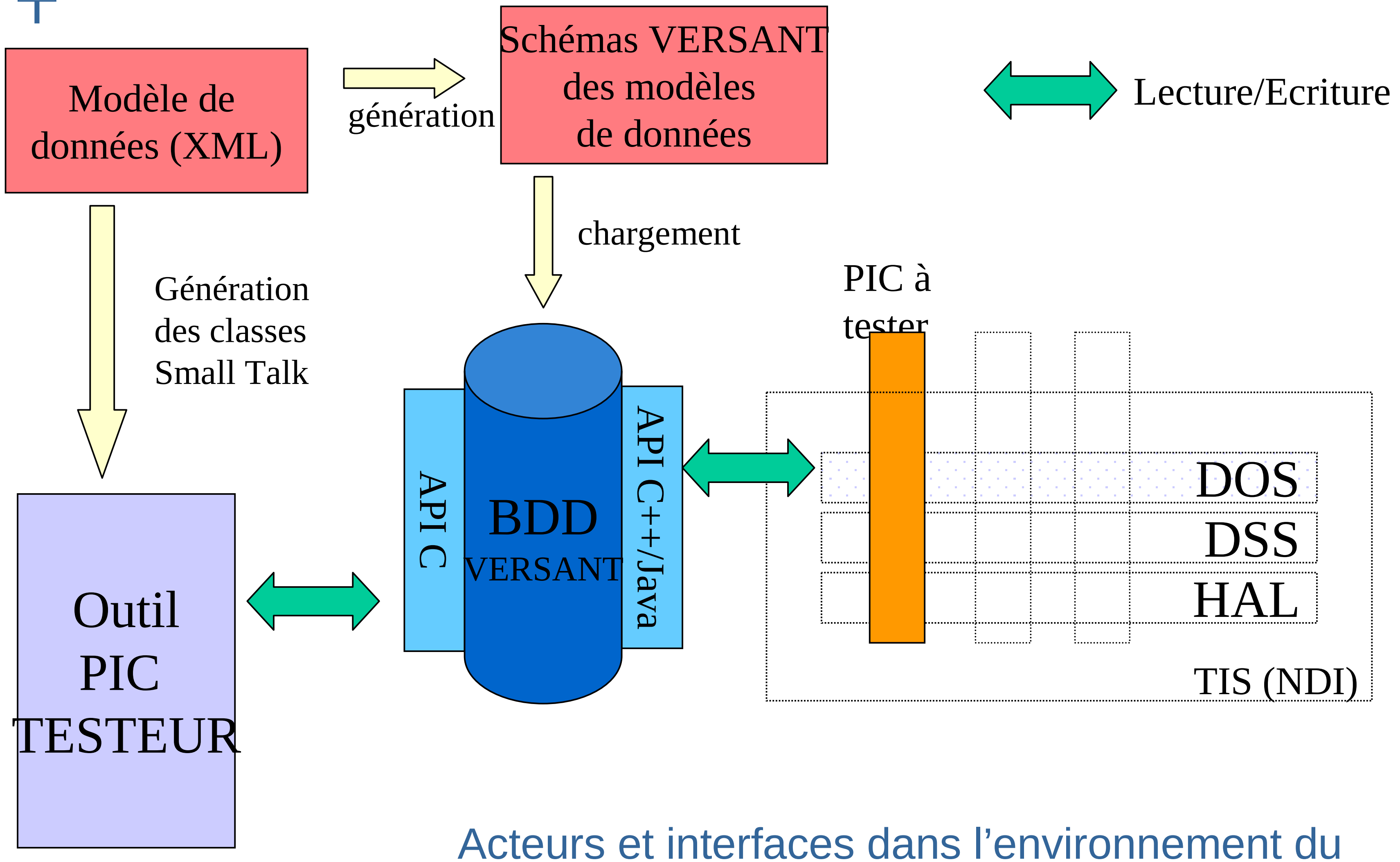
THALES

PIC structure



THALES

Environnement et interfaces



Acteurs et interfaces dans l'environnement du

PIC TESTEUR