

# TOOLS SUPPORTING IMPLEMENTATION OF CAPE-OPEN INTERFACES



**Michel PONS**


**Chief Technology Officer**

**Paper #489d, 3rd US CAPE-OPEN conference,  
AIChE'06 Topical , San Francisco, November 16, 2006**



# Outline

- ◆ **Needs**
- ◆ **Tools available**
  - ⇒ **Tester Suite**
  - ⇒ **Wizards**
  - ⇒ **Logging Tool**
- ◆ **Conclusion & perspectives**

CO  LaN



# Needs

- ◆ **Users' need: delivery of reliable, seamless interoperability**
  - ⇒ **Facilitate relationship with support teams if problems arise**
- ◆ **Developers' need: reduce learning curve**
  - ⇒ **Lessen cost of adopting CAPE-OPEN**
- ◆ **CO-LaN's goal: accelerate adoption**
  - ⇒ **Get more components and environments available with CAPE-OPEN interfaces**

# Solution proposed by CO-LaN

## ◆ Simplify processes

- ⇒ of developing a CAPE-OPEN compliant component
  - Wizards develop most of the code needed around a CAPE-OPEN component automatically
- ⇒ of testing compliance with CO standards
  - A Tester Suite analyzes the CAPE-OPEN interfaces displayed by a component
- ⇒ of analyzing communication between a PMC and a PME
  - A CAPE-OPEN Logging and Testing Tool reports on whatever transactions take place between a PME and a PMC



## Support tools: a major commitment from CO-LaN

- ◆ **Unit Wizards: ~25 K\$ development cost for CO-LaN**
  - ⇒ Provides wrappers for Unit Operations components quickly
- ◆ **Tester suite: ~ 90 K\$ development cost for CO-LaN**
  - ⇒ Checks CO compliance of software components
- ◆ **Logging Tool: ~ 33 K\$ development cost for CO-LaN**
  - ⇒ Analyzes communication

# Unit Wizard purpose

- ◆ **Make it simpler and faster for an engineer to build a CAPE-OPEN compliant Unit Operation model using the Microsoft COM version of the interface standards**
  - ⇒ **Simpler because:**
    - **No need to know as much about COM as an engineer trying to develop a unit operation from scratch would.**
  - ⇒ **Faster because:**
    - **The tool generates a complete source code project, to which the engineer only needs to add a user interface form, a calculation routine and a validation routine.**



# Programming languages covered

## ◆ Visual Basic 6.0

⇒ Developed by AspenTech for CO-LaN

- Version 0.93 compliant freely available from [www.colan.org](http://www.colan.org)
- Version 1.0 compliant available to CO-LaN membership

## ◆ C++

⇒ Developed by IFP for its own purposes

- Available to CO-LaN membership

## ◆ Fortran 90

⇒ Contracted by TOTAL for its own purposes

- Available to CO-LaN membership

## ◆ Delphi

⇒ Developed by SASOL for its own purposes

- Available as open source





# Demo of CAPE-OPEN Unit Operation Wizard 1.0 for Visual Basic

CO  LaN





# CO Tester Suite: general purpose

## Self-testing of PMCs

- ◆ **Help to software development**
  - ⇒ **Preliminary debugging**
- ◆ **Screening before use**
  - ⇒ **Provide details on a software component**
    - **Property Package chemical compounds, properties, phases, etc...**



# CO Tester suite: technical description

- ◆ **Stand-alone Windows application**
- ◆ **Provided with Install and Uninstall**
- ◆ **Developed in Visual Basic 6.0**
- ◆ **Graphical User Interface available**
- ◆ **Contextual help**



# CO Tester main features

Launch of Basic test

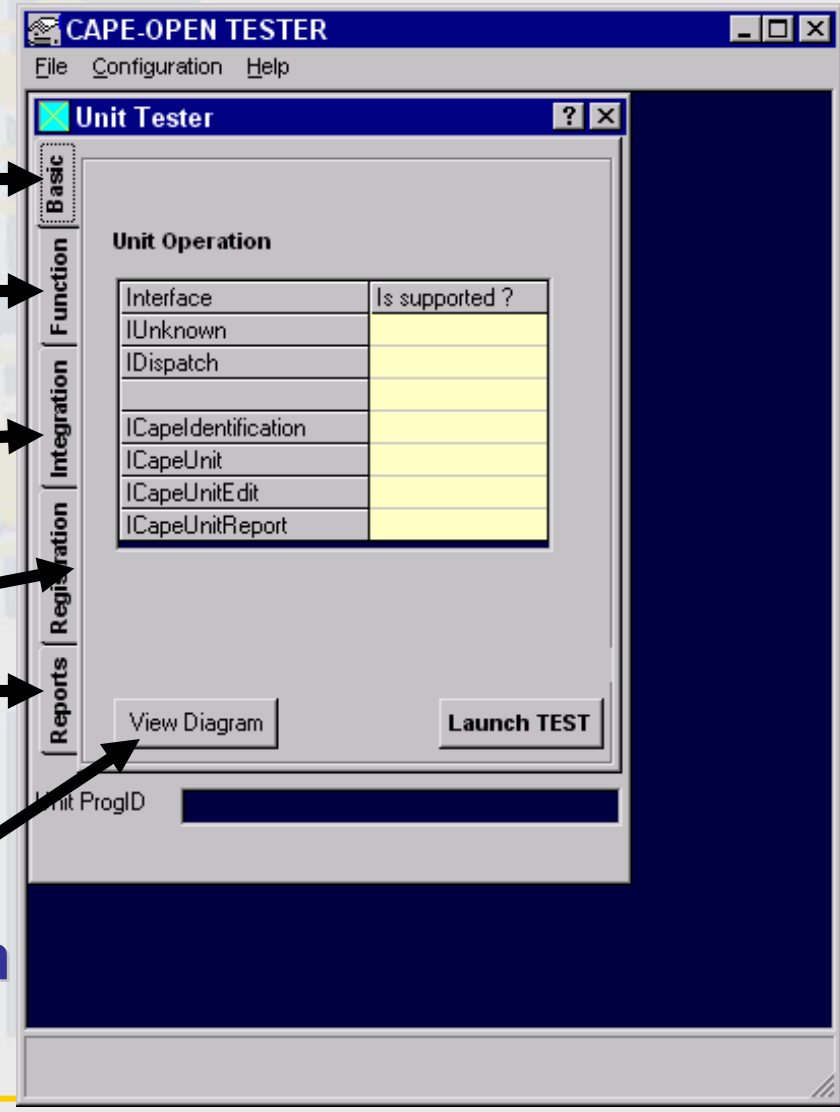
Launch of Function test

Launch of Integration test

Selection of component to test

Access to reports

Corresponding Interface Diagram



# CO Tester reports

```
File Edit : D:\CAPE-OPEN\TESTER\CAPEOPENVALIDATIONNR... X
File
*****
CAPE-OPEN VALIDATION TESTS: BASIC
*****
Time: 31/10/00 15:10:22
IDL: VERSION 0.9
*****
STARTING TEST
Instantiating : ATCOMixNSplit.MixNSplit
COM Interfaces are supported by this Unit Operation
ICapeIdentification Interface is supported by this Unit Operation
ICapeCapeUnit Interface is supported by this Unit Operation
ICapeCapeUnitEdit Interface is supported by this Unit Operation
ICapeCapeUnitReport Interface is supported by this Unit Operation
*****
```

**Unit Tester** [?] [X]

**Basic** | **Function** | **Integration** | **Registration** | **Reports**

Unit BASIC test report available

View Report

View Report

View Report

View Report

Unit ProgID: CO.Fiber.1

Reporting done in plain ASCII text files

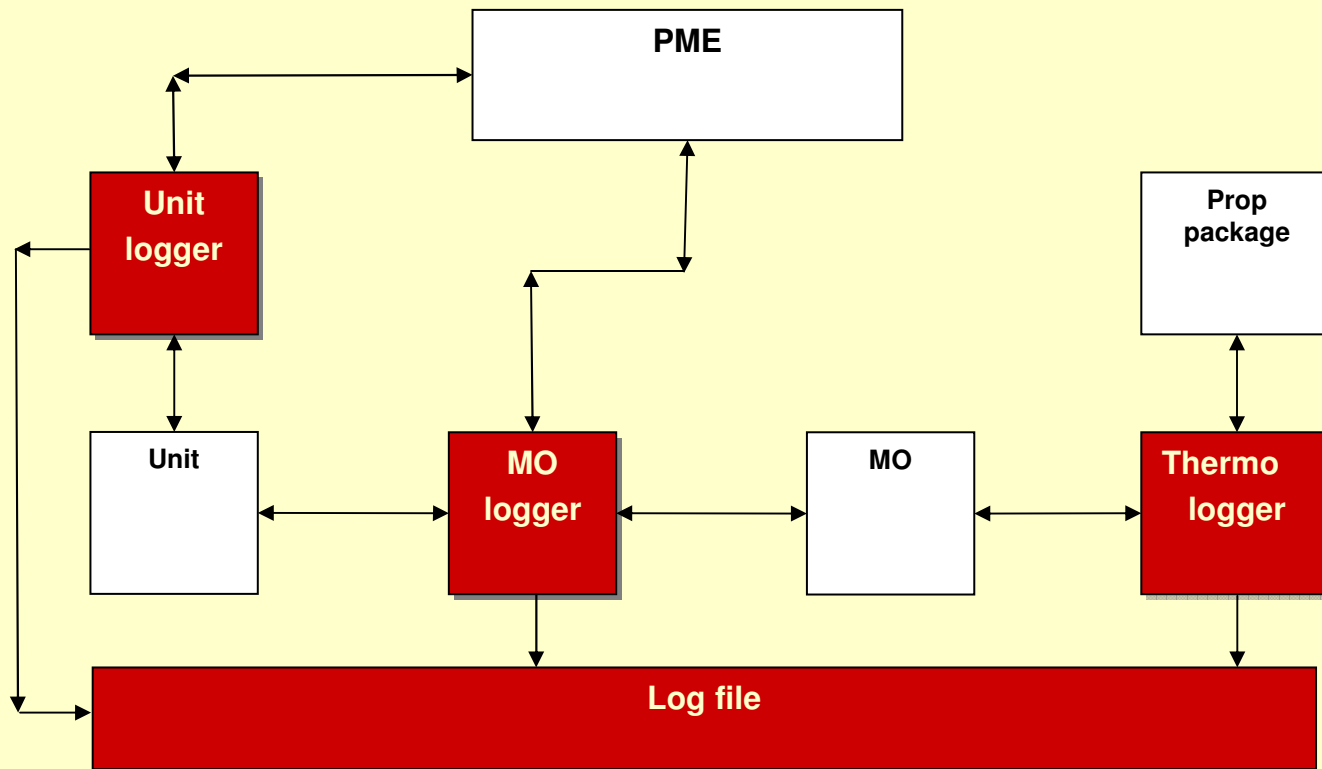
# CO Tester in perspective

- ◆ **Targeted interfaces**
  - ⇒ **UNIT, THRM, PPDB, SMST, MINLP**
- ◆ **Downloadable from CO-LaN website**
- ◆ **Applicable to COM components only**
- ◆ **Development**
  - ⇒ **Initially done by IFP**
  - ⇒ **Then contracted by CO-LaN to:**
    - **Adduce GmbH**
    - **ProSim SA**
    - **University of Catalunya**
- ◆ **An on-going development from CO-LaN**



# CO Logging and Testing Tool (COLTT)

**COLTT components inserted in between actual components and PME**



# CO Logging and Testing Tool

The screenshot shows the 'PROTOTYPE CAPE-OPEN Logging and Testing Configuration Utility' window. It has three tabs: 'Components', 'Log Files', and 'Real Time Logging'. The 'Components' tab is active. On the left, there is a search area with a 'Category' dropdown set to 'All Components', a 'Component Name' text box, and radio buttons for 'Show Components' (Logged, Unlogged, All). Below this is a tree view of components, including 'Thermo Systems', 'Standalone Property Packages', and 'Unit Operations'. The 'Distillation shortcut' component is selected. On the right, the 'Component Detail' panel shows fields for Name, Description, About, Vendor, CAPE-OPEN Version, and Component Version. The 'Enable Logging' checkbox is checked. At the bottom, there are buttons for 'Clear all logs', 'Close', and 'Help'.

**Enabling logging on one or several components**



# Detailed log of any communication

Log generated by C:\Program Files\AspenTech\Aspen Plus 2004.1\Engine\xeq\apmain.exe  
using configuration from C:\Documents and Settings\Michel PONS\Bureau\CAPE  
tools\COLTT\CAPE-OPENLogs.ini

ClassFactory : Loading Xist implemented by  
c:\PROGRA~1\HTRI\Shared\HTRICO~1.DLL

ClassFactory : Created instance of Xist successfully

ClassFactory : Logging enabled for Unit Operation Xist

Unit : Call to Initialize

Unit : Return from Initialize - 0x0

← Error code returned by method

Unit : Call to put\_ComponentName

Unit B1 : Return from put\_ComponentName - 0x0

Unit B1 : Call to Load

Unit B1 : Return from Load - 0x0

Unit B1 : Call to put\_simulationContext

Unit B1 : Return from put\_simulationContext - 0x0

Unit B1 : Call to get\_ports

Unit B1 : Return from get\_ports - 0x0

Port Collection : Call to Count

Port Collection : Count Is 4

← Value returned by method

Port Collection : Return from Count - 0x0

Port Collection : Call to Item requesting Item 1

Port Collection : Return from Item - 0x0

Port : Call to get\_ComponentName

Port get\_ComponentName returns HotInlet

Port HotInlet : Return from get\_ComponentName - 0x0





# COLTT in perspective

- ◆ **Prototyping phase**
  - ⇒ **Check of feasibility**
- ◆ **Specification of final tool**
- ◆ **Phase I**
  - ⇒ **Check of workability on 50+ interoperability situations**
- ◆ **Phase II**
  - ⇒ **Resolving difficulties discovered in Phase II**
- ◆ **Phase III - current**
  - ⇒ **Developing tool up to specification**

# Conclusion & perspectives

- ◆ **Numerous aspects of CAPE-OPEN technology implementation and use already covered**
  - ⇒ **Development, testing, use & debugging**
- ◆ **On-going actions by CO-LaN to facilitate the development of reliable CO components**
  - ⇒ **Maintenance of Wizards, Tester Suite**
  - ⇒ **Finalization of CO Logger**
- ◆ **Your feedback appreciated in pointing out how these tools can be made more user-friendly**





Thank you  
Questions?

CO  LaN

