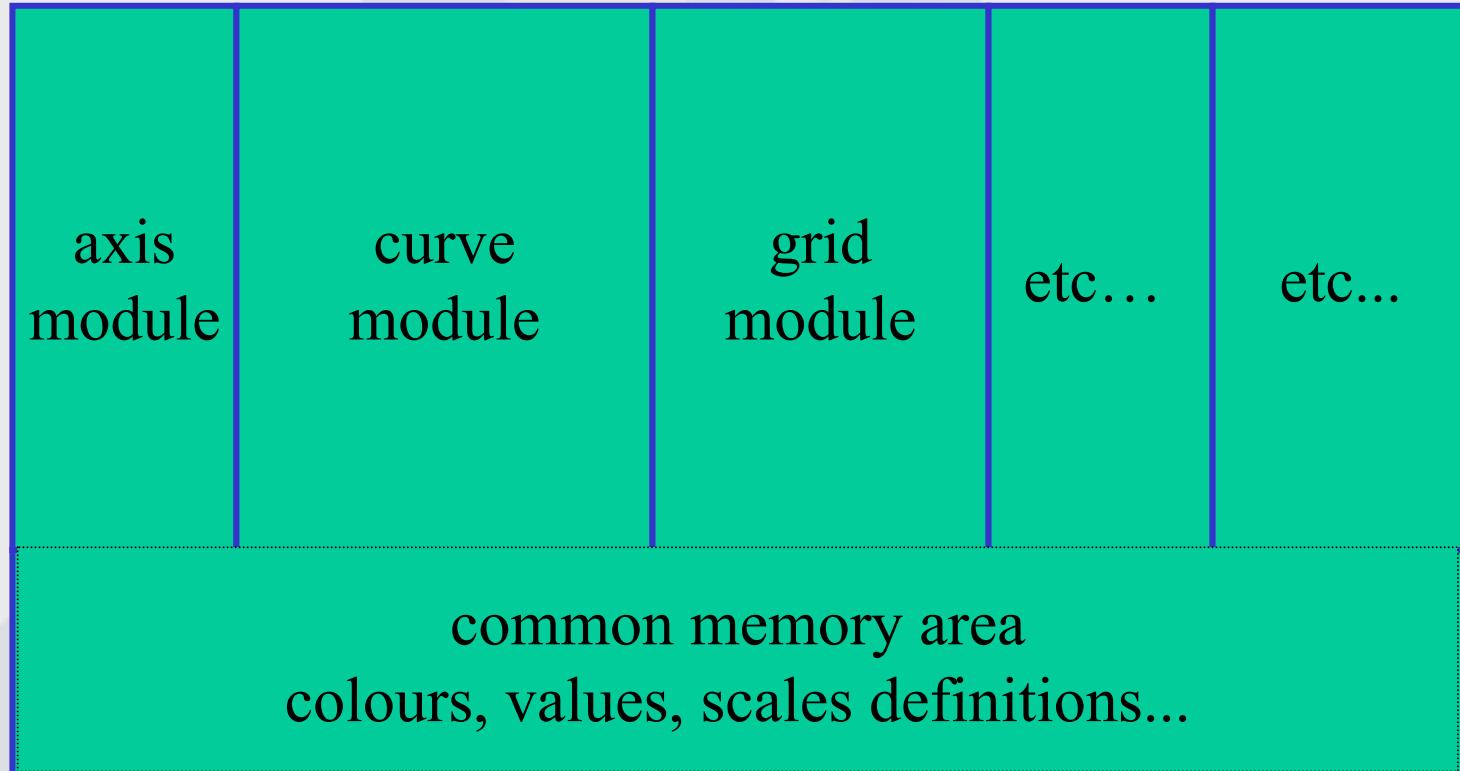


2.1 CAPE OPEN Concepts

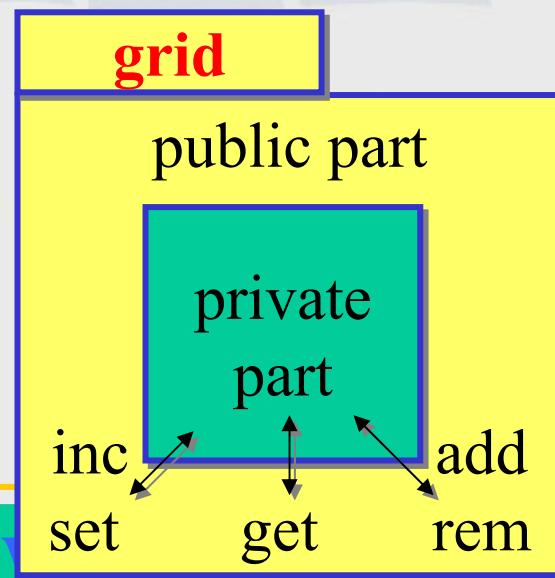
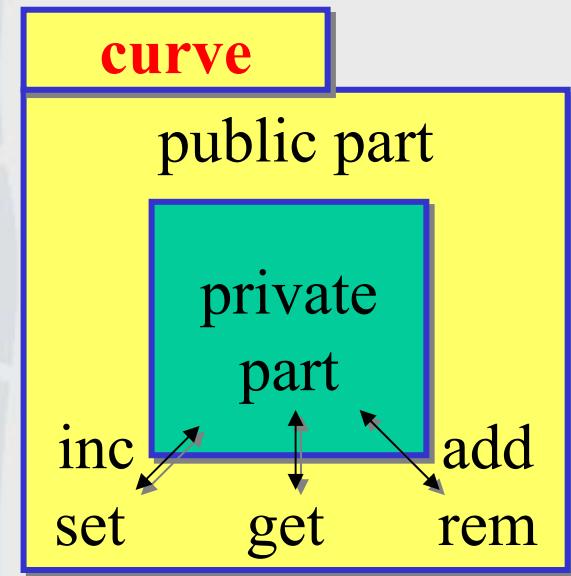
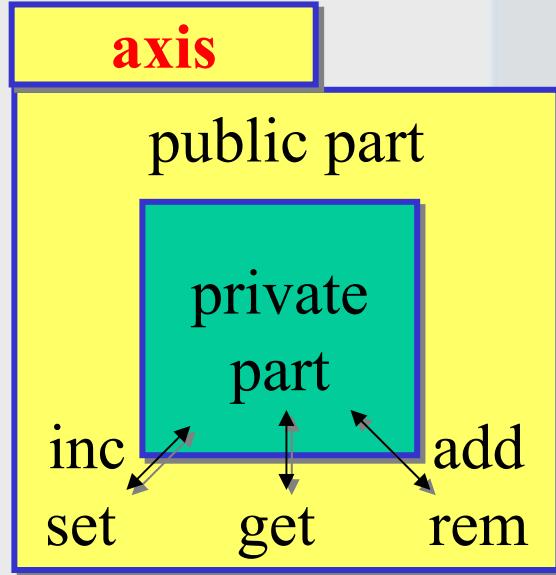
Bertrand Braunschweig



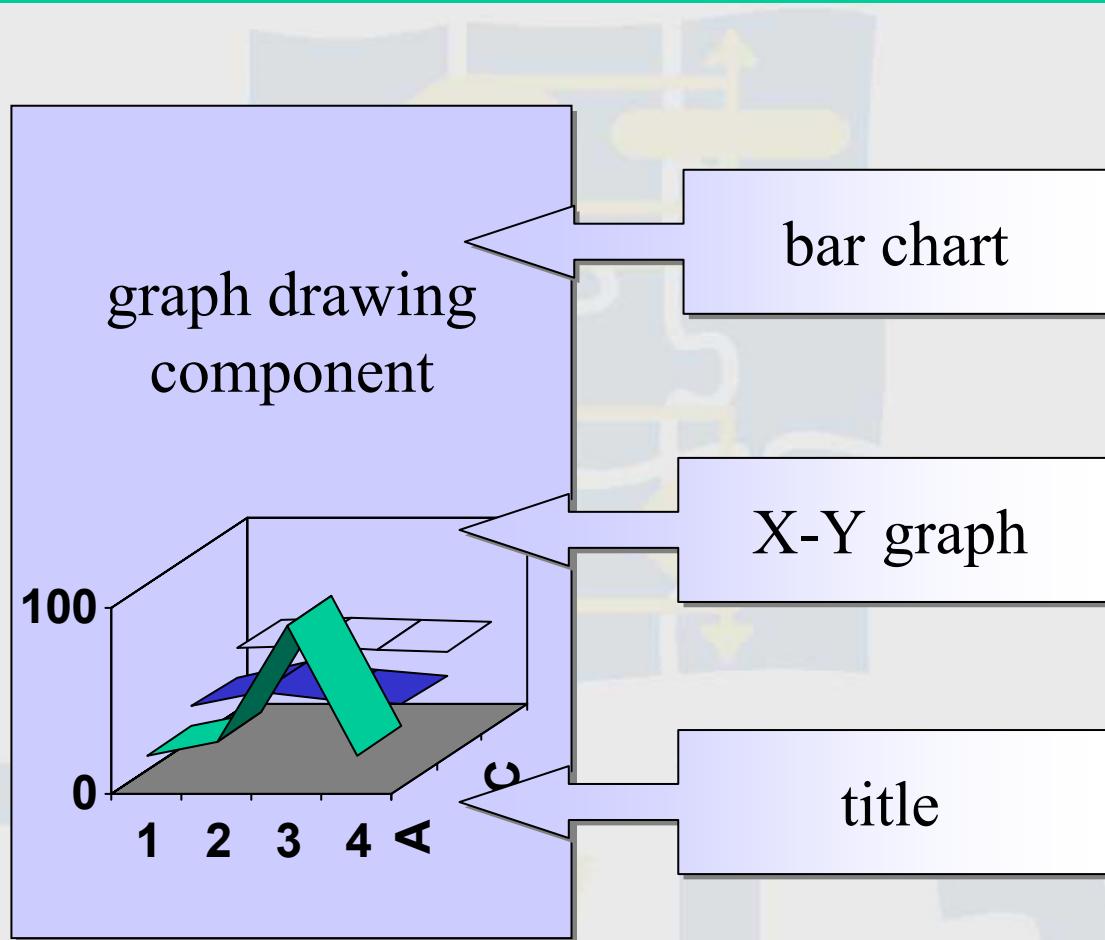
Monolithic “COMMON”



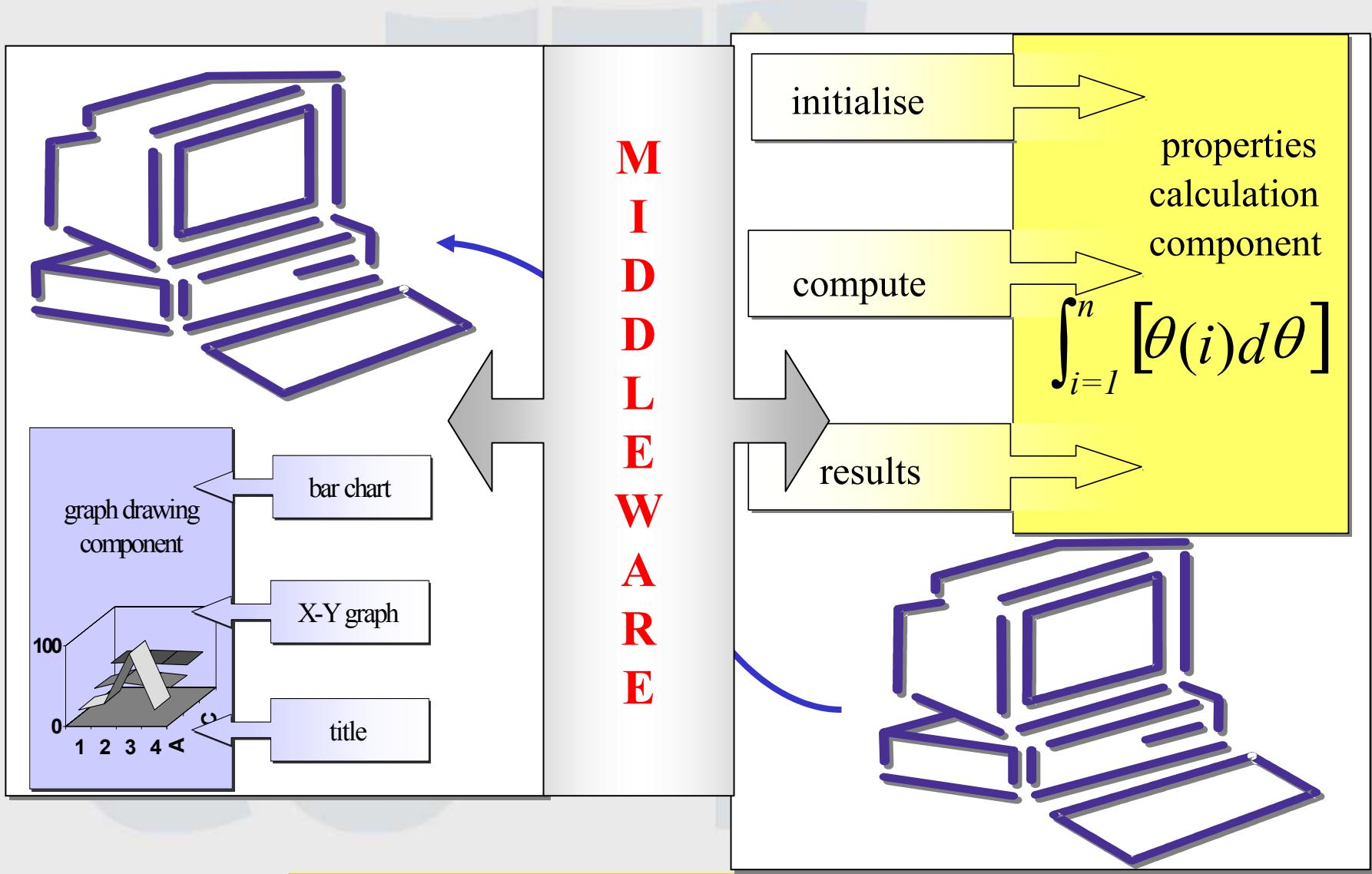
software objects



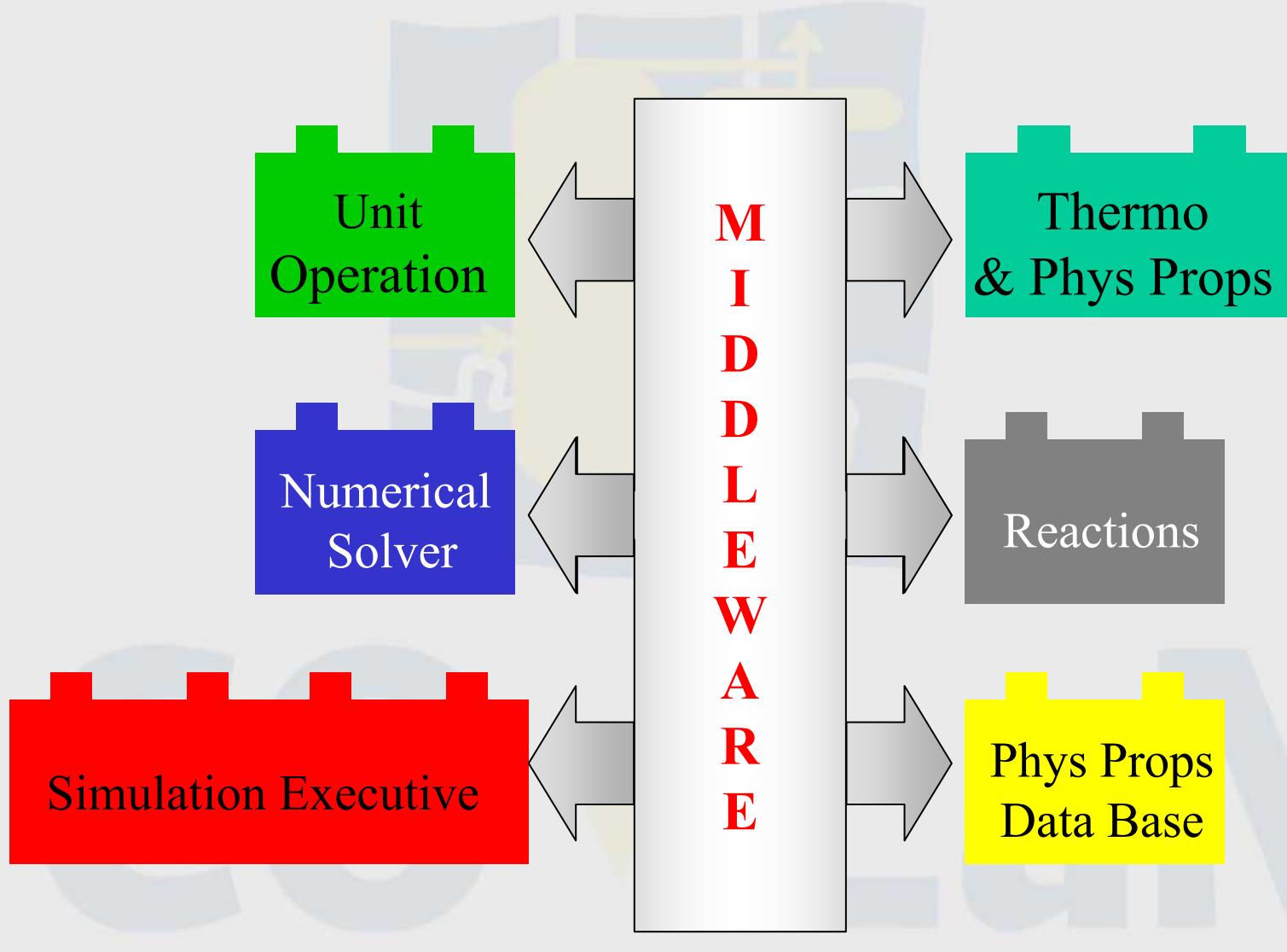
software components



distributed software components



CAPE-OPEN Components



Models & Middleware for distributed components

- ▼ Microsoft's COM/DCOM
- ▼ Microsoft .NET
- ▼ OMG's CORBA component model
- ▼ Sun' Java-based J2EE (EJB)
- ▼ Web Services and Service-Oriented Architectures (SOAs)
- ▼ FIPA's multi-agent standards
- ▼ etc..



CAPE-OPEN choice

▼ Microsoft's COM/DCOM

⇒ Microsoft .NET

▼ OMG's CORBA component model

▼ Sun' Java-based J2EE (EJB)

▼ Web Services and Service-Oriented Architectures (SOAs)

▼ Software fortresses

▼ FIPA's multi-agent standards

▼ etc..



What Is COM?

▼ COM is a standard that provides for:

- ↪ Language independence.
- ↪ Location independence.

▼ COM Interfaces

- ↪ Objects implement multiple interfaces
- ↪ Clients can select a specific interface to use

▼ In addition

- ↪ It guarantees a common behaviour for all objects through the interface IUnknown.
- ↪ It handles the components management processes (instantiation, reference counting,...)
- ↪ It provides standard error/exception handling.



COM Interface

▼ Contract between objects

- ↪ Defines group of functions supported
- ↪ Defined in a language called IDL

▼ Strongly typed

- ↪ Each interface is unique and has a unique IID
- ↪ Confusion between interfaces cannot happen accidentally
- ↪ Defines binary standard through which objects communicate

▼ Robust under change.

- ↪ Objects can support additional interfaces over time without breaking existing clients



COM IDL example

Interface ICapeUnitPort: IDispatch

{

[propget, id(1), helpstring("type of port, e.g. material, energy or information")]

HRESULT **portType**([out, retval] CapePortType* portType);

// Returns the direction in which the object connected to this

// port is expected to flow

[propget, id(2), helpstring("direction of port, e.g. input, output or unspecified")]

HRESULT **direction**([out, retval] CapePortDirection* portDirection);

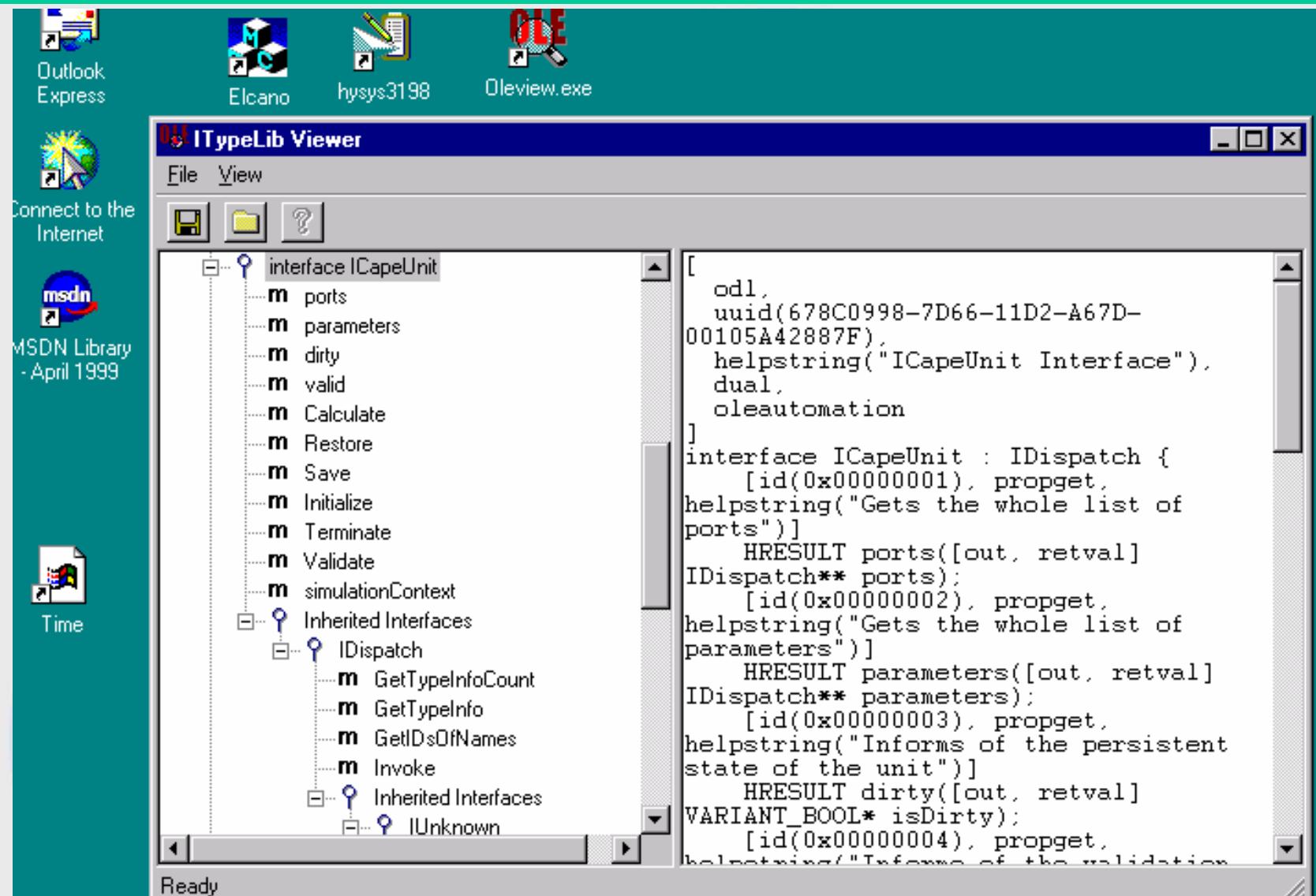
// Returns to the client the object that is connected to this port

[propget, id(3), helpstring("gets the object connected to the port, e.g. material, energy or information")]

HRESULT **connectedObject**([out, retval] CapeInterface*
connectedObject);

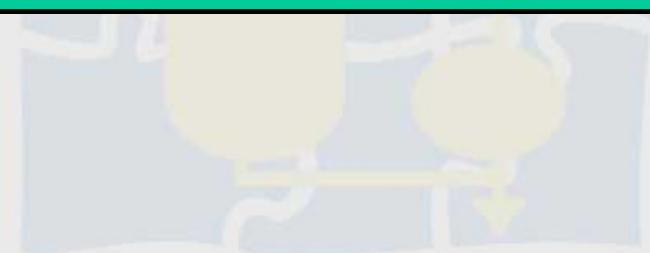


IDL definitions are stored in a Type Library





No CORBA presentation



COvLaN



The CAPE-OPEN Architecture Main Interfaces

Common Services

Unit Operations

Physical and Thermodynamic Properties

Solvers





Common Services

CO²LaN



Common Interfaces

Parameters

Collections

Identification

Error Handling

Persistence

Utilities

Types and undefined values



Unit Operations



Overview of CO 1.0 Interfaces: Unit Operations

Hybrid Units

Dynamic Units

Unit Operations
Steady-State

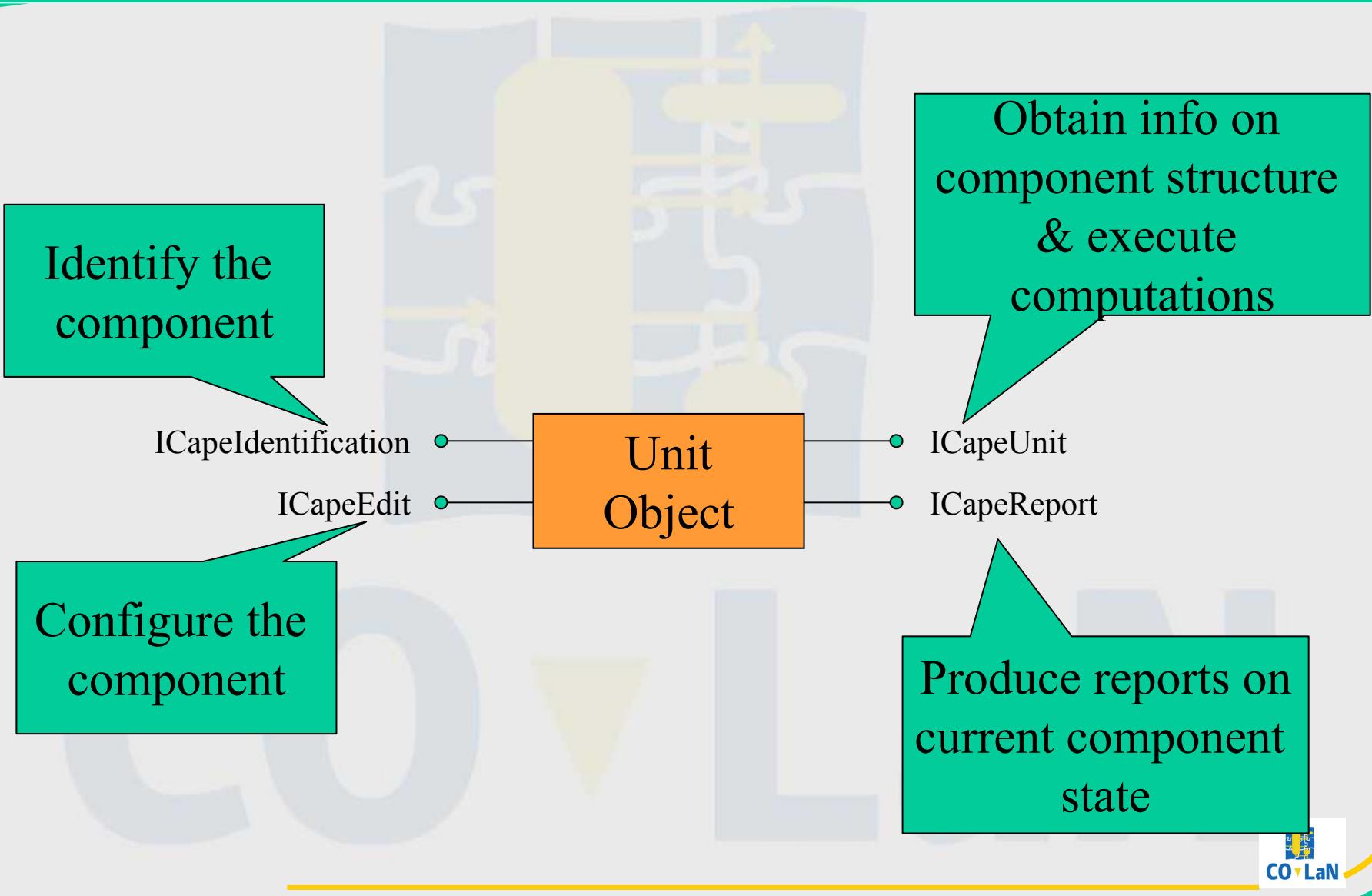


CAPE-OPEN UNIT OBJECTS

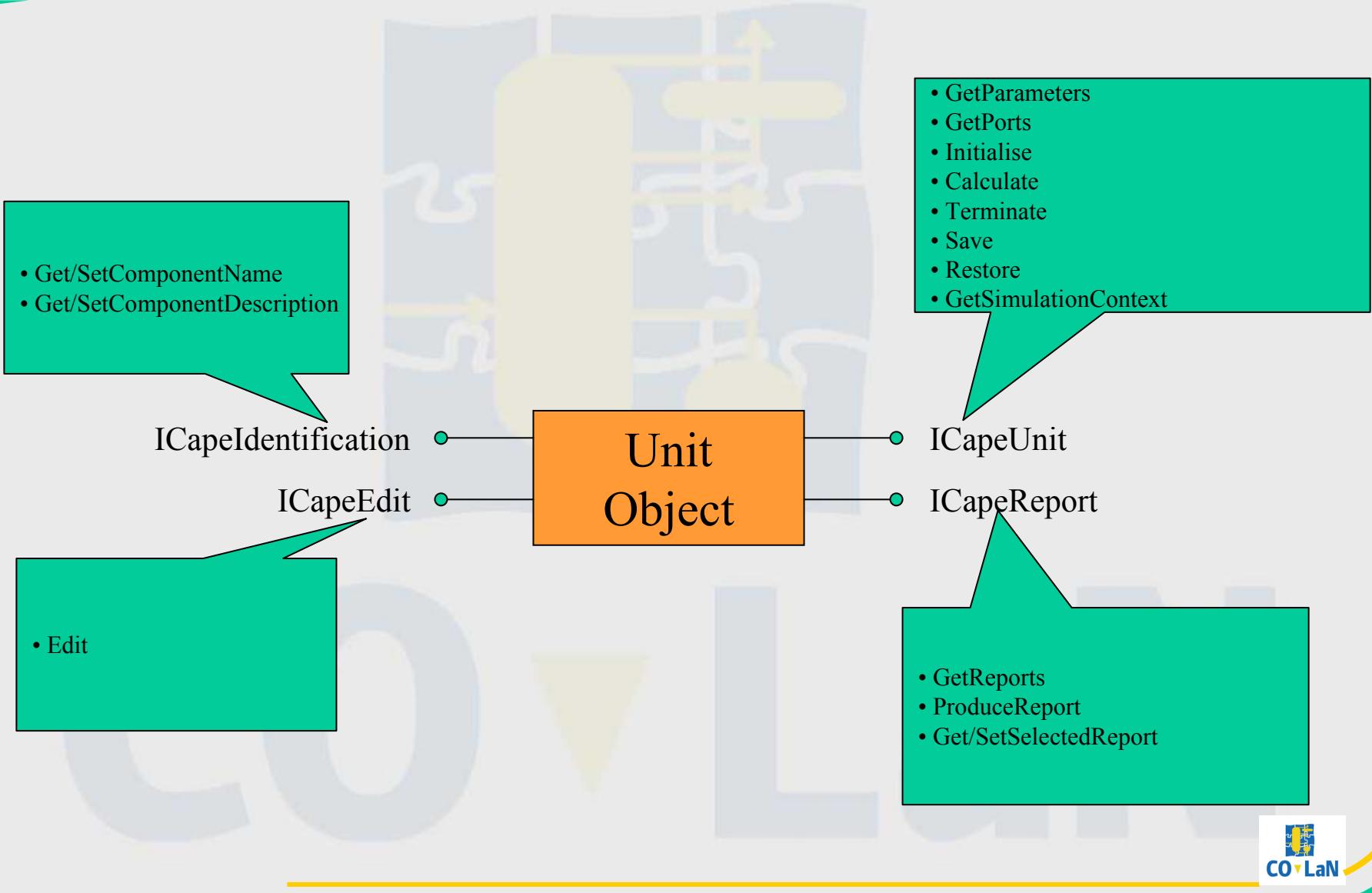
- ▼ A “CO unit object” may be
 - ⌚ a single unit operation
 - ⌚ a plant sub-section
 - ⌚ a whole plant
- ▼ ...connected to its environment via “ports” carrying material, energy or information
 - ⌚ input ports
 - ⌚ output ports
- ▼ ...and characterised by “parameters”
 - ⌚ input parameters
 - ⌚ output parameters
- ▼ Unit object behaviour
 - ⌚ given input port information and input parameter values
 - ⌚ compute output port information and output parameter values



CAPE-OPEN Unit Object Interfaces



CAPE-OPEN Unit Object Interfaces



Thermo & Phys Props



Overview of CO 1.0 Interfaces: Physical Properties Services

Petroleum
Fractions

Thermodynamic and Physical
Properties

Physical Properties
Data Bases

Reactions



The main elements of ‘Thermo’

▼ Thermo systems

↪ can create ...

▼ Property packages

↪ containing ...

▼ Equilibrium servers

▼ Calculation routines

↪ to serve ...

▼ Material objects



Thermo System

- ▼ Manages creation and selection of Property Packages
- ▼ Configuration of a PP is outside the scope of CO specs
 - ↪ Selection of components models and phases is done by proprietary methods



Property Package

▼ Self-contained collection of

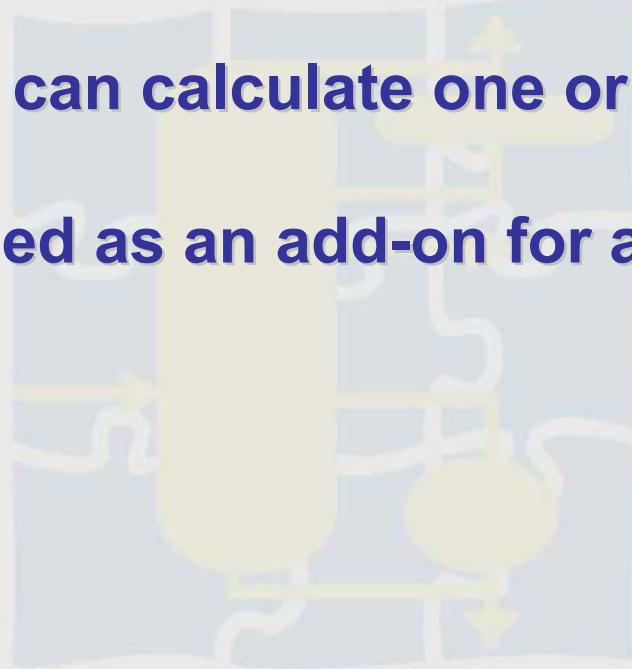
- chemical compounds (pure substances),
- Models/correlations/data for mixture properties
- Phases
- Equilibrium calculation procedures

▼ Normally a small subset of all the components and models available in a Property System



Calculation Routine

- ▼ A routine that can calculate one or more physical properties
- ▼ May be supplied as an add-on for a PP

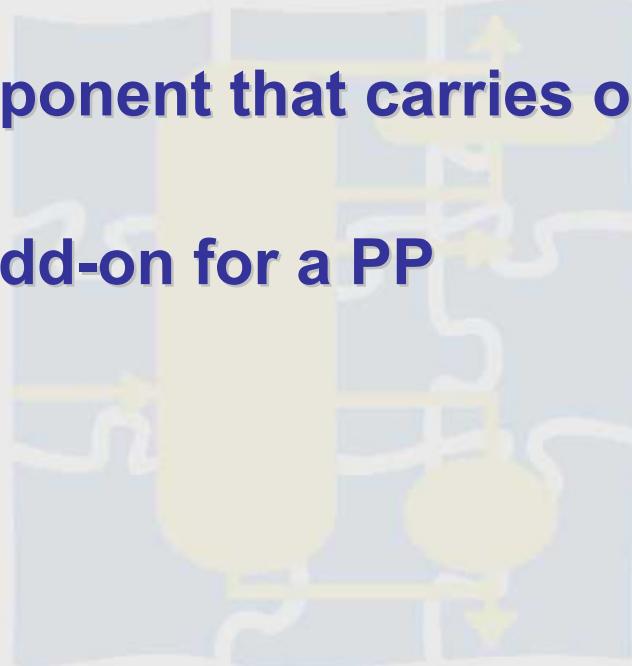


CO₂LaN



Equilibrium Server

- ▼ Software component that carries out phase equilibrium calculations
- ▼ Could be an add-on for a PP



CO₂LaN

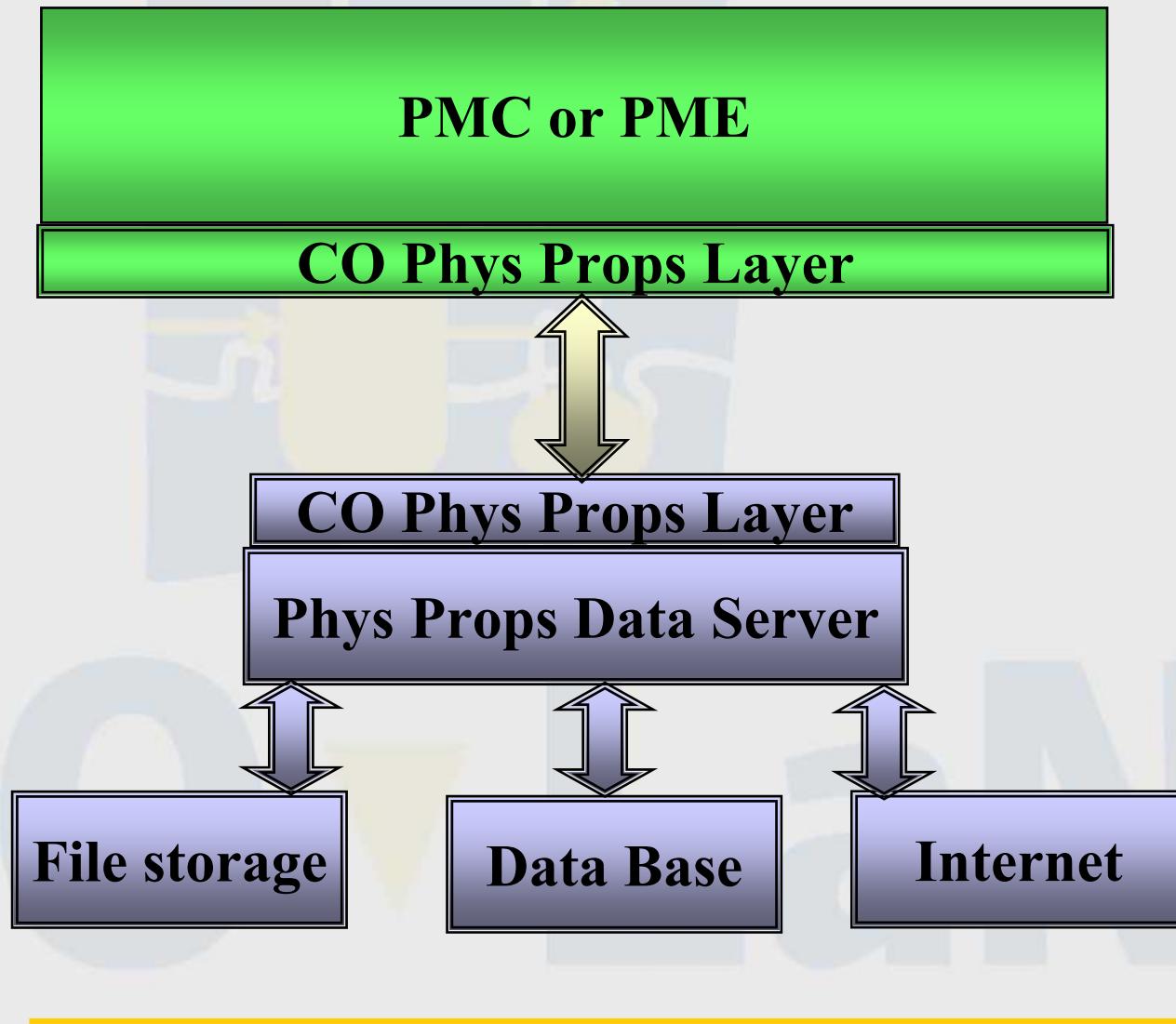


MaterialObject

- ▼ Container for properties of a material
 - ⌚ Components, phases, pressure, temperature compositions, other properties...
 - ⌚ Reference to a PP
- ▼ Used by client of Property Package to specify input and collect output from calculations



Physical Properties Data Bases (PPDB)



Physical Properties Data Bases (PPDB)

▼ Interface for PP Data Bases:

- ➲ Physical property data at discrete values of the state variables (temperature, pressure, composition)
 - measured, correlated or estimated values
- ➲ Parameters of model equations to be used for calculating data at any desired state

▼ The PPDB standard is made of three interfaces

- ➲ DB management
- ➲ Access to properties in tables
- ➲ Access to model parameters



Reactions

▼ Consistent reaction modeling across CAPE-OPEN PMEs

▼ Reaction Modeling Scope:

- ⦿ Support kinetic and equilibrium reactions
- ⦿ Support electrolyte reactions
- ⦿ Support for any reaction model
- ⦿ Support formulation of reaction equations by a client
- ⦿ Support Reaction model parameter estimation



Reactions Interfaces Design

▼ Primary CAPE-OPEN Components:

↪ Reactions Package

- defines system of reactions involving specified compounds existing in specific phases
- able to calculate reaction properties for these reactions

↪ Reactions Package Manager

- Manages Reaction Packages
- May allow Reaction Packages to be created and/or edited

▼ PME software objects

↪ Reactions Object

- supports exchange of reaction property values between clients and components



Petroleum Fractions Interfaces

▼ Extensions of Thermo and Unit interfaces

▼ ICapeThermoPetroFractions implemented on the Material Objects

- ↪ Allows setting and getting refinery properties (bulk, curves and component properties)
- ↪ Allows characterising a set of petroleum fractions (e.g. estimate Tc, Pc, etc)

▼ ICapeUnitTypeInfo implemented on the UNITS

- ↪ Informs COSEs whether a UNIT will require characterization of petroleum fractions (e.g. FCC)





Numerical Solvers



co
LaN



Overview of CO 1.0 interfaces: solvers services and clients

PEDR

Optimisation
MILP, MINLP

PDAE
Solvers

Solvers
LAE, NLAE, DAE



CAPE-OPEN scope for numerical solvers

▼ Variety of core model-based activities

- ↪ Steady-state & dynamic simulation
- ↪ Steady-state optimisation
- ↪ Parameter estimation and data reconciliation

▼ Both “modular” and “equation-orientated” systems

▼ Emphasis on “large-scale” problems



CAPE-OPEN Problem Objects

- ▼ Fundamental principle: complete separation between
 - ⌚ the description of the problem being solved
 - ⌚ the code used for its solution
- ▼ Describe different types of mathematical problems as different classes with formally defined interfaces

Mathematical Problem Type	CAPE-OPEN Problem Object
Nonlinear algebraic equations	Equation Set Object (ESO)
Differential-algebraic equations	Differential Algebraic ESO (DAESO)
Partial differential-algebraic equations	Partial Differential Algebraic ESO(PDAESO)
Mixed integer nonlinear programming problems	minlp object



CAPE-OPEN Systems

$$\left\{ \begin{array}{c} \text{CAPE-OPEN} \\ \text{System} \end{array} \right\} \equiv \left\{ \begin{array}{c} \text{Problem} \\ \text{being solved} \end{array} \right\} + \left\{ \begin{array}{c} \text{Numerical code} \\ \text{used for solution} \end{array} \right\}$$

- ▼ Each CAPE-OPEN System has method(s) for solving the problem
- ▼ Final solution of the problem is placed in the CO Problem Object





The CAPE-OPEN Standard: What it permits

CO²LaN



Example of use 1

- ▼ A physical and thermodynamic properties calculations PMC developed by a supplier, can be used the same way within several CO-PMEs.
- ▼ e.g. Infochem's Multiflash, can be used the same way in Aspen+, gPROMS, Pro/II, Indiss or Hysys.
- ▼ The user saves the time needed to configure the properties calculations parameters for those environments, and gets consistent results by using the same methods and data.
- ▼ This is simply obtained by wrapping the thermo server with CAPE-OPEN standard interfaces.



Example of use 2

- ▼ A CO-compliant PME can transparently use several physical properties and thermodynamic servers for one model.
- ▼ e.g. Hysys can be configured to use Aspentech's Properties Plus, or Infochem 's Multiflash, or IFP 's proprietary thermo.
- ▼ This can be through replacing a single thermo server for the whole flowsheet, or even by combining different servers for different sections of the flowsheet (with precautions on the enthalpy basis).
- ▼ Thus, the modeller can easily try out diverse methods and choose the best
- ▼ This is obtained by introducing the CO « Thermo » API in the PME.



Example of use 3

- ▼ A Unit Operation model such as a proprietary chemical reactor model, developed by an operator or a process licensor, **can be used transparently in CO-compliant PMEs.**
- ▼ e.g. IFP 's FIBER (FIxed BEd Reactor) generic reactor model can be used the same way in most commercial PMEs without any change, whitout any coding or compiling.
- ▼ The process licensor can easily serve clients who demand the use of a specific PME in their contracts.
- ▼ This is obtained from putting the reactor model to the Unit Operation standard: introduction in a flowsheet, connection of input-output ports, specification of parameters, validity checking, calculation, publication of results.



Example of use 4

- ◆ Reciprocally, a modeller who uses a PME with CO Unit Operations sockets can **seamlessly include foreign unit operation models** by selecting from a list of available CO-compliant Unit Operations.
- ◆ This, the model designer can easily test several equipment models and choose the best equipment (a compressor, a heat exchanger, a pump etc.) for a specific process.
- ◆ This imposes that all equipment models are available on the user's machine. in the future, this will be possible though component identification services developed for the internet.
- ◆ Equipment manufacturers should take advantage of this facility.



Example of use 5: GO:CAPE-OPEN



■ gO:CAPE-OPEN overview

- Introduce advanced gPROMS models within CAPE-OPEN compliant steady-state flowsheeting packages e.g.
 - ASPEN PLUS™
 - HYSYSTM™
- Use consistent physical properties throughout
- No programming required
 - retain advantages of gPROMS-based modelling

Example of use 6: APECS

NETL's Advanced Power and Energy Co-Simulation (APECS)



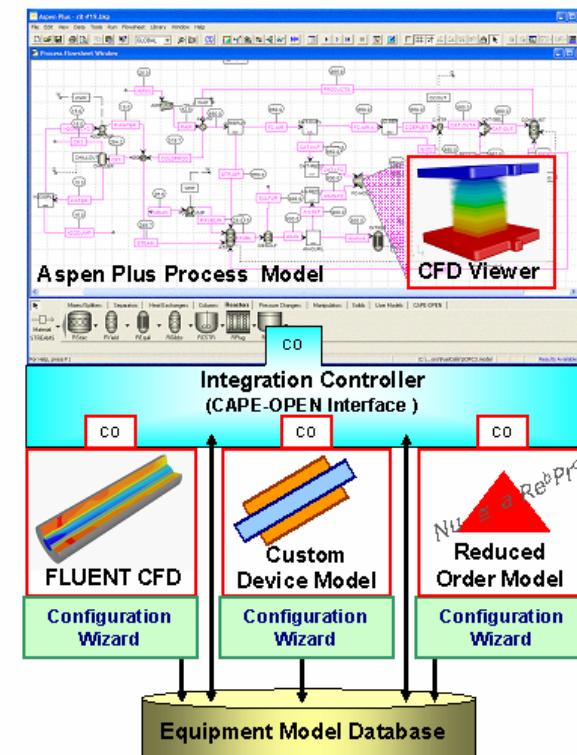
ALSTOM

CERC
West Virginia University



Major Components and Features

- Process Models
 - Aspen Plus®
- Equipment Models
 - FLUENT®
 - Custom Device Models
 - Reduced-Order Models (ROM)
- Integration Controller
 - CAPE-OPEN (CO) Interfaces
 - Unit Operations, Physical Properties, Reactions
- Configuration Wizards
 - FLUENT®
 - Custom Model** and ROM**
- Model Database
- CFD Viewer
- Solution Strategies
 - Speed (ROM)
 - Accuracy (CFD)
- Remote Execution
 - Windows/Linux
 - Serial/Parallel



CAPE-OPEN Meeting SEZ/NETL/August 24-25, 2004



Other uses

▼ More than 10 published interfaces

- ⇒ numerical solvers
- ⇒ chemical reactions
- ⇒ physical properties data banks
- ⇒ etc.

▼ Same kind of facilities as presented in other examples



2.1 CAPE OPEN Concepts

Bertrand Braunschweig

