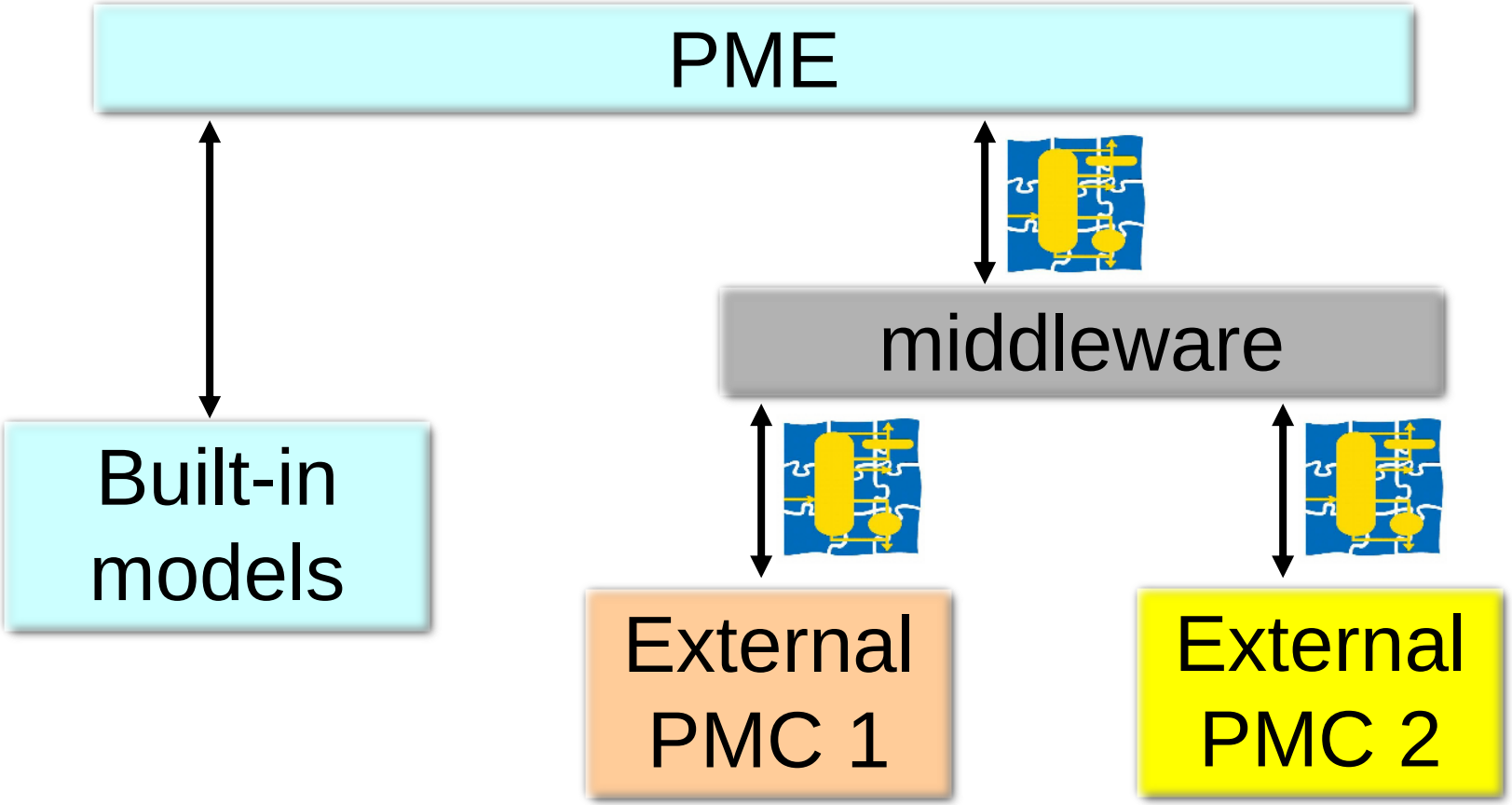
COBIA – PHASE I

Jasper van Baten – AmsterCHEM
Michel Pons – CO-LaN
Bill Barrett – USEPA
Michael Hlavinka – BR&E
Mark Stijnman – Shell Global Solutions

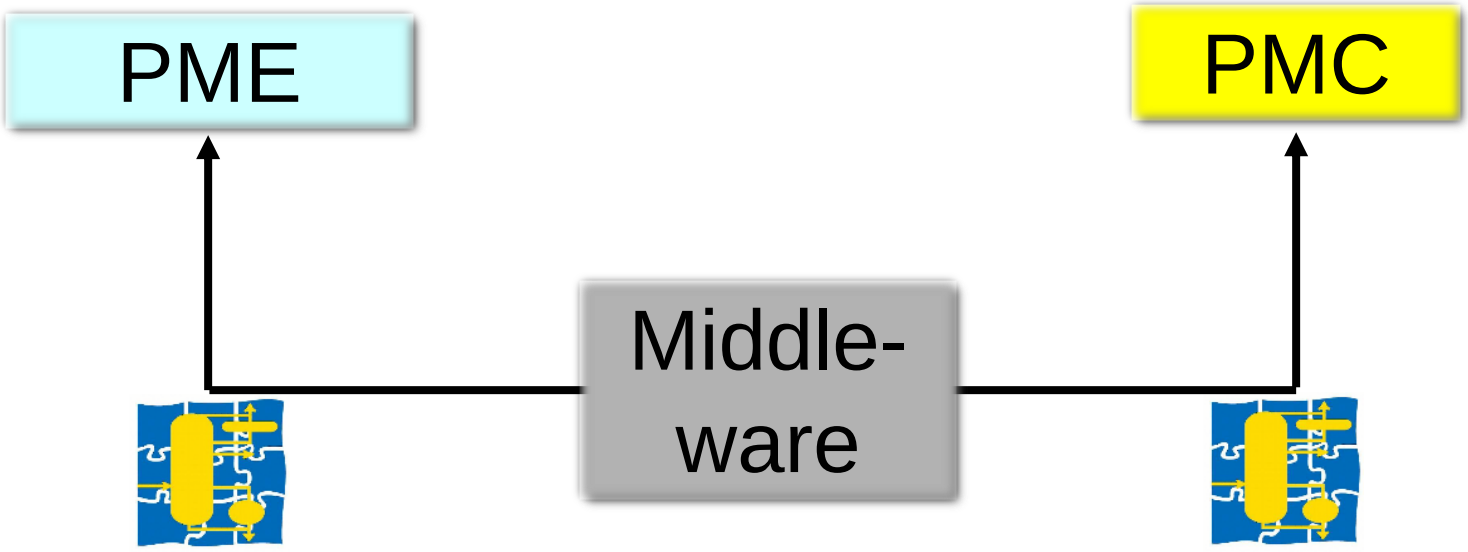
PRESENTATION OUTLINE

- Introduction
- Targets for new middleware
- Evaluation of targets in Phase I
- Enumerate Phase I deliverables
- Conclusions

CAPE-OPEN: INTRODUCTION



IMPLEMENTATION, INTERFACE, MIDDLEWARE



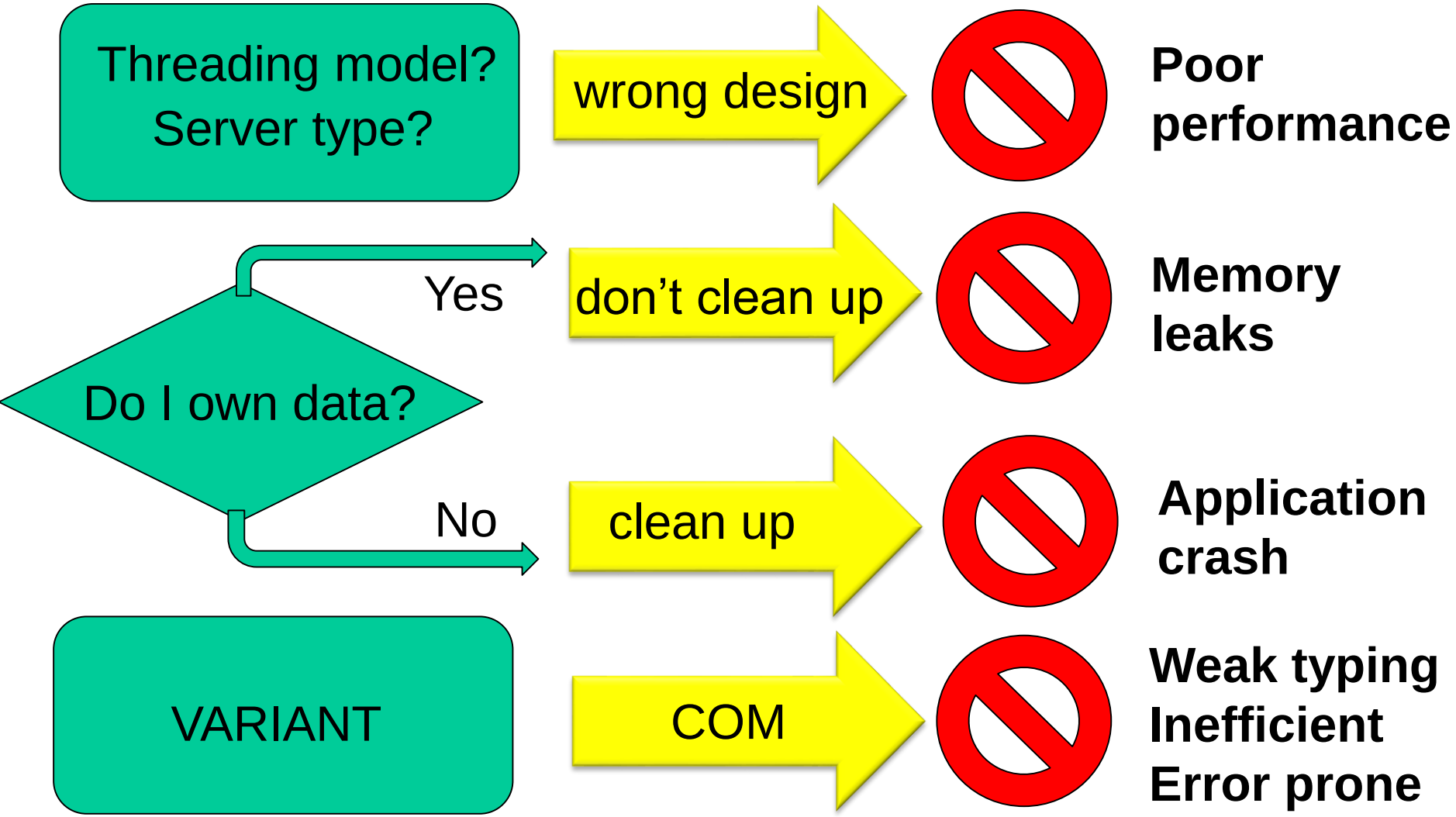
Boolean ICapeUnit::Validate(String message)

CAPE-OPEN MIDDLEWARE

- COM:
  - Common Object Model
  - Microsoft, Windows (built-in)
- CORBA
  - Common Object Request Broker Architecture
  - Platform independent
  - Requires extern ORB software

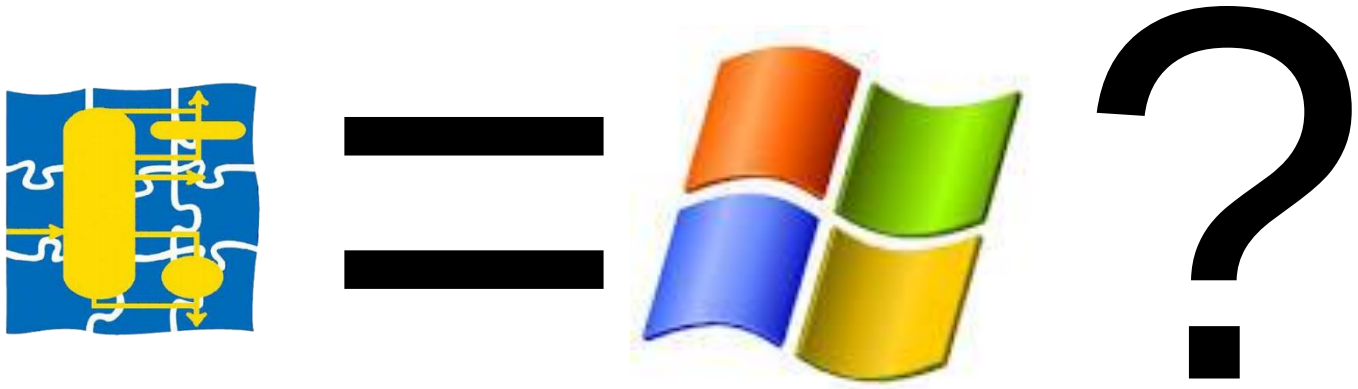
COM much more used – nearly no CORBA implementations

COM PROGRAMMING REQUIRES SKILL



OTHER COM ISSUES

- Windows bound, cannot be ported



COM was introduced in 1993

COM fading out?

TARGETS FOR A NEW MIDDLE WARE

- Platform independent, independent of specific vendor (Microsoft)
- Easier on programmers
- Strong data typing
- More efficient
- Less error prone code
- Open source
- Fully COM compatible

COBIA

CAPE-**O**PEN **B**inary **I**nterop **A**rchitecture

Phase 1:

native, C++, in-process, Windows, thermo 1.1

COM interop *(proof of concept)*

Phase 2:

entire interface set, code generation, IDL based

(full functionality, no marshalling)

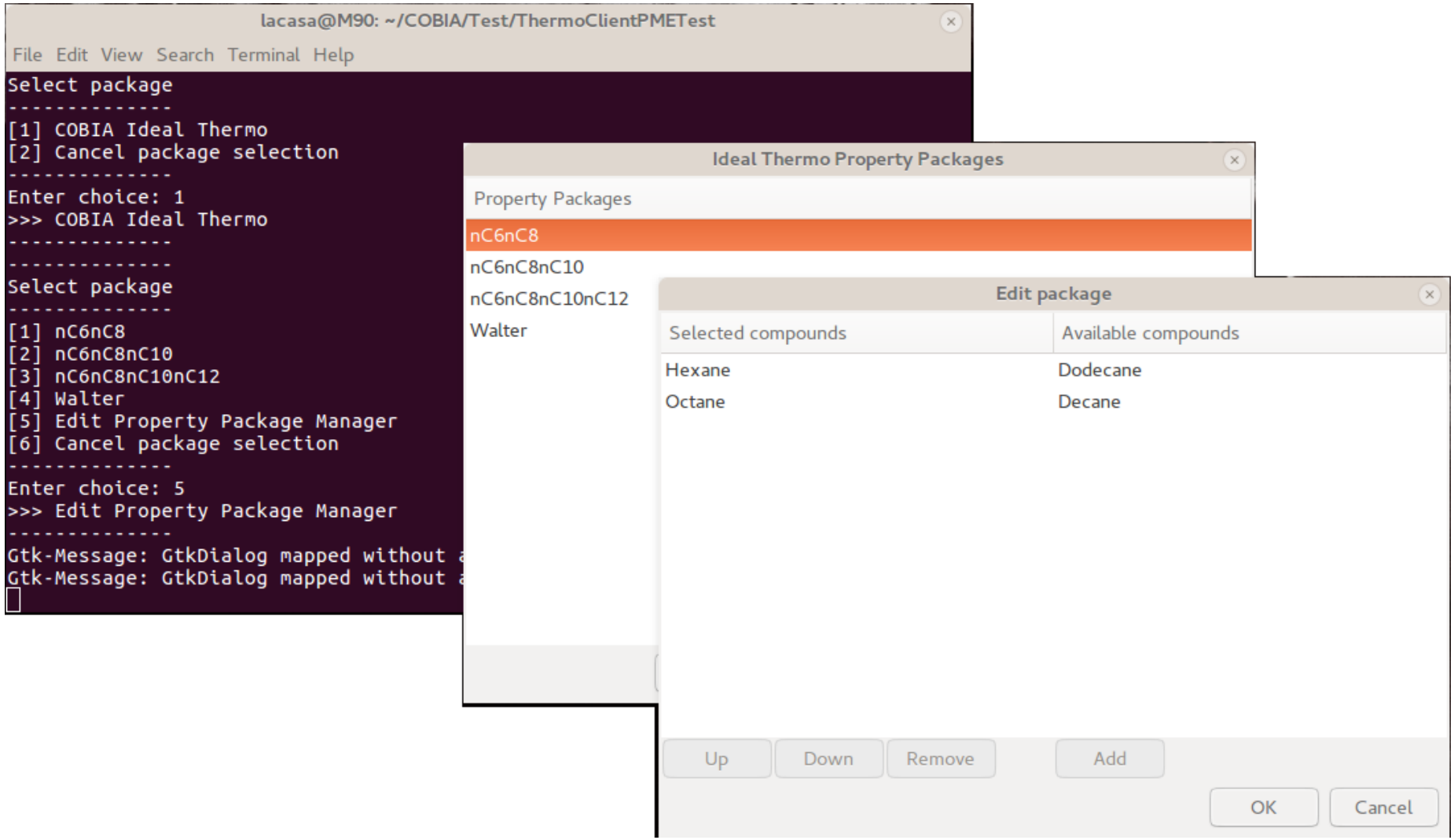
Phase 3:

other platforms, inter-platform interop

PLATFORM INDEPENDENCE

- Openness requires independence of particular OS vendor
- Applications exist on Mac + Linux
- Market seems to go to web-based
- C binary interface (not C++)

PLATFORM INDEPENDENCE



EASIER ON PROGRAMMERS

```
#include <COBIA.h>

class PropertyPackage :
  public CapeOpenObject<PropertyPackage>,
  public CapeUtilitiesAdapter<PropertyPackage>,
  public CapeIdentificationAdapter<PropertyPackage>,
  public CapeThermoCompoundsAdapter<PropertyPackage>,
  public CapeThermoPhasesAdapter<PropertyPackage>,
  public CapeThermoPropertyRoutineAdapter<PropertyPackage>,
  public CapeThermoEquilibriumRoutineAdapter<PropertyPackage>,
  public CapeThermoUniversalConstantAdapter<PropertyPackage>,
  public CapeThermoMaterialContextAdapter<PropertyPackage> {
```

EASIER ON PROGRAMMERS

```
//ICapeIdentification

std::wstring packageName; //On Windows, wstring

void getComponentName(/*out,retval*/CapeString name) {
  name=packageName;
}

void putComponentName(/*in*/const CapeString name) {
  if (name.empty()) {
    throw cape_open_error(COBIAERR_InvalidArgument);
  }
  packageName=name;
}
```

EASIER ON PROGRAMMERS

```
//ICapeIdentification

std::wstring packageName; //On Windows, wstring

void getComponentName(/*out,retval*/CapeString name) {
  name=packageName;
}

void putComponentName(/*in*/const CapeString name) {
  if (name.empty()) {
    throw cape_open_error(COBIAERR_InvalidArgument);
  }
  packageName=name;
}
```

Strong argument typing

EASIER ON PROGRAMMERS

```
//ICapeIdentification

std::wstring packageName; //On Windows, wstring

void getComponentName(/*out,retval*/CapeString name) {
  name=packageName;
}

void putComponentName(/*in*/const CapeString name) {
  if (name.empty()) {
    throw cape_open_error(COBIAERR_InvalidArgument);
  }
  packageName=name;
}
```

Exception throwing

EASIER ON PROGRAMMERS

```
//ICapeIdentification
```

Generated stub code

```
void getComponentName(/*out,retval*/CapeString name) {

}

void putComponentName(/*in*/const CapeString name) {



}
```

STRONG DATA TYPING AND EFFICIENCY

- Data types are passed as interfaces
- Data is strongly typed, e.g. CapeArrayDouble

```
void get(CapeDouble *&data,CapeSize &size);
CapeResult setsize(CapeSize size,CapeDouble *&data);
```

STRONG DATA TYPING AND EFFICIENCY

- Wrapped up on C++ interfaces:

```
void GetSinglePhaseProp(…,
  /*out,retval*/CapeArrayDouble results) {
  //we can set a scalar directly
  double T;
  results.set(T);
  //we can set a vector like this
  size_t nCompounds;
  results.setsize(nCompounds);
  for (size_t i=0;i<nCompounds;i++) {
    results[i]=10;
  }
  //or directly access the data
  memcpy(results.data(),X,sizeof(double)*10);
```

STRONG DATA TYPING AND EFFICIENCY

- **CapeArrayDoubleImpl**
  Derives from std::vector<double>
- **CapeArrayDoubleAdapter**
  Wraps around existing std::vector<double>
- **ConstCapeArrayDouble**
  For outbound data, wraps around fixed size double*
- **CobiaArrayDouble**
  Middle-ware implemented version

STRONG DATA TYPING AND EFFICIENCY

- Declare:

```
CapeArrayDoubleImpl propValue;
```

or

```
std::vector<double> data;
CapeArrayDoubleAdapter propValue(data);
```

- Use:

```
propPack.GetSinglePhaseProperty(...,propValue);
...
propPack.GetSinglePhaseProperty(...,propValue);
```

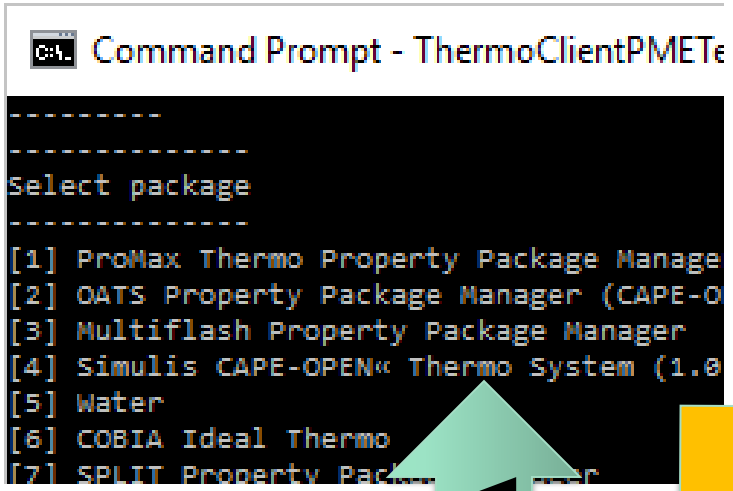
LESS ERROR PRONE

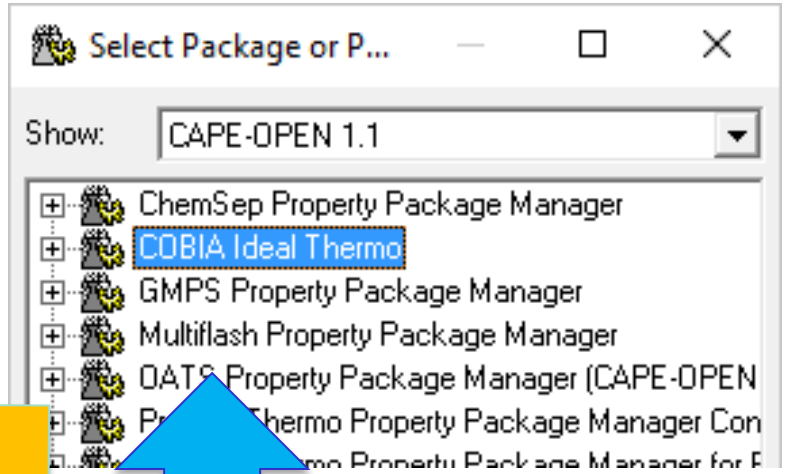
- No explicit allocations or de-allocations
- All responsibility in C++ wrapper classes
- No type checking
- C++ exception handling on wrapper level
  - callee can throw exception
  - caller can catch exception

COM COMPATIBILITY (WINDOWS ONLY)

- COM PMCs are usable from COBIA PMEs
- COBIA PMCs are usable from COM PMEs
- Efficiency is close to COM-COM interop

COM COMPATIBILITY (WINDOWS ONLY)

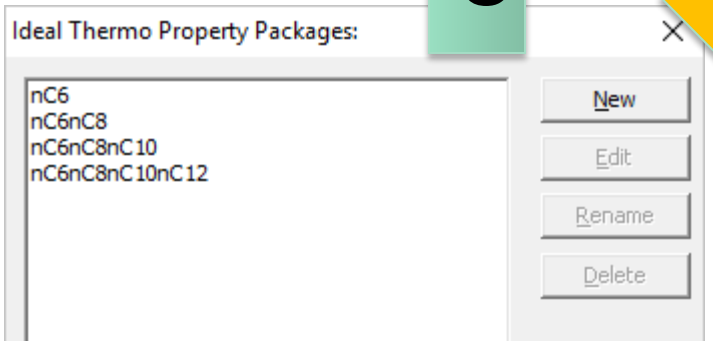


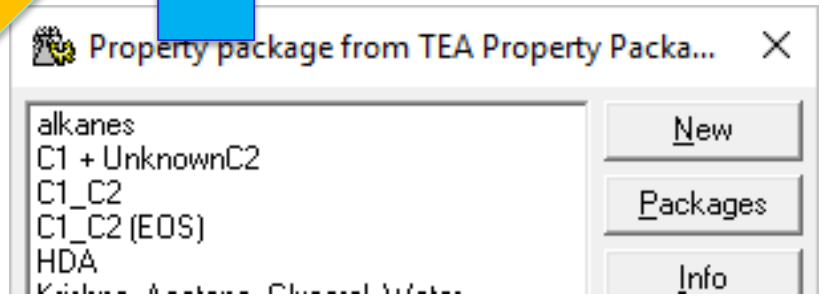


COBIA

COMBIA

COM





TARGETS FOR A NEW MIDDLE WARE

- Platform independent, independent of specific vendor (Microsoft)
- Easier on programmers
- Strong data typing
- More efficient
- Less error prone code
- Open source
- Fully COM compatible

COBIA PHASE I DELIVERY (1/6)

- Registry
  - Per-user + machine wide
  - XML based
  - C++ interfaces to registry
  - Testing coved by Unit Testing
  - Interacts with COM registry
    - COBIA PMCs appear in COM registry
    - COM PMCs appear in COBIA registry

COBIA PHASE I DELIVERY (2/6)

- Data types
  - Several implementation flavours
  - String encoding UTF-16 (Windows), UTF-8
  - Testing covered by Unit Testing
  - Used in Test PMC
  - Used in Test PME
  - Additional interface binding in COMBIA

COBIA PHASE I DELIVERY (3/6)

- ✓ C++ Language binding
  - ✓ Selected interfaces
    - ✓ All thermo 1.1
    - ✓ Identification
    - ✓ Utilities
    - ✓ Simulation environment + diagnostics
  - ✓ Prototyped stub code
  - ✓ Tested in TestPMC + TestPME

COBIA PHASE I DELIVERY (4/6)

- ✓ COM/COBIA interop (COMBIA)
  - ✓ Tested in PMC + PME
- ✓ Test PME
  - ✓ Command line based
  - ✓ Tested vs Test PMC
  - ✓ Tested vs COM interop, various PMCs

COBIA PHASE I DELIVERY (5/6)

- ✓ PMC registration utility
- ✓ Compiles on variety of platforms
  - ✓ MSVC Windows x86, x64 (*)
  - ✓ Intel Windows x86, x64 (*)
  - ✓ GCC/MinGW Windows x86, x64 (*)
  - ✓ GCC/Ubuntu-linux x86, x64

(*) Interop verified

COBIA PHASE I DELIVERY (6/6)

- ✓ Test PMC
  - ✓ Revised Ideal Thermo Library
  - ✓ COBIA binding as PPM
  - ✓ Tested vs TestPME
  - ✓ Tested vs COM interop, various PMEs
- ✓ Deliveries verified by M&T SIG

CONCLUSIONS

- COBIA Phase 1 delivered
- Tailored to the needs of CAPE-OPEN
- Easier coding, less error prone, more efficient
- Paves the way to Linux, Mac, ….

- But: not integrated into OS, requires installer!

PLEASE TRY!

- All code is available to CO-LaN members
- Trying is encouraged!
- Feedback is welcome!

(code resides on CO-LaN code repository)