A Yokogawa Company

The Multiflash CAPE-OPEN COBIA Interface

Behnam Salimi (KBC)
Richard Szczepanski (KBC)
Jasper van Baten (AmsterCHEM)

CAPE-OPEN Annual Meeting October 2020

All about EXCELLENCE

Overview

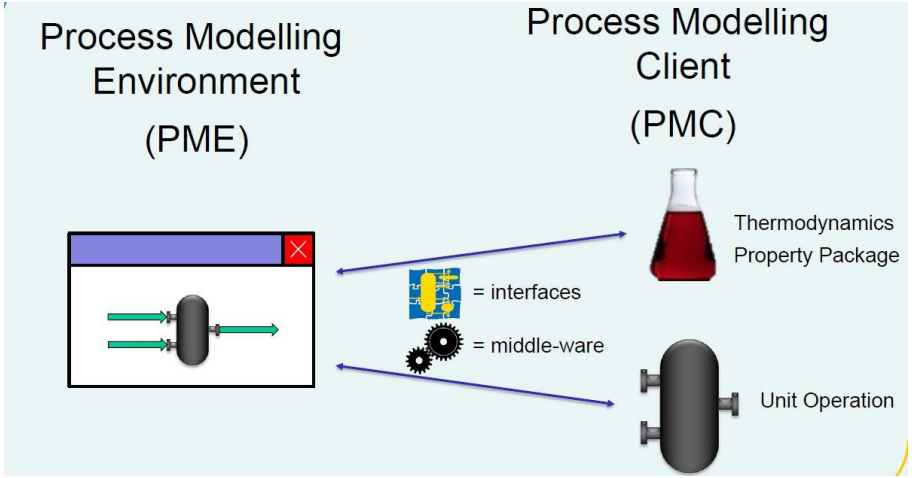
- New release of Multiflash: Multiflash 7.2
- COBIA vs. COM
- Implementation of COBIA based MF CAPE-OPEN Interface
- Performance of COBIA vs COM
- Effect of COMBIA
- Conclusions

Multiflash 7.2

- Release by end of 2020
- Multiflash dll
  - Extension of threadsafe models
  - MF web API
  - Mercury model review
  - Better compatibility of models with other simulators
  - Extension of Python interface
  - Many other developments mostly concerning GUI
- CAPE-OPEN Interface
  - Cobia based interface
  - Thermo 1.0 support by COM interface

KBC
A Yokogawa Company

CAPE-OPEN Binary Interop Architecture



Middle-ware:

- Glue to make everything work
- Defines (format of) data types
- Implements component registration
- Implements component instantiation
- Takes care of interoperability: calling convention and marshalling

COBIA vs. COM

- Multiplatform / dependence on operating system
- No dependence on commercial products
- Data handling
- Strong data typing
- More efficient
- Better error handling
- Support for legacy COM-based CAPE-OPEN
- Better support for future developments of CAPE-OPEN

COBIA Interface Implementation

- COBIA Software Development Kit (SDK):
  - Set of tools to create and test software that utilises COBIA
  - To compile the source code of interfaces developed using COBIA IDL
  - To register COBIA components
  - To test developed software
  - It also includes examples, code generators, portions of the COBIA source code, etc.

- User Interface:
  - COBIA_CodeGen.exe (Command line app)

or

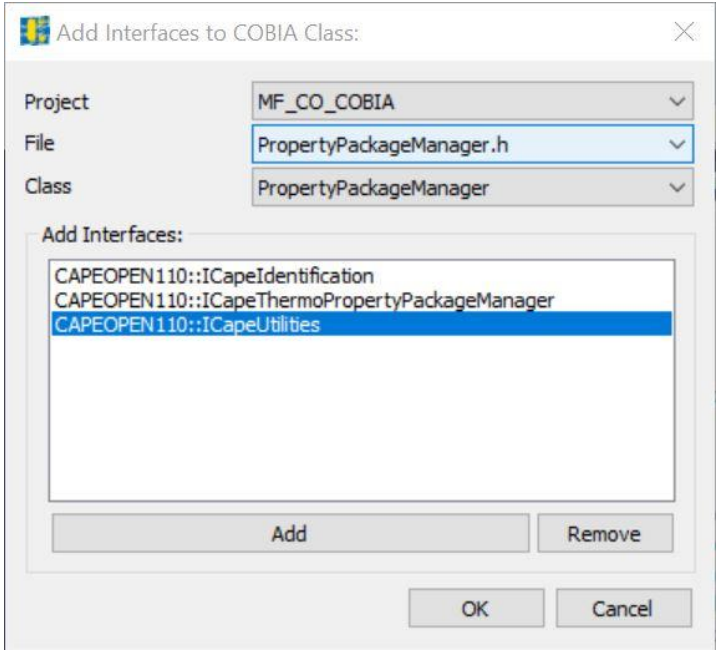
  - AmsterCHEM COBIA Class Wizard Add-in for Visual Studio

AmsterCHEM COBIA Class Wizard

- Add-in for Visual Studio to help develop COBIA PMC
- Based on COBIA Code Generation Interface
- Generates classes and the definitions for all the functions in the classes.
- The COBIA Wizard does NOT generate ready to run PMCs!
- It provides a skeleton with Interfaces and Methods.
- The actions in the methods still have to be provided by the developer.
- Help from example document for creation of Unit Operation using the Class Wizard.

PropertyPackageManager Interfaces

- Right-click PropertyPackageManager.h
- Select Implement CAPE-OPEN Interface on COBIA Class



Click add and select the following interfaces:

CAPEOPEN110::ICapeIdentification
CAPEOPEN110:: ICapeThermoPropertyPackageManager
CAPEOPEN110:: ICapeUtilities

COBIA based CAPE-OPEN object

Easier on Programmers: Interface adapter

```
class PropertyPackage :
        public CapeOpenObject<PropertyPackage>,
        public CAPEOPEN_1_2::CapeIdentificationAdapter<PropertyPackage>,
        public CAPEOPEN_1_2::CapeUtilitiesAdapter<PropertyPackage>,
        public CAPEOPEN_1_2::CapeThermoMaterialContextAdapter<PropertyPackage>,
        public CAPEOPEN_1_2::CapeThermoCompoundsAdapter<PropertyPackage>,
        public CAPEOPEN_1_2::CapeThermoPhasesAdapter<PropertyPackage>,
        public CAPEOPEN_1_2::CapeThermoPropertyRoutineAdapter<PropertyPackage>,
        public CAPEOPEN_1_2::CapeThermoEquilibriumRoutineAdapter<PropertyPackage>,
        public CAPEOPEN_1_2::CapeThermoUniversalConstantAdapter<PropertyPackage>,
        public CAPEOPEN_1_2::CapePersistAdapter<PropertyPackage> {
```

COBIA based CAPE-OPEN object

Easier on Programmers: Generating Stub Code

```
//CAPEOPEN_1_2::ICapeIdentification

void getComponentName(/*out*/ CapeString name) {
}
void putComponentName(/*in*/ CapeString name) {
}
void getComponentDescription(/*out*/ CapeString desc) {
}
void putComponentDescription(/*in*/ CapeString desc) {
}
```

COBIA based CAPE-OPEN object

Easier on Programmers: Error handling

```
//CAPEOPEN_1_2::ICapeIdentification

void getComponentName(/*out*/ CapeString name) {
name = this->name;
}
void putComponentName(/*in*/ CapeString name) {
If (name.empty()){
throw cape_open_error(COBIAERR_InvalidArgument)
packageName = name;
}
void getComponentDescription(/*out*/ CapeString desc) {
}
void putComponentDescription(/*in*/ CapeString desc) {
}
```

BasePropertyPackage (Multiflash PP)

- Started with existing COM-based code (BasePropertyPackage)
- Getting rid of COM specific code and reuse the rest of it

```
//allocate constant BSTR values
STR_MOLECULARWEIGHT=SysAllocString(L"molecularWeight");
```

- Conversion of data types and use COBIA Unified data types:

  COM: LONG, BOOL, BSTR, OLECHAR, …
  COBIA: CapeInteger, CapeBoolean, CapeCapeStringImpl, ..

- Thread safe coding

  Interface class to Lock/Unlock

Remarks on implementation

- The Positive
  - AmsterCHEM COBIA Class Wizard makes it easy to generate the skeleton and framework for the classes selected
  - The available adapter classes are easy to use
  - Easier error handling
  - Less error prone and more efficient
  - Reusing the existing COM based code for many methods
- The challenges
  - Which interfaces should be selected
  - Conversion of COM based code to COBIA (type conversion, data allocation, …)
  - Documentation and examples on COBIA such as the one to develop a Unit Operation
  - Multithreading
  - Test and checking interoperability (future)

Performance benchmarking

- Five different scenarios:

A. COBIA PME / COBIA PMC

B. COM PME / COM PMC

C. COM PME – COMBIA - COBIA PMC

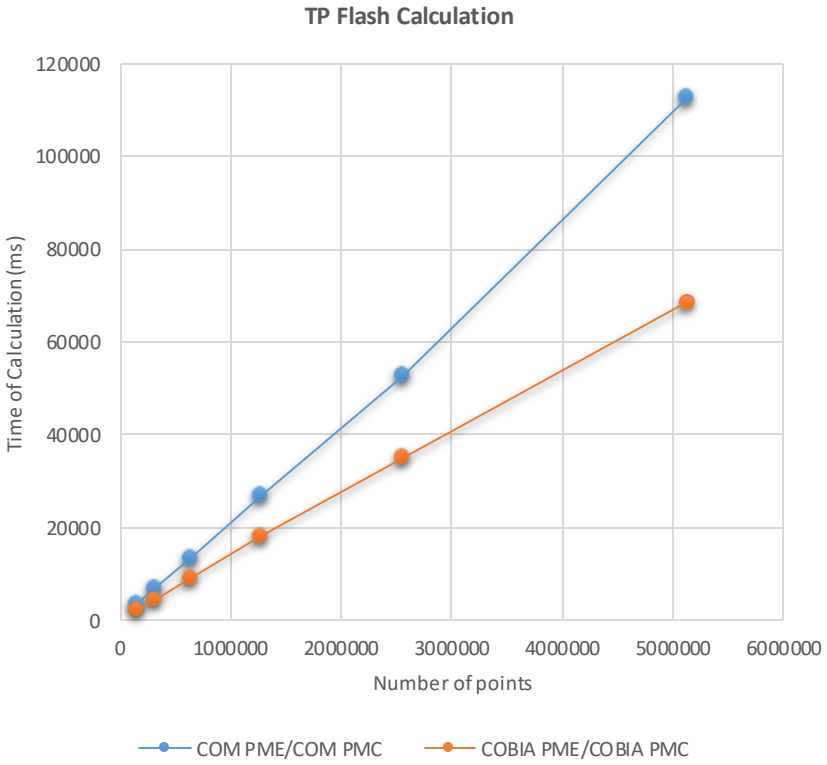
D. COBIA PME – COMBIA- COM PMC

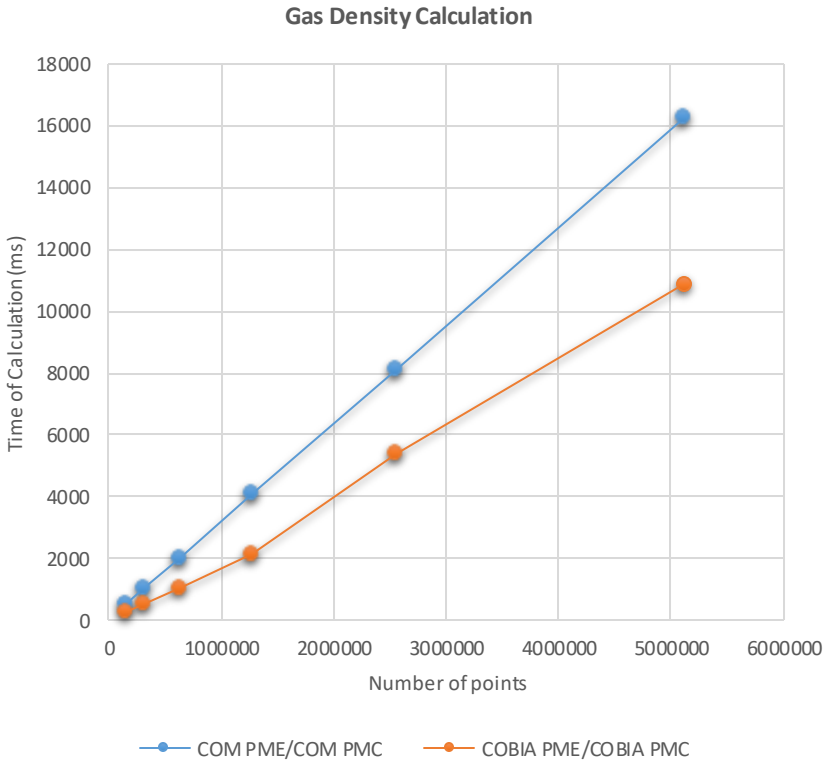
E. Native MULTIFLASH

Performance benchmarking

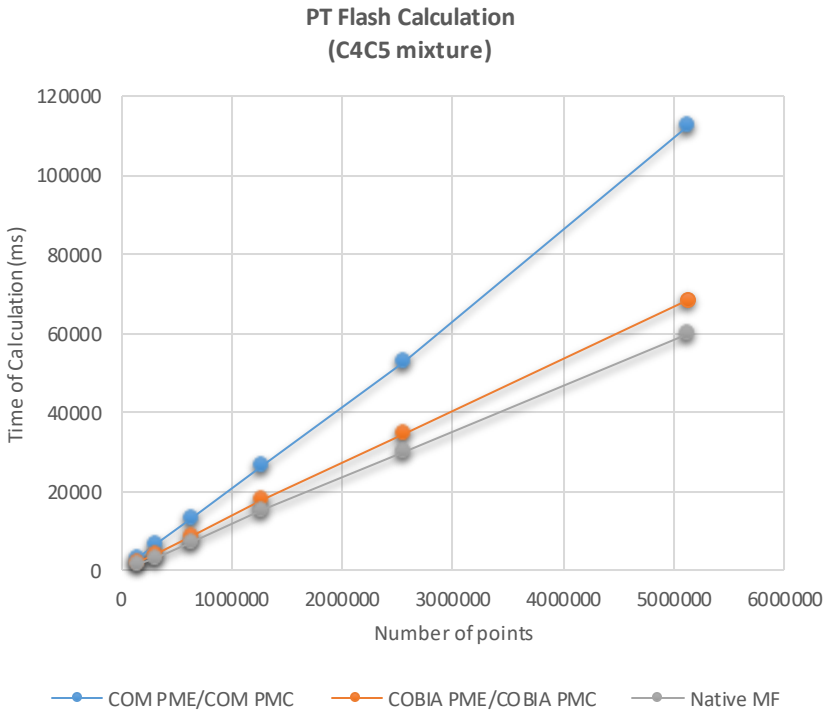
- Test runs using COBIA/COM command line test PME and Multiflash as COBIA/COM PMC.
- Multiflash 7.2 and Peng-Robinson model
- Runs with different number of TP flash calculations(from 160,000 up to 5,120,000 TP flashes)
- Runs for single phase property calculation (gas density) at the same T,P conditions tested for TP flash runs
- Check the effect of number of components: Binary mixture(C4C5) & Crude oil fluid (14 Components)
- Comparison with native MF for TP flash calculations

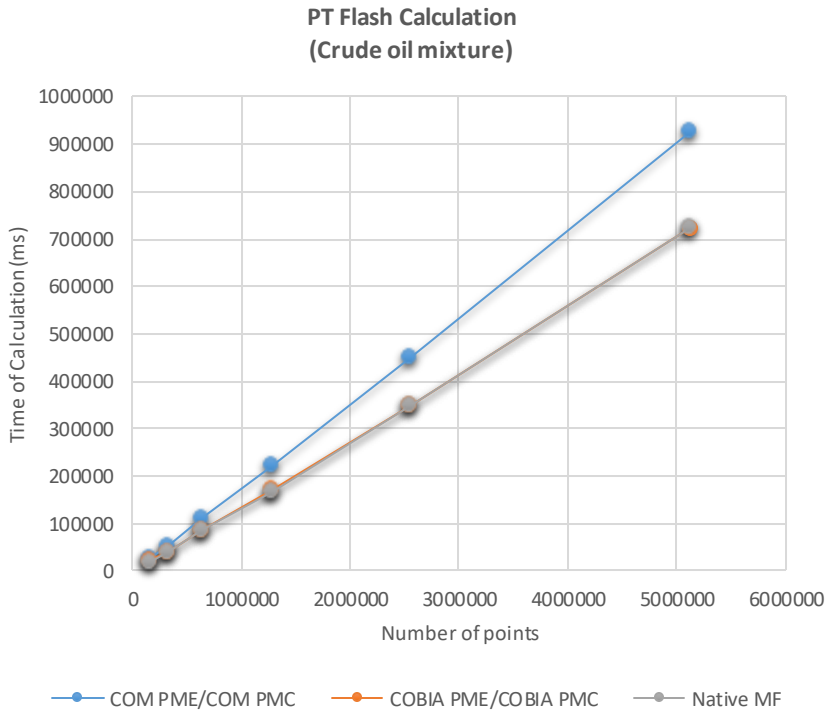
COBIA vs COM performance: (A) & (B)





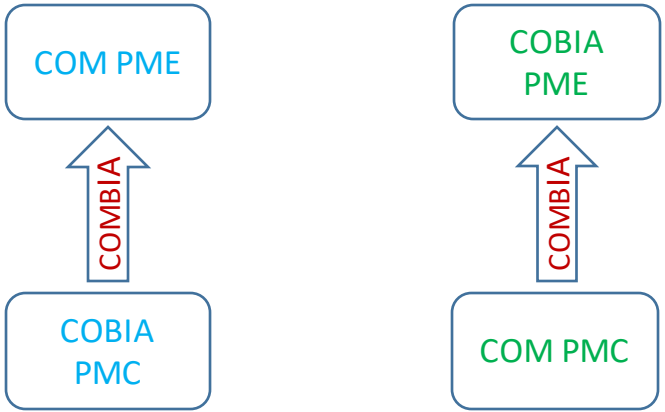
Effect of number of the components: (A) & (B) & (E)





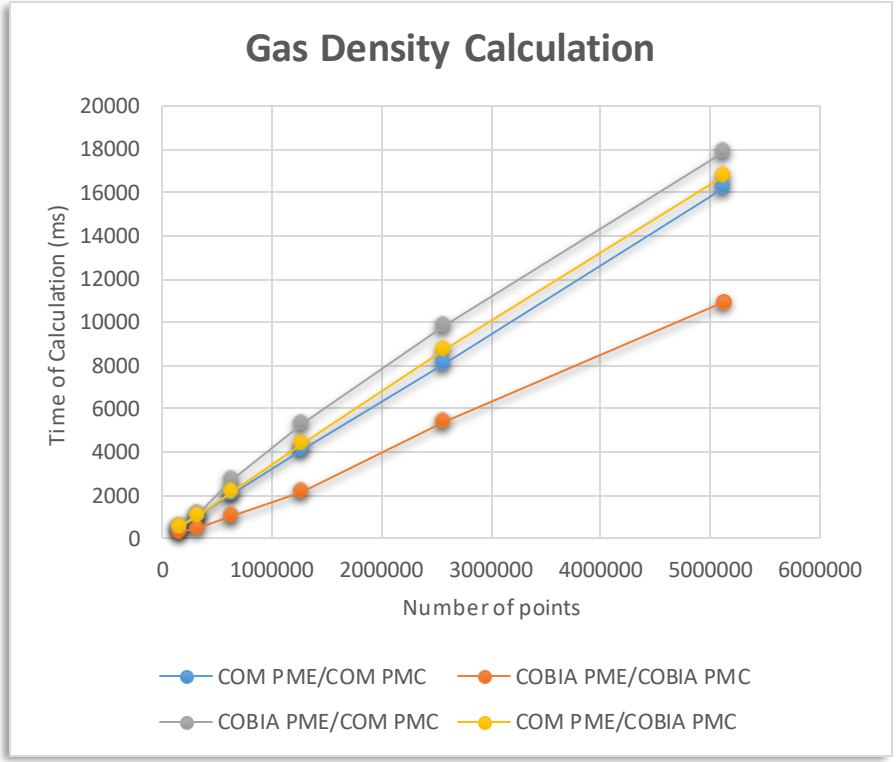
COMBIA

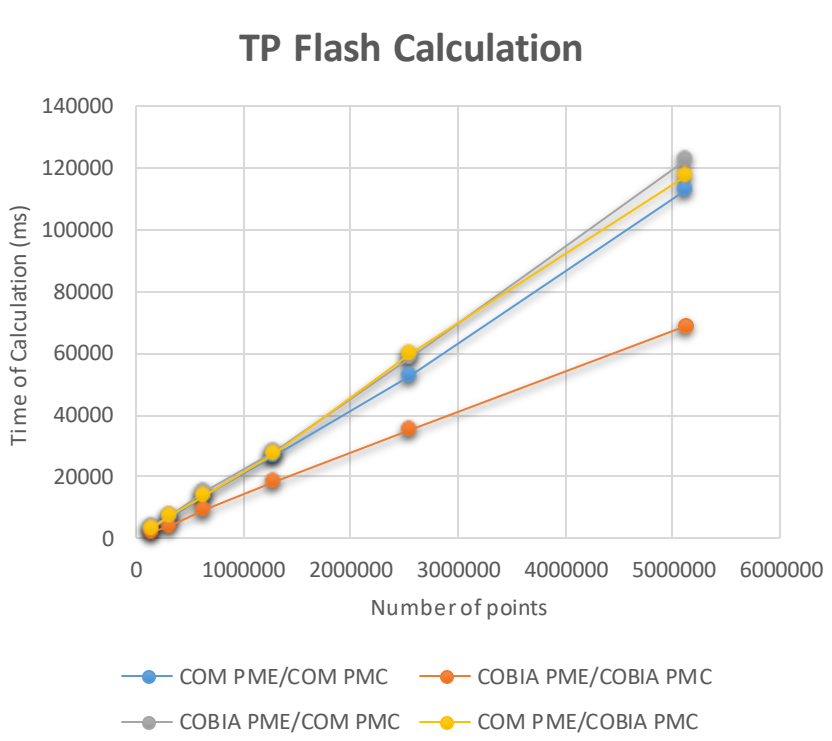
- COMBIA translates COM ↔ COBIA calls



- To use COBIA PMCs directly from COM PMEs and vice versa
- COM PMCs automatically visible to COBIA PMEs
- COBIA PMC registration also registers a COM object (on Windows)

Effect of COMBIA on performance





Conclusions

- COBIA PME/COBIA PMC vs COM PME/COM PMC:
  - For gas density calculations, (A) is 1.4 to 2 times faster than (B).
  - For TP flash calculations, (A) is 1.4 to 1.7 times faster than (B).
  - The larger the number of calculation points the smaller the difference in speed.
- Effect of COMBIA:
  - There is a little difference between (B) , (C) and (D).
  - Although COMBIA has slight effect on speed but the bottleneck is COM.
  - Although very close but (D) is slower than (C).
  - (A) is the fastest among all four options.

- Five different interactions:
  - A. COBIA PME / COBIA PMC
  - B. COM PME / COM PMC
  - C. COM PME – COMBIA - COBIA PMC
  - D. COBIA PME – COMBIA- COM PMC
  - E. Native MULTIFLASH
- Effect of number of components:
  - The larger size of problem the less pronounced will be effect of COBIA or COM.
  - For TP flash calculations runs of the crude oil mixture, (E) is slightly faster than (A).
  - For TP flash calculations runs of the crude oil mixture, (E) is about 1.3 faster than (C).

Thank You!

www.kbc.global

Behnam.Salimi@kbc.global