Methods and Tools SIG Report 2013

SIG Leader

Bill Barrett

SIG Membership

Bill Barrett
US EPA

Jasper van Baten
AmsterCHEM

Bjørn Maribo-Mogensen
Technical University of Denmark

Tony Garratt
Reaction Design

Daniel Wagner

Jorge Martinis and
Michael Hlavinka
Bryan Research and Engineering

Loic d'Anterroches
Céondo, Ltd.

Marc-Olivier Andrez
Process Systems Enterprise Limited

David Jerome

Krishna Penukonda
SIMSCI/INVENSYS

M&T SIG Charter

- Improve integration, and expand utilization of Computer-Aided Process Engineering (CAPE) applications within the enterprise through identification and resolution of existing cross-cutting issues with the CAPE-OPEN platform, develop mechanisms for use of CAPE within other application domains, and incorporate advances in information technology into the CAPE-OPEN platform.

- Key responsibilities
  - Resolve issues with the common interface specifications.
  - Develop and maintain standards and protocols for CAPE-OPEN implementations.
  - Incorporate advances in information technology into the CAPE-OPEN protocols.
  - Identify novel uses of CAPE and provide standards for utilizing CAPE within these applications.

M&T SIG Current Projects

- Review M&T Integrated Guidelines and Common Interface Specifications
  - Identify issues exposed through implementation
  - Provide errata and clarification documents
  - Develop best practice guidance.

- The M&T SIG is currently working on the following interface specifications:
  - Parameters
  - Identification
  - COSE
  - Collections
  - Errors
  - Persistence
  - Utilities
  - Flowsheet Monitoring
- CAPE-OPEN Object Model development

PARAMETER Common Interface

- Errata and clarification document under development
- Dimensionality
  - Formalizes the definition of the dimensionality object as a real-valued array.
  - Up to 10 dimensions
    - Indices 0 through 8 are base dimensions: length, mass, time, electrical current, temperature, mole, luminous intensity, angle.
    - Index 9 – indicates absolute or relative value
    - Index 10 – indicates natural log dimensionality
  - Dimensionality of non-dimensionable parameters (integer, option, boolean) should be returned as an zero-element array.
  - Missing Dimensionality for dimensionable parameters should be treated as dimensionless by the PME.

Parameter Common Interface, cont'd

- Static nature of parameters
  - Static properties of a parameter may include Mode, Dimensionality, Upper/Lower Bounds, Default Values, Option List, and/or whether the value of a parameter is restricted to values in the Option List.
  - Parameter Owner is the only actor that can change the static elements, and these changes can only be made during creation or an edit operation.
  - When changes to the static aspects of a parameter can occur is not specified, so PMEs do not know when it is necessary to re-scan the parameter collection.
  - The parameter value is the only property that can be changed using the CAPE-OPEN interfaces. This can only be changed if the parameter has a mode of *CapeParamMode.CAPE_INPUT* (0) or *CapeParamMode.CAPE_INPUT_OUTPUT* (2).
  - The Parameter Owner can only change the value property of a parameter of mode *CapeParamMode.CAPE_OUTPUT* (1) during a calculation.

Parameter Common Interface, cont'd

- **Array Parameters**
  - There is a consensus amongst CAPE-OPEN developers that the *ICapeArrayParameterSpec* interface is unnecessarily 'general' and that this generality makes it difficult to work with in its entirety.
  - Array parameters are clarified to be:
    - Homogenous element types, *e.g* an array of doubles.
    - No arrays of arrays
    - The *ICapeArrayParameterSpec.ItemsSpecification* shall return an array of *ICapeParameterSpec,* having the same upper/lower bounds, default value, etc.
    - The *ICapeParameterSpec.Type* of the elements of this array should not be CAPE_ARRAY
    - *ICapeParameterSpec.Dimensionality* of the elements of the array should be the same as the *ICapeParameterSpec.Dimensionality* of the array parameter itself.
  - Consumers must support one (1)-dimensional arrays (*ICapeArrayParameterSpec.NumDimensions* == 1). Support for two (2)-, three (3)-, and higher dimensional arrays are optional.

Collection Common Interface

- SIG Final Draft is out for Peer Review.
- Variant Value for *ICapeCollection.Item* method clarified
  - If Integer:
    - Lower Bound for Collection Index is 1.
    - Valid Range: 1 to *ICapeCollection.Count*, inclusive.
  - If String – match to the requested element's value of its *ICapeIdentification.ComponentName* property is case insensitive.
  - Error condition when index requested does not correspond to any item
    - In the case of an integer index that is out of bounds, *ECapeOutOfBounds* should preferably be returned.
    - In the case of a wrong parameter name, *ECapeBadArgument* should preferably be returned.
    - As with all routines that return an *ICapeInterface*, if an error is raised, the caller should not assume that the returned value is valid and the caller should not attempt to release the returned interface.
    - If the method does not return an error, the caller may assume that the method returns a valid (non NULL) interface.

Collection Common Interface, cont'd

- **Naming of Collection Members**
  - The collection needs to ensure that the *ICapeIdentification.ComponentName* property of each object within the collection is unique.
- **Use of Non-CAPE-OPEN mechanisms to access Collections**
  - Implementation of middleware-based accessors is not covered under the CAPE-OPEN standard (e.g. for…each style enumeration using *IEnumXXX* in COM).
  - Nothing in the standard precludes their implementation or use along with the access mechanism provided by the CAPE-OPEN standards and mandatorily implemented.

Simulation Context COSE Interface

- SIG Final Draft is out for Peer Review.
- *ICapeCOSEUtilities::NamedValueList* (section 3.6.3)
  - Only one NamedValue specified in the specification (FreeFortranChannel).
    - Additional named values can be specified by the PME.
    - The M&T SIG will maintain a list of NamedValues using in the CAPE-OPEN context.
    - CO-LaN needs to identify an appropriate forum.
  - Empty NamedValue List
    - Should be a string array containing zero(0) elements – to be defined by clarification of M&T guidelines.
  - Typographical error will be corrected in specification document.

Identification Common Interface

- *Errata and clarification document under development*
- *ICapeIdentification.ComponentName* (section 3.5.1)
  - Uniqueness of *ICapeIdentification.ComponentName property*
    - Collection Owner (*i.e.* the CAPE-OPEN object for parameters, the unit operation for ports, or the PME for Unit Operations and Material Object collections) is responsible for ensuring that the *ICapeIdentification.ComponentName* properties are unique for all collection elements.
  - Minimum and Maximum Length are not specified.
    - Minimum length will be one alphanumeric character.
    - First character of the name must not be whitespace.
    - No maximum length limit set, but should be based on string limitations for the chosen middleware.
  - White space in names – work in progress.
  - Character sets – Issue for M&T guidelines clarification.

Flowsheet Monitoring Interface

- Currently being reviewed and edited by the M&T SIG.

Error Common Interface

- The following issues have been identified
  - Complexity of COM implementation
    - Choice of not using the COM *SetErrorInfo*/*GetErrorInfo*
    - Requirement was instead to force CAPE-OPEN objects to expose all possible CAPE-OPEN error interfaces.
    - Most CAPE-OPEN objects only expose *ECapeRoot* and *ECapeUser* and most only return the *ECapeUnknown* HRESULT
  - Logging tools are required to identify the cause of problems.
- Error Handling Issues to be addressed under the CAPE-OPEN Object Model
  - Possible Structured Exception Handling
  - Improving debugging capabilities

Persistence Common Interface

- No current issues with Persistence.
- Persistence will be covered as part of the CAPE-OPEN object model development.

Utilities Common Interface

- Identified issue to be addressed:
  - Return of *S_FALSE* HRESULT when an edit does not modify an object.

M&T Guidelines Issues

- Issue currently in Peer-Review
  - .NET Interoperability Guidelines Errata and Clarifications
    - Type compatibility issues between CAPE-OPEN objects derived from different imports of CAPE-OPEN type libraries or from manual creation of CAPE-OPEN interfaces in .NET environments.
  - CAPE-OPEN LaN developing a .NET Primary Interop Assembly (PIA) that provides a universal set of .NET-based CAPE-OPEN interfaces.
  - CO-LaN recommends making use as much as possible of Microsoft .NET Framework 4.0 and above when developing CAPE-OPEN PMEs.

M&T Guidelines Issues, cont'd.

- Current Issues Being Address:
  - Note that a distinction needs to be made between the CAPE-OPEN interface specifications and the implementation of these specifications in a particular middleware platform.
    - A number of the issues identified are COM implementation specific.
      - *E.g.* VARIANTs
    - Some items relate to expected behaviour of objects outside of the middleware used.
      - *E.g.* ComponentName uniqueness
  - Current Issues with COM:
    - Transport of zero-length arrays in Variant-wrapped SAFEARRAYs (VT_EMPTY or VT_ARRAY with zero elements).
    - Integer types in COM – 16 bit and 32 bit are currently used.
  - Character sets used in implementation
    - Given CAPE-OPEN is an international standard care needs to be taken to be as inclusive as possible in allowable character sets.

CAPE-OPEN Object Model

- Problems solved by Object Model
  - Cross platform interoperability.
  - Responsibility for memory allocation
  - Marshaling between different platforms such as native, .Net and JAVA.

- Concerns:
  - PMC development could still be quite complicated.
  - PME vendor standpoint, this was another middleware platform that would need to be supported in addition to all of the other ones (e.g. COM and CORBA).

CAPE-OPEN Object Model

- Project Scoping – GOALS
  - System and compiler independent language bindings
  - Efficient in connections between objects
  - Strengthening data typing (reduce/eliminate the need for the variant type)
  - Middleware does memory allocation

- Other Questions
  - How to do in Linux – shared libraries?
  - Will we miss out on custom interfaces?
  - How do we keep the object model from having the complexity of other middleware platforms such as COM? (limit only to early binding?)
  - Who will write the idl compiler and registration tool?
  - QA and Security issues
  - Does the object model make more sense than using something that already exists such as .NET?

2013 Deliverables

- Object Model:
  - Develop design criteria for the CAPE-OPEN Object Model.
- Common Interfaces and Integrated Guidelines:
  - Review the common interface specification documents to identify issues exposed through implementation.
  - Review of Integrated Guidelines to identify issues exposed through implementation.
  - Address errata, provide clarifications, and develop best practices guidance to address the issues identified in (a.) and (b.).
- Flowsheet monitoring:
  - Complete SIG draft version the Flowsheet Monitoring Specification

2014 Deliverables

- Object Model:
  - Provide a roadmap for development of the CAPE-OPEN Object Model.
  - Develop a prototype implementation of the CAPE-OPEN Object Model for one computational platform so that benefits may be seen and interoperability along with COM demonstrated.
- Common Interfaces and Integrated Guidelines:
  - Continue the review of the common interface specification documents to identify issues exposed through implementation.
  - Continue the review of Integrated Guidelines to identify issues exposed through implementation.
  - Address errata, provide clarifications, and develop best practices guidance to address the issues identified in (a.) and (b.).
- Flowsheet monitoring:
  - Complete adoption of the Flowsheet Monitoring Specification

Preliminary Deliverables for 2015

- Integrated Guidelines:
  - Update the integrated guidelines document to specify an Object Model that will allow CAPE-OPEN to be implemented and utilized across multiple platforms including Microsoft Windows, Unix/Linux based systems, and computational clusters.
  - Include in the updated integrated guidelines document all interim guidance that has been developed to date and provide guidelines for resolving other outstanding issues related to the CAPE-OPEN standards.
- Object Model:
  - Test the implementation of the CAPE-OPEN Object Model for one computational platform.
- Common Interfaces:
  - Update common interface specifications to reflect the CAPE-OPEN Object Model.

Longer Term Objectives

- 2016:
  - Object Model:
    - Deliver final implementation of the CAPE-OPEN Object Model for one computational platform.
  - Business Interfaces:
    - Coordinate development of business interfaces (such as thermodynamics and unit operations) with the respective special interest groups.
- 2017:
  - Interface Specifications:
    - Publish final versions of the common interface specifications based upon the CAPE-OPEN Object Model.
    - Help other SIGs finalize their CAPE-OPEN Object Model interface specifications.

Ongoing Activities

- Common Interface Conference Calls
  - First Wednesday of each Month at 11 AM Eastern US Time.
- Object Model Conference Call
  - Last Wednesday of the month at 10 AM Eastern US Time.
- Goal is to complete majority of Common Interface Work by the end of the calendar year and consolidate calls.
- Please contact either SIG Leader or CTO if you are interested in participating:
  - Bill Barrett – barrett.williamm@epa.gov
  - Michel Pons - technologyofficer@colan.org