The MOSAIC Approach – Self-Made CO-UOs Without Programming Knowledge

Gregor Tolksdorf, M.Sc.  |  Faculty of Process Sciences  |  CAPE-OPEN 2015 Annual Meeting
Modular MOdel Specific Ation on Documentation Level - Application in a Web Based Modeling Environment.

www.mosaic-modeling.de
Self-Made CO-UOs Without Programming Knowledge | G. Tolksdorf | CAPE-OPEN 2015 Annual Meeting

www.mosaic-modeling.de
MOSAIC-Team

Scientific supervisor:
Prof. Dr.-Ing. G. Wozny,
Head of Chair of Process Dynamics and Operation, TU Berlin

Current Developer:
Dr.-Ing. E. Esche
Dipl.-Ing. S. Fillinger
Dipl.-Ing. V.A. Merchan-Restrepo
M.Sc. G. Tolksdorf
Agenda

1. Motivation – The MOSAIC Approach
2. Unit Operations & Physical Properties
3. Code Generation
4. Application Example
5. Summary & Outlook
Self-Made CO-UOs Without Programming Knowledge

MOTIVATION –

THE MOSAIC APPROACH
MOSAIC – Why?

Models and their application change over time.

Problems:
• Documentation is outdated
• Reimplementation is error-prone

Eutopia: Process Engineer == Programming Expert

MOSAIC offers:
• Transparent model documentation
• Automatic code generation
MOSAIC + CAPE-OPEN Advantages

- Encourages systematic modeling
- Eliminates redundant, error-prone manual implementation
- Enables usage of platform-independent models in virtually any simulation software based on
  - Equations (z.B. PSE, …)
  - Flowsheeting (z.B. Aspentech, Pro Sim SA, Honeywell Process Solutions, Amsterchem, …)

Vision: « Model once, simulate anywhere »

*Image “Hand being bandaged as injury” courtesy of Stuart Miles / FreeDigitalPhotos.net*
Agenda

1. Motivation – The MOSAIC Approach
2. Unit Operations & Physical Properties
3. Code Generation
4. Application Example
5. Summary & Outlook
Self-Made CO-UOs Without Programming Knowledge

UNIT OPERATIONS &

PHYSICAL PROPERTIES
MOSAIC – Unit Operations

A Unit Operation in MOSAIC is…

\[ N_{out,c} = N_{in,c} \quad p_{out} = p_{in} + dp \quad T_{out} = T_{in} \]

…an equation system with ports.

MOSAIC-Ports translate variables and are connection points for streams.
The Definition of Ports:

Connector connects variables from the System to the Interface

Interface defines the Stream Type (Material, Energy, etc.)
MOSAIC – Units and Flowsheets

Unit Equation System

<table>
<thead>
<tr>
<th>Interface to Stream</th>
<th>Interface (Stream type)</th>
<th>Interface to Stream</th>
</tr>
</thead>
<tbody>
<tr>
<td>Con</td>
<td>nect</td>
<td>or</td>
</tr>
</tbody>
</table>

MOSAIC Stream

Optional Connectors for Conversion
MOSAIC – Units and Flowsheets

MOSAIC Unit Operations: „Equation-based flowsheeting“
MOSAIC – Physical Properties

(External) physical properties in MOSAIC are...

![Function](image)

...variables that are calculated by external, language-specific functions.

Example code (using Amsterchem’s „Matlab CAPE-OPEN Thermo Import“):

```matlab
function[std_p_LV_iALL] = fun_12513_co_vapor_pressure_function(std_T)
global co_handle;
std_p_LV_iALL = capeOpenTDepProp(co_handle,'vaporPressure',std_T);
end
```
Agenda

1. Motivation – The MOSAIC Approach
2. Unit Operations & Physical Properties
3. Code Generation
4. Application Example
5. Summary & Outlook
Self-Made CO-UOs Without Programming Knowledge

CODE GENERATION
MOSAIC – Code Generation

MOSAIC offers code generation for several predefined languages:

- general purpose programming languages
- specific environments for simulation
- optimization languages
MOSAIC – Code Generation

MOSAIC users can define code generators for **new** languages.

Example:
Code generator for Amsterchem‘s „Scilab CAPE-OPEN Unit Operation“
Agenda

1. Motivation – The MOSAIC Approach
2. Unit Operations & Physical Properties
3. Code Generation
4. Application Example
5. Summary & Outlook
Self-Made CO-UOs Without Programming Knowledge

APPLICATION EXAMPLE
Application Example - Simple Membrane Separation

MOSAIC:

*van Baten, Taylor, Kooijman: „Using Chemsep, COCO and other modeling tools for versatility in custom process modeling“, AIChE 2010*
Application Example - Simple Membrane Separation

MOSAIC:

*van Baten, Taylor, Kooijman: „Using Chemsep, COCO and other modeling tools for versatility in custom process modeling“, AIChE 2010
Application Example - Simple Membrane Separation

MOSAIC:

*van Baten, Taylor, Kooijman: „Using Chemsep, COCO and other modeling tools for versatility in custom process modeling“, AIChE 2010
Application Example - Simple Membrane Separation

MOSAIC:

```plaintext
1 // Scilab output for algebraic equation systems
2 // Copy this code into Amsterchems Scilab Unit Operation
3 // Code generated by the help of MOSAIC
4 // Please enter the following inlet ports in the Ports tab:
5 // feed_port
6 // Please enter the following outlet ports in the Ports tab:
7 // permeate_port
8 // retentate_port
9 // Please enter the following parameters in the Parameters tab:
10 // f0_greek_theta default value: 0.6
11 // f1_greek_alpha default value: 70.0
12 // function[Y] = getFunVal(X,ITER,PARAMS)
13 // Calculate the function value of a normalized equation system.
14 // read out variables
```

*van Baten, Taylor, Kooijman: „Using Chemsep, COCO and other modeling tools for versatility in custom process modeling“ , AICHE 2010
Application Example - Simple Membrane Separation

COCO/COFE:

*van Baten, Taylor, Kooijman: „Using Chemsep, COCO and other modeling tools for versatility in custom process modeling“, AIChE 2010
Application Example - Simple Membrane Separation

COCO/COFE:

*van Baten, Taylor, Kooijman: „Using Chemsep, COCO and other modeling tools for versatility in custom process modeling“, AIChE 2010
Application Example - Simple Membrane Separation

COCO/COFE:

*van Baten, Taylor, Kooijman: „Using Chemsep, COCO and other modeling tools for versatility in custom process modeling“, AIChE 2010

Self-Made CO-UOs Without Programming Knowledge | G. Tolksdorf | CAPE-OPEN 2015 Annual Meeting
Agenda

1. Motivation – The MOSAIC Approach
2. Unit Operations & Physical Properties
3. Code Generation
4. Application Example
5. Summary & Outlook
Self-Made CO-UOs Without Programming Knowledge

SUMMARY & OUTLOOK
Summary & Outlook

MOSAIC

• is an equation-based modeling and code generation tool

• covers unit operations and physical property calls

• automatically generates code for various programming languages, including
  • MatLab/SciLab Cape-Open UnitOperations by Amsterchem
Summary & Outlook

Vision:
« Model once, simulate anywhere »  
by creating CO-UOs with MOSAIC

Next steps:
• C++ code generation for Cape-Open unit operations  
• Direct delivery of a Cape-Open unit operation shared library (DLL)
Thank you very much for your kind attention.

www.mosaic-modeling.de

Acknowledgement:

This project is
• supported by the Cluster of Excellence 'Unifying Concepts in Catalysis'
• coordinated by the Technical University of Berlin and
• funded by the German Research Foundation.
Self-Made CO-UOs Without Programming Knowledge

BACKUP SLIDES
Goals and Resulting Characteristics

Main goals:
- Less errors
- Less effort
- More cooperative work
  - Improved reuse
  - Improved portability

Resulting characteristics:
- Highly modular modeling concept
- Define Platform Independent Models (PIM) in the documentation level using an enhanced symbolic notation
- Use of PIM and code generation to Platform Specific Models (PSM)
- Support web-cooperation
  - Store and share all model elements in a web database
Modular Modeling Concept – The Editors

MOSAIC editors and model elements:

- **Notation**
  - What symbols/variables are allowed?
- **Equation**
  - What equations will be used?
- **EquationSystem**
  - How will the equations be combined? What functions will be used?
- **Evaluation**
  - What are the design, state, and iteration values? How does the problem solving code look like?
Enhanced Symbolic Notation I - Variables

Notation editor:

Example:

\[ p_{o,i=2,j=4}^{LV,I} \]
Enhanced Symbolic Notation II - Equations

Equation editor:

\[
\frac{d}{dt} HU_i^m = F_{in}^m \cdot x_{i,in = 1} - F_{out}^m \cdot x_{i,out = 1} + r_i
\]
Enhanced Symbolic Notation III – Equation Systems

Equation system editor:
Enhanced Symbolic Notation IV – Instantiated Equations

Evaluation editor:
Code Generation I – What language do you prefer?

Choose of a list of supported target platforms:
Code Generation II – Show me what you got!

Take a look at the generated code:
MOSAIC Ports – let’s get connected

MOSAIC external Ports:

Y – internal variable
A – variable connected to input T
B – variable connected to output P
MOSAIC Ports – let’s get connected

MOSAIC external Ports:

- **Direction:**
  - In or Out

- **Interface:**
  - Which variables will be presented?
    - > naming, dimension, engineering unit, direction
      - e.g. p, scalar, bar, out

- **Connector:**
  - How are internal variables and external interface variables connected?
    - > e.g. A <-> T, B <-> P
CAPE-OPEN and MOSAIC I – Physical Properties

Physical properties in MOSAIC:
Variables to be calculated by external functions, e.g.

\[ p_i^{LV}(T) \]

Supported target platforms for „CO physical properties“ code generation:

• Matlab

• gPROMS
CAPE-OPEN and MOSAIC I – Physical Properties

MOSAIC Example – CO function:

\[ h^L(T, p, \gamma_2-1, \gamma_2-2) = \text{CO Calculate Molar Liquid Enthalpy} \]

Description: Liquid enthalpy with CAPE OPEN

No. of usages: 13
CAPE-OPEN and MOSAIC I – Physical Properties

MOSAIC Example – Matlab code:

```matlab
function [std_h_L] = fun_12514_co_liquid_enthalpy_function(std_T, std_p, std_x_i1, std_x_i2)
    global co_handle;
    std_h_L = capeOpen1PhaseProp(co_handle, 'enthalpy', 'liquid', std_T, std_p, [std_x_i1, std_x_i2]);
end

function [std_p_LV_iALL] = fun_12513_co_vapor_pressure_function(std_T)
    global co_handle;
    std_p_LV_iALL = capeOpen1DepProp(co_handle, 'vaporPressure', std_T);
end
```
Summary

**MOSAIC**

- A modular equation based modeling tool
- Implemented in Java, using XML/MathML
- Provides automatic code generation for specific platforms (e.g. Matlab, C++)
- Can use the concept of ports
- Supports CO physical properties in code generation (Matlab, gPROMS)

MOSAIC is not

- Designed to be a full solver / process simulator
- A programming language
- A computer algebra system (CAS)
Thank you very much for your kind attention.

Acknowledgement:

This project is
• supported by the Cluster of Excellence 'Unifying Concepts in Catalysis'
• coordinated by the Technical University of Berlin and
• funded by the German Research Foundation.