POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Design and Validation of an Object-Oriented Modelica Library for Dynamic Chemical Process Simulation

Laurea Magistrale in Automation and Control Engineering - Ingegneria dell'Automazione

Author: Federico Esposito

Advisor: Prof. Alberto Leva

Academic year: 2024-2025

---

1. Introduction

Dynamic modeling plays a central role in the system engineering of chemical plants. Beyond its relevance for control design, dynamic plant-wide models are routinely used to analyze interactions among unit operations, assess recycle loops, and support architectural decisions involving inventories and throughput constraints. In industrial practice, these needs are traditionally addressed using commercial process simulators [1, 2, 7]. While robust, these tools rely on closed architectures that limit flexibility when customized modeling structures or direct access to the underlying equations are required.

Over the last decades, Object-Oriented Modeling (OOM) and acausal languages like Modelica have emerged as a promising alternative. By promoting modularity and decoupling physical equations from numerical solvers, OOM facilitates the construction and maintenance of large-scale models that evolve over time [4, 5]. However, when moving from individual units to reactive chemical plants at system level, a structural difficulty emerges. Reaction-driven dynamics are inherently species-based, local to individual units, and contribute simultaneously to multiple balance equations. This representation does not map straightforwardly onto the stream- and port-based abstractions typically employed in object-oriented frameworks.

This thesis addresses this gap by designing *OpenProc*, a modular library that reconciles the flexibility of equation-based modeling with the pragmatic abstractions of process engineering. The framework is validated through the implementation of the Tennessee Eastman benchmark process [3], demonstrating that open-source OOM can achieve the necessary robustness for system-level analysis.

2. The Structural Challenge

While OOM languages are mathematically fully capable of representing chemical systems, a fundamental difficulty emerges at the architectural level. This stems from a mismatch between domain physics and standard software abstractions. Standard frameworks organize interactions through *port-based* connectors, where components exchange conserved quantities (mass, energy, momentum) along well-defined boundaries. This paradigm works exceptionally well for transport phenomena (e.g., fluid flow in pipes), which map naturally onto connections, but breaks down for chemical reactions.

The main conceptual difficulty lies in the fact that reactions are inherently *species-based* and act as *local* transformations of composition. In a strictly object-oriented formulation, reaction kinetics are typically handled through *expandable connectors*, where the reaction interface is dynamically constructed as the union of participating species represented by effort–flow pairs. While formally elegant and maximally generic, this design shifts the complexity to the wiring level. Key modeling choices (such as stoichiometry and species participation) become distributed across implicit connection equations rather than being encapsulated within explicit process-level constructs.

This results in "fragile" models where the reaction network is no longer identifiable as a first-class engineering element. As conceptually illustrated in *Figure 1*, intermediate species often exist only locally (e.g., produced and consumed within a reactor) and never cross a boundary. Yet, standard port-based abstractions force these species to be exposed in the stream interfaces or managed through complex implicit equations. Consequently, small process changes (such as adding a side reaction or modifying a stoichiometric coefficient) often require non-local code refactoring and extensive manual bookkeeping of array indices [6]. This increases the cognitive load on the modeler and reduces the robustness of plant-wide models.

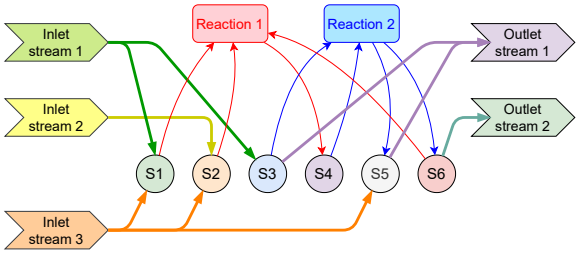

Figure 1: Conceptual illustration of the structural mismatch. Reaction-driven couplings are inherently local and do not map straightforwardly to stream interfaces, creating fragility in standard OOM.

3. The Proposed Approach

Equation-based object-oriented modeling languages make it possible to represent a chemical plant as a hierarchy of interacting objects with strict encapsulation. A fully object-oriented formulation would push this idea to its logical extreme, enforcing uniform abstraction patterns across all components. However, such a stance is only partially aligned with established process-engineering practice. Engineers typically reason in terms of flowsheets and material streams rather than abstract object hierarchies. At plant level, readability and semantic clarity often outweigh the benefits of maximal software abstraction.

For this reason, the approach adopted in this work is explicitly *flowsheet-driven*. Object-oriented mechanisms are used as an enabling technology rather than as a goal in themselves. The objective is to preserve the familiar workflow of industrial simulation while exploiting the transparency of equation-based modeling. In the proposed workflow, the flowsheet is the main modeling object. A plant model is constructed by instantiating unit operations and interconnecting them through stream interfaces, while reaction-driven phenomena remain local to the units. Conservation laws are enforced locally within components and globally through the connection structure, without requiring explicit equation wiring by the user.

This design deliberately relaxes strict OO purity. In particular, encapsulation is softened in favor of semantically rich connectors that expose physically meaningful variables. This reflects a conscious trade-off: reduced object autonomy is accepted in exchange for improved readability and easier modification of plant topology. Importantly, this choice shifts the burden of abstraction from the user to the modeling framework. As a consequence, typical modeling actions (such as changing feed composition or restructuring recycle loops) can be performed locally, without global refactoring. This property is essential in practice, where process models are frequently revised during design and analysis.

Within this workflow, material streams are treated as semantic entities rather than anonymous numerical vectors. In addition to transporting flow rates, streams explicitly encode chemical identity. To support this, the library relies on a set of *utility functions* that automate the mapping between local component lists and stream variables. This prevents silent inconsistencies when connecting heterogeneous units and removes the fragility associated with purely

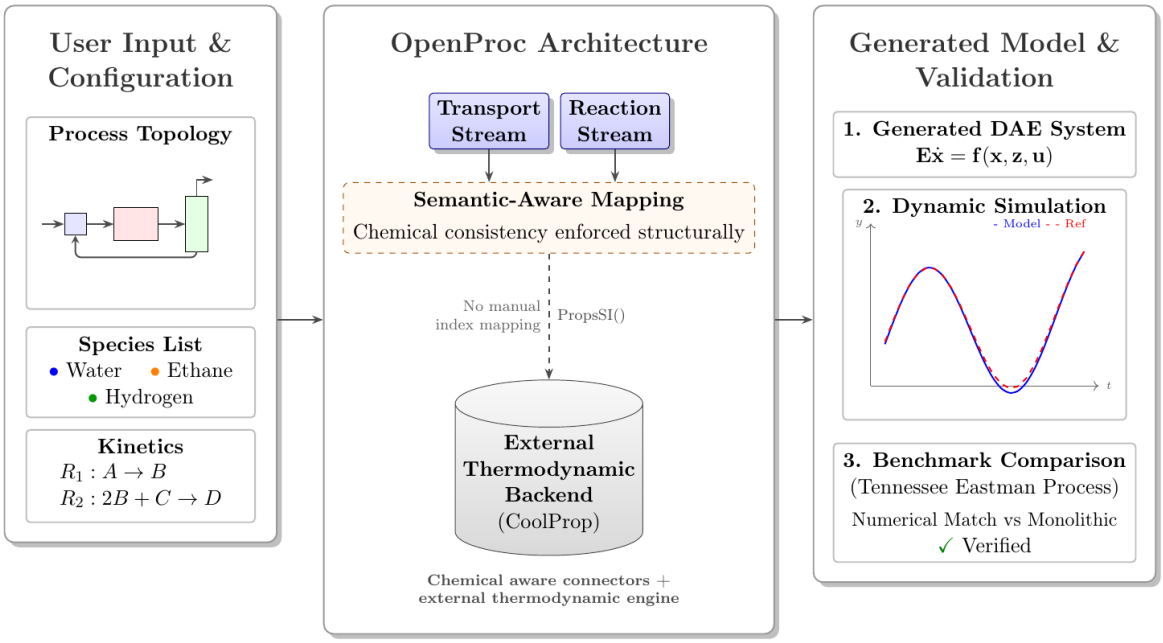

Figure 2: Overview of the proposed *OpenProc* architecture. The framework decouples user input (topology and kinetics) from the underlying DAE generation. Thermodynamic consistency is enforced via semantic mapping and an external C++ backend, enabling robust dynamic simulation.

positional indexing.

Unit operations are formulated in a chemically agnostic way at the library level: their governing equations depend only on generic conservation principles. To ensure computational efficiency and robustness, thermodynamic property evaluations are not hard-coded in Modelica but delegated to an external C++ backend (based on *CoolProp*) via a custom interface. This hybrid architecture enables full component reuse across different plants and operating conditions, while preserving structural consistency. The overall architecture is depicted in *Figure 2*, which highlights the structural decoupling between the user-defined flowsheet and the external thermodynamic computation.

A clear separation is thus maintained between the generic modeling library and case-specific definitions. The core library provides reusable unit operations and stream interfaces, while all application (dependent information—such as species sets and parameters) is confined to a dedicated *Case Layer*. From the modeler's perspective, this mirrors standard industrial practice: the plant is assembled graphically from known units, and process data are configured without modifying internals. This workflow provides the methodological foundation for the case study presented in the next section. The Tennessee Eastman process is used to demonstrate how a complex plant can be simulated within this framework without departing from established process-engineering intuition.

4. A representative example

This section shows how the flowsheet-driven modeling stance established in *Section 3* materializes in a plant-scale model, using the Tennessee Eastman (TE) benchmark as a representative case. The objective is to document model structure and workflow, making the correspondence between the process flowsheet and the object-oriented implementation immediate and readable.

The TE benchmark represents a multi-unit chemical plant featuring reaction, phase separation, recycle compression, and downstream stripping, organized as a conventional process flowsheet (*Figure 3*). In the proposed stance, this flowsheet is not merely a documentation artifact but the primary blueprint for the model architecture. The Modelica model is assembled by instantiating one object per unit operation and connecting them through typed stream interfaces, reproducing the same topology as the

PFD (*Figure 4*). This yields a one-to-one correspondence between flowsheet structure (units and streams) and object structure (components and connectors). Consequently, plant-wide behaviour is expressed by composition rather than by a monolithic set of equations. The construction workflow is deliberately flowsheet-centric: the model is built through graphical placement and connection, while unit parameters and operating conditions are provided through component parameterization rather than by editing connection equations.

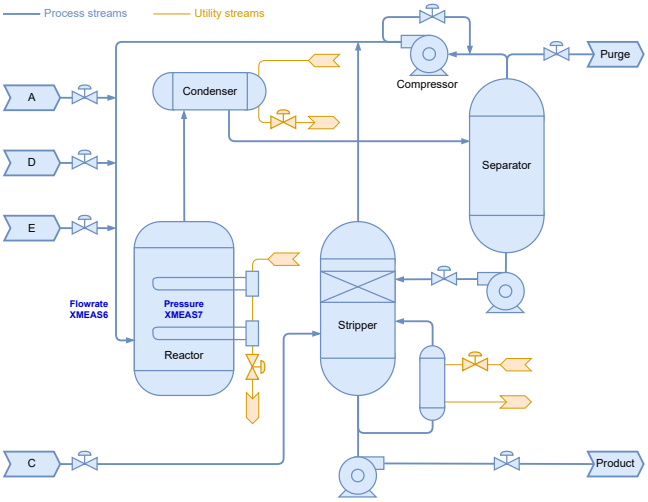

Figure 3: Simplified flow diagram of the Tennessee Eastman benchmark process (XMEAS 6 and XMEAS 7 are the variables shown in *Figures 5* and 6, *Section 5*).

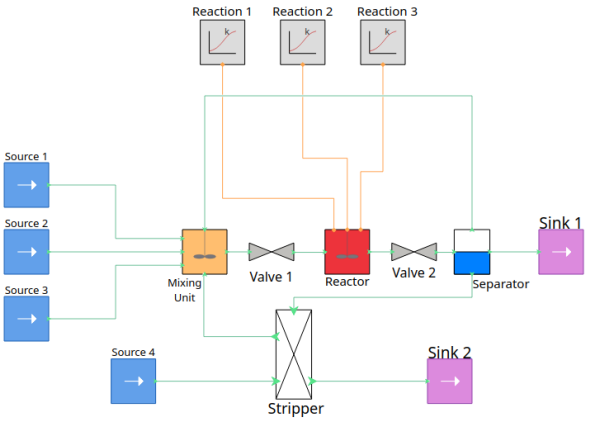

Figure 4: Modelica diagram of the Tennessee Eastman process model, constructed according to the proposed flowsheet-driven, object-oriented structure.

To support this modular composition, material transport between unit operations is represented by a *semantic stream interface* carrying flow variables, intensive properties, and explicit chemical identity. *Listing 1* reports the essential structure of the transport connector, highlighting the explicit species identifier field used to attach chemical meaning to the composition vector.

```
connector TransportStream
  parameter String iupacNames[:] = {"
      Water"};
  Modelica.Units.SI.Pressure p;
  flow Modelica.Units.SI.MolarFlowRate
      wm;
  stream Modelica.Units.SI.Temperature T
      ;
  stream Modelica.Units.SI.MoleFraction
      y[size(iupacNames, 1)];
end TransportStream;
```

Listing 1: `TransportStream` connector

Species sets are not fixed globally: they are specialized at the case level, so that each stream instance is chemically self-describing and components can enforce compatibility structurally. This decouples process topology from chemical definition and prevents silent inconsistencies when interconnecting heterogeneous streams. Case-level specialization is performed without modifying library components, by defining specialized connector types and redeclaring the corresponding ports in unit instances. An example is shown in *Listing 2*, where a two-species stream is defined and then used to specialize a generic source model through redeclaration.

```
connector Tstream_AB
    extends TennesseeEastmanLibrary.
        Types.TransportStream(
    iupacNames = {"A", "B"}
    );
end Tstream_AB;

model Source_AB
    extends TennesseeEastmanLibrary.
        General_components.Source(
    redeclare Case.Interfaces.Tstream_AB
        outlet
    );
end Source_AB;
```

Listing 2: Case-level specialisation via redeclaration.

In the TE flowsheet, different units naturally operate on different subsets of components (e.g., vapor-phase species in reaction and recycle, liquid products in stripping). The semantic stream mechanism allows these heterogeneous subsets

to coexist transparently: units remain generic, while the case package defines which species each connection can carry.

Complementing this transport architecture, reaction-driven dynamics are handled locally at the unit level, following standard process-engineering intuition: kinetics are a constitutive element of the reactor, not a plant-level coupling. Accordingly, reaction rate laws are encapsulated in dedicated kinetic blocks that interact with the reactor through a reaction-specific interface, while transport between units is handled exclusively by material streams. This separation isolates responsibilities: the reactor enforces conservation, while kinetic blocks compute source terms and heat generation from the local thermodynamic state. *Listing 3* shows an excerpt of a TE kinetic block implementing an empirical partial-pressure rate law, expressed locally and without introducing any global reaction logic.

```
equation
Rate = if noEvent(port.p[1]>0 and port.p
    [2]>0 and
port.p[3]>0) then
(1/3600)*alpha*port.Vvap*
exp(44.06 - 42600/(Rkcal*port.T))*
(port.p[1]*1e3)^1.08*(port.p[2]*1e3)
    ^0.311*(
port.p[3]*1e3)^0.874
else 0;
end Kinetics_ACDtoG;
```

Listing 3: Kinetic block implementing a reaction rate law, expressed locally at unit level.

Finally, the framework proves flexible enough to accommodate "grey-box" or hybrid elements when mandated by the benchmark specification. For instance, the TE stripper requires an empirical formulation that does not follow standard component balances. As shown in *Listing 4*, this implementation is easily embedded within the same semantic connector architecture, maintaining the overall flowsheet consistency.

```
for i in 1:nCompLiquid loop
  phi[i+nCompVapor] = PC_1[i]*(Tstr-
      Tref)^3 + PC_2[i]*(Tstr-Tref)
      ^2 + ...;
end for;
F5.wm + F10.wm + F4.wm + F11.wm = (
    der(NG) + der(NH));
```

Listing 4: Excerpt of empirical unit model embedded in the flowsheet-driven framework

As an overall outcome, the TE case study confirms that a flowsheet-driven stance can be realized within an equation-based, object-oriented model without sacrificing locality or transparency. The resulting modular structure exposes local dynamics and inter-unit couplings explicitly, providing a natural basis for system-level control design without requiring additional model restructuring.

5. Simulation results

This section reports representative open-loop simulation results obtained with the proposed modular formulation and compares them against reference trajectories available in the literature. The objective is twofold: first, to verify that the flowsheet-driven, object-oriented reconstruction preserves the physical dynamics of the Tennessee Eastman benchmark; second, to assess the numerical implications of the modular architecture without entering toolchain-specific implementation details that are inessential to the scope of this paper.

All simulations are performed in open loop, starting from the nominal operating point. This choice is intentional: the Tennessee Eastman process is known to be open-loop unstable, and even small discrepancies in model structure or numerical handling would rapidly amplify, leading to diverging trajectories. Successfully reproducing the reference dynamics therefore constitutes a stringent validation of the proposed modeling approach.

Figure 5 reports the open-loop dynamic response of the reactor feed rate (XMEAS 6), representative of fast hydraulic dynamics. The trajectory obtained with the modular formulation is superimposed to the reference implementation reported in the literature. The two curves are indistinguishable, confirming that fast-scale dynamics are preserved despite the architectural reformulation.

Figure 6 shows the corresponding open-loop response of the reactor pressure (XMEAS 7), which captures slower plant-wide dynamics dominated by recycle interactions and the well-known snowball effect. Also in this case, the modular formulation reproduces the same instability trends and time constants reported in the reference literature, confirming that the proposed structure preserves the essential plant-

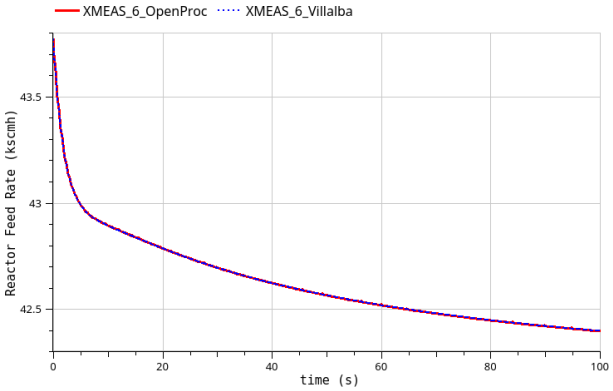

Figure 5: Open-loop dynamic response of the reactor feed rate (XMEAS 6). The trajectory obtained with the proposed modular formulation is superimposed to the reference implementation reported in the literature, showing identical fast-scale dynamics.

wide feedback mechanisms.

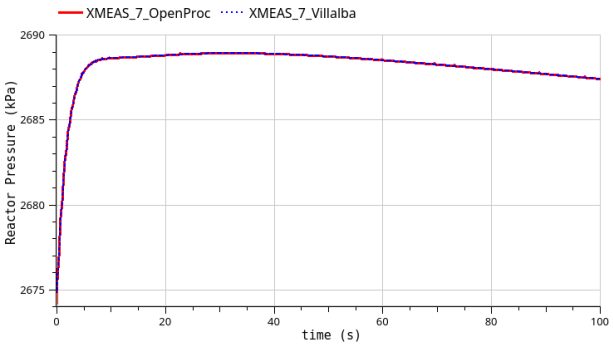

Figure 6: Open-loop dynamic response of the reactor pressure (XMEAS 7). The modular formulation reproduces the same slow-scale behaviour and instability trends reported in the reference literature, confirming that plant-wide feedback and recycle effects are preserved.

In addition to the comparison with literature trajectories, a direct internal comparison was performed between the modular flowsheet-driven model and a monolithic implementation obtained by transcribing the same constitutive equations into a single, hard-coded system. The two formulations followed identical solver trajectories, with the same number of solver steps and Jacobian evaluations, and produced indistinguishable state evolutions over the entire simulation horizon.
The modular formulation exhibited a moderate increase in CPU time. Over-simplifying the discussion to avoid toolchain-related details that are inessential to this paper, this overhead can be attributed to how modular, connector-rich code is currently expanded and processed at the level of individual solver steps. The fact that the number of solver steps and Jacobian evaluations remains unchanged indicates that the numerical problem itself is identical, and that the observed overhead is structural rather than physical. This observation suggests clear opportunities for improved exploitation of sparsity and structural information in future toolchain developments.

6. Conclusions and future work

This work addressed a structural mismatch between reaction-driven, species-based representations naturally adopted in chemical engineering and the port-based abstractions commonly used in equation-based object-oriented modeling. Rather than pursuing full OO purity at the cost of accessibility, we adopted a deliberately flowsheet-driven stance in which object-oriented mechanisms act as enabling technology. The Tennessee Eastman case study shows that plant-wide, reaction-dominated dynamics can be represented within an equation-based framework without altering the underlying process equations and without sacrificing structural transparency. The resulting models remain readable at flowsheet level, localize reaction-driven complexity within unit operations, and are directly usable for system-level engineering tasks, including control-oriented studies.
From a computational perspective, the comparison with a monolithic implementation indicates that the observed overhead is structural rather than physical: solver trajectories and Jacobian evaluations remain unchanged, while moderate runtime differences arise from the current handling of modular, connector-rich formulations. This suggests that further performance gains are primarily a toolchain issue rather than a modeling one, and that improved exploitation of sparsity and structural information represents a promising direction for future developments. In parallel, future work will explore the scalability of alternative modeling stances, including more strictly OO formulations, in order to better understand the trade-offs between abstraction purity, usability, and computational efficiency at plant scale.

References

[1] Aspen Technology, Inc. Aspen hysys simulation basis, 2023. Process simulation software.

[2] AVEVA Group. AVEVA PRO/II Simulation, 2023. Process Engineering Software.

[3] J.J. Downs and E.F. Vogel. A plant-wide industrial process control problem. *Computers & Chemical Engineering*, 17(3):245–255, 1993. Industrial challenge problems in process control.

[4] Marek Mateják. Chemical 2.0: Free open-source Modelica library. In *Proceedings of the 16th International Modelica & FMI Conference*, pages 55–60, Lucerne, Switzerland, September 2025.

[5] Modelica Association. *Modelica® - A Unified Object-Oriented Language for Physical Systems Modeling, Language Specification Version 3.5*. Modelica Association, 2021.

[6] Priyam Nayak, Pravin Dalve, Rahul Anandi Sai, Rahul Jain, Kannan M. Moudgalya, P. R. Naren, Peter Fritzson, and Daniel Wagner. Chemical process simulation using openmodelica. *Industrial & Engineering Chemistry Research*, 58(26):11164–11174, 2019.

[7] Siemens Process Systems Engineering. gPROMS Platform, 2023. Advanced Process Modeling Environment.