Material Production Process Modeling with Automated Modelica Models from IBM Rational Rhapsody

John Batteh[1] Jesse Gohl[2] James Ferri[3] Quang Le[4]
Bill Glandorf[5] Bob Sherman[6] Rudolfs Opmanis[7]

[1,2]Modelon Inc., USA, {john.batteh, jesse.gohl}@modelon.com
[3,4]Virginia Commonwealth University, {jkferri, leq2}@vcu.edu
[5]Procter and Gamble, USA, glandorf.wm@pg.com
[6]Occam Systems Inc., USA, bob@occamsystemsinc.com
[7]Tom Sawyer Software, USA, rudolfs@tomsawyer.com

Abstract

This paper describes a method to author dynamic simulation models in Modelica from a manufacturing architectural model structure in SysML. Modelica models are generated from IBM Rational Rhapsody and simulated using Modelon Impact. Following a brief overview of the overall modeling approach and tool coupling, the Modelica modeling is detailed for computing process throughput and processing time. Following the model overview, a sample application for the production of the pharmaceutical ingredient atropine is demonstrated.

Keywords: SysML, Rhapsody, process modeling, pharmaceuticals, manufacturing

1 Introduction

The increasing complexity of modern manufacturing systems has created a paradigm shift from the traditional document-centric approach in systems engineering to Model Based System Engineering (MBSE). MBSE provides stakeholders access to an authoritative thread of information describing the system design throughout the product lifecycle. Complementary to MBSE, Multidisciplinary Design Analysis and Optimization (MDAO) focuses on creating an analysis model to demonstrate properties or outcome of the intended system by leveraging high performance computing and dynamic simulations. However, there is little effort in bridging the gap between the two approaches. We propose a method to author dynamic simulation models from a structural model. Specifically, we are developing infrastructure to automatically generate simulations in the Modelica language from a manufacturing architectural model structure in SysML (SysML 2022). This digital coupling allows for the unified design vision be developed with analytical rigor throughout the product lifecycle.

This work is part of a U.S. Defense Advanced Research Projects Agency (DARPA) project to build a digital infrastructure that the chemicals and materials industry needs to improve efficiencies. The goal of this project is to make it easier for chemical manufacturers in the U.S. to produce critically needed medicines. The models and workflow described in this paper allow manufacturers to explore production duration, synchronization, requirements, and constraints to improve and optimize chemical synthesis processes. Addressing challenges in the chemical industry to encourage domestic production of these medications will also pave the way for manufacturing a broad range of active pharmaceutical ingredients (APIs) for drugs in the U.S., instead of overseas. These include in-demand APIs related to COVID-19 and several others on the federal government's Strategic National Stockpile list.

This paper focuses on just the modeling and simulation portion of the project. Following a brief overview of the overall approach and tool coupling to generate Modelica models from SysML using IBM Rational Rhapsody (IBM 2022), the Modelica modeling is detailed for computing process throughput and processing time. An overview of the modeling approach and key components is provided. Following the model overview, a sample application for the production of atropine is demonstrated with simulations in Modelon Impact (Modelon 2022).

2 Modeling Process Overview

This section provides a high level overview of the modeling process to generate an executable Modelica model starting from an MBSE system model in IBM Rational Rhapsody. Figure 1 illustrates key components of the modeling process and the various software tools involved. The overall steps in the modeling process are as follows for the drug manufacturing use case:

- Based on research regarding the process to create a particular pharmaceutical ingredient, a SysML model is created in IBM Rational Rhapsody that outlines the manufacturing process steps.

 10.3384/ECP21186158 DOI

- Based on the SysML model and an equipment database for chemical plants, a fully parameterized Modelica model is automatically created using Tom Sawyer Software technology for the manufacturing process based on a Modelica model library for the various process primitives
- The resulting Modelica model is then simulated in Modelon Impact using its simulator API
- Key simulation results, including process throughput and process time, are then returned to Tom Sawyer Software for visualization and also for use in an optimization process regarding the manufacturing process and equipment allocation

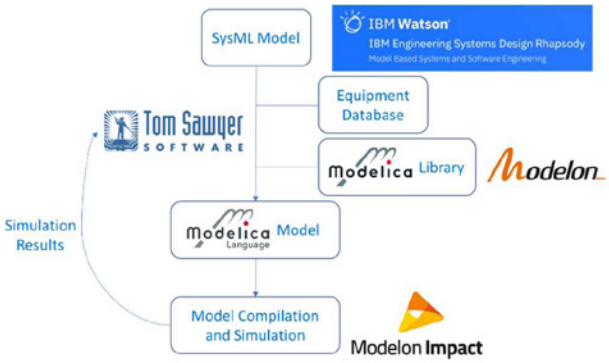


Figure 1. The modeling process overview.

This modeling process will be outlined in detail in separate future publications.

3 Modelica Models

As outlined in Section 2, the intended studies supported by this work are focused on the process layout for analyzing and optimizing the process time and yield for batch based material processing. The degrees of freedom for such an optimization are not only individual steps' boundary conditions (parameters) but also the overall process topology. Since the process steps required for manufacturing a given pharmaceutical ingredient are not wholly documented in public references and since the models are expected to be used in massive optimization loops, the models are intentionally implemented to be computationally fast due to low fidelity implementations and the use of time events.

The distinct stages of the process are simulated as individual components in the Modelica model. These stages compute their status (complete or not) and the batch material characteristics based on the behavior of the specific stage. The material characteristics and status feed successive stages. This data that is shared between components are described in Section 3.2. Details about specific implementations are provided in Section 3.3. Finally, examples of a fully assembled process are given in section 3.4.

3.1 Material Data Record

The properties of individual species are maintained within a data record that can be customized to each process. The records contain an array of the species names which are necessary for the process along with arrays of the species properties. The material data record approach is used in lieu of the more rigorous and detailed medium model approach to provide material information and thermophysical properties due to the lack of rigorous species information at various stages of the manufacturing process.

```
record Base
  extends Modelica.Icons.Record;
  final parameter Integer ns=size(species_names,1) "number of species";
  parameter String species_names[:] "species names";
  parameter Modelica.SIunits.MolarMass MW[ns] "Molecular weights (kg/mol)";
  parameter Modelica.SIunits.Density rho[ns] "density";
  parameter Modelica.SIunits.ThermalConductivity lambda[ns] "thermal conductivity";
  parameter Modelica.SIunits.DynamicViscosity mu[ns] "absolute viscosity";
  parameter Modelica.SIunits.SpecificHeatCapacity cp[ns] "specific heat";
  parameter Modelica.SIunits.DiffusionCoefficient D[ns] "diffusion coefficient";
  parameter Modelica.SIunits.SpecificEnthalpy Hv[ns] "latent heat of vaporization";
  parameter Modelica.SIunits.Temperature Tv[ns] "vaporization temperature at standard conditions";
  parameter ASKPSP.Types.MatterState state[ns] "state of matter";

  annotation (...);
end Base;
```

Figure 2. The material data record.

As an example, the process simulated in Section 3.4, uses the species listed in Table 1.

Table 1. An example process data record species list.

Index	Species
1	DCM
2	Tropine
3	Methanesulfonic acid
4	DCM+Tropine
5	DCM+Methanesulfonic acid
6	DCM+Tropine+Methanesulfonic acid
7	DCM+Tropine methanesulfonate

This record is instantiated as an `inner` component at the top level of any simulation model for access by model components via an `outer` relationship.

3.2 Component Interfaces

Material Interface

The individual "steps" in the production process implies an explicit ordering to the stages. This approach allows the interface between Modelica model components to use a causal interface with distinct inputs and outputs that define the material state.

```
connector ProcessInput "Process stage input material characteristics"
  input Integer species_index "Material species identifier index";
  input Modelica.Units.SI.Mass mass "Material mass";
  input Modelica.Units.SI.Temperature T "Material temperature";

  annotation (•••);
end ProcessInput;
```

```
connector ProcessOutput "Process stage output material characteristics"
  output Integer species_index "Material species identifier index";
  output Modelica.Units.SI.Mass mass "Material mass";
  output Modelica.Units.SI.Temperature T "Material temperature";
  annotation (•••);
end ProcessOutput;
```

Figure 3. A process step's material data input and output.

These connectors represent the material type, quantity, and state that is shared between process stages. Because all of the steps represent processes that operate on batches of material (e.g. heating, stirring), these connectors are instantiated in interface base classes.

The first of these is `ComponentSingle` that supports components that operate on a single batch of material. In this case a single input and output is needed.

```
partial model ComponentSingle
    "interface for a single material input component"
  extends MaterialData;
  ASKPSP.Interfaces.ProcessInput pIn annotation (•••);
  ASKPSP.Interfaces.ProcessOutput pOut annotation (•••);

  annotation (•••);
end ComponentSingle;
```

Figure 4. The single component base class.

To support processing steps that operate on multiple batches of material (e.g. mixing and reactions), the `ComponentArray` base class is used.

```
partial model ComponentArray
  "interface for a triggered component with an array input"
  extends MaterialData;
  parameter Integer nu2(min=0) = 0 "Number of input connections"
     annotation (•••);
  ASKPSP.Interfaces.ProcessInput pIn[nu2] annotation (•••);
  ASKPSP.Interfaces.ProcessOutput pOut annotation (•••);

  annotation (•••);
end ComponentArray;
```

Figure 5. The array component base class.

Notice that the dynamically incremented connector size parameter is named `nu2`. This naming is to avoid conflicts with the connector size parameter for the trigger inputs, discussed in the next section. Components that extend from these base classes are required to include at least three equations to balance the three variables mass, temperature (`T`), and species_index on the output connector `pOut`. The ancestor base class for both of these base classes is the `MaterialData` model that instantiates an `outer` instance of the `process_data` record described in Section 3.1.

```
partial model MaterialData
  "interface for a material handling component"
  outer parameter ASKPSP.ProcessData.Base process_data;

  annotation (•••);
end MaterialData;
```

Figure 6. The material data base class.

Logical Interface

In addition to the material data, stages also share their status with adjacent steps. This Boolean status is the local variable `complete` and is an output from all stages that serves as the trigger for successive stages to start their process. Because a stage could depend on previous steps (for example when process steps occur simultaneously), the input `trigger` is a Boolean array. Processing for a stage begins when all trigger inputs are true. Thus, all components include a scalar Boolean output named `complete` and an array Boolean input named `trigger`. These Boolean inputs and outputs are collected in a base class named `TriggeredComponent` that includes the logic to consider all input triggers to start the process step. The diagram and text of this base class appears in the following figure.

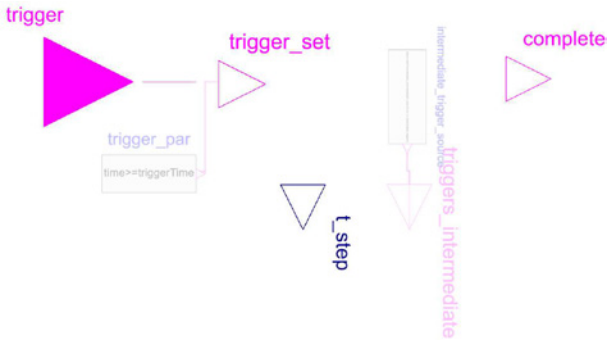


Figure 7. The triggered component base class diagram.

The base class includes the parameter option `paraOption_trigger` which allows the same component to be used either with external trigger connectors or from a parameter for absolute trigger time, `triggerTime`. The `triggered` internal Boolean variable becomes true when all triggers are true. This also starts the internal timer by setting the discrete variable `startTime` to the time instant of the trigger.

Fixed Time

The required duration for some stages can be specified directly. For example, stirring stages are specified as "stir the material for x seconds". These stages extend from the fixed `TimeComponent` base class which includes the process time parameter `tau`. Notice the state is complete when the simulation time exceeds the process time after that stage starts.

10.3384/ECP21186158 DOI

```
partial model TimeComponent "base class for a time based component"
  extends TriggeredComponent;
  parameter Modelica.Units.SI.Time tau "time for process";

equation
  t_step = tau;
  complete = time >= startTime + tau;
end TimeComponent;
```

Figure 8. The fixed time component base class.

Prescribed Rate

Other stages operate at prescribed rates. The processing time required for these stages is computed based on a rate equation using the ratio of the total quantity to be processed and the processing rate. For example, the duration of mass transfer is computed as the total mass to be transferred divided by the rate of transfer. For these stages the `RateComponent` base class is used.

```
partial model RateComponent "base class for a rate based component"
  extends TriggeredComponent;
  Modelica.Units.SI.Time tau "time for process";

equation
  t_step = tau;
  complete = time >= startTime + tau;
end RateComponent;
```

Figure 9. The rate component base class.

3.3 Process Stages

Elemental Stages

The following components represent fundamental stages used to describe an overall material handling process. These elemental components are responsible for computing the time required to complete their stage based on the established equations that describe the process physics. In addition, each stage is responsible for computing the exit state mass, temperature, and species index. Some stages can also produce a change of species (e.g. reactions, mixing, and drying). Due to the limited information available regarding each process step and the species mix, mass conservation is handled based on the incoming mass but also based on a yield input or parameter to allow the output mass to be calculated.

Time and Hold

The `Time` and `Hold` components represent simple timed stages that require a fixed time delay. Both of these components extend from the `TimeComponent` base class and add equations for the output connector's contents. These components behave as pure delays with no additional modification to the material. The same material species leaves that entered, no mass is stored in the component, and no heat is transferred. When the stage is complete, the mass and the temperature at the outlet become the same as the inlet.

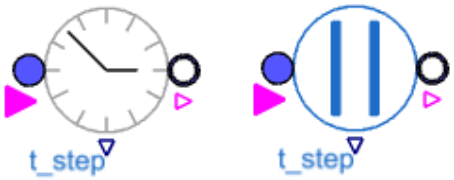


Figure 10. Time and Hold components' icons.

Parameter Rate

The `FixedRate` component describes the time required for handling mass at a fixed rate. It adds a parameter for the fixed mass flow rate and species the outlet connector similar to the `Time` and `Hold` components discussed above. When the stage starts (`triggered` becomes true), the processing time is computed based on the incoming mass divided by the fixed mass flow rate from the parameter.

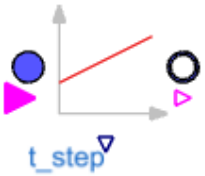


Figure 11. The FixedRate component icon.

Stir

The `Stir` component is another trivial component used to simulate the duration required to stir the material for a fixed amount of time. It specifies the outlet state in the same way as the components above.

Figure 12. The Stir component icon.

Mix

The `Mix` component extends from the `ComponentArray` base class. It is used to simulate stages where multiple materials are combined over a fixed amount of time. The new, outgoing species is specified by a parameter. The outgoing mass is conserved as the sum of incoming masses and the temperature of the outlet material is the mass-average of the incoming materials (note: not a rigorous conservation of energy due to lack of detailed information about the process species at each stage).

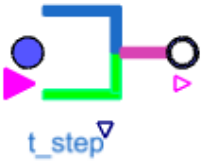


Figure 13. The mix component icon.

Reaction

The `Reaction` component is similar to the `Mix` component because it combines multiple materials to produce a new outgoing species. It also extends from the `ComponentArray` base class in order to support multiple incoming materials. In addition to the `species_index_out` parameter, there are parameters for the reaction time (`tau_reaction`), yield fraction, and reaction temperature change (`dT`). The yield fraction reduces the outgoing mass to account for losses and incomplete reactions. The temperature difference applies an additional offset to the outgoing temperature from the mass averaged inlet temperatures.

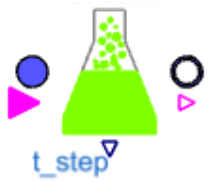


Figure 14. The Reaction component icon.

Volumetric Rate

The `VolumetricRate` component simulates a process that applies a fixed volumetric flow rate to the incoming material. This component also demonstrates the use of the "process_data" record. The density of the incoming material, `rho`, is selected based on the incoming species index. This density is then used together with the incoming mass, to compute the volume of the material that must be transferred by this stage. When the stage starts (`triggered` becomes true), the process time is updated based on the ratio of the material volume, `Vf`, with the fixed volumetric flow rate, `Q`.

Figure 15. The Volumetric rate component icon.

Liquid Transfer

The `LiquidTransfer` component is similar to the `VolumetricRate` component except the volumetric flow rate is computed assuming hydraulic flow through a smooth pipe driven by a constant power pump. The Fanning friction factor (Incropera 2007) relates the pressure drop to the mean fluid velocity and the Reynolds number characteristic of the flow (Munson 2009).

$$Re = \frac{4\rho Q}{\pi \mu D} \quad (1)$$

$$\Delta p = \frac{2\rho f L v^2}{D} \quad (2)$$

$$f = 0.0791 \cdot Re^{-1/4} \quad (3)$$

$$Q = v \cdot A \quad (4)$$

$$P_b = Q \cdot \Delta p \quad (5)$$

For Re – Reynolds number, ρ – fluid density, Q – volumetric flow rate, μ – dynamic viscosity, D – pipe diameter, Δp – pressure drop, f – Fanning friction factor, v – mean fluid velocity, A – pipe cross-sectional area, Pb – pump power

The unknown variables in these five equations are Re, Δp, Q, f, and v, which can be solved from these five equations. The volume of the fluid, Vf, is directly computed from the incoming mass and density. With the solution of the volumetric flow rate, Q, the duration can be directly computed.

Dissolution

The `Dissolution` component computes the duration required to fully dissolve a specified mass of solute in a stirred, fixed volume vat. The solute can be specified either as a fixed parameter, or it can enter from a conditional connector. The time for dissolution is calculated based on mass transfer analysis for a stirred vessel.

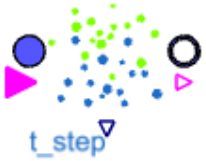


Figure 16. The Dissolution component icon.

Heat Transfer

As with the `Dissolution` component, the `HeatTransfer` component assumes a liquid in a fixed volume, stirred vessel. The formulation starts with Newton's law of cooling and the First Law of Thermodynamics to relate the heat capacity, heat transfer rate, and the material temperature for a constant pressure process (Castellan 1983).

$$\Delta H = Q = C_p \Delta T = C_p (T - T_0) \quad (5)$$

$$\dot{Q} = C_p \dot{T} = hA(T_{ht} - T) \quad (6)$$

T – material temperature, C_p – heat capacity at constant pressure, h – heat transfer coefficient, A – heat transfer area, T_{ht} – temperature of the heat transfer element. From these equations, the solution to the initial value problem for $T(0) = T_0$ is:

$$T(t) = T_{ht} + (T_0 - T_{ht}) e^{-t\frac{hA}{C_p}} \quad (7)$$

The final time, t_f, can be calculated from this equation to heat (or cool) the material to the final temperature, T_f.

$$t_f = -\tau \log\left(\frac{T_{ht} - T_f}{T_{ht} - T_0}\right) \quad (8)$$

 10.3384/ECP21186158 DOI

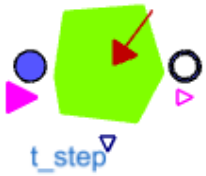


Figure 17. The heat transfer component icon.

Drying Fixed Rate

The `DryingFixedRate` component accounts for two physical phenomena. First is the addition of heat to evaporate a fraction of the incoming material stream. Second is the optional change of species to the outlet stream. This change depends on whether the evaporate or the dessicate (remainder) continues to the outlet or is discarded. This implementation allows the same component to be used for both a drying operation as well as a distillation step. The component assumes heat is added at a fixed rate defined by a parameter `Qmax`. The process time accounts for the duration to add sensible heat to raise the temperature of both the evaporate and dessicate and to add the latent heat of the evaporate. The fraction of the incoming stream that evaporates is specified by a parameter. From this information, the masses of both the evaporate and dessicate are known. From these masses and the heat capacity at constant pressure of each species, the total energy required for the sensible heat necessary to raise the temperature of both species from the incoming temperature to the vaporization temperature of the evaporate species can be directly computed. The total heat required also includes the latent heat of vaporization. This heat is directly computed from the evaporate mass and the evaporate species specific latent heat of vaporization. The duration of this step is then directly computed from the fixed rate of heating, `Qmax`.

This component includes an optional Boolean flag that allows the user to select whether the process is a drying or distillation step. This selection determines which species continues to the outlet stream. An assumption of this component is that the species index of the incoming stream is the evaporate species.

Figure 18. The Drying component icon.

Filter/Wash Base Class

Both the `Filter` and `Wash` component use Darcy's law to model the flow rate of a solvent through a permeable solid. This law relates the volumetric flow rate of a liquid to the hydraulic permeability, cross sectional area, dynamic viscosity, flow length, and pressure difference.

$$\boldsymbol{Q = \frac{\Delta V}{\Delta t} = \frac{kA}{\mu L}\Delta p} \tag{9}$$

Q – volumetric flow rate, ΔV – volume, Δt – duration, k – hydraulic permeability, A – cross sectional area of the filter path, μ – dynamic viscosity, L – length of the filter path, Δp – pressure difference

For gravity driven filtrations, this equation can be rearranged to use hydraulic conductivity, fluid density, and gravity in place of the permeability and dynamic viscosity.

$$\boldsymbol{\frac{K}{\rho g} = \frac{k}{\mu}} \tag{10}$$

K – hydraulic conductivity, ρ – density, g – gravitational constant

These equations are implemented in the `FilterWash` base class model to calculate the filtration time.

$$\Delta t = \frac{\Delta V \rho g}{A \Delta p} \cdot \left(\frac{L_f}{K_f} + \frac{L_c}{K_c} \right) \tag{11}$$

L_f – filter length, K_f – filter conductivity, L_c – cake path length, K_c – cake conductivity

In this case, L_f/K_f is assumed to be a constant and is specified as a parameter. L_c is computed from the solid volume, packing efficiency, and cross sectional area. The outlet mass is the solute mass computed from a fixed yield fraction of the filtration.

The `Filter` component extends from the `FilterWash` base class and assumes the incoming material already contains the solvent and solute. The solvent mass is based on the yield fraction of the filtration and is used to compute the volume. The filtration is also assumed to be isothermal, so the outlet temperature is the same as the inlet.

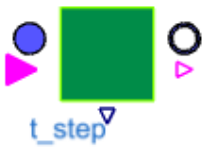


Figure 19. The Filter component icon.

Like the `Filter` component, the `Wash` component also extends from the `FilterWash` base class. In this case a new connector is added for a separate inlet for the solvent. The solvent volume is explicitly computed from the solvent mass and density. The outlet temperature for the `Wash` component is specified by a parameter. An additional condition for completion for this stage is that all the solvent must be delivered. Because prior stages hold their outlet mass at zero until they are complete, the

`Wash` stage uses the inlet solvent mass greater than epsilon to indicate completion. This implementation prevents the stage from completing until all the solvent mass has been transferred.

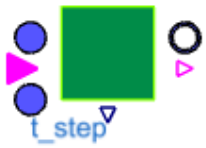


Figure 20. The Wash component icon.

Phase Cut

The `PhaseCut` stage represents a process separation step that splits the material on the input connector into two separate outputs based on a fixed fraction (`usable_fraction`). An additional `yield_fraction` parameter is defined to account for losses that occur during the separation process.

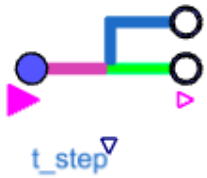


Figure 21. The PhaseCut component icon.

End

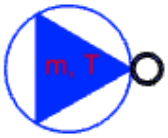
The `End` component performs two functions. First it records the overall completion time in the discrete variable `t_final`. This value is updated when the `End` component is triggered. The second function the component performs is to terminate the simulation. The additional condition for termination guarantees the simulation does not terminate before the solver has updated all outputs (e.g. `t_final`). Notice the final product species, mass, and temperature is available on the input connector.


```
model EndStep "end step for process"
  extends ASKPSP.Processes.Interfaces.TriggeredComponent(
    final paraOption_trigger=true);

  parameter Integer nu2(min=0) = 0 "Number of input connections"
    annotation (•••);
  parameter Modelica.Units.SI.Time t_eps=1e-6
    "Epsilon time to terminate the simulation after completion"
    annotation (•••);

  Modelica.Blocks.Logical.TerminateSimulation terminateSimulation(
    condition=complete and time >= startTime + t_eps)
    annotation (•••);
  Modelica.Blocks.Interfaces.RealOutput t_final(start=0,fixed=true)
    "final processing time"
    annotation (•••);
  ASKPSP.Interfaces.ProcessInput pIn annotation (•••);

equation
  complete = triggered;
  t_step=0;
  when triggered then
    t_final = time;
  end when;

  annotation (•••);
end EndStep;
```

Figure 22. The End component icon and implementation.

Source

The `Source` component provides a fixed amount of material (mass), at a fixed temperature, as a specific species. It does not include an output `complete` signal because it is assumed to be a source of material. If the delivery time must be considered at the start of a process, one of the transfer components can be placed immediately after the source.




```
model Source_mT
  parameter Modelica.Units.SI.Mass mass "mass";
  parameter Modelica.Units.SI.Temperature T "temperature";
  parameter Integer species_index "species index";
  Interfaces.ProcessOutput pOut annotation (•••);
equation
  pOut.T=T;
  pOut.mass=mass;
  pOut.species_index=species_index;
  annotation (•••);
end Source_mT;
```

Figure 23. The Source component icon and implementation.

Composite Stages

In many cases during material handling, multiple stages are often associated with each other. For this reason, it is convenient to create composite stages that combine these steps into a single component. This approach to create composite stages reduces the effort required for the automated code generation process.

Heat Transfer and Dissolution

In many cases of dissolution during material processing, the instructions specify simultaneous heating during the dissolution. The elemental `Dissolution` component however assumes an isothermal process (the outlet material maintains the same temperature as the inlet). For this reason, a composite component was created that includes both these steps in parallel. This component

starts with a mixing stage that combines the solute with the inlet material. After this stage is complete the heating and dissolution stages can start. The overall stage cannot complete until both stages are done.

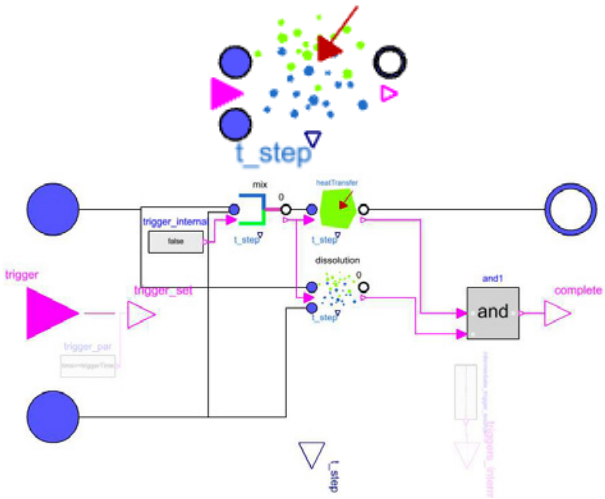


Figure 24. The Heat transfer and Dissolution component.

Add Liquid and Wash

Because washing steps require the addition of a liquid, the `AddLiquidAndWash` convenience component combines both the `Wash` component with the `VolumetricRate` liquid transfer component. Because washing can commence immediately, no mixing component is necessary. The volumetric transfer component only needs to generate its completion signal to complete the composite step.

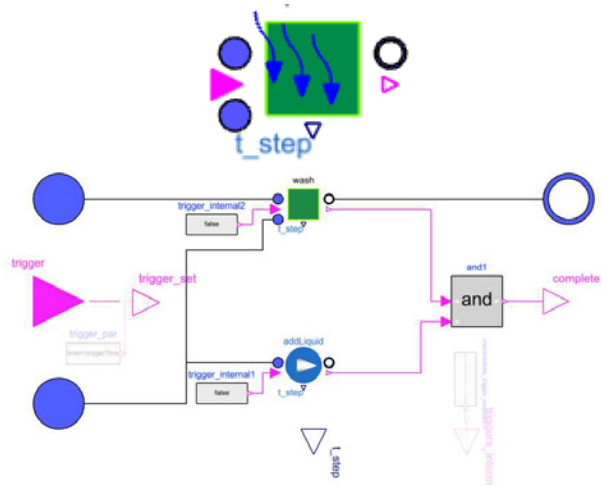


Figure 25. The AddLiquidAndWash component system.

3.4 Complete System Configuration

To demonstrate the usage of these components, the following model was created. This model is used to simulate the synthesis of tropine methanesulfonate, which is a precursor in the synthesis of atropine. The process starts by mixing species dichloromethane (DCM) to tropine as indicated by the species indices, 1 and 2 as listed in Table 1, as shown on the `source_DCM` and `source_Trop` components. The `liquidTransfer` and `solidTransfer` components compute the duration required to transfer the supplied materials to the mixing vessel (`mix`).

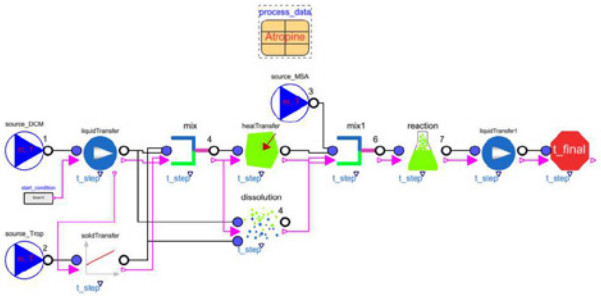


Figure 26. The Tropine methanesulfonate synthesis system.

Notice the `mix` component gets its material from `liquidTransfer` and `solidTransfer` and starts its process when both these steps are complete. Because it is based on the `TimeComponent` base class, it also includes a process time, tau for the additional duration after it starts. Finally the mix component also changes the species index to 4, which is the identifier for the mixture of DCM and Tropine (see Table 1).

After mixing, the mixture is heated and the tropine is dissolved in solution. This process is simulated by the `heatTransfer` and `dissolution` components. Notice the DCM liquid transfer is connected to the main material inlet of the dissolution component. The tropine material outlet connector is connected to the `dissolution` component solute material inlet. This approach allows the dissolution step to be computed for cases when the materials are not already mixed.

Alternatively the dissolution component also includes the parameter option to premix the solvent and solute, in which case the solute connector is disabled and the solute species can be specified via parameter. In this case the materials are already mixed so the dissolution component only needs to indicate when dissolution occurs. This demonstrates the flexibility of the models to be connected in multiple ways to accommodate different processes. This version of the library does not account for thermal effects of dissolution so there is no feedback from the heating step to the dissolution step. In this case they occur in parallel and are connected to the next `mix1` component.

The `mix1` component accounts for the time to complete both the heating, dissolution, and to add the third component, methanesulfonic acid, that is the reagent for the reaction. The outlet is the new species 6, DCM, tropine, and methanesulfonic acid.

The `reaction` component is another fixed time component that accounts for the reaction time based on a specified duration as a parameter. The outlet of this component is the final species 7, DCM and tropine methanesulfonate.

The final step in this sub-process is to transfer the liquid from the reaction vessel. This step is simulated with an additional liquid transfer component, `liquidTransfer1`. When this step is complete, it triggers the end step which records the overall duration and terminates the simulation. The progression of completion the stages can be tracked throughout the process by observing the `complete` local Boolean variable for individual stages as shown in the following figure.

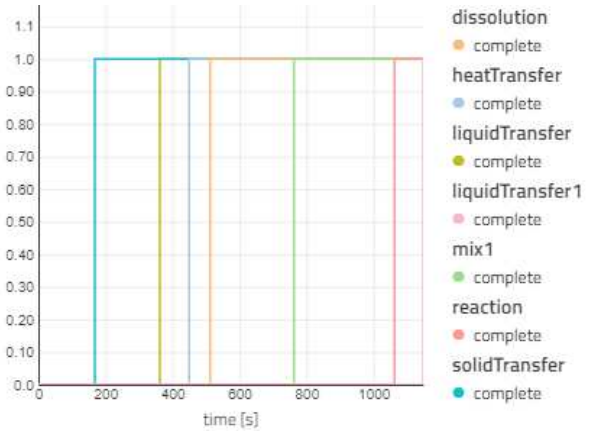


Figure 27. Atropine precursor process, completion progression.

The final results are shown in the following image. The overall process time (`t_final`), the mass, temperature (`T`), and final species index are visualized.

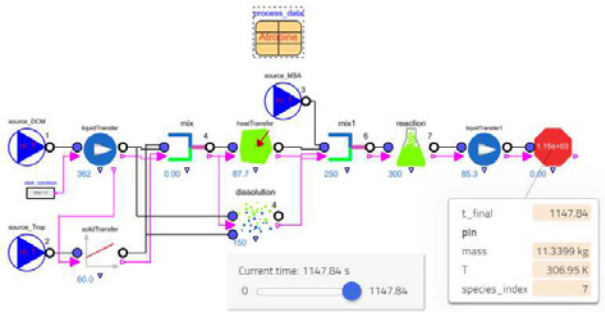


Figure 28. Precursor final results.

A larger process system is shown in the following image. This system simulates the final stages in the synthesis of atropine starting from the precursor created by the previous process. The stages are numbered according to the identifiers used in the original synthesis instructions. Again, the final results are indicated in the diagram and the progression of individual stages completion can be tracked by plotting the time trajectory for the Boolean variable, `complete`, shown in Figure 30.

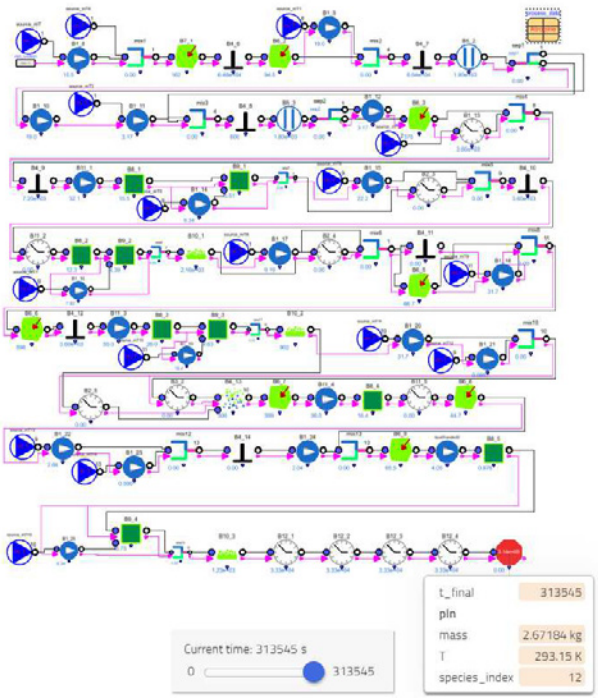


Figure 29. Atropine synthesis system model, and final results.

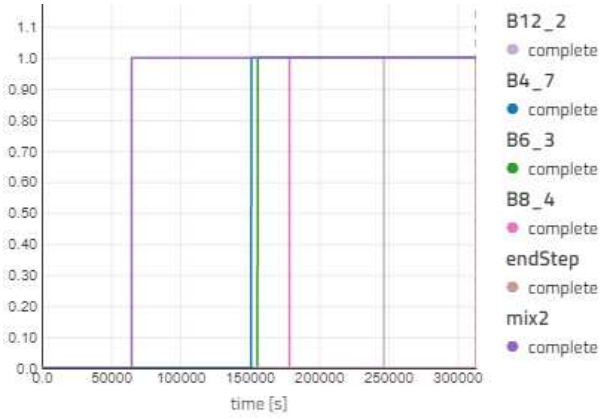


Figure 30. Atropine synthesis, completion progression.

4 Conclusions

This paper focuses on the modeling and simulation capability for the DARPA project to build a digital infrastructure to improve efficiencies in the chemicals and materials industry. Based on the work in the DARPA project to build SysML models that describe the manufacturing process for given pharmaceutical agents, simulation models for process throughput and yield are automatically generated in Modelica using the model library detailed in the paper. The models are simulated in Modelon Impact using its simulator API and key results returned to the software stack for use in visualization and

optimization processes. A sample use case for the synthesis of atropine is demonstrated.

This paper focuses on the model library developed to support this digitalization approach and an initial use case. Ultimately this project will support efforts to analyze the chemical supply chain and identify capability gaps in pharmaceutical manufacturing for key drugs.

Acknowledgements

This work has been supported by U.S. Defense Advanced Research Projects Agency (DARPA).

References

Castellan, Gilbert W. (1983). *Physical Chemistry*. 3rd ed. The Benjamin/Cummings Publishing Co., Inc. ISBN: 0-201-10386-9.

IBM (2022). *IBM Engineering Systems Design Rhapsody*. URL: https://www.ibm.com/products/systems-design-rhapsody

Incropera, Frank P., David P. Dewitt, Theodore L. Bergman, and Adrienne S. Lavine (2007). *Fundamentals of Heat and Mass Transfer*. 6th ed. John Wiley & Sons, Inc. ISBN: 978-0-471-45728-2.

Modelon (2022). *Impact*. URL: https://www.modelon.com/modelon-impact/.

Munson, Bruce R., Donald F. Young, Theodore H. Okiishi, and Wade W. Huebsch (2009). *Fundamentals of Fluid Mechanics*. 6th ed. John Wiley & Sons, Inc. ISBN: 978-0470-26284-9.

SysML (2022). *SysML Open Source Project - What is SysML? Who created SysML?* URL: https://sysml.org/