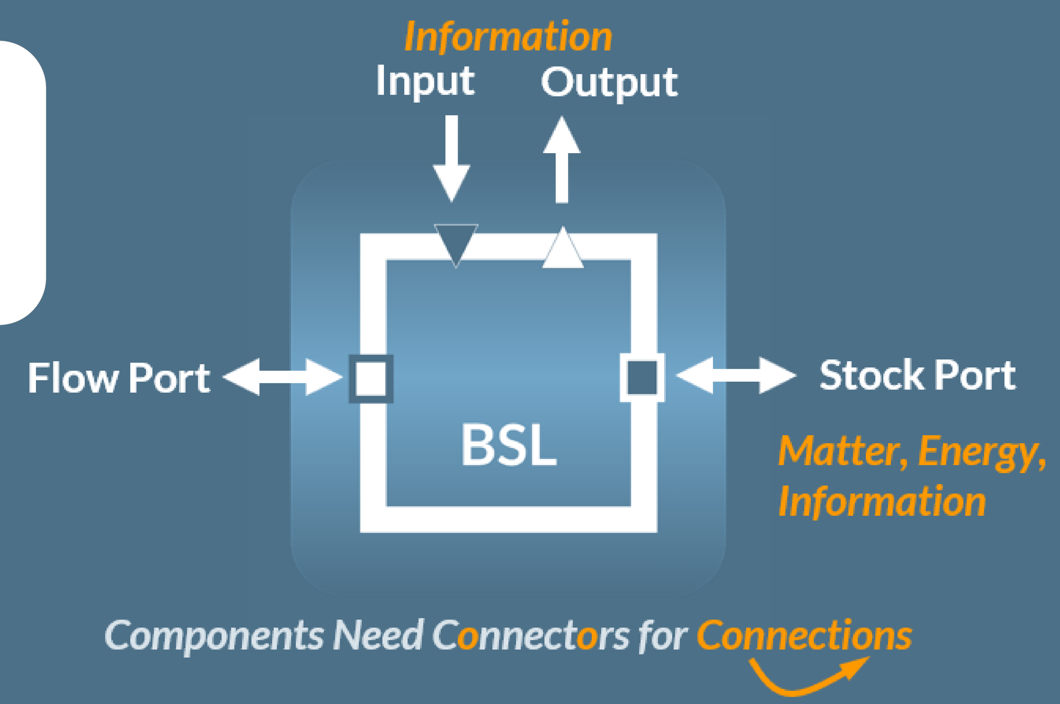
“System dynamics modeling software packages are ‘modularity unfriendly’.”

Elmasry and Größler (2018)

Hierarchical, Component-Based Modeling Using the Cyber-Physical Modeling Language Modelica

Guido Wolf Reichert ⟨gwr@bsl-support.de⟩



Hierarchical Modeling Absorbs Structural Complexity

A complex “system of systems” might best be modeled by a “model of models.” Pre-built models in Modelica are called *components*. In an object-oriented modeling approach components are connected to each other via interfaces called *connectors* (Figure 1).

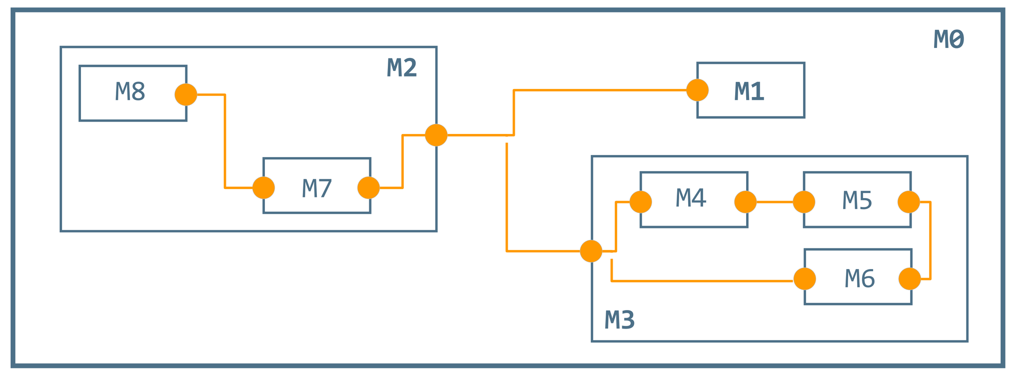


Figure 1. Schematic diagram of nested model structure

In a nested model, there are *connections* between higher-level *outside* and lower-level *inside* connectors (e.g., M2.c and M7.c) as well as connections between inside connectors of components at the same level (e.g., M8.c and M7.c).

In System Dynamics Four Different Connectors Are Needed

Looking at an elementary stock and flow structure (Figure 2) we may realize that next to *information input* and *output connectors*, we will also need *“physical” connectors* for the transport of matter, energy or conserved information (e.g., orders) between systems.

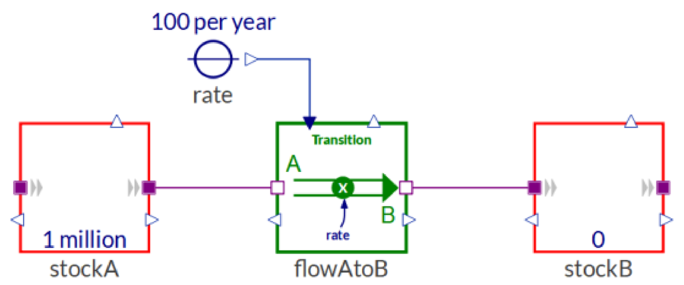


Figure 2. Basic stock and flow structure

In the Business Simulation Library (BSL) for Modelica there are *flow* and *stock ports*. Different from earlier approaches (Powers, 2011) flows are *not mere connections* but components with two flow port connectors (e.g., pull-push processes). Thus, we can split the structure and embed it within two separate subsystems (e.g., push supply chain) as shown in Figure 3.

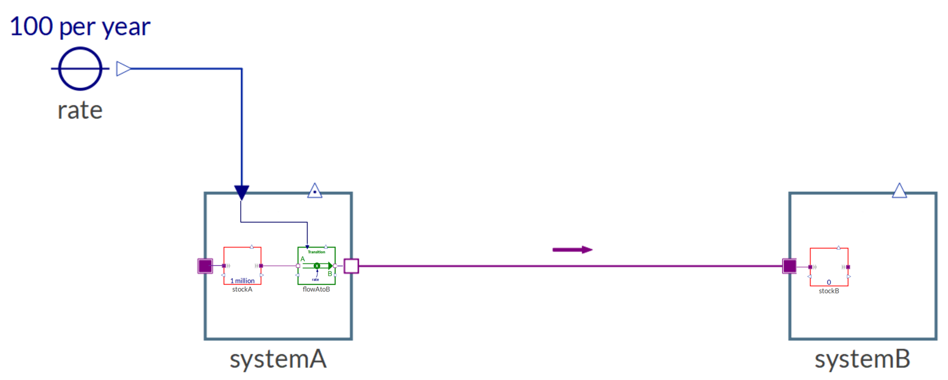


Figure 3. Basic stock and flow structure as hierarchical system

Everything Is a Circuit—Cyber-Physical Modeling in a Nutshell

The generalized flow of energy is proportional to the flow of substance-like *extensive quantities* (e.g., amount of substance, electric charge, momentum, entropy). Across energy domains differences in an *intensive quantity* (e.g., chemical potential, electric potential, velocity, absolute temperature) can be seen as causing the flow of the extensive quantity.

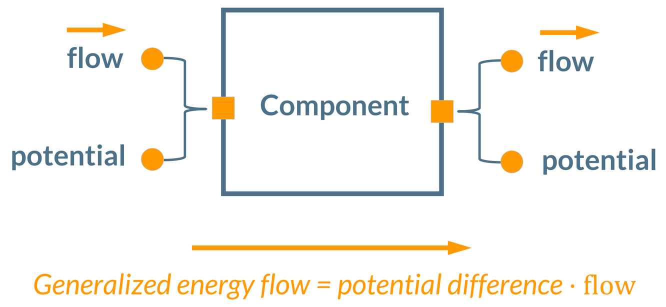


Figure 4. Quadripole representation of a lumped-component model

Henry M. Paynter (1961) made use of this in *bond graph* modeling and Modelica combines flow variables and potential variables in a single *acausal connector* (the squares in Figure 4). Most components can accordingly be understood as *quadripoles* from electrical engineering.

Stock and Flow Connections Are Governed by Kirchhoff’s Laws

If a component has *in* and *out* ports, then for any *connection set* the sum of flows must add up to zero (Kirchhoff’s First Law; Figure 5). In other words, what flows out must flow in. Modelica automatically adds algebraic equations to enforce this.

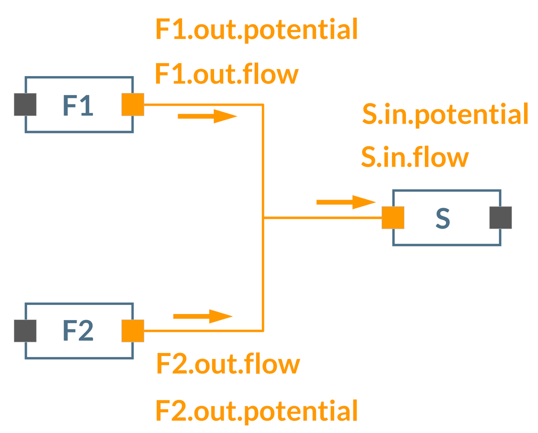


Figure 5. Potentials and flows in a connection set

The potential variable of acausal connectors can be used to transmit the *amount contained* within a stock in a stock and flow connection set; Kirchhoff’s Second Law in Modelica enforces that the potential is equal for any connector in a connection set. A basic population model thus just needs two connections for processes of *exponential growth and decline* (Figure 6).

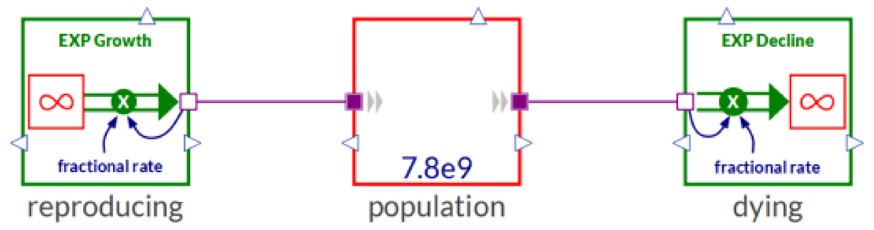
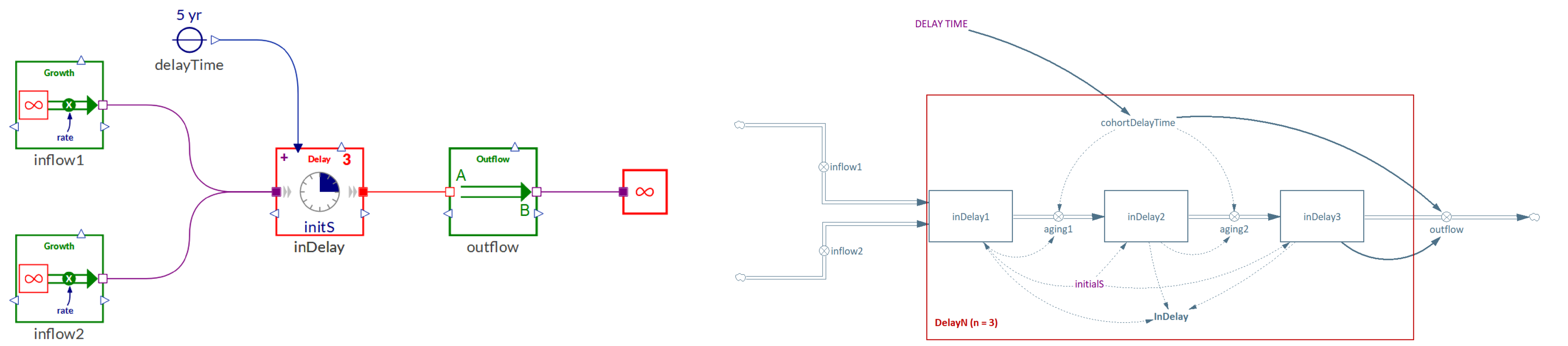


Figure 6. Component-based model of population dynamics

Dynamic Stocks Versus Flows With Hidden Stocks

In many system dynamics environments higher-order delays are modeled with stocks being hidden within a delayed flow. This leads to *redundant input of information* and is *error prone* as filling and draining the stock of material being delayed in a consistent fashion is everything but easy. There is *less clutter* in the Modelica diagram (Figure 7a) and a *higher data-ink ratio*, but most importantly the explicit delay structure (Figure 7b) is simply embedded within a *dynamic stock*, which sets the rate of the connected outflow via a *special stock port*. This implementation avoids redundancies and is less error prone. The special character of this structure is succinctly visualized by conspicuous ports (red) and the `Outflow` class's icon that is a mere *pipe*—lacking the typical paddle wheel pump of regular flow classes.



(a) Third-order delay with multiple inflows **(b)** Explicit structure for third-order delay comparable to BSL’s `DelayN` class

Figure 7. Third-order delay with multiple inflows modeled as a stock with internal dynamics (dynamic stock)

Reaching Higher Levels Of Abstraction

Using *composite connectors* any stock can have a switchable *stock information connector* reporting the *rate of inflow*, the *amount* in the stock, the *mean residence time*, the *net rate of flow*, and the *rate of outflow*. Such compact connectors can be used to model a coflow structure with essentially two connections, that can be filled and drained like a regular stock in a generic “managing the workforce” structure (Figure 8), or to build quantitative causal loop diagrams (Figure 9). *Policy converters*—a recommendation by Morecroft (1982)—make diagrams more easily comprehensible and documentable.

The availability of modern differential algebraic equation (DAE) solvers in Modelica environments allow to robustly solve simultaneous equations, which may prevent implementing *quantitative causal loop modeling approaches* like MARVEL (van Zijderveld, 2007). A loop of elasticities (B1) in an exemplary model of world dynamics presents no challenge to DAE-solvers.

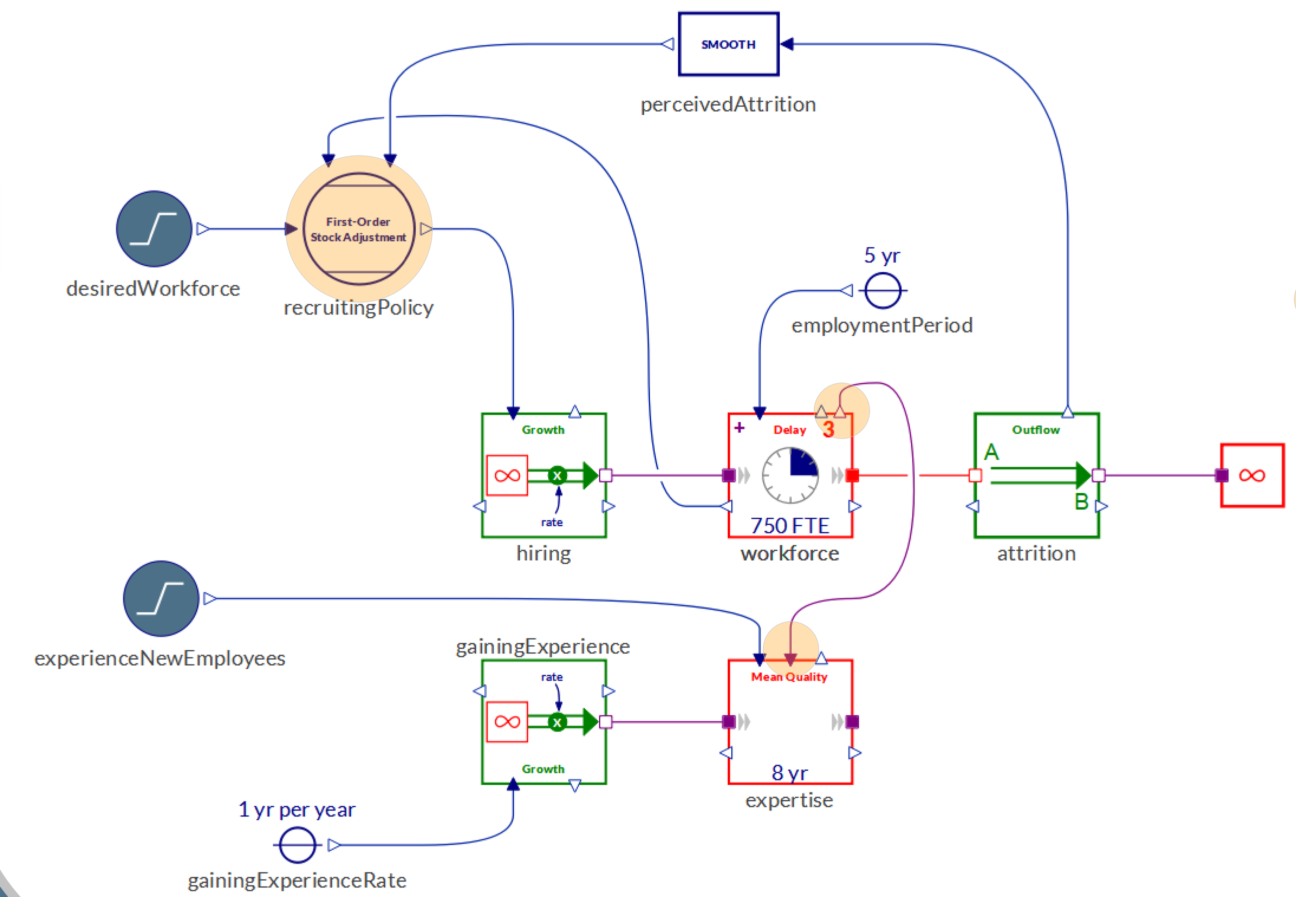


Figure 8. HinesCoflow and FirstOrderStockAdjustment in a generic structure

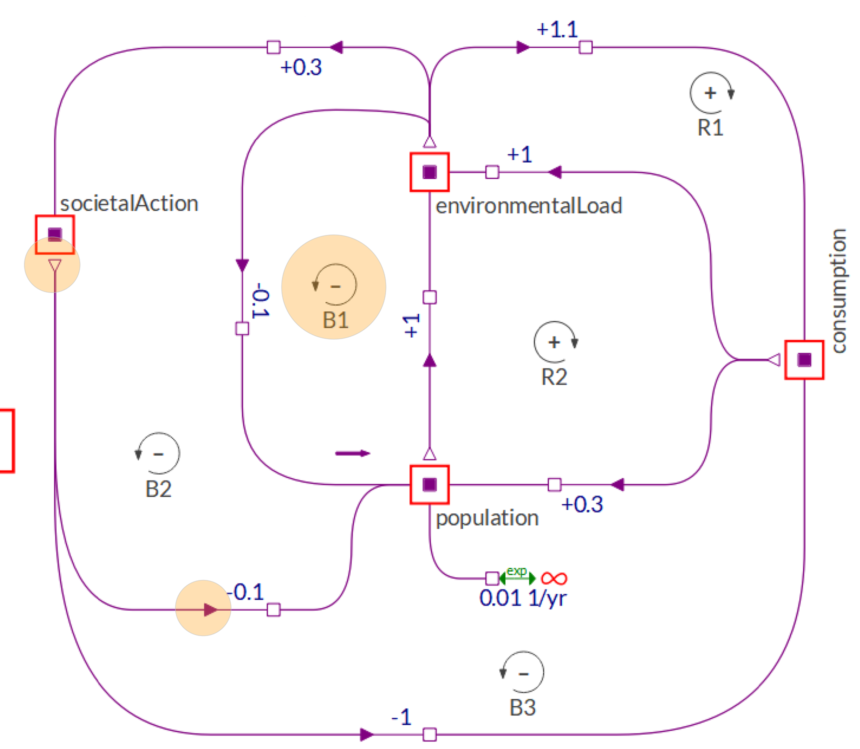


Figure 9. Quantitative causal loop model of world dynamics

We are gaining the full benefits of object-oriented development, including the kind of reusability that Sotaquira and Zabala (2004) missed in existing object-oriented approaches. We can: (a) inherit and modify structure to build new components or to store scenarios; (b) easily and thoroughly test partial models; (c) re-use pre-built components to more efficiently—and more reliably—build large-scale models; (d) use structural parameters to build more general classes, e.g., switch between using a constant or a continuous-time input; (e) replace model components or subsystems with another variant, as we know that its interfaces match up, e.g., switch between different policies in a model; (f) better document and present our models, as each class is documented separately and links to the documentation of its components; (g) mix discrete-event and continuous time simulation in our models; (h) have better support for collaborative modeling; (i) use the standardized and industry-level functional mockup interface (FMI) for model exchange *and* co-simulation.

From a more idealistic perspective, we feel that Modelica fulfills the promise of systems science in that we may mix elaborate physical models from different domains and system dynamics on the same modeling canvas.

Elmasry, A, Größler, A. 2018. Supply chain modularity in system dynamics. *System Dynamics Review* **34**(3): 462–476.
Morecroft, JDW. 1982. A Critical Review of Diagraming Tools for Conceptualizing Feedback System Models. *Dynamica* 8(1): 20–29.
Paynter, HM. 1961. *Analysis and Design of Engineering Systems*. MIT Press, Cambridge, MA.
Powers, R. 2011. *An Object-oriented approach to managing model complexity*. Master’s Thesis. The University of Bergen.
Sotaquíra, R, Zabala, GCA. 2004. Reusability in System Dynamics: Current Approaches and Improvement Opportunities. In *Proceedings of the 22nd International Conference of the System Dynamics Conference*. Oxford, England, System Dynamics Society.
van Zijderveld, EJA. 2007. Marvel – principles of a method for semi-quantitative system behaviour and policy analysis. In *Proceedings of the 25th International Conference of the System Dynamics Society and 50th Anniversary Celebration*. Boston, MA, System Dynamics Society.

BSL MANAGEMENT SUPPORT
Business Simulation · Learning · Management Science