

Assemblies™: Lowering the Barriers to System Dynamics

By Karim Chichakly¹, Bob Eberlein¹, Billy Schoenberg¹, and Steve Peterson²

Abstract

Many early attempts at building models reach a dead end because of improper formulations, a lack of understanding, or a fear of the algebraic equations involved. Assemblies were developed to increase people's modeling success and reach people who are too scared to start or don't have the time for the extensive training necessary. By reducing blank-page syndrome and the requirement to develop explicit equations and algebra to best practice formulations and a few simple choices, the chances for success are improved. This not only allows current practitioners to build models and model prototypes more rapidly but reaches people who like the idea of systems thinking but haven't been able to successfully build simulating models.

Background

The steps to building usable system dynamics models have long been recognized as difficult and fraught with missteps. From putting down something on a blank canvas to finding a way to formulate the subtle relationship between three variables, model building is a slow and meticulous process. In Appendix O ("Beginners' Difficulties") of *Industrial Dynamics* (Forrester 1961) there is a section titled "Automatic Formulation and Evaluation." In it, Forrester is making the point that there is no rote set of actions that result in a useful model, but that the modeler needs to fall back on their own understanding and intuition to get things done. In many respects, Forrester's answer to how to build a good model was to be smart and courageous. Unfortunately, few of us are Jay Forrester, and we've long sought tools to help the formulation process.

In introducing STELLA[®], Richmond (1985) articulated five generic flow processes fundamental to the construction of models. Richmond called these "atoms of structure." Büttner (1986) coined, or perhaps more accurately documented, the term "molecule" to refer to these constructs, and identified a synergy molecule that has largely been lost to time. In *An Introduction to Systems Thinking*, Richmond (1992) provided more complete formulations for the generic flow processes and included ways these could be built up into more complete dynamic structures, using the word "template" to refer to them.

Building on all of this, and using the name "molecules," Hines and Eberlein (1996) provide a categorization of most useful structures and an inheritance framework that formalized the relationship between them. This work has been continually refined by Hines (2023) to give more detail on both where molecules come from and where they can be used. Hines et al (2011), using physical objects arranged on a table, shows how approaching these molecules as replaceable components can lead to ease of model creation, increased insight, and better dynamic

¹ isee systems, inc., Lebanon, NH, USA

² Dartmouth College, Hanover, NH, USA

understanding. While models could be constructed with higher confidence and speed using this method, models could sometimes become unwieldy and difficult to understand. This is a concern for our method as well.

We believe five things are needed to improve the success of beginner modelers: (1) accessing structures as chunks, not as individual components, (2) best practice formulations at their fingertips (with specific problem-domain instantiations), (3) suggestions of most likely equations to avoid fear of equations, (4) inferring units directly from the equations so the model is always unit consistent, and (5) keeping the model in a simulatable state – and simulating it constantly – so the user gets immediate feedback of the model’s behavior after every change.

Chunking (Miller 1956) uses the brain’s ability to recognize and use patterns versus seeing the individual parts of an object. It has been used successfully to enhance learning in many areas (Fountain and Doyle 2012) and, especially relevant to us, to teach beginners software development. Molecules and templates represent best practice formulations for System Dynamics models that can be chunked into a model.

There are limited equation forms that appear in most model equations, allowing software to suggest one of these forms and be right most of the time. This removes the need for the user to figure out which equation to use each time. In addition, the equation chosen by the user implies what the units must be, so the user doesn’t need to separately enter them. Finally, by setting equations when the equation is known (including constants) and by encouraging the user to enter a canonical equation form as they edit the stock-flow diagram, the model will always simulate, giving immediate feedback to the user.

We have not found attempts more recent than listed above to address the beginner’s difficulty in knowing and choosing best practice formulations for model development, nor any other modeling environment that integrates the selection and placement of such structures with immediate simulation to give instant feedback on the model’s changed behavior. We thus believe our solution is novel and original. Having worked with many beginner modelers, we believe these capabilities can make a great difference in their modeling success.

Challenges to an integrated software solution

Historically, the best way of using predefined structures has remained to study them and then put them into models using the basic building blocks of stocks, flows, connectors, and auxiliaries. To integrate the use of these structures seamlessly into System Dynamics software, there are four challenges that must be addressed.

The first is flow, in the creative development sense (Csikszentmihalyi 1975, 1990). The importance of this was recognized in Richmond (1985) and positioned STELLA as a way of achieving creative flow by minimizing the disruptions between building and seeing results. Modern GUI-based System Dynamics software with instant simulation goes a long way toward

doing that. But even with the automated display of results as model assumptions are changed, there remains a break between putting the structure together and seeing the results. The solution to that, which we articulate below, does require a tradeoff between expedience and user control, much like autocomplete will at times slow one down when writing a paper.

The second challenge is contextualizing the structure being added to a model to the model's purpose and problem domain. Variable names, units of measure, and even geometry don't always line up nicely with their ultimate presentation. None of these are fundamental issues; it is quite easy to change variable names, units of measure, and layout, but those steps are time consuming. In many respects, the second challenge is derived from the first – contextualization needs to be as easy as possible to enable creative flow. To address this challenge, we provide several adaptations of the placement process as described below.

The third challenge is discoverability, which is closely related to cognitive load. Hines (2023) often makes the point that the difference between an experienced and novice modeler is that the experienced modeler has many structures they have used before at their fingertips while the novice has to search and consider which would be appropriate. In addressing discoverability, we are looking at ways to make that search process as easy as possible while enabling, but not requiring, that consideration.

The final challenge is what to name these substructures. “Molecules,” while fundamentally equivalent to what we are working with, has a well-defined pedigree and it could be confusing to reuse that name. “Molecules” also do not as explicitly call out the five generic atoms or templates identified by Richmond (1992) and we wanted to be able to do that. “Atoms” suggest stocks and flows rather than their composition which seems counterproductive and “templates” is already taken.

For each challenge, we will discuss how we approached it and the considerations that went into that approach. We do not present them in the order listed above, but first look at the name, then contextualization, then flow, then discoverability. These latter three are interrelated and changes in one necessitate adjustments to the others.

Assemblies

There are a variety of words that could be used to describe pieces of model structure that are commonly used in model development. “Structures” itself is the most generic, but other choices include patterns, idioms, bundles, constructs, blocks, and fabrications. We chose “assemblies” because they are assembled out of stocks and flows, and they normally are not sufficiently complete to be useful models. This puts them in the intermediate stage between basic components and a finished product. Additionally, “assemblies” has not been used for other things in the field, so people will not have a preconceived notion of what they might be.

Note that we are not proposing new generic structures. Assemblies is a holistic feature that gathers known generic structures together in a searchable, highly usable, and integrated way, as described in the following sections. The goal is to make these readily available and easily accessed for inclusion in any model at any stage of development. By making these available and easy to connect together, we open the world of modeling to people who are unable to start a model (facing blank-page syndrome – the inability to start when faced with an empty page with infinite possibilities) or, once started, do not know how to create proper formulations. These are both problems we have observed many, many times in beginner modelers.

Contextualization

Contextualization is the process of going from a generic Assembly to a specific instance appropriate to the model in which it will be used. The approach used builds on the “Template-Based Model Wizard” (Chichakly 2004) that was available in prerelease versions of STELLA Version 9, but did not ever make it into an official release. The wizard would give access to the templates as defined in Richmond (1992, Business Edition) and take the user through an adjustment process to work with the current model. The key things needing adjustment are variables names and units. Both are changed through a substitution process so that, for example “demand for xx” and “supply of xx” can become “demand for pretzels” and “supply of pretzels”. Similarly, “units for xx” might get replaced by “grams”.

The use of “xx” in the above paragraph is intended to be illustrative, but actually points to an important issue in abstraction. Assemblies are intended to be quite generic, but presenting generic concepts leads to confusion for many people. For example, the “Coflow Productivity” assembly has the stocks *Resource* and *Quantity*. In Richmond (1992) these are presented as *Cumulative Success* and *Self Confidence*. While those stocks are not generic, the resulting diagram is easier to understand as shown in Figure 1.

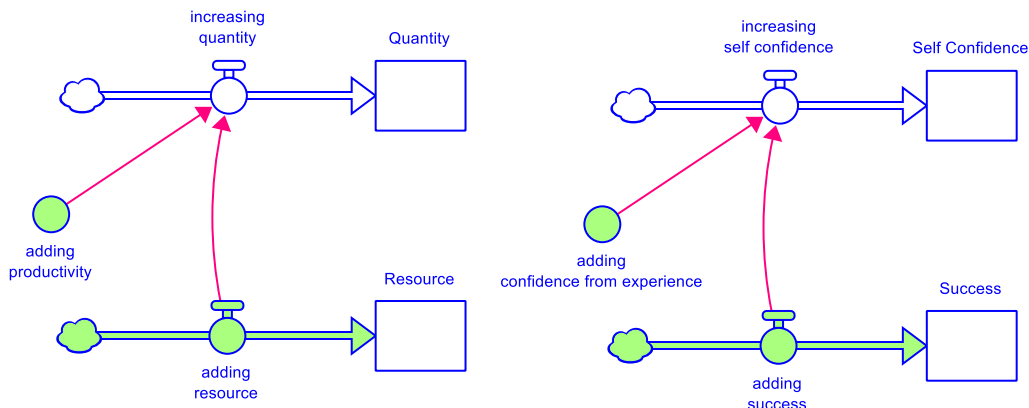


Figure 1. Generic versus specific example labeling of an assembly.

Put another way, it is easier for most people to generalize from a sequence of examples, than to understand the generalization directly. Presenting a number of examples every time would be

tedious, but it is straightforward to provide examples for each Assembly so users can scroll through them as they build familiarity. This has the added advantage that some of those prepackaged examples may be close to the intended use, making life even easier for the user as we demonstrate in the Appendix.

Figure 2 shows how this movement between generic and the specific examples is done for the “Aging Chain” assembly. As different options are selected in the dropdown menu, the names of the stocks and flows adjust from the generic *Resource* to the specific example (such as *Workforce*). The example selected is then displayed in the assembly. This allows the user to quickly view the assembly in context, as well as see the breadth of ways that the assembly might be put in context.

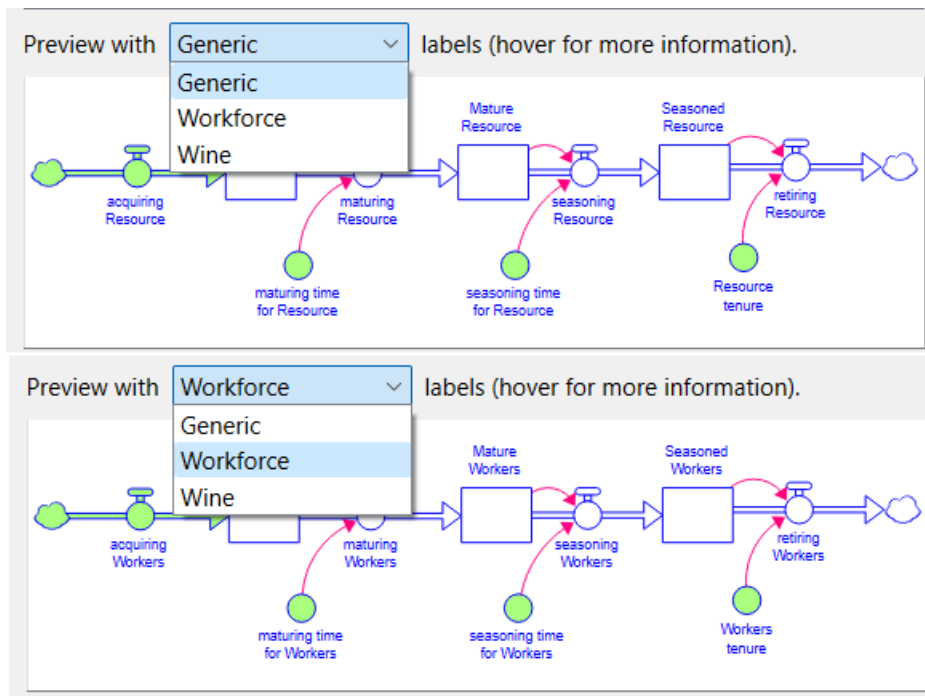


Figure 2. Switching between examples before selecting an assembly for addition to a model.

Once the user settles on an assembly, and the example to use, the names of that specific version will appear and allow further customization (see Figure 3 for the Workforce aging chain).

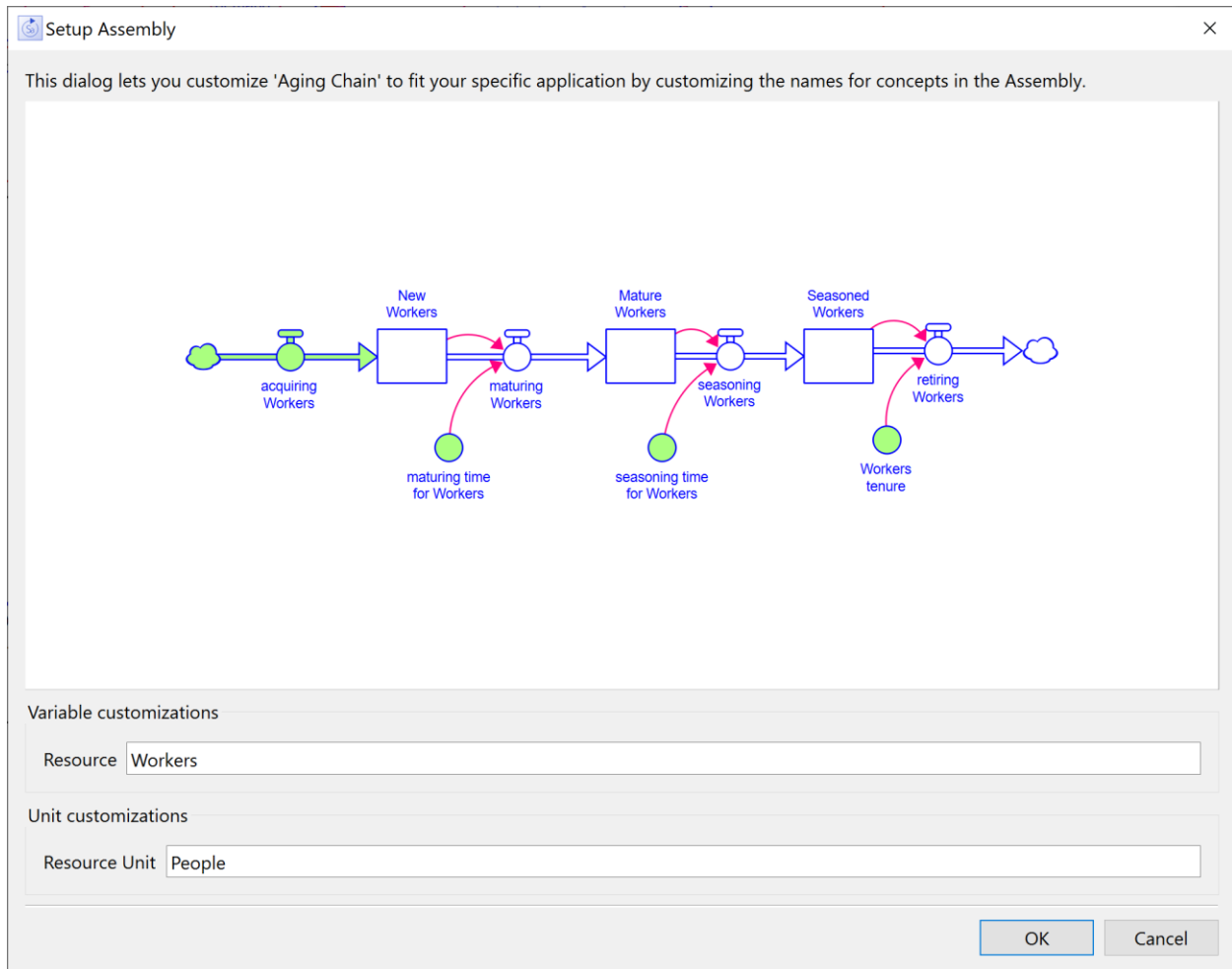


Figure 3. Changing the instance of the assembly also changes the names and units in the placed assembly.

Assemblies are fully valid structures with no errors in the units of measure. Once an assembly has been dropped (done by clicking OK in the dialog shown in Figure 3 and then clicking on the location in the model diagram to place the assembly), it is both running and ready to be connected to other model structure.

By convention, variables in an assembly that you may wish to connect to – either from another assembly or from structure you created – are styled with light green. This is something done for the core assemblies discussed below and something users can do in preparing their own assemblies as discussed in the section on extensions.

Suggest View: Setting Equations Easily to Keep the Model Running

Richmond (1985) emphasized the importance of letting users continually make progress without running into stumbling blocks. In 2001, Richmond was training executives to build stock and flow maps that did not simulate. A strong believer in the need for simulation, Richmond also knew these executives would not dedicate the time necessary to become proficient modelers. He

proposed a STELLA version that would always be simulating based only on the stock and flow map creation by using default constants and operations. Work developing these capabilities began in earnest in the spring of 2002, but was curtailed by his untimely death that summer.

In a nutshell, what Richmond envisioned was unimpeded creative flow, and Suggest View works toward achieving that. As each stock is put down, it is automatically given an initial value and there is a place to name its units. As each flow connects to a stock, it is given a value and it changes the stock as shown in Figure 4. Units of measure are only requested for stocks. Units of measure for all other variables can be inferred with a high degree of accuracy based on the way things are connected (and ultimately the equations used).



Figure 4. Adding stocks and flows with simulation occurring at each step.

Every time a variable is added, it is given an equation, and, as connections between variables are added, the equations are updated either automatically, if there is only a single input, or based on suggestions as shown in Figure 5, if there are multiple inputs.

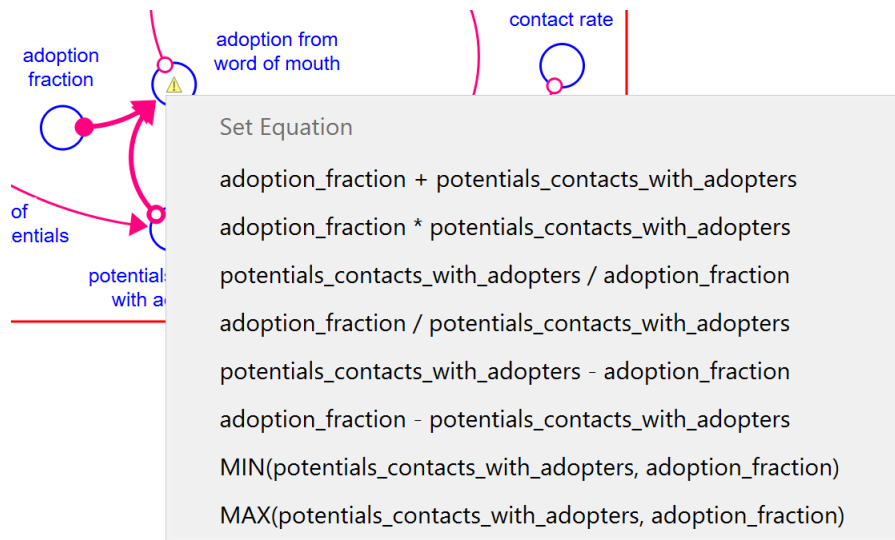


Figure 5. Suggestions for setting *adoption from word of mouth*. This appears when one variable is connected to another (in this case, *adoption fraction* was connected to *adoption from word of mouth*).

Constants (variables with no inputs and stock initial values) are set to 100 by default, but both the right-click menu and the equation panel suggest other choices (see Figure 6). Any arbitrary value can also be entered in the equation panel. Variables with just one input are set to that input.

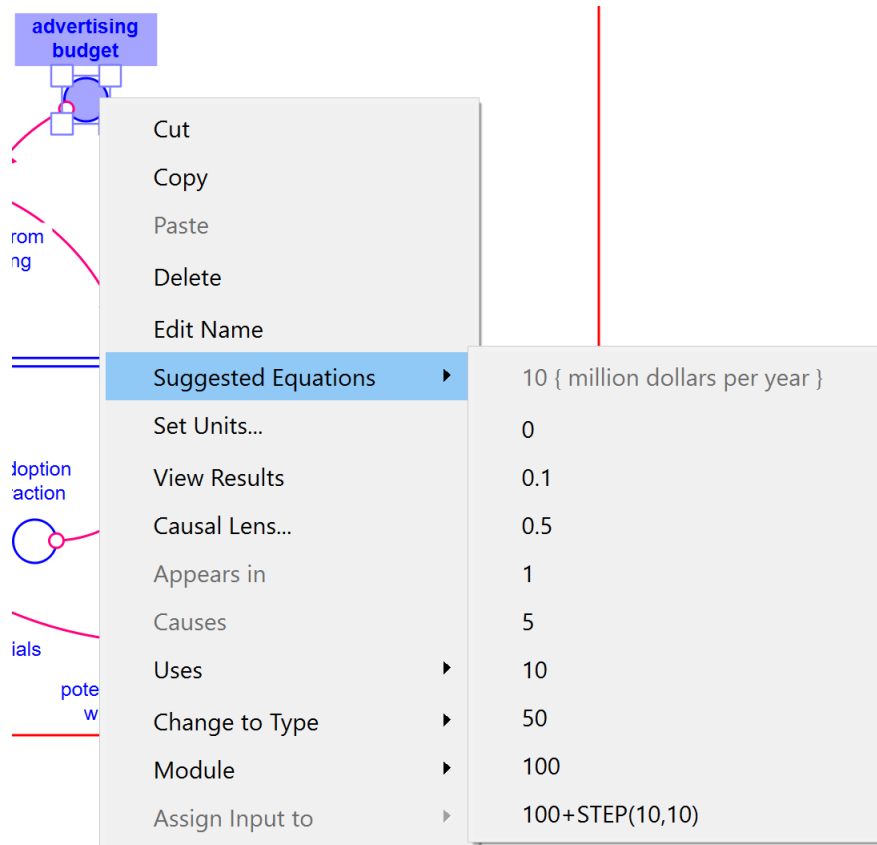


Figure 6. The suggested values for a constant. Note the existing equation appears at the top of all suggestions.

These features not only speed up the model building process, but they allow assemblies to be quickly integrated into larger, working model structures. When the assembly is put down, it is already simulating, but disconnected. The color coding of inputs as light green makes it clear where connections into the assembly should be made, and most often these connections are from a single variable so the software can automatically set the equation. This is demonstrated more fully in the Appendix.

Equation suggestions will not allow the construction of every possible equation. Suggest supports only the basic operators +, -, *, / and the functions MIN, MAX, and STEP. However, it is very easy to edit the equation in the equation panel, as you would for more traditional model development. It is also very easy to switch back and forth between Suggest View and the more traditional Model View with the click of a dropdown as shown in Figure 7.

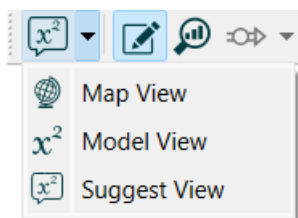


Figure 7. Switching in and out of Suggest View.

In addition to the capabilities directly related to Suggest View, we added many other features to increase creative flow. It is now possible to flip, horizontally or vertically, selected structure in a diagram, making it easier to adjust the geometry of an assembly to the model in which it is being placed. It is also possible to replace the use of a variable, or the variable itself, to decrease clutter, as shown in the Appendix. In addition, the renaming capability described for assembly placement above can also be applied to any selection of variables. These changes are in addition to ongoing development in Stella aimed at making it easier to style and organize model diagrams, equations, and other content.

Making Assemblies Discoverable

We considered two paths to discoverability. The first was through organizing and filtering content to make it easier to narrow down a search. The second was to simplify the provided set of assemblies so that it was relatively straightforward to see all the available choices.

In our original conceptualization of Assemblies, we were going to include most of those in the Molecules (Hines 2023), as well as the Templates of Richmond (1992), and some other common constructs that would ease the burden on the modeler. The notion was that rather than having to adapt an assembly that was close but not exactly what was desired, the adaptation could be made available directly.

With this in mind, we built up a hierarchical organization of assemblies along with the inclusion of a name and keyword search filter as shown in Figure 8. The categories of Adjustments, Chains, and Flows can have subcategories (though they don't in the core set of Assemblies). The filter will operate on the names, explicit keywords, and the example specific definitions. In figure 8, we see that there are four Assemblies related to work.

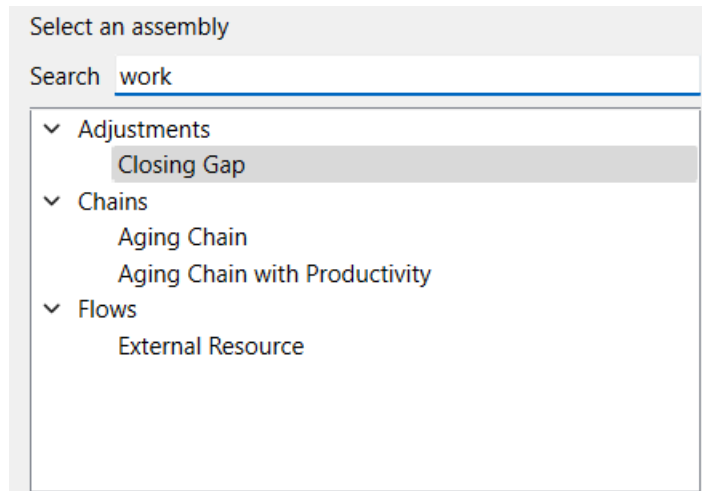


Figure 8. Assembly organization and search filtering.

The advantage of this comprehensive approach is that it is more likely users will be able to find something very close to what they want. The disadvantage is cognitive overload. When we implemented a broad range of Assemblies in this manner it quickly became overwhelming. The difficulties in finding the right thing using the hierarchy and filtering approach was not much different from that of the full set of Molecules which use a different hierarchy and substantial cross referencing.

In the end, we used simplification as the means of improving discoverability. The selected set of assemblies are a minimal set of common constructs that would allow users to build anything we might ask of them in an introductory modeling course. Some of them, like the flow templates of Richmond (1992), were made more complete. Some, notably the dimensionless multipliers, include several variations because even though they are fundamentally the same thing, how to get from one to another is not obvious to most people. In terms of naming conventions, we tried to stick as closely as possible to the terminology used in isee systems' documentation and supporting materials.

Core Assemblies

The core assemblies included with Stella are intended to be sufficient for those getting starting building and working with System Dynamics models. They are organized into three categories: Adjustments, Chains, and Flows. Within each category some of the entries are simply variations on others because it takes some experience to make such a variation. For example, Allocation Quality and Allocation Quantity are fundamentally the same, but the logic of swapping demand and supply is not easily grasped.

- Adjustments
 - Allocation Quality
 - Allocation Quantity
 - Closing Gap

- Combing Effects
- Constrained Inflow
- Constrained Outflow
- Dimensionless Multiplier
- Dimensionless Multiplier Diminishing
- Dimensionless Multiplier Elasticity
- Dimensionless Multiplier Extended
- Stock Adjustment
- Chains
 - Aging Chain
 - Aging Chain with Productivity
 - Capacity Limited Production
- Flows
 - Coflow Attribute
 - Coflow Productivity
 - Compounding
 - Draining based on Rate
 - Draining based on Time
 - External Resource

Extensions

The way assemblies have been implemented in Stella, it is possible to add to them by appropriately configuring models and placing them in a specific location. This can be done by the user, or by others wanting to make available convenient and usable formulations they have used.

This approach, effectively building a library, flies in the face of the design decision we made around Assemblies to keep the number available as small as possible. However, it lends itself very nicely to specialized or problem-area-focused formulation issues. Consider, for example, tracking straight-line depreciation for accounting purposes. In Stella, this can be done using a conveyor with linear leakage and a leak fraction of 1. Such an assembly would not be of much interest for most modelers, but for someone trying to incorporate a proper income statement and balance sheet into a model it becomes a necessity.

Another way to look at this is that when trying to get some insights into an interesting dynamic problem, a relatively inexperienced modeler is probably best off with a minimal set of tools – the core Assemblies included in Stella. But when trying build confidence with specific audiences or incorporate details of specific processes, it is nice to have a more comprehensive library that will take more time to explore but will also have something close to a usable solution.

We are still working on the details of the library development process.

Conclusions

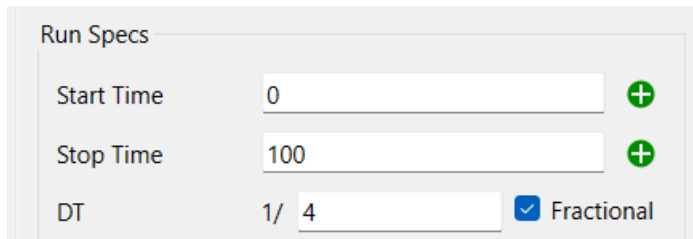
We anticipate Assemblies will allow more people to build simulating models, especially people who understand the concepts, but are unsure how to formulate a model. However, there also exists a greater chance for abuse using Assemblies than there is without them. It is very easy to connect random pieces of simulating structure together to create a model that delivers meaningless results. Our hope is that new training materials will be developed to help people properly choose and configure the assemblies they need for their specific model. In addition, just as we do today with models built from the ground up and we will have to do with AI-generated models, these models will need to be reviewed for correctness.

In a sense, Forrester is right that useful understanding of complex systems requires courage, intellect, and perseverance. But the tools available today are vastly more complete and usable than what he was working with when writing *Industrial Dynamics*. On the one hand, that is a humbling observation, on the other it shows how the types of insights Forrester had are within reach of a broader audience today than they ever have been. Assemblies and the continuous development process they are embedded in open the doors of understanding to still more people. While we may not truly be able to reach the other 98% (99.8%?), we are moving in the right direction.

Appendix: Example

Let's walk through the steps of what it is like to build a model using Assemblies and suggested equations. We will build a slightly nonstandard bass diffusion model using the Compounding and Dimensionless Multiplier assemblies. There is a video of this example including the recovered stock at <https://youtu.be/4oHOugrYEDk>.

1. Set the model to run from 0 to 100 days with a DT of $\frac{1}{4}$. Using assemblies is a complement to, not a replacement for, the standard system dynamics method, and thinking about reference modes and the time horizon remain an important part of model conceptualization.



Run Specs

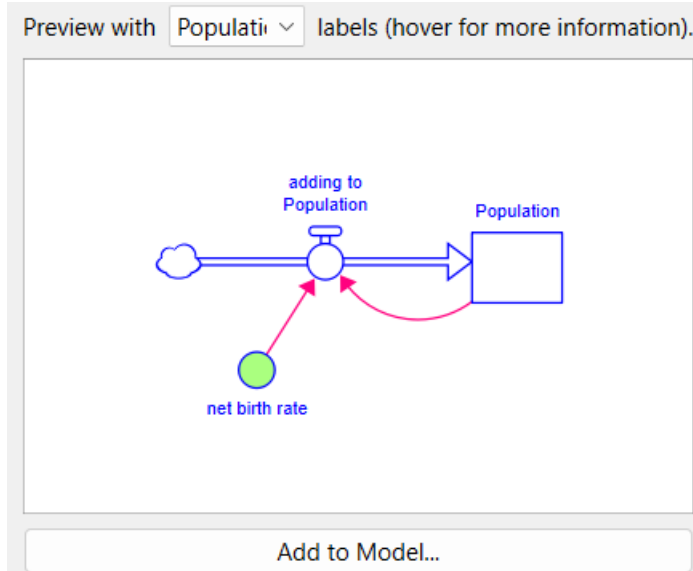
Start Time: 0

Stop Time: 100

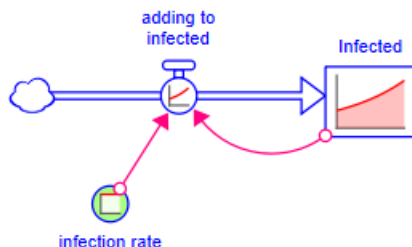
DT: 1/4

Fractional

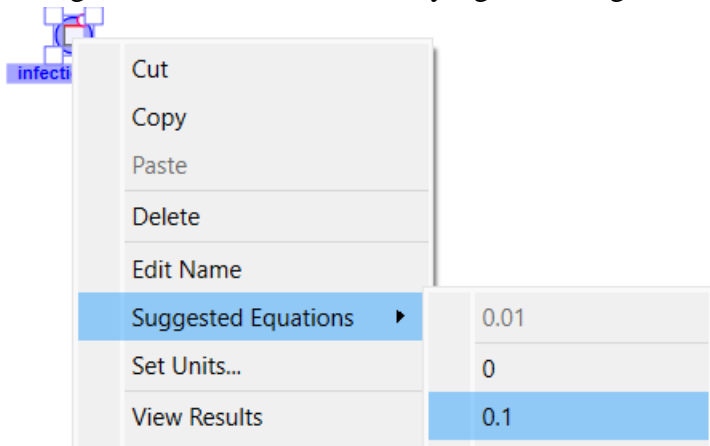
2. Select the population example of the Compounding assembly.



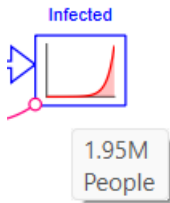
3. Add this to the model changing *Population* to *Infected* and *net birth rate* to *infection rate*. The units are already set from the population example.



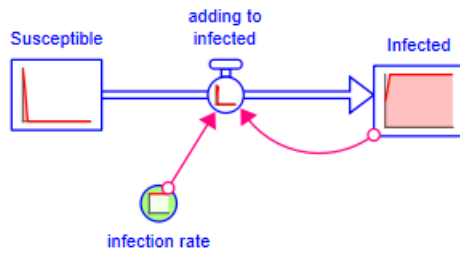
- Change the infection rate to 0.1 by right clicking and selecting this from the menu.



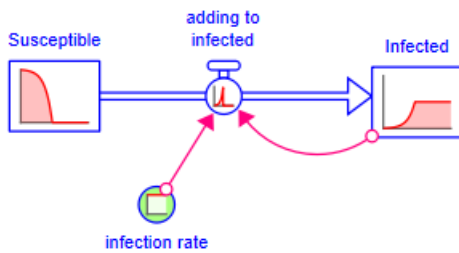
The behavior is quite extreme.



- Add *Susceptible* as a stock to the left of the flow.

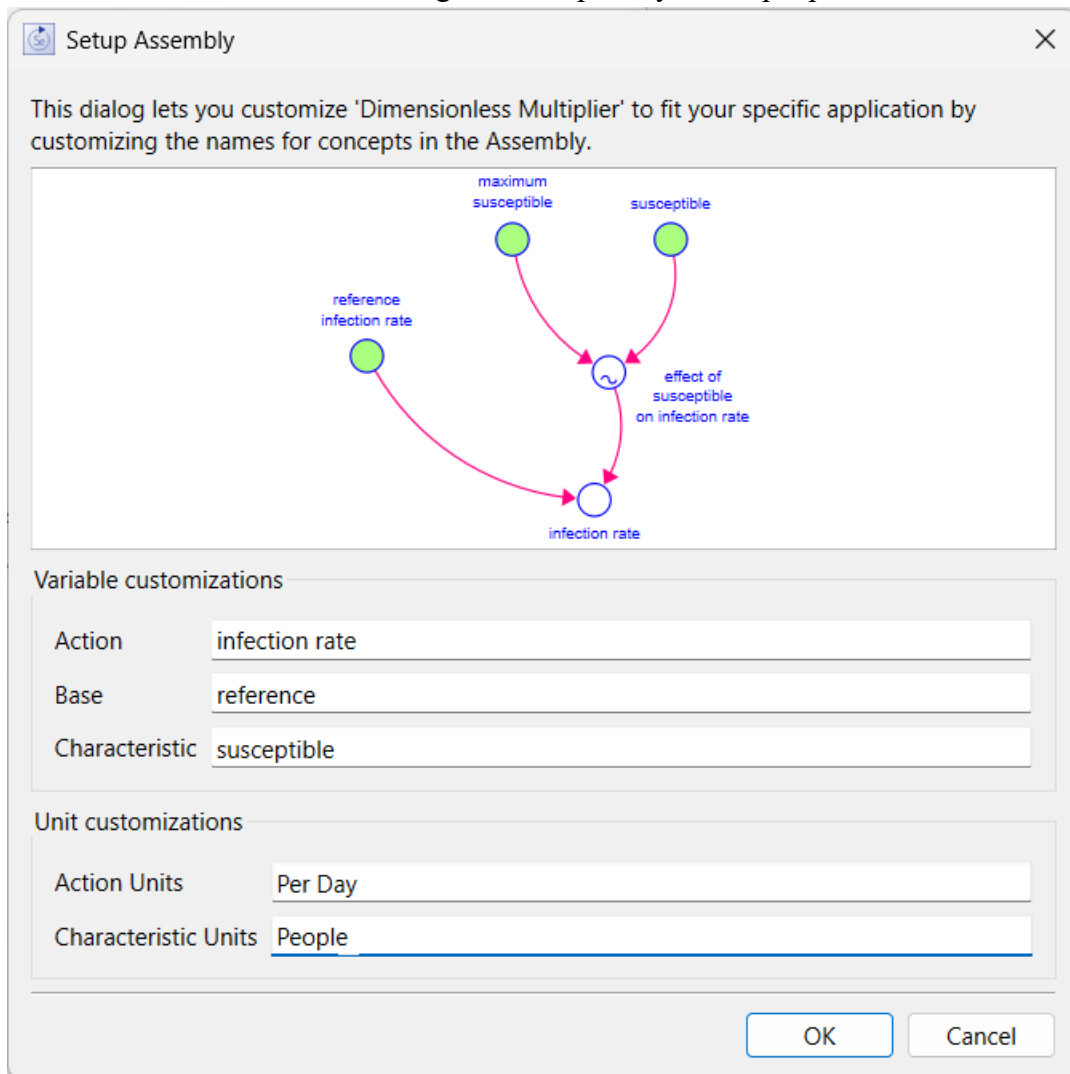


- Right click to change *Infected* to start at 1 (instead of 100) so that we can see some dynamics.

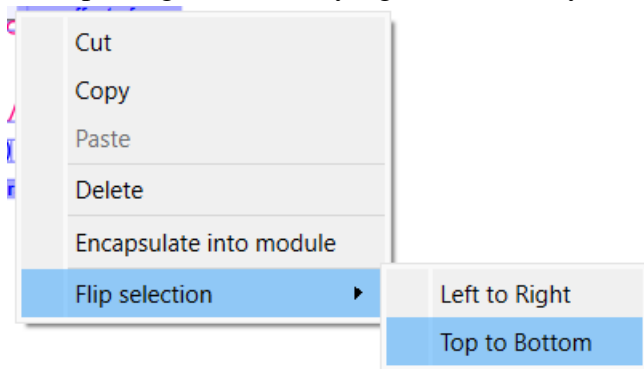


Only the nonnegative nature of the susceptible stock stops infections.

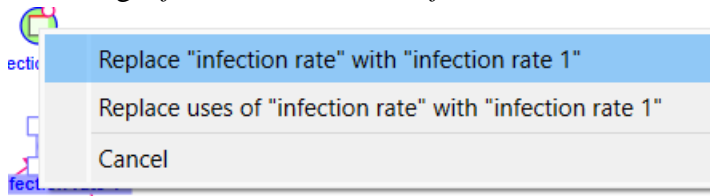
7. Add the Dimensionless Multiplier assembly configured for crowding, renaming action, base, and characteristic and setting units to “per day” and “people”.



8. After placing the assembly right click on any selected element and flip vertically.



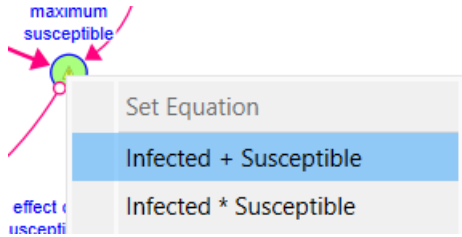
9. Ctrl+drag *infection rate 1* onto *infection rate* and select replace.



10. Right click to change *reference infections* to have a value of 0.1.

11. Ctrl+drag *Susceptible* onto *susceptible 1* and replace.

12. Connect *Infected* to *maximum susceptible*, then *Susceptible* to *maximum susceptible*. On the second connection a menu will come up, select the sum of the two.



It is now possible to experiment with the model using Stella Live and, of course, extend and refine it.

References

- Büttner, Peter (1986). “Molecule and Generic Structure for Synergy” in the Proceedings of the 1986 Conference of the System Dynamics Society,
<https://proceedings.systemdynamics.org/1986/proceed/buttn403.pdf>
- Chichakly, Karim J (2004), “Template-Based Model Wizard”, internal isee systems memo
- Csikszentmihályi M (1975). *Beyond Boredom and Anxiety*. Jossey-Bass Publishers
- Csikszentmihalyi, Mihaly (1990). *Flow: the psychology of optimal experience* New York: Harper Collins
- Forrester, Jay W (1961). *Industrial Dynamics*, MIT Press/System Dynamics Society
- Fountain, S.B., Doyle, K.E. (2012). “Learning by Chunking”, In: Seel, N.M. (eds) *Encyclopedia of the Sciences of Learning*. Springer, Boston, MA, pp 1814–1817.
https://doi.org/10.1007/978-1-4419-1428-6_1042
- Hines, James H. and Robert Eberlein (1996). “Molecules for Modelers” in the Proceedings of the 1996 Conference of the System Dynamics Society,
<https://proceedings.systemdynamics.org/1996/proceed/papers/eberl149.pdf>
- Hines, James H., Thomas Malone, Paulo Gonçalves, George Herman, John Quimby, Mary Murphy-Hoye, James Rice, James Patten, Hiroshi Ishii (2011). “Construction by replacement: a new approach to simulation modeling”, *System Dynamics Review*, Vol 27 No 1, pp 64-90
- Hines, James H. (2023). *Molecules of Structure* <https://sdmolecules.org/>
- Miller, George A. (1956). “The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information”, *Psychological Review*, Vol 63 No 2, pp 81-97
- Richmond, Barry (1985). “STELLA: Software for Bringing System Dynamics to the other 98%” Proceedings of the International Conference of the System Dynamics Society,
<https://proceedings.systemdynamics.org/1985/proceed/richm706.pdf>
- Richmond, Barry (1992). *An Introduction to Systems Thinking*, isee systems, inc.
- Richmond, Barry (1992). *An Introduction to Systems Thinking Business Edition*, isee systems, inc.