

Systems Thinking

Themes

1. Introduction to Systems Thinking
2. Static/Structural Complexity
3. Dynamic/Behavioral Complexity
4. Analytical/Evaluation Complexity
5. System Dynamics Modeling

Introduction to Systems Thinking

A system is simply a group of elements that are connected or related in some way. There are many forms of systems – physical (an ecosystem, a weapons system), conceptual (the Metric System, a betting system), even combinations (a nation’s justice system, a traffic system) – but all are functional, or they would not be identified as systems. That is, a system is worth thinking about only if the behavior of the collective is more meaningful than that of its components.

Many systems (in fact most) are comprised of smaller systems, sometimes referred to as subsystems. The distinction between systems and subsystems is highly subjective, since the appropriate frame of reference depends upon the situation. For example, a cow is a biological system. Within cows are many subsystems: their circulatory systems, nervous systems, digestive systems, etc. If the issue is cow population, it makes little sense to refer to subsystems; the things being counted are cows. On the other hand, if the issue is a disease that affects cows, it may be best to discuss the affected subsystems, or even more specific details. Yet again, if the issue is the stability of a large food chain, the entire cow population may be only a small component of the overall system.

In short, systems thinking really means properly bounding the issue under consideration. This may sound trivial, but it can be difficult in practice. To analyze large problems more easily, we frequently use models. All too often, however, models that are oversimplified or incomplete lead to erroneous conclusions. This may derive from poor design, but more often, it comes from the applying a model to a different system than that for which it was designed. This can be quite insidious: a model which was appropriate in the past may become increasingly inadequate if the problem changes over time. For this reason, it is crucial to carefully consider the system(s) involved in any analysis.

Everyone is familiar with systems. Consciously and without realizing it, people regularly ascribe events to all kinds of systems, real and imagined, and interpret them accordingly. But if this is so, we must be pretty good at it. Do we need training in “systems thinking?” Yes, because there is a large class of systems we have trouble understanding. We label these systems “complex,” and resign ourselves to only the most cursory explanations of their behavior. A good example is weather prediction. What will be the high temperature on March 11, 2004? We can predict exactly how far away the Earth will be from the Sun on that day... but not (reliably) whether it will rain. Why is planetary revolution less complex than terrestrial weather? What makes a system complex?

The terms “complexity” and “complex systems” have a variety of definitions, but below, we will discuss three aspects of complexity, which can be thought of as generating three types of complex systems:

- Static complexity – associated with problems of complex structure
- Dynamic complexity – associated with problems of complex behavior
- Analytical complexity – associated with problems that are difficult to evaluate

Finally, we will briefly introduce System Dynamics, a modeling technique developed to facilitate the analysis of complex systems.

Static/Structural Complexity

The first form of complexity is, ironically, pretty simple. Called static or structural complexity, this is just the idea that the more elements comprise a system, the more complicated it becomes.

For a crude example related to common human experience, consider sorting socks after doing the laundry. (Note: even if you just throw all of your socks in a drawer, you still have to face a version of this problem to find a matching pair when you get dressed.) The task is trivial: pick up any sock, then find its mate. For an ordinary laundry load this doesn't take long. Even trying to sort 30 pairs of socks is only a little bothersome; at first you may have to hunt around a bit, but it gets progressively easier. But consider sorting 300 pairs of socks, or 3 million. Obviously, at some point it becomes overwhelming. Technically, the task remains as easy as ever. But increasing the number of variables, parameters, or “steps” required to finish does make the job take longer, so it is said to increase the static complexity.

In general, however, static complexity alone rarely causes much difficulty any more, for two reasons. The main reason is that today we have fast computers, and repeating simple steps over and over, with incredible speed and precision, is exactly what computers do best. Tireless, and never intimidated by large numbers or tasks, computers can crunch through problems that unaided humans could never complete.

The other reason that statically complex problems are rarely truly difficult today is that, in most cases, we know how to solve them. The fields of Operations Research and Computer Science have continually developed better algorithms for cutting through static complexity, and reducing even enormous problems to comparatively simple ones. Mathematical programming, for example, is the art of using a few mathematical equations to represent a system, and then solving those equations simultaneously to find feasible or optimal solutions. As long as the relationships between system elements are relatively well-known, these techniques, coupled with ever faster and more powerful computers, can solve problems of almost any degree of static complexity.

Dynamic/Behavioral Complexity

If static complexity is not really what makes systems like weather so confounding, we must turn to dynamic complexity. Also called behavioral complexity, this is basically the degree to which the outputs (behavior) of a system are difficult to connect to the inputs.*

The definition may not sound striking, but the concept is critical. Intuitive understanding, formal modeling, and rational decision-making are all fundamentally based upon our expectations about how system behavior will change in response to specific changes in inputs. If our beliefs about these relationships are incorrect, we cannot use available information to make wise decisions. This is actually fairly common: there are numerous examples of the process, sometimes called “policy resistance,” of believing we understand a problem, taking steps to try to remedy it, but failing (and too often making things worse): anti-lock brakes in cars make drivers more aggressive, decreasing safety; fire-suppression policy results in more severe forest fires; low-tar cigarettes lead smokers to smoke more, increasing carcinogen intake, etc. Except in very special cases, however, there is not really an aspect of the system deliberately resisting the intended solution. Rather, our understanding of the system – specifically, the reasons for its behavior – is inadequate. Why? Many factors contribute to dynamic complexity. Six of the most important (commonly present but unrecognized or misunderstood) are introduced below.

* For the concept of dynamic complexity and some characteristics of dynamically complex systems, we are indebted to John Sterman. For more information, see [Business Dynamics: Systems Thinking and Modeling for a Complex World](#) (Irwin/McGraw-Hill, 2000).

Dynamics – over a long enough time frame, almost any aspect of a system can change; in a short enough period, none do. Unfortunately, key elements can change at dramatically different rates, making them difficult to judge. Causes and effects may be distant in time, and their relationships may change by the time their connections are recognized.

Coupling – components of a system are said to be “coupled” if changing one affects the other. Tightly coupled elements make it almost impossible to affect the intended target exclusively; any change also causes some other changes, each of which may cause still more changes, ... with each induced effect being less intentional and therefore potentially less predictable, recognizable, and controllable.

Feedback – the condition created by a causal loop of couplings. When a system involves feedback, a known change to a factor can eventually travel back and exert an additional change to the original factor. A single feedback loop can result in significantly greater or lesser effects than expected, if the feedback is not anticipated. Multiple loops, acting in different time frames, can result in practically unpredictable behavior.

Nonlinearity – when the changes in one or more outputs are not proportional to changes in the inputs. This is caused by coupling or feedback, and is primarily problematic because people rarely anticipate – or fully understand – nonlinear relationships.

Chaos – for our purposes, this is basically unpredictable behavior from a deterministic system. Chaotic systems are characterized by extreme sensitivity to initial conditions and unpredictable, aperiodic evolution (with some stable structure) instead of convergence to a steady-state. The classic description of chaotic behavior is the “butterfly effect.”

Adaptation – if the capabilities or preferences of actors within the system can evolve, (i.e., they learn from or actively respond to events), self-organization, self-selection, and co-evolution can result in sophisticated “emergent” behavior.

Failure to recognize these characteristics in a system may lead to underestimation of the dynamic complexity of the system. Consequently, predictions of system behavior may be seriously flawed, while unwarranted confidence in these predictions (stemming from the belief that the “mechanics” of the systems are understood) may be maintained.

Finally, note that dynamic complexity is entirely independent of static complexity. Dynamic complexity can arise even in systems of extremely simple structure; Edward Lorenz “discovered” chaotic behavior (1963) while working with a system of only three equations (next page). To his surprise (and dismay), instead of converging to a solution, the system exhibited the chaotic behavior plotted in Figure 3.1.1.

$$\frac{dx}{dt} = a \cdot (y - x)$$

$$\frac{dy}{dt} = rx - y - xz$$

$$\frac{dz}{dt} = xy - bz$$

Note that Lorenz originally used $a = 10$, $r = 28$, and $b = 8/3$. He was attempting to model the circulation of warm and cool air masses, to improve weather prediction.*

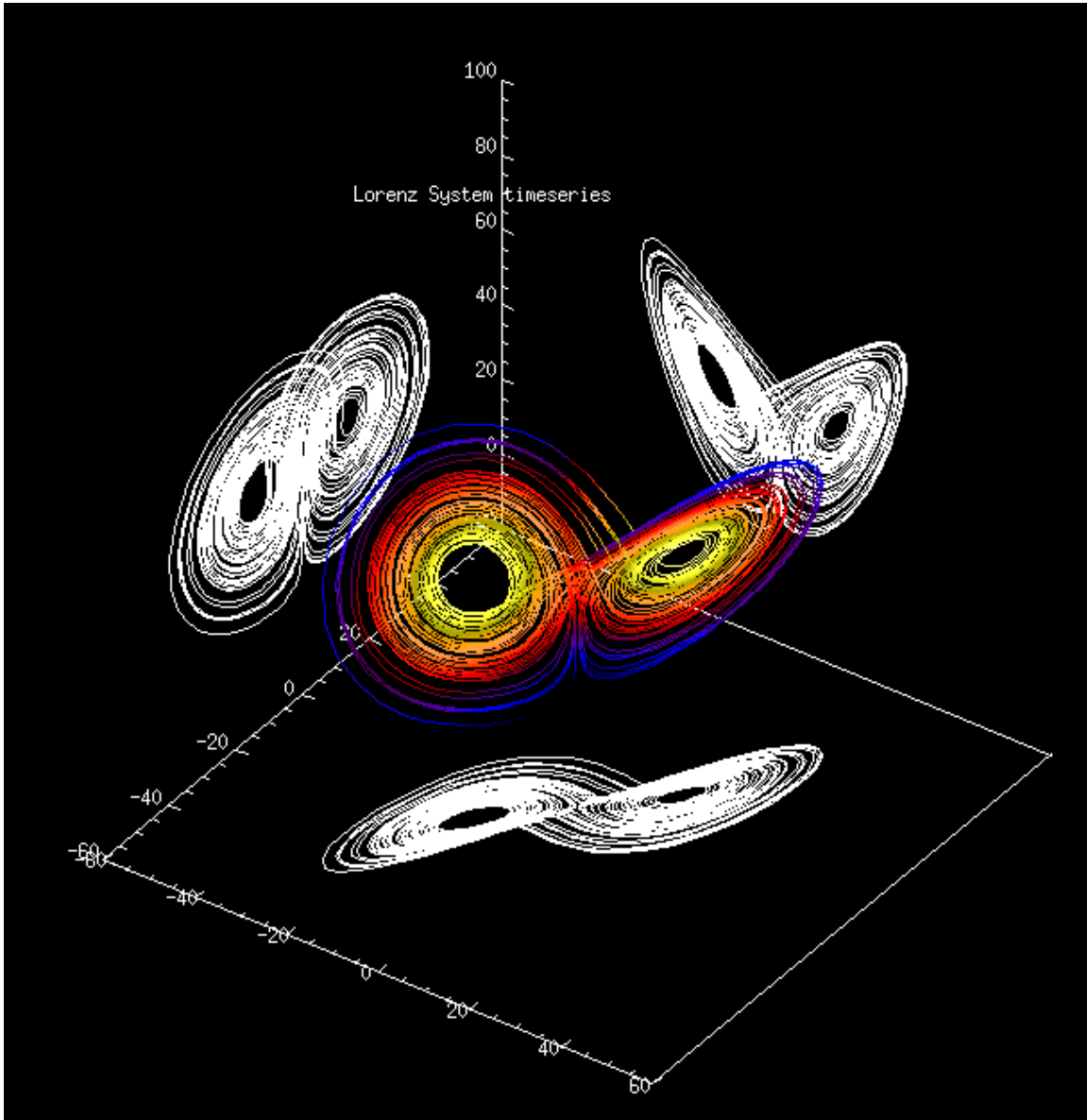


Figure 3.1.1: Plot of 3-D Lorenz Attractor (center, color) and 2-D Projections †

* Robert S. Houghton, *A Chaotic Paradigm* [<http://www.ceap.wcu.edu/Houghton/thesisM/Ch3.html>].

† Courtesy of Seth A. McGinnis [<http://ucsub.colorado.edu/~smcginni/science/poster.html>].

Analytical/Evaluation Complexity

The final type of complexity we wish to discuss is analytical complexity. Also known as evaluation complexity, this is the notion that a system is more complex, the more difficult it is to “solve” in terms of a given policy. In other words, systems exhibiting analytical complexity involve value disputes and multidimensionality, complicating the evaluation of and choice between policy options.

Analytically complex systems are characterized by trade-offs. For example, consider a simplified version of the air traffic control system, in which the only public values are safety and efficiency. It is common knowledge that delays keep getting worse, year after year. Suppose a policy was proposed to reduce the minimum required separation between aircraft, to try to accommodate more traffic (in the same amount of airspace) and thereby cut down on delays. Overall, would such a policy be beneficial?

The answer would be difficult to determine, even considering only safety and efficiency. We would need to decide how to evaluate each value independently, and how to trade them off. This might be easier said than done, because various groups would probably prefer different definitions of each. Safety, for example, could be measured in terms of conflicts (when one aircraft comes within the minimum separation distance of another), accidents, fatalities, or accident costs. Efficiency could be measured in terms of flights per day, average delay, maximum delay, passengers per hour, etc. Moreover, groups that could not agree on the very definitions of “safety” and “efficiency” would be even less likely to agree on the minimally acceptable levels of each, or the appropriate trade-off between them. Thus, even if the air traffic system were so simple (or well-modeled) that we could accurately estimate any effects, this problem would still be complicated.

Analytical complexity is different than static or dynamic complexity in that it really arises not from the system itself, but from a larger context. This is actually an indicator that our concept of “the system” needs to be expanded. For the above example, the best procedure would be to step back from the details of the air traffic system, and begin researching the preferences of key groups of stakeholders. It does not make sense try to determine the appropriate minimum separation distance between aircraft until we have obtained an acceptable definition of society’s preferences regarding safety, efficiency, and the trade-off between them.

So to summarize: statically complex problems are large, but not necessarily difficult; dynamic complexity is what actually leads to “complex” system behavior; and analytical complexity generally suggests that the system in question is improperly defined.

System Dynamics Modeling

At this point, you may be wondering how to proceed if the problem you wish to address appears to involve significant dynamic complexity. For such cases, there do exist advanced modeling techniques. One of these, called System Dynamics, was developed to facilitate the understanding and analysis of dynamic systems. Of course, proficiency in System Dynamics modeling requires more advanced classes, but the following is a brief introduction to the general approach.

Causal Loop Diagrams – the heart of System Dynamics modeling, these diagrams define the key relationships in the system. To build a causal loop diagram, we must first identify the main elements we want to track. For example, suppose we’re running a farm, and we want to keep track of our chickens. Our major elements would be the chickens, the way we gain chickens (eggs), and the way we lose chickens (road crossings).^{*} Now we are ready to draw a causal loop diagram; we just have to connect our main elements with a loop depicting each process (gaining and losing chickens):

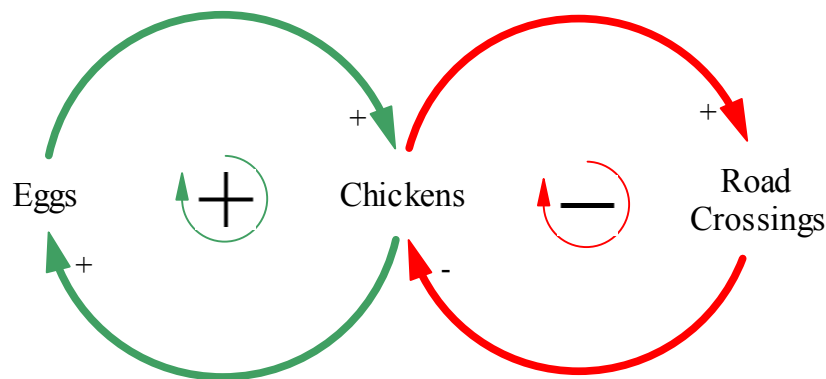


Figure 3.1.2: Causal Loop Diagram (created in Vensim[®])

There are two feedback loops. On the left is a loop describing how we gain chickens: more chickens lay more eggs (hence the + at the tip of the arrow), then more eggs leads to more chickens (+). The right loop describes losing chickens: more chickens leads to more road crossings (+), but then more road crossings kills more chickens (this time a -). Now for a trick: in any feedback loop, we can simply count up the number of + and - symbols; even means a “reinforcing” loop (more each time you go around), while odd means a “balancing” loop (less each time you go around). Logically, we find that our chicken-gaining loop is reinforcing (hence the big + symbol in the middle), while chicken-losing is a balancing loop (hence the big - symbol in the middle). Starting with such a causal loop diagram helps to clarify the processes driving system behavior.

^{*} For this illustrious example, we are again indebted to John Sterman.

Although balancing and reinforcing loops themselves are straightforward concepts, the system behavior that various combinations of them produce can be extremely complex (especially if time lags are involved). Thus, System Dynamics modeling builds complex behavior from combinations of simple causes. Figure 3.1.3, below, illustrates some modes of behavior common in dynamic systems. Each of these behaviors can arise from just a few reinforcing and/or balancing loops.

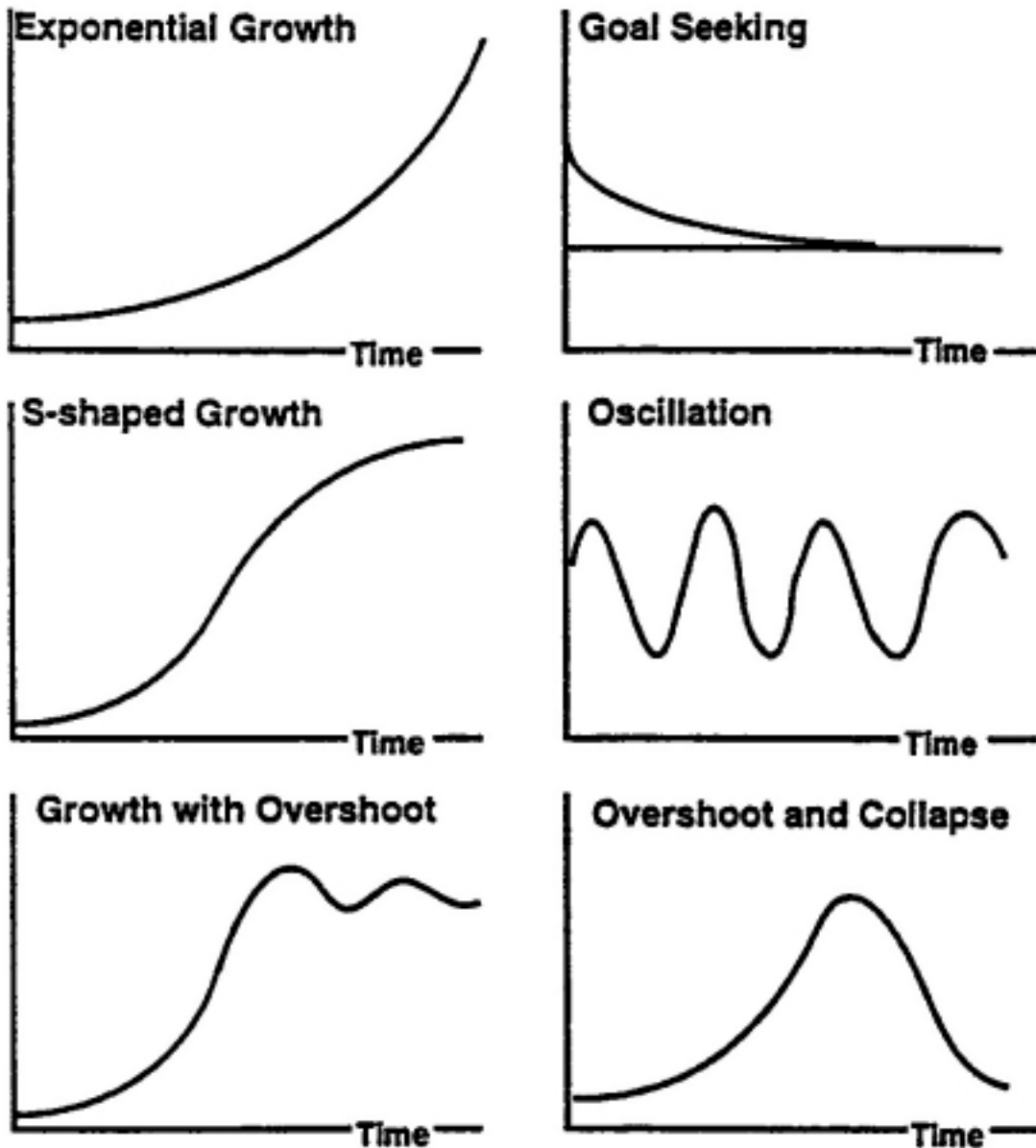


Figure 3.1.3: Common Modes of Behavior in Dynamic Systems

Stocks and Flows – the main elements of the actual model, which we’re ready to design now that we have a causal loop diagram. First, we have to decide which of our elements are “stocks,” and which are “flows.” Stocks are quantities that accumulate, such as money in a bank account, employees in a firm, or components in a warehouse. Flows are the sources and sinks that add to or deplete stocks. The following figure depicts stocks and flows using four different representations, all of which are equivalent.

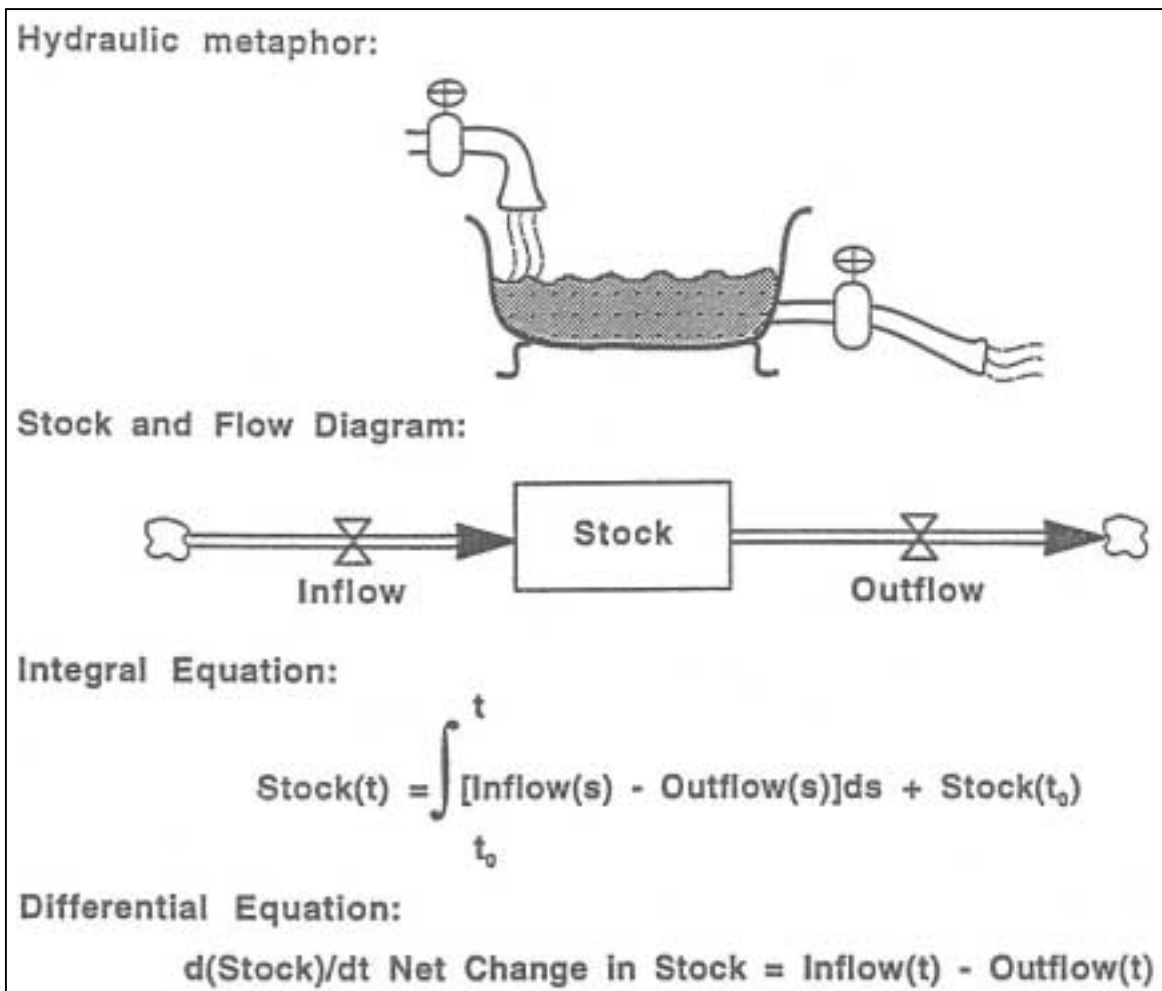


Figure 3.1.4: Four Equivalent Representations of Stocks and Flows

For the chicken model, chickens and eggs are stocks, while road crossings would be a flow. With this information, we can begin graphically building our model. After adding a few more variables that helped complete the picture as we developed it (hence we call these variables “auxiliaries”), we ended up with the following:

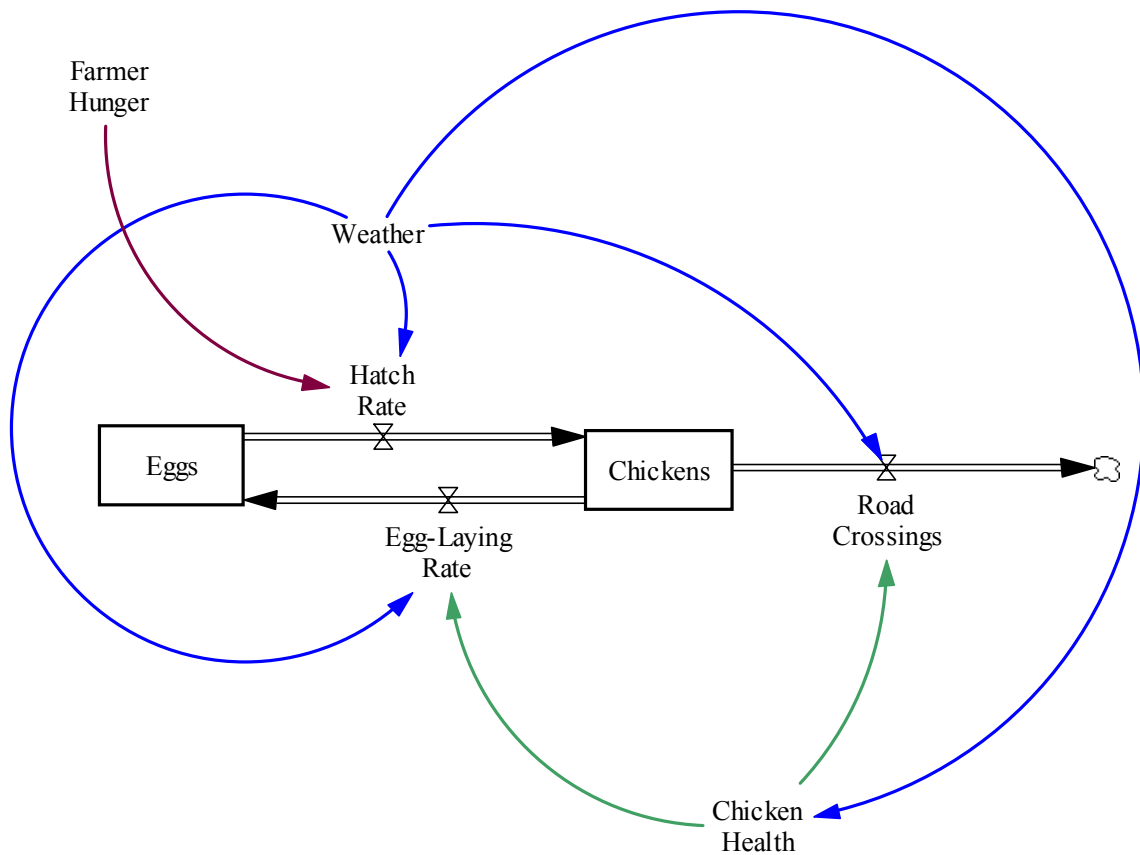


Figure 3.1.5: Graphical System Dynamics Model (created in Vensim[®])

Clearly, this is a little more involved than the causal loop diagram. But the main elements are unchanged: the reinforcing loop between chickens and eggs, and the balancing loop between chickens and road crossings. [Vensim[®] conventions: stocks are boxes, flows are valves on “pipe” arrows (connecting stocks to other stocks, or to “clouds” when the stock flows to/from a place outside the bounds of the model), and auxiliaries are simply floating words, connected by arrows to the system elements that they affect.]

This model was developed by simply placing the elements we had already defined – Chickens, Eggs, and Road Crossings – and proceeding logically. The Road Crossings flow from Chickens goes to a cloud, since we didn’t want to model Chicken Heaven (or wherever chickens go when they die). Next, we had to add a flow into Chickens, lest our stock simply deplete and we remain out of chickens forever. We already knew this flow had to come from Eggs, so we added one and named it the Hatch Rate. Then all we needed was a flow into Eggs, which we already knew had to come from Chickens, so we added one and named it the Egg-Laying Rate.

At this point, all of the elements we initially identified were placed and connected, but the model was not quite complete, because we needed to add any factors that we wanted to have affect the rates. So we thought about it and added three auxiliaries: Chicken Health, Weather, and Farmer Hunger, connected to the rates as shown. Finally, we decided that Weather would also probably affect Chicken Health – a new feedback loop (one which we had not expected!) – so we added that arrow, completing the model.

Equations – once the graphical model is complete, the last step would be to go through and define the equations governing all of the relationships, though we will not actually do this for the chicken model. Vensim[®] (or any other System Dynamics software) would assist us in this, by displaying each intersection of model elements that it expects us to describe using equations, and then checking that the units all match up.

Once the model is complete and all of the equations are defined, it would be time to run it. The usual procedure for a System Dynamics run is to prepare a few graphs, and then watch them fill in as the model elements change over time. For the chicken problem, for example, we might want a plot of Chickens over time, and possibly one of Eggs as well. The last step would be to calibrate and validate the model, making sure that all known/predictable behavior develops as expected. This step can be difficult, and is more effective the more data is already available for this purpose.

Finally, the real power of the System Dynamics modeling approach: now we could begin adjusting things and monitoring the effects. For example, we could begin selling eggs, or regularly buying chickens. Or we could investigate the effects of a change in weather on our chicken population. In serious models, this enables a great deal of “what-if” analysis, investigating policies and their propagation through the system’s feedback loops, for very little additional work on the part of the modeler. Thus, a sufficient System Dynamics model is a very powerful analytical tool.