Evaluating Systems *as* Systems And Using that Knowledge to Inform Program Evaluation

Jonny Morell

jamorell@jamorell.com

blog

YouTube

Abstract

Our evaluation models have two personae. One is the logic of the program we are evaluating. The other is the system logic that is embedded in the relationships depicted in the program model. This is not a distinction that we usually make. The premise of this article is that we would do well to make this distinction because appreciating the system logic can provide insight into the program. The argument proceeds through four examples with different system attributes. Each illustrates how knowing the system logic can inform the program logic. The system attributes used in the examples are: 1) causal chains, 2) stocks and flows, 3) attractors and equilibria as behaviors of complex systems, and 4) network structure. The article concludes with variables that characterize systems. It may never be necessary to employ more than a few of these characteristics, but it is worthwhile to have a sense of the broad range of the possibilities. Section headings in the table are 1) rates and magnitudes, 2) boundaries, 3) system architecture, 4) change, and 4) resistance to change and sustainability,

Scope of this Article

There is much talk about how our programs are (or should be) thought of in terms of systems, and quite a bit of progress is being made toward that end. There is a difference though between:

- evaluating programs in terms of systems, and
- evaluating the logic of the systems themselves,

The first is the notion of system as it is used in common discourse, e.g. a car as a mechanical system, the body as a biological system, a green energy development program as a socio-technical system. The common theme is that the system has recognizable parts that fit together to make something happen.

The second refers to relationships that are abstracted from the details of the real-world system and labeled and understood on their own terms. They have their own vocabulary and their own logic (Meadows, 2008; Williams, 2010). To take a simple example, a description of a thermostat might be: a device that controls the temperature of the air. It measures the air temperature and tells the furnace when to turn on and turn off. The term "feedback" does not have to be part of the explanation of what the thermostat does. But feedback as a systems concept is operating, and not only with the thermostat, but also in countless other scenarios.

The purpose of this document is to take a stab at evaluating systems in its second sense. The effort is based on my belief that systems logic is embedded in the domain specific models that we construct to guide program evaluation, and that it is worthwhile to look at each separately. (I use the term "model" rather than "logic model" because the latter term diverts attention from the long history of models as they are used to guide inquiry. (Box, 1979; Frigg & Hartmann, 2018; Rogers, 2012)).

Left out of this document is a consideration of systems as they are treated in the fields of ecology and evolutionary biology (Morell & LeGros, 2025). There is good reason to include systems like these, but doing so would lead the discussion too far afield.

To make my case I use simple examples that are accompanied by (what I hope are) a few intuitively obvious system behaviors. In point of fact, real-world cases would have much more elaborate models, accompanied by numerous system behaviors that can be very technical and mathematical. But this article is not a primer on models or system behavior. It is an explanation of why it can be worthwhile to abstract system logic from domain-specific program logic. Simple examples are best to provide this explanation. Those simple examples will cover:

- causal chains,
- stocks and flows,
- complex systems with an attractor / equilibrium focus and,
- network development and structure.

Why bother? Because understanding the system that underlies a program will help us understand the program. An illustration is presented in Example 1.

Example 1: Causal Chains

Imagine a program with a causal chain showing outcomes at three points in time, with the third outcome being the most important. The program works. That third outcome appears. But in Universe One, the logic is the top of Figure 1, while in Universe Two, the logic is the bottom of the figure.

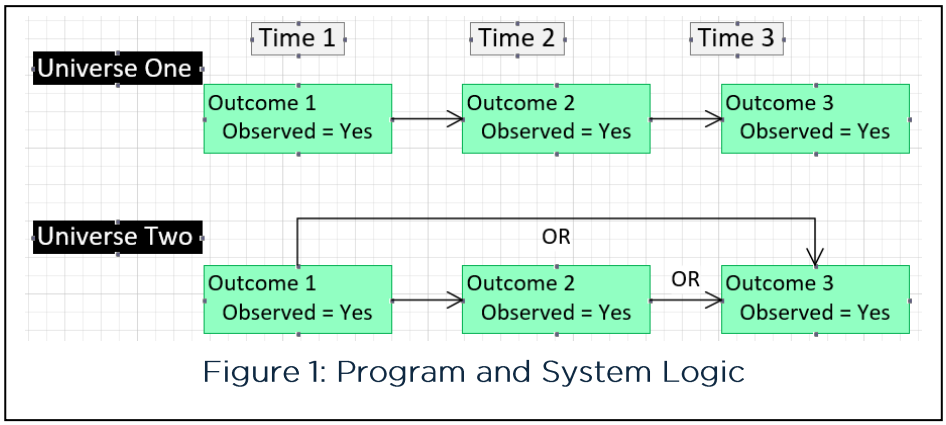


Figure 1: Program and System Logic

From a program evaluation standpoint, let's say that the evaluators assessed whether the outcomes occurred and which causal paths were operating. That's great program evaluation, but it's not a system evaluation.

A system evaluation would focus on the logic of causation and would reveal how radically different the programs are in Universe One and Universe Two. In Universe One the program is fragile because there is only one path from the earliest outcome to the last. In Universe Two the program is robust because Outcome 3 stands in a 1:many (many = 2) relationship with its precursors, with each of those "many" relationships capable of effecting the desired change.

In this simple example the evaluators would surely understand both the programmatic and the system message. What they might not appreciate is that they were looking at two qualitatively different findings, one dealing with program logic and one dealing with system logic. As models get more elaborate, the implications of system logic for program logic become less obvious and less intuitive.

Example 2: Stocks and Flows

An innovative program is being fielded to teach math. There is an elaborate model to guide the evaluation but consider a snippet of that elaborate model (Figure 2). The top model is as it was originally constructed. The bottom model has the exact same architecture, but it casts the model in systems terms. These are both

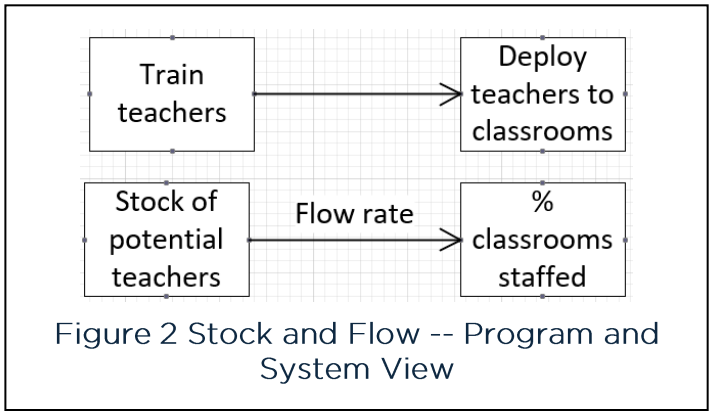
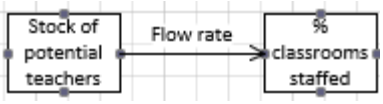


Figure 2 Stock and Flow -- Program and System View

entirely reasonable and complementary ways to do the evaluation, but they provide different theoretical views of the program and suggest a different set of questions (Table 1).

Table 1: Example Questions Suggested by Program and System Views	
	
Does the training adequately equip teachers to work in the classrooms?	What is the relationship between training capacity and the population of trainees?
How do the teachers use the curriculum in practice?	How quickly do trainees get deployed to classrooms once they complete their training?
Do students' math scores improve?	What is the ratio of qualified teachers to the number of classrooms that have to be staffed?

Example 3: Complex Systems with an Attractor Space and Equilibrium Focus

An attractor is a set of values to which a system will tend to over time.[1] With this definition in mind, consider the population of teachers in classrooms as depicted in (Figure 2). At any given time there will be variation in:

- the number of teachers entering training,
- the number of teachers dropping out of training,
- the number of teachers completing training but declining their teaching jobs,
- the number of teachers leaving their jobs once deployed in classrooms, and
- the number of classrooms that have to be staffed.

Measured over time, each of these will have a distribution, with each distribution having a mean, variance, and shape. One could also calculate the joint distribution of these variables for the program as a whole, or for parts of the program. Of course one could just look at any of the distributions in purely statistical terms. But with respect to understanding how the program functions within its environment, it also makes sense to interpret that data as describing the program's equilibrium state to which it (or parts of it) are attracted over time. So doing would lead to questions such as:

[1] For the sale of simplicity, this explanation is adapted from a more technical definition that is provided in the glossary of Complexity Explorer. The full definition is: "In dynamical systems, an attractor is a value or set of values for the variables of a system to which they will tend towards over enough time, or enough iterations. Examples include fixed-point attractors, periodic attractors (also called limit cycles), and chaotic (also called "strange") attractors ". https://www.complexityexplorer.org/explore/glossary/6-attractor#gsc.tab=0

- What is it about the program and its educational setting that generates that equilibrium?
- How stable is that equilibrium, i.e. is it deeply embedded in a particular state (imagine the bottom of a well), or likely to move around a lot (imagine a shallow valley)?

While this example deals with only two specific behaviors of complex systems, it also conveys a more general message, to wit, that that it is the behaviors of complex systems that matter, not whether something is or is not a complex system (Morell, 2024)

Example 4: Network Focus

Imagine a program designed to set up small local health clinics throughout a region. Again there is an elaborate model to guide the evaluation but consider the snippet in Figure 3. That model is a network (it has nodes and edges) but it is not interesting enough to influence the evaluation. Rather, the standard questions apply, e.g.

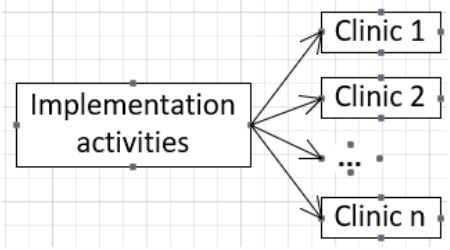


Figure 3 Clinic Deployment – Original Plan

- Is each clinic set up on schedule?
- Are clinics sufficiently staffed and equipped?
- What services and treatments do the clinics offer?
- Do potential patients actually visit their neighborhood clinic?
- What is the quality of medical care?

Funders and program designers originally assumed that each clinic would operate strictly on its own. But consider the model in practice, as it may well evolve over time (Figure 4). It assumes that even though the clinics were set up as stand-alone organizations, it is likely that a network will develop among them. They have come to trade expertise, staff in a pinch, and patient referrals. Unlike Figure 3, weError! Reference source not found. now have a network that is worthy of evaluation effort. Some of the new evaluation questions would be program related, i.e. the list presented for Figure 3 with a few additional questions such as:

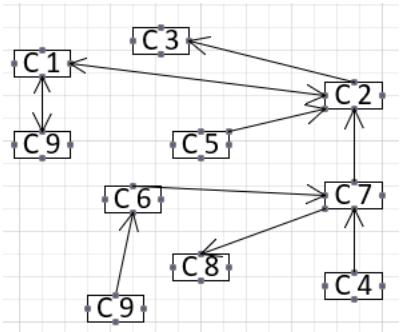


Figure 4 Network Structure Among Clinics as Relationships Evolve

- Does trading staff and expertise change the quality of care?
- Do patient referrals even out capacity limitations?

But in addition to these program-related questions, network logic could also be asked. For instance:

- How robust (or fragile) is the network?
- What are the network's centrality measures?
- Is the network structure stable or does it change frequently?
- How much has quality of care come to depend on the network?
- What was it about the clinic's operating environment that led to this specific network?

Again there is nothing wrong with either of these models, but they do present a different theoretical view of the program, one focusing on program logic and one on system logic.

Choosing What Model to Construct

Where does the system model come from? We construct it, just as we would any other model. We construct it to guide the inquiry that will best serve our needs. Consider Figure 5. The top model is the one we used in Example 4 (Figure 4). Its focus is on the cooperation network structure among clinics.

The bottom is a stock and flow extension of Example 2. The model assumes that implementation has been successful and depicts stock and flow behaviors from the regional patient population through to the capacity of each clinic. Both models are entirely reasonable and legitimate, but they guide different inquiry.

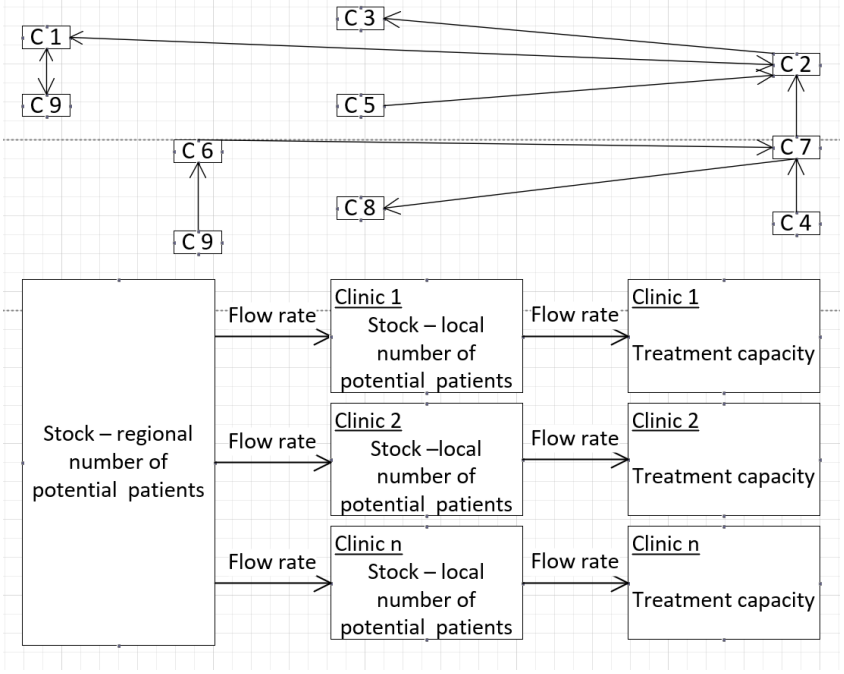


Figure 5: System Models: Network compared to Stock and Flow

Sometimes the System/Program Distinction Blurs

The program/system distinction advocated above is useful but not clean. There are scenarios where a program model contains a system model. For instance, consider Example 4. A clinic establishment program like that might well have cooperative

arrangements among the clinics as one of its objectives. In that case, the bottom model (systems view) would be part of the program model, perhaps as an outcome model at the far right. In fact, one might argue that blurring the program/system distinction is highly desirable because it means that funders and program designers recognize the system implications of their actions.

What are the touch points between system evaluation and program evaluation?

Given that systems can be evaluated on their own terms, as can programs, how does system evaluation relate to program evaluation? The answer can be discerned in all the scenarios presented above – causal chains (Figure 1) math scores (examples 2 and 3), and clinic deployment (example 4, Figure 3, and Figure 4). Table 1 summarizes system evaluation / program evaluation relationships. A theme running through the table is that the systems concepts involved apply to multitudes of settings but have specific messages for domain-specific efforts at effecting change.

Table 1: Example Questions Suggested by Program and System Views

Example	Location	Implications of System Logic for Program Logic
Causal chains	Example 1 Figure 1	It is one thing to collect data on whether outcomes appear and what causal paths are operating. It is something else to appreciate that patterns of causal relationships determine whether a program is fragile or robust. It is the system logic that speaks to this question. The systems implications are obvious in the simple case shown in Figure One. But imagine a model with ten elements and connections in many different directions, and including 1:1, 1:many, many:1 and many:many relationships; and with a variety of "AND", and "OR" relationships. In such cases, discerning the system logic, and determining its implications for the program, would require dedicated effort.
Math scores	Example 2, Figure 2.	Figure 2 shows a stock and flow logic that was abstracted from a program that was designed to improve kids' math scores. The program logic involved measuring teacher training and math scores. The system logic involved numbers of potential teachers and the rates at which they flowed through the system. Those stocks and flows would have consequential implications for staffing classrooms, and if teachers were not available, math scores for the entire population of students would not improve no matter how good the

Table 1: Example Questions Suggested by Program and System Views		
Example	Location	Implications of System Logic for Program Logic
		innovative math curriculum was. While the program logic and system logic are conceptually different in the example, they are close enough that a program evaluation would probably also consider stock and flow issues. But consider a more realistic model with flows across many elements, multiple flow rates, and multiple stocks. In such a scenario it would not be obvious what the implications of the system logic were for the program logic. For instance. Are two different stock levels on a critical path? Are there minimum or maximum flow rates that would break the system? These kinds of questions may have major implications for success paths in the program logic. Answers to these kinds of questions could be discerned, but only with dedicated effort that would not normally be part of a program evaluation.
Math scores	Example 3	Example 2 extends the math score scenario by incorporating two complex system behaviors – attractors and equilibria. It deals with the distribution rates for many aspects of the stock and flow model, e.g. numbers of teachers entering training, dropping out of training, or quitting once they began teaching. The focus is on the means, variances, and distribution shapes for these parameters. While these characteristics can be interpreted in statistical terms, there is also a systems logic lens that can be applied. That systems lens would interpret the joint distributions (for some or all of the parameters) in terms of the shape of the attractor states and the stability of the equilibria that characterize how the program lives in its environment. Looked at this way, the data can shed light on a program's sustainability and functionality in different settings. For instance, an unstable equilibrium that dwells in a shallow attractor space may not be sustainable, while a stable equilibrium in the same attractor space might have long-term staying power. In a sense, attractor spaces and equilibria can be thought of as contributing to the kind of knowledge that is sought in realist evaluation – what works, for whom, and under what circumstances (Pawson, 2013).

Table 1: Example Questions Suggested by Program and System Views		
Example	Location	Implications of System Logic for Program Logic
Clinic deployment	Example 4, Figure 4.	Any evaluation would look at a number of health-care specific measures, e.g.: 1) the operations and health care provided by each clinic, 2) network patterns among the clinics, and 3) the consequences of those network patterns for clinic viability and health care quality. But from a system point of view, one could also assess network parameters that have nothing to do with the specifics of the clinics' work, but which have consequences for the networked system of clinics. Some examples. Degree centrality is an indicator of the measure of the local influence that each node, aka clinic, has. Eigenvector centrality is an indicator of influence. Closeness centrality is an indicator of the alacrity with which a clinic can interact with all the others. In addition to centrality measures, there are characteristics of networks that speak to their robustness or fragility with respect to breakage of edges between nodes. Examples include distributed centrality and small-world structure.

What to Measure When Evaluating a System

Table 2 is a laundry list of characteristics of systems that can be measured. As in any inquiry, the fact that something can be measured does not mean that it should be measured. Again as with any inquiry, start with the model and an understanding of the work to be done, work from there in terms of what aspects of the model will yield good data and how hard it is to get that data.

This list represents judgement calls about what counts as a characteristic of a program model and what counts as a characteristic of a system model. For instance I categorized mental models and assumptions as program related and left them out of this discussion.

Table 2: Observables that Characterize Systems	
Rates and Magnitudes	
Distributions of magnitudes and rates.	All of the items in this table are susceptible to change over time. So rates and distributions become data. How frequent is an event? Is the distribution linear? Does it have an inflection point? Is there a discontinuous break? Is it a smooth curve with a fat tail? And so on. These data are important because rates and distributions are consequential parameters of system functioning.
Boundaries	
Where are the boundaries?	In any setting there will be many possibilities for setting system boundaries, i.e. for deciding what is in and what is out. Some of this determination is conceptual – what boundaries will help us understand what a program is doing, why, and what outcomes ensue? Some is practical – what data can we get, when can we get it, and how much will it cost?

Table 2: Observables that Characterize Systems	
Movement across boundaries	There is much talk in the evaluation literature about how systems interact with other systems. This is true, they do. But the question is how it relates to evaluating a system, not the program. One answer to this question is traffic. What other systems provide input into our system? How frequent is the input? How consequential? How much traffic is there from our system to others? A second answer is network structure for the traffic that is observed. Is our system central in the network? At the periphery? And so on.
Architecture and interior	
Network view	Because programs are comprised of entities and relationships among them, they can be understood as networks. As such one can look at program models with respect to measures of centrality, density, path length, and so on.
Logic model view	
Nesting	Systems are nested. One implication of this is that evaluators have to decide on the right scale for their work. Another implication is that decisions need to be made about whether nested systems should be included in the analysis. It may be necessary to do so, but caution is needed. For one thing, including a nested model assumes a causal relationship between the lower-level model and the one above it, That adds a methodological burden. But beyond that, it's risky to assume that subsystems roll up into higher level systems in a deterministic manner.
Characteristics of relationships – information that is almost always available.	The ability of a system to function is largely determined by the ways in which elements are connected. One could look at a conventional logic model and ask system questions such as: • How many relationships are characterized as 1:1, 1:many, many:1, and many:many?

Table 2: Observables that Characterize Systems	
	• How many critical paths are there? The more, the lower the likelihood that the program will succeed. • How many relationships are specified, but with undetermined consequences? For instance models often show inputs into a cloud of possible change, but without any specific change being specified. The greater the number of such relationships, the greater the possibilities for a host of complex system behaviors.
Characteristics of relationships – information that should be available but never is.	• When an element has multiple inputs, are those connections "and", "or", or "and/or"? The greater the number of "and" relationships, the lower the likelihood that the program will succeed. • How important is each relationship to the success of the program? The greater the number of important relationships, the lower the likelihood that the program will succeed. • How much confidence is there in each relationship? The lower the overall confidence in the system, the less likely the program is to succeed. If it were up to me, I would insist that program designers added this information, at least for critical parts of the model. I would also ask them to identify parts of the model where they truly did not know the answer.
Change	
Note: Any time during the lifecycle of a change effort can be considered the start from which measurement is taken, with both a forward-looking and backward-looking point of view.	
Predictable change	Many systems will change over time in planned, or at least predictable ways. Predicting a change is an aspect of program theory. The greater the number of changes built into the theory, the less likely that theory is to be proved correct. Complicated patterns have a way of not working out.
Unexpected change	Needless to say, there is plenty of unplanned and unanticipated change.
Regular, recurring change	Systems can change their structure and operation in regular, predictable ways, and then change back again. I.e., they go through cycles.

Table 2: Observables that Characterize Systems	
	One example I can think of is a system when it is providing routine services, and that same system at grant writing time. Frequency and period of change matter.
Stigmergent change	Stigmergy is > a mechanism of indirect coordination, through the environment, between agents or actions. The principle is that the trace left in the environment by an individual action stimulates the performance of a succeeding action by the same or different agent. Agents that respond to traces in the environment receive positive fitness benefits, reinforcing the likelihood of these behaviors becoming fixed within a population over time. > Stigmergy is a form of self-organization. It produces complex, seemingly intelligent structures, without need for any planning, control, or even direct communication between the agents. As such it supports efficient collaboration between extremely simple agents, who may lack memory or individual awareness of each other. (Wikipedia) Stigmergy is a fundamentally different change mechanism from the kind of planning and direction that we assume takes place at many different loci in a system. The content and amount of stigmergent change in a system says a lot about how it operates and evolves.
Resistance to change / sustainability of change One is the evil twin of the other. Which is which depends on your point of view.	
Lifetime	Sometimes a system maintains itself long after the reason for its existence goes away. (Assessing something like this is highly value loaded, but let's assume that reasonable criteria can be defined.) At other times systems go extinct before their work is done. (USAID comes to mind.) Lifetime can be measured.
Self-organization	"Self-organization" is a phenomenon in which, when a system is perturbed, it returns to its equilibrium state *without* information that comes from outside its boundaries. It is the systems' internal mechanisms that bring it back into equilibrium. The perturbation might come from external sources, but the

Table 2: Observables that Characterize Systems	
	mechanisms and processes that lead back to equilibrium are internal. If a system is observed over a long enough period of time, perturbations are sure to occur, which means that the dynamics of self-organization can be observed. Sometimes self-organization will bring the system back to equilibrium. Sometimes it will not. Either way, this is an important aspect of systems behavior.

References

Box, G. E. P. (1979). Robustness in the strategy of scientific model building. In R. L. Launer & G. N. Wilkinson (Eds.), *Robustness in Statistics* (pp. 201–236). Academic Press.

Frigg, R., & Hartmann, S. (2018). Models in Science. *The Stanford Encyclopedia of Philosophy*. https://plato.stanford.edu/archives/sum2018/entries/models-science/

Meadows, D. H. (2008). *Thinking in Systems:* . Chelsea Green

Morell, J. A. (2024). *How Does Knowledge of Complex System Behavior Help us do Better Planning and Evaluation?* . https://ecdan.org/session9-event/

Morell, J. A., & LeGros, T. (2025). Ecology and Evolutionary Biology's Unique Contribution to Evaluation. *Journal of Multidisciplinary Evaluation, 21*(49). https://journals.sfu.ca/jmde/index.php/jmde_1/article/view/1055

Pawson, R. (2013). *The Science of Evaluation: Realist Evaluation: A Realist Manifesto*. Sage

Rogers, K. (2012). Scientific Modeling. *Encyclopædia Britannica*. https://www.britannica.com/science/scientific-modeling

Williams, B.,
Hummelbrunner, Richard (2010). *Systems Concepts in Action: A Practitioner's Toolkit*. Stanford University Press