SYSTEM LANGUAGE: UNDERSTANDING SYSTEMS

George Mobus[1]
University of Washington Tacoma, Institute of Technology, 1900 Commerce St. Tacoma WA, 98402
Kevin Anderson
University of Washington Tacoma, Institute of Technology, 1900 Commerce St. Tacoma WA, 98402

ABSTRACT

Current languages for system modelling impose limitations on how a system is described. For example system dynamics languages (e.g. Stella) assume that the only concern in modelling a system is its dynamics which can be expressed in stocks, flows, and regulators only. A language for describing systems in a general framework provides guidance for the analysis of real systems as well as a way to construct models of those systems suitable for simulation. The language being developed, system language (SL) for lack of a catchier name, consists of:

- A set of lexical elements, terms that represent abstractions of components and entities that are found in all dynamic, complex systems to one extent or another - e.g. regulator, process, flow, boundaries, interfaces, etc.
- A syntax for constructing the structure of a system including:
    - describing the boundary and its conditions (including expansion of boundaries as needed)
    - describing the hierarchical network of connections and relations (e.g. system of systems)
    - describing interfaces and protocols for entities to exchange flows
    - describing the behaviour of elements in the system (e.g. functions)
    - providing specific identifiers naming the abstract lexical elements (e.g. electrical power flow)
    - providing a set of attributes appropriate to the nature of the element (e.g. voltage, amperage, etc.)
- A semantics that establishes patterns of connectivity and behaviour including:
    - distinction of material, energy, and message (communications) flows
    - laws of nature to be observed, e.g. conservation principles and second law of thermodynamics
    - imposes process-oriented abstraction on subsystems (similar to object-oriented modularization)
    - establishes rules for interfacing entities through flows
    - provides higher order organization and functions such as:
        - adaptivity (a capacity to vary behaviour in response to environmental changes within limits)

[1] Corresponding author: gmobus@uw.edu

- agency (an ability to make decisions – operational, logistical, tactical, or strategic in the framework of a hierarchical cybernetic governance system)
- evolvability (an ability to add or modify functionality either through foresight or by chance)

The language pragmatics is drawn from general systems theory as explicated in a set of principles of systems science. The language is formal and imposes rules of expression and construction that assure the resulting descriptions correspond with the nature of real systems in the world.

It is proposed that SL can capture the essence, structure, and dynamics of any real physical system. For example the first author has used this language to describe the human brain and its relation with the body and environment meta-system. The construction involves analysing the brain as a hierarchical cybernetic governance system (HCGS) that manages the internal operations of the body (operations and logistical management) and its near-term interactions with the environment (tactical management). The human brain has been additionally shown to provide strategic management (coordination with a world that might be in the future!) A very similar analysis has been applied to organizations and their management.

A recent survey of existing modelling languages has revealed only a limited capacity for these languages to support the SL framework. Existing languages generally support basic system dynamics and/or agent based (e.g. for explorations of emergent behaviour) approaches. Some are extensible through additional programming in languages like Java but do not directly address some of the more important features of SL. These results indicate that a new language should be developed to provide native support for SL directly. One immediate advantage of doing so is that the development approach can include support for massive multi-processing so that extremely large systems models can be developed. Modelling the human social system (HSS) would be an obvious target to help us understand our apparent predicament from a systems perspective.

Keywords: System modelling, formal systems science, system ontology, modelling language, system dynamics, agent-based modelling, language of thought, systemese

INTRODUCTION

Why can everyone with a healthy brain perceive, and agree among themselves, objects that have behaviours and interact with other objects in their environment? Why can people having different native natural languages learn to speak other languages and translate concepts from one language to another (not necessarily easily)? Many linguists and philosophers of mind assert that the human brain ‘speaks’ an internal, private language referred to as Language of Thought (LoT) or mentalese (Fodor, 1975; Pinker, 1997; Schneider, 2011). They consider LoT to be universal, having a basic ontology that is the basis of thinking, manipulation of LoT symbols in working memory, and through the attachment of somewhat arbitrary abstract symbols, words, the production of a public

language. The attachment of abstract symbols, which are themselves patterns in prefrontal cortex of the human brain, to the LoT ontological elements is accomplished through learning in the context of a cultural setting (native language).

To posit the existence of a common mentalese, however, is not sufficient to explain the universal aspect of human perceptions. What is needed is a clearer understanding of what the mentalese ontology really is. One of us (Mobus) in a forthcoming work has advanced the thesis that the mentalese ontology is actually *systems ontology*. That is, the human brain is predisposed architecturally to communicate within itself in what we will call 'systemese.' In other words, all humans think, subconsciously, in a systems-based LoT. Under the premise that the world is composed of systems of systems and that all systems share certain principles of organization, function, and behavioural outputs regardless of their material composition, it then follows that evolution would select for brain processing that allowed the formation of internal representations matching the world's nature.

Given this thesis, it makes sense then to ask what is this system ontology and in what form should we consider the lexicon, i.e. the symbols used in the language, and issues of syntax? The first case is one of finding what amounts to an 'upper ontology'[2] for systems, a set of words (e.g. in English) that map to a basic set of symbolic terms that are to be found universally in all languages and consist of patterns that conform to systems theory. The idea that there might exist such a common ontology has been explored by Anna Wierzbicka and Cliff Goddard (Wierzbicka & Goddard, 2014) in terms of a minimal set of semantics that are common across 'all' natural languages and are used as a base set of terms to define other terms in a language[3]. They have identified a set of terms, called "semantic primes," that they claim are common to all investigated languages (meaning that cross language interpretation is straightforward)[4].

Work such as this lends credence to the notion that there are a set of basic terms which can be used to describe fundamental concepts that are universal in human thinking (and thus in language symbols). The claim that we make is that there is a set of terms that map to the attributes of systems and that this set is the basis for human perception of what exists and works in the world. In this paper we will introduce an approach to identifying what those terms might be (in English) as an approach to systemese as mentalese with the idea of developing a universal language of systems that can be understood in any natural language and thus form the basis for rigorous systems thinking for everyone.

---

[2] Upper ontologies are explored in computer science, especially in domains of knowledge for the purposes of tagging on-line resources to improve searches within that domain. See Wikipedia article: https://en.wikipedia.org/wiki/Upper_ontology

[3] See the Wikipedia article: Natural Semantic Metalanguage https://www.griffith.edu.au/humanities-languages/school-humanities-languages-social-science/research/natural-semantic-metalanguage-homepage

[4] C.f. the Chart of Semantic Primes (in English) https://www.griffith.edu.au/__data/assets/pdf_file/0005/636890/NSM_Chart_ENGLISH_v16_05_2015_Greyscale.pdf

Language allows agents to share concepts. Natural languages evolved in human beings to provide a way for humans to share complex concepts about situations in their lives and projections about future situations (Deacon, 1997; Tomasello, 2014). Linguists have long analysed the forms and functions of human languages looking for commonalities among categories of languages (e.g. Aryan or Sino-Tibetan). There is some deep structure shared by all languages that allows humans from one culture to grasp concepts from another culture. We posit that there exists this *deepest* form of language that contains constructs (lexical primitives and syntax) that describes the world in a fundamental way and that that description is of systems (semantics).

There is another complementary approach to consider. By examining a wide variety of systems and system types to see what commonalities we might find we derive a set of principles that provide a framework for situating our language semantics (Mobus & Kalton, 2014, and see below). Guided by these principles and clues from psycholinguistics, neuroscience, and biological science one of us (Mobus) developed a formal structure, reported herein, for a definition of system. That, in turn, provides a means of identifying ontological elements along with the syntax required to develop a formal language of systems.

In this paper we review the basic concepts involved in developing a language that can be used to describe systems models so that a wider array of non-domain experts can appreciate the structures and functions as well as allow the simulation of the models to see their dynamics. We then introduce a formal structure definition of a general system which we claim provides the structure of system ontology – a framework in which the lexicon can be used to describe any general system and which captures the description in a formal structure that lends itself to computer simulation.

BACKGROUND

The Justification for a Language of System and its Need in Human Communications

A Universal Language that Describes Real Systems and is the Basis for Interpersonal Communications

Human beings share ideas about the world primarily through *symbolic* language[5]. The vast majority of this sharing comes through the naturally evolved forms of what we therefore call natural language.

Some of what we share is in the form of a formal language. Mathematics and logic are the premier examples. They have strict rules for the use of symbols (even the construction of symbols is rule based) and the meaning of conclusions arrived at by their rule-based

[5] This is as opposed to simpler indexical (pointing) and iconic (pictograms and pantomime) expression. Symbolic language has the additional property of being recursively constructed, i.e. sentences can contain sub-sentences or sentential phrases (c.f. Tomasello, 2014).

manipulations. Formal languages are unambiguous (if you know how to properly use the symbols).

A natural language, by contrast, only vaguely seems to have a formal structure. Some linguists argue for a universal innate grammar (Chomsky, 1965; Pinker, 1997) as a built-in (i.e. genetically determined) propensity that is actualized with experience in development in one's native culture. But the actual syntactical practices can obscure this built-in grammar. For example, the order of noun-verb-noun, or agent-action-patient can vary among different 'species' of languages. Certainly the sounds employed for various terms (words), including syllabic emphasis, tone, and prosody, vary considerably even within language families. Still there are patterns of similarity across natural languages. There are nouns, verbs and other categories of words that function similarly in all languages. This would suggest strongly that all human brains are wired to recognize and vocalize aspects of reality that are fundamental to human experience. Were it otherwise translations between natural languages would not be possible. Moreover, an evolutionary argument would conclude that since language has become our primary path to fitness (through cooperation within groups) it must be based on some framework that gives individuals a basis for common experience and expression otherwise we would not be here. Our assertion is that the structure of reality is systemness (Mobus & Kalton, 2014, Chapter 1). Thus the human brain is evolved to perceive and think systems. It is likely that the scope of systems thinking is still limited (Mobus, in preparation). People appear to perceive discrete objects as systems (simple and complex alike) even when they cannot directly perceive internal operations. They generally can perceive connections between objects that are contiguous in time and space (e.g. causal relations: Mobus, 1999). Many seem capable of finding more tenuous relations between objects, especially more complex objects, that are more distal from one another and the observer. For example most people grasp the relation between rain and plant growth from observing what happens in a drought compared with what happens in a wet season.

However, many more people have difficulty dealing with much more complex systems (like the economy) and their interrelations. They have difficulty perceiving all of the relevant objects (subsystems) and their interrelations as well as having difficulty projecting long-term complex phenomena (e.g. what happens when the population grows and the consumption of a fixed, finite energy source does likewise). Science, may have been "invented" to circumscribe the difficulties individuals have with perceiving systems of great complexity and time/space scope. But even the sciences have tended to restrict themselves to related sets of phenomena owning to the apparent need for individual scientists to specialize in order to make progress. Science has made considerable progress in its various disciplines when formal languages are used. The perception and communications of systems, though a natural aspect of human thinking could use the same approach.

A language that can capture the descriptions of system structures and functions seems obviously to require some formality. At the same time such a language needs to relate to natural languages in a way that allows any such language to express the nature of systems in a way that can be appreciated even by non-mathematicians. One of us (Mobus) has

argued elsewhere that the natural process of thinking (both conscious and sub- or pre-conscious) involves that of transitions of mental states (the holding of concepts in working memory) that conform to the syntax of systemese. This applies to the case of bottom-up construction from perception to conception and to top-down conception driving recall of perceptions (processing ideas in systemese). The temporal arrow of causality enforces the sequential symbol processing-like train of thinking and language production as things like the actor-action-patient relations are encoded by reinforcement of that ordering among systems. Neurons and neuronal networks are evolutionarily designed to capture just this relation (Mobus, 1994).

The claim is that systemese is the most fundamental language, applicable to all aspects of the sensible universe, and that the brain has evolved to use it as its private and subconscious mentalese. That, in turn, is the underlying architecture of natural and formal languages. This is the reason that human beings can grasp the organization of the world and share their conceptions with one another.

A System Language as a Guide to Analysis

Philosophers of mind may assert that the language one speaks (natively) guides and constrains the kinds of thoughts one can have. Indeed a strong claim is that the language affects perception itself.

A language based on systemness, if reasonably complete, should allow one to formulate thoughts about the world they observe that are true statements[6]. The language should help one think about the world forming veridical statements about it. Systemese should be a general purpose language that helps speakers formulate descriptions of the world that capture the truth of phenomena.

In particular the use of systemese to analyse systems helps analysts deconstruct the details of a complex system ensuring the capture of the essential elements, their relations, their dynamics, and their evolution over time. This follows from the nature of systems in that they are recursively complex, i.e. have internal structure comprised of subsystems that can themselves be similarly analysed. Sentences formed in all languages have this internal complexity and human thinking always seems to be curious about what must be inside of something that makes it work! With a formal structure forming the framework for valid statements in systemese, the thinking process is made explicit. The analyst is guided in forming questions about internal structures and relations and guided in discovering what those are. The collection of true statements about a system and its composition (as well as dynamics) constitutes a formal knowledgebase and model of the system.

Building Models

[6] When we say 'true' we will generally mean a 'fuzzy true', that is, a statement has a degree of truth or falsity in the general case. Statements in formal systems that admit to the excluded middle and other requirements of logics simply resort to the ordinary sense of truth (or false).

System Language

The synthesis in the systems approach is to construct models of the systems we have analysed. Mathematical and computational models allow us to simulate a system and vary the conditions (environment, internal variables, etc.) to see what happens to the system, its behavior, under those conditions and at some future time. These are what we call formal models and, as with all models, are necessarily[7] abstractions of the real systems. To construct these models requires formal languages, mathematics and computer programming languages, for example.

Human beings construct mental models of systems in the world. These are implemented in mental representations encoded in neuronal networks in the neocortex of the brain. These mental models are expressed in natural language, at least the "public" expressions are. But as mentioned above, the actual internal expression, which includes subconscious ones, is in terms of mentalese. As with formal models being abstractions of the real systems, mental models are also abstract. Here the reason is both that of computational capacity and of perceptual limits. When perceiving objects and actions in the world, the real system provides its own set of perceptual details as is cogent to the perceiver. But when recalling and exercising a mental model only the vague outlines of the objects and actions will be available, at least casually.

A language of systems, as developed below, provides a way to construct (express) models of real systems in both a formal way and in mentalese (quasi-formal). In both cases the veracity of the model (how true its predictions are) depends on the competency of the modeller but not on any constraints of the language.

Compiling the Knowledgebase

The use of a system language that contains symbols and syntax that reflect how the world is constructed and works, that is it has terms for all of the meaningful aspects of the world and rules for constructing true statements, provides the basis for constructing statements that capture that reality. Using it to guide deconstruction (analysis) helps to ensure the discovery of all relevant elements (if followed rigorously). Statements thus captured constitute a knowledgebase for the system of interest along with links to the entities in the environment that matter.

Simulation and Experimentation

With the formal model and the knowledgebase assembled, it is possible to compile those statements into executable computer codes. With these and sufficiently powerful computers it is possible to run faster than real-time simulations in which we can do experiments under varying conditions to make predictions (or at least anticipations) about the future behavior of the system should those conditions come to pass.

---

[7] The necessarily abstract, meaning losing some supposed unnecessary details, arises from the constraints of computational resources. No model ever quite replicates the details of the real system. What we seek is the replication of the essential behavior of the real system as seen in the dynamical changes in critical variables.

Even the mental models are useful; we call it thinking about the future. Mental models are tricky things. Most of our thinking (model processing) actually takes place subconsciously, and may have a degree of efficacy of which we are not completely aware (Mobus, in preparation). Our intuitions and judgements are largely affected by the operation of subconscious models (what is also called our "tacit" knowledge). Some people have better modelling capacity than others and so have unusually good intuitions. The most dramatic forms of subconscious models producing good results are those so-called aha experiences, sudden insights that seem to come out of nowhere but are actually the result of our subconscious mind working on the problem.

All of these "benefits" could be realized with the "right" language. To summarize, the overarching claim is that the Universe is a system of systems and that systemness (or systemicity) is a set of properties and conditions that can be identified by a science of systems. A consequence of this systemness is that living brains evolved to be able to model systemness in degrees of detail commensurate with their complexity (i.e. human brains can perceive more details about larger more complex systems than worms). That is brains and their capacity to represent the real world are a reflection of the very systemness of the world in which they evolved. The language of mental models is the language of systems. From an evolutionary point of view this results from the simple fact that the more competent a brain is in representing the systems it encounters in the world the more fit it is to survive and replicate.

What we seek is to identify and formalize that language and derive all of the above stated benefits. We are hardly the first. As mentioned above a number of researchers have sought a language of systems that could fulfil this role. However, we are not convinced that there has been an adequate language yet developed.

Examples of the Limits of Current Modelling Languages

The limits of modelling a complex system involving multiple kinds of flows with a language like system dynamics (SD) became apparent to one of us (Mobus) when he attempted to build a model of a complete energy production system based on a renewable source, the sun. Solar photovoltaic panels are touted as potential replacements for fossil fuel-based electrical generation. Solar energy is clearly renewable insofar as the time scale of human civilization is concerned. If it is possible to capture solar energy efficiently it is possible, in theory, to convert our economy to this "green" source of energy that is not a source of carbon pollution and global warming.

But every energy source that we employ must produce a net positive amount of power after taking into account the power consumed to extract/convert the energy from a "raw" source. Fossil fuels can be converted to considerable power via heat engines where the energy released through combustion (oxidation) is extraordinary and can drive massive machinery, for example electrical generators, to do useful work – the basis of our material economy. Such engines produce much more power than was consumed in the

process of extracting the fuels and refining them for use in the engines. Thus the net energy return is generally quite high[8].

The purpose of Mobus' modelling attempt was to consider the net energy production of solar photovoltaic production of electricity. Along with several graduate students he attempted to build models using the SD language Stella and it became clear very early on that the nature of the problem did not resolve easily in the language of SD. We were trying to model the flows of materials and energies separately and this required considerable effort to keep track of how both moved through the system (a manufacturing and installation system that used photovoltaic energy to run the operations while providing export of sufficient electricity to satisfy a social need). Because of the need to account for all forms of energy and material (e.g. emergy and exergy as well as raw inputs) the models soon became extraordinarily complex. But the real problem was an inability to represent processes as subsystems and abstract them so that their details could be ignored (until needed)[9].

The real problem with various modelling languages is that they *are* modelling languages! That is, they were developed for purposes within a domain of interest to model phenomena that the developers thought were important. For example DYNAMO[10], an SD-based language, was developed initially by Jay W. Forrester to answer questions about the dynamic properties of systems that were essentially free of adaptive or evolutionary capabilities. All agency in these models is covered by straightforward control equations. Some would argue this isn't even agency as it is understood in the agent-based modelling world. We will briefly address this in the section below on semantics.

---

[8] However, the work needed to extract resources that are increasingly difficult, i.e. deeper wells and mines, means that more energy is needed to recover each unit of energy for the economy. When oil was first being extracted, the energy return on energy invested (EROI) was roughly 100:1. The EROI on shale oil is estimated at < 15:1 and tar sands extract is even less. So even fossil fuels are providing less net energy to the economy today. Hall, et al (2012) have estimated that it takes an EROI of 20:1 to provide adequate net energy to society just to maintain our technological lifestyle.

[9] Subsequent to Mobus' efforts the SD-based modeling language Simile was released. It boasts an ability to encapsulate sub-models based on the same approach used in object-oriented computer languages (specifically Java). This would seem to answer one of the problems but one of us (Anderson) is still working on resolving the issue.

[10] c.f. https://en.wikipedia.org/wiki/DYNAMO_(programming_language)

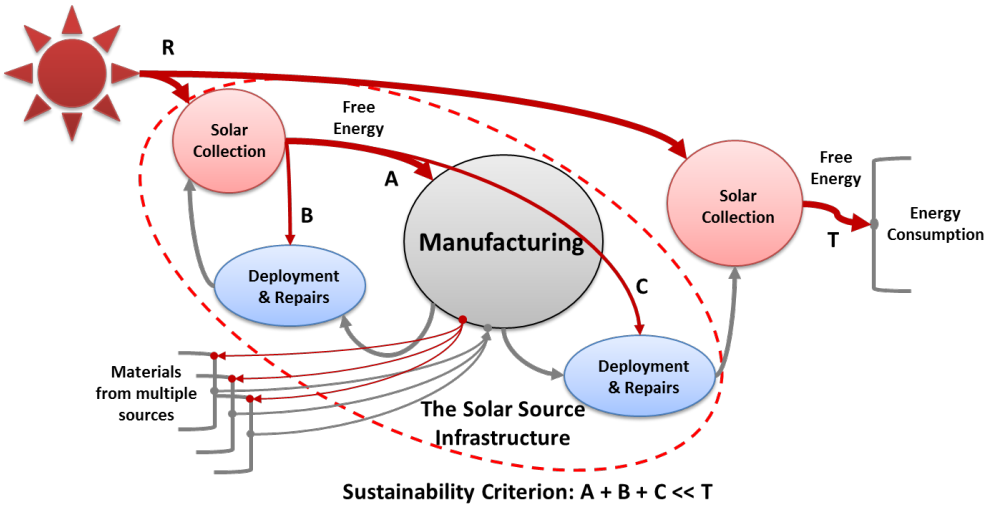

Figure 1. A model of a renewable energy production system requires encapsulating sub-processes. This model shows how a system language based on process semantics and differentiation of material/energy flows can simplify the design. Further deconstruction of the sub-processes (ovals) would explicate the complete model.

Modelling is an important aspect of systems science, unquestionably. But should a language developed just to do modelling also be used for other purposes in systems science? Specifically, should the language designed for modelling guide the deeper thinking about a system as a phenomenon? Borrowing from Abraham Maslow's Law of the Instrument, "…if the only tool you have is a hammer, to treat everything as if it were a nail,[11]" we assert that a language of systems must come from the principles of what makes systemness itself. The experience with trying to make SD work for an alternative energy system model acted as a wakeup call that the language of systems must be developed from the principles of systems science so as to be general enough to capture the essence of every kind of system. The questions a modeller might want to ask cannot be the basis for such a language. On the other hand, the language must allow modellers to ask the questions in which they are interested. Thus the approach to SL is to ask what is true for all systems in terms of components (lexicon), and how the components interact (syntax), and what does it all mean vis-à-vis systemness (semantics). Rather than invent a language that satisfies certain modelling requirements Mobus sought to find a language that expressed systemness itself and that could also serve to build models of systems.

Dynamics

Real systems are structured as hierarchies of subsystem and sub-subsystems, in which the lower-level systems often operate on shorter time scales than higher-level ones. For example the low-level operations of a manufacturing company work in real-time with time constants of minutes, whereas the accounting system gathers and summarizes data from operations on daily or even weekly time constants. Most SD-based languages for

[11] See the Wikipedia article: https://en.wikipedia.org/wiki/Law_of_the_instrument

modelling only support a single Δt (lowest-level) and so spend inordinate amounts of simulation time processing conditions at higher levels in the hierarchy that hardly change from time step to time step.

Agents

SD modelling languages do not directly support the construction of decision processes except through functions written as extensions. They do not have a built-in model of an agent so the modeller is forced to construct any agency by writing functions that emulate agency. Nor is there a way to create populations of agents, often used in social system modelling, though the general desire is to see what the dynamics of a population look like. Agent-based modelling languages are actually not much better since there is not a lot of agreement on what agency entails. Some support rule-based agent decision makers. Some provide a framework for implementing adaptive agents. But none have acknowledged the fact that there are perfectly good and consistent models of agents in the form of complex adaptive systems (CAS).

In part the problem stems from varying definitions of adaptivity itself. For example the literature is rife with conflation of adaptivity and evolvability. In part this can be attributed to the fact that biological evolutionist talk about adaptations in species referring to the fact that a *species* is "adapted" to a given environment, the econiche. But the word adaptive applies to the capacity of any individual in a species to alter its internal states in response to relatively minor changes in the environmental conditions, for example being able to adapt to lower temperatures by increasing shivering, or adapting to increased stresses by increasing response capacity (Mobus, 1994). Evolvability (and subsequent evolutionary process) speaks to the changes in the genotype that are reflected in the phenotype capacity to adapt. For example a variant on a gene that allows a possessor to adapt to an even lower temperature than its conspecifics would become more widespread if the average temperature went down so that the environment would select for individuals with this variant. A number of writers in the past have treated adaptivity as a kind of evolvability leading to the confusion between the two. The result is that many "adaptive" agents are actually the product of evolution as opposed to being merely adaptive.

Adaptive agents have to have a realistic capacity to learn (Mobus, 1999) based on their on-going experiences. Their decisions have to show modifications based on what they have learned and not be merely a result of a genetic algorithm.

Higher-order Models

Some languages support extension libraries for specific model constructions (such as the agent model discussed above). These are usually sub-model constructions that can be imported into a larger model. Some languages, such as Simile, follow the object-oriented language approach of 'inheritance' in which a base module can be inherited and modified as needed for particular functionality. However, most of these libraries address common

sub-models. For example a model may import a population growth model, modifying the parameters according to the actual population characteristics.

SYSTEM LANGUAGE

We now present an abbreviated description of SL, its theoretical basis and the fundamentals of its lexicon, syntax, semantics, and pragmatics. SL is being developed as part of a larger project to provide a holistic method for doing systems analysis. SL provides the guidance for constructing descriptions of analysed systems that are suitable for compiling to a computer model for simulation.

A Mathematical Structure Defining a System

A formal language is based on the existence of a formal structure into which elements of the language fit (see Pragmatics section below). For example a computer programming language is based on a formal structure involving arithmetic, logic, and conditional flow control (e.g. IF-THEN). These are realized in an actual computer architecture based on the theory of computation (e.g. the Turing Machine formalism and the von Neumann architecture).

The following definition is excerpted from Mobus (in preparation, 2). The development of this approach was inspired by Klir (1969). However, the purpose of this approach is to provide a structure for "holding" the details of a system description.

A system $S_{i,l}$ is an 8-tuple:

$$S_{i,l} = \langle C_{i,l}, N_{i,l}, Src_{i,l}, Snk_{i,l}, G_{i,l}, B_{i,l}, T_{i,l}, H_{i,l}, \Delta t_{i,l} \rangle \quad \text{(Eq. 1)}$$

where i and l are indexes. i is a subsystem index and l is the level of organization. Both are 0 for the initial system of interest.

C is a set of components along with membership functions in the event the set is fuzzy, i.e. the components may have partial inclusion.

$$C_{i,l} = \{(c_{i.1,l}, m_{i.1.l}), (c_{i.2,l}, m_{i.2.l}), (c_{i.3,l}, m_{i.3.l}), \cdots (c_{i.k,l}, m_{i.k.l}), \cdots (c_{i.n,l}, m_{i.n.l})\}_l \quad \text{(Eq. 2)}$$

is the set of components at level l and i is the component index from the level above (if any). The components of $C_{i,l}$, e.g. $(c_{i.k,l}, m_{i.k,l})$ use the dotted integer index that keeps track of the lineage of a component. That is, $i.k$ is the k^{th} component belonging to the i^{th} component in the level above (i.e. l-1). The $m_{i.k,l}$ are membership functions for fuzzy sets. A component might be a member of a given system only partially or only part of the time. If the set is crisp then all $m_{i.k,l}$ are equal to 1.

Components of a system may themselves be subsystems, i.e. having sufficient complexity to warrant further deconstruction. That is:

$$c_{i,l} = S_{i,l+1} \quad \text{(Eq. 3)}$$

is the i^{th} component treated as a new system of interest at the l+1 level. Equation 3 describes the recursive structure of system hierarchies.

The recursion cannot go on forever, obviously. What stops it? We have identified several stopping conditions, some semi-formal, others a matter of choice by the analysts. As an example of semi-formal stopping rules we use the simplest process rule. This means that a component, $c_{i,l}$, $l >> 1$, needs no further deconstruction because it is either merely combining two inputs to produce a single output (has a simple transformation function), or it is splitting one input into two outputs. This applies to material, energy, and messages alike. It also requires that there are no internal decision rules beyond the transformation function. A third simple component is a "raw" stock being used simply as a buffer and without regulating controls.

Informal stopping conditions include a judgment that the component's inner workings are already well known and specified outside of the system deconstruction. For example transistors do not need further deconstruction as components since their specifications are given. Similarly, an organic molecule in a biophysical system need not be further deconstructed. Figure 2 depicts a system deconstruction tree resulting from the recursion.

$$N_{i,l} = \langle C_{i,l}, E_{i,l} \rangle \quad \text{(Eq. 4)}$$

is a graph with vertices, $\left(c_{i.k,l}, m_{i.k.l}\right) \in C_{i,l}$, and directed edges, $(e_{i.k,l}, cap_{i.k,l}) \in E_{i,l}$. Edge $e_{i,k,l}$ is the vertex pairs, $(c_{i,k,l}, c_{i,o,l})$, where $k \neq o$ and the direction is assumed from k to o.

$cap_{i,k,l}$: $C_{i,l} \times C_{i,l} \rightarrow \mathbb{R}_{\infty}$, is a capacity function describing the flow rates. The actual function will be generally complex in that flows are usually fluctuating as a function of several different factors. Alternatively $cap_{i,k,l}$ may simply provide the max flow rate possible.

N is a flow network graph that describes the flows of material, energy, or messages (or directed forces) between all of the internal components of the SOI at a given level. In implementation N is an augmented graph, meaning that its labels are actually tables of attributes and values.

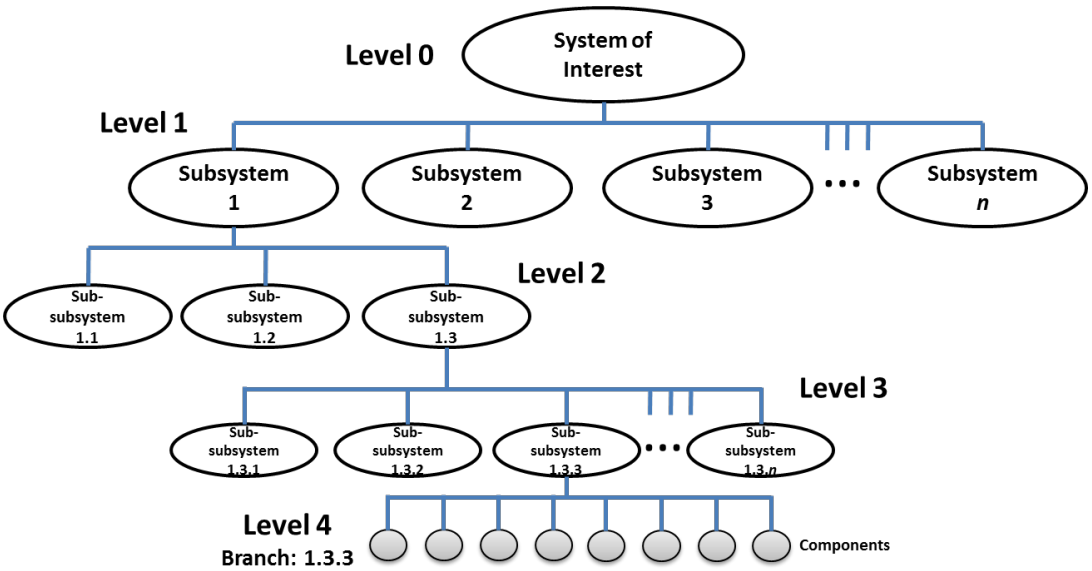

Figure 2. A system can be seen to be a tree structure of subsystem. Here subsystem 1 at level 1 has been deconstructed into three sub-subsystems (note the dotted numbers used for identification). Sub-subsystem 1.3 (level 2) is further deconstructed into a larger number of sub-sub-subsystems. One of these, 1.3.3, is further deconstructed into a set of components that do not need further deconstruction. The process of deconstruction is recursive, but can be pursued in either a depth-first or breadth-first manner.

G is a tri-partite flow graph defined as:

$$G_{i,l} = \langle (C'_{i,l}, Src_{i,l}), (C''_{i,l}, Snk_{i,l}), F_{i,l} \rangle \quad \text{(Eq. 5)}$$

where:
$C'_{i,l}, C''_{i,l} \subset C_{i,l}$, are the subsets of components within $\boldsymbol{C}_{i,l}$ that receive inputs from the source elements $\boldsymbol{e}_{i,k,l} \in \boldsymbol{Src}_{i,l}$ and send outputs to the sink elements $\boldsymbol{e}_{i,j,l} \in \boldsymbol{Snk}_{i,l}$ respectively. $\boldsymbol{F}_{i,l}$ is the set of directed flow edges as was the case for $\boldsymbol{N}$ above. Edges are of the form: $(f_{i.k,l}, cap_{i.k,l}) \in F_{i,l}$ ($\boldsymbol{cap}_{i,k,l}$ is the capacity function from above). Edge $\boldsymbol{f}_{i,k,l}$ is the vertex pair, $(\boldsymbol{e}_{i,k,l}, \boldsymbol{c}_{i,o,l})$, $\boldsymbol{e}_{i,k,l} \in \boldsymbol{Src}_{i,l}$ and $\boldsymbol{c}_{i,o,l} \in \boldsymbol{C'}_{i,l}$, or $(\boldsymbol{c}_{i,o,l}, \boldsymbol{e}_{i,k,l})$, $\boldsymbol{e}_{i,j,l} \in \boldsymbol{Snk}_{i,l}$ and $\boldsymbol{c}_{i,o,l} \in \boldsymbol{C''}_{i,l}$. As above, $\boldsymbol{k} \neq \boldsymbol{o}$ and the direction is assumed from $\boldsymbol{k}$ to $\boldsymbol{o}$. Nodes $\boldsymbol{e}_{i,k,l}$ specify those in the **e**nvironment (sources and sinks) relative to the SOI.

B is also a multi-set describing the boundary conditions of the system. For example ***B*** includes a listing of the *interface* objects that connect external sources and sinks to internal processes and stocks. An interface is a very special kind of sub-*process* that "penetrates" the boundary providing a channel for the passage of flows or a reception/production of a force. Interfaces often involve some kind of "protocol" or mutual signalling mechanism that allows sources, systems, and sinks to interact in a systemic way. Interfaces, being processes in their own rights, can involve complexities of

their own. What makes interfaces special is that they cross the boundary of the larger system and that of a sub-process that is generally involved with either acquiring an input resource or pushing out a product/waste.

The boundary, $\boldsymbol{B}$, at level l, then is a set, $\boldsymbol{P}$, of properties and a set of interfaces, $\boldsymbol{I}$. That is:

$$\boldsymbol{B}_{i,l} = \langle P_{i,l}, I_{i,l} \rangle \quad \text{(Eq. 6)}$$

where $\boldsymbol{P}$ is the set of properties and the second set is the set of interfaces. The exact form of P is still an object of research. At present it includes such properties as porosity (0 being completely non-porous) and "fuzziness", meaning the degree to which it is easily perceived (0 being able to identify and locate in space the separator between inside and outside).

By analogy with the set $\boldsymbol{C}$ above, the interfaces are components in a subsystem and themselves subsystems. That is, every $\boldsymbol{r}_{i,l} \in \boldsymbol{I}_{i,l}$ is an $\boldsymbol{S}$ itself but modified by a protocol object. That is:

$$r_{i,l} = \left(S_{i,l+1}, \varphi\right) \quad \text{(Eq. 7)}$$

There is a reason for treating interfaces as different from other system components due to their special role in crossing boundaries. Interfaces do not typically alter the flow, i.e. do not transform the substances as a process does to create products. In the case of interfaces as subsystems, the ϕ parameter is called a protocol which is an algorithm for letting the flow across the boundary in an ordered fashion.

The sets $\boldsymbol{T}$ and $\boldsymbol{H}$ are more difficult to describe within the scope of this paper.

$\boldsymbol{T}$ is the set of transformation rules for the subsystems in $\boldsymbol{S}$. That is, for each $\boldsymbol{c}_{i,l} \in \boldsymbol{C}_{i,l}$ there is a formula, $\boldsymbol{t}_{i,l}$, that describes the transfer function of that component for transforming inputs to outputs. These may be expressed in any suitable form, such as ODEs or computer codes.

$\boldsymbol{H}$ is a super complex object that records the history of the system, or its record of state transitions, especially as it develops or evolves. For example, brains learn from experience and as such their internal micro-structures change over time. This is called memory and the current state of $\boldsymbol{T}$ is based on all previous states. Some simple systems, like atoms for example, may have a NULL $\boldsymbol{H}$; that is there is no memory of past states. As just mentioned, on the other hand, brains (and indeed all biological systems) have very rich memories. $\boldsymbol{H}$ is an augment for $\boldsymbol{T}$ and all variables associated with elements in $\boldsymbol{N}$ and $\boldsymbol{G}$. The most succinct explanation of $\boldsymbol{H}$ is that it is the time series data of all state variables of the system averaged over some appropriate time window. This too is an area of research to pursue. The best model for $\boldsymbol{H}$ would be the human brain, particularly the neocortex, where memories are encoded, stored, and retrieved for use.

Finally, the last element in ***S*** is Δt, a time constant relevant to the level of the system of interest. In general, higher levels in the hierarchy of organization have larger time constants; the activities take longer than those at lower levels. Δt is generally an integer multiple of the lowest level time constant. In discrete time simulation it is the time step over which the model of that level is computed.

Figure 3 shows graphically what a system definition entails.

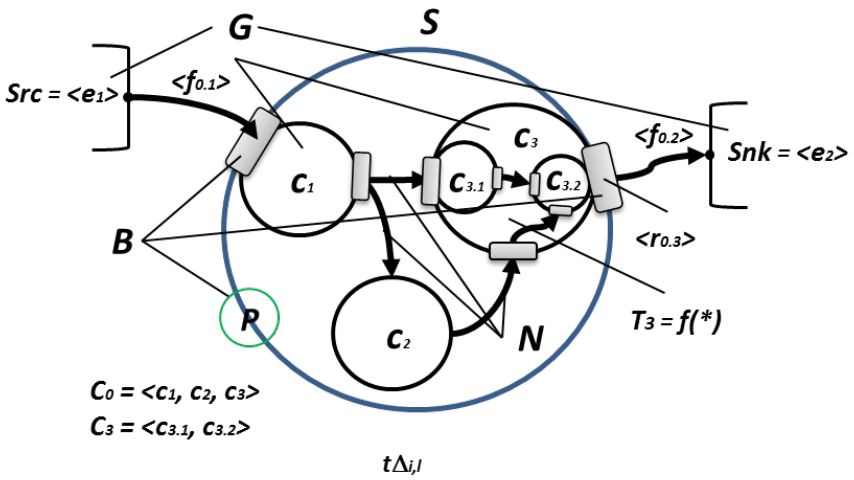

Figure 3. A graphical representation of a system formal definition shows the elements of each of the members of the *S* 8-tuple (except *H*). *C* is the set of components. *N* is the internal map of the flows (connections) between components. *G* is the network of flows (connections) between external entities and the components with which they are connected. B is a set that contains descriptors of the interfaces through which flows pass and of the boundary itself, e.g. "porous". *T* is the set of transformation formulas.

By having a formal definition such as this it is now possible to apply a great many mathematical tools to analyse many aspects of the system.

Lexicon

The fundamental symbols in the language constitute the lexicon. As with natural languages this is the set of words that are used to construct sentences in the language. These sentences describe the structures and functions of systems. They do so in a hierarchical fashion reflecting the fact that subsystems are component parts of systems and vice-versa, systems are components in larger super-systems. In the system language the frame of reference for all sentences is the system of interest (SOI) and its immediate environment of interacting sources and sinks. Figure 4 shows a set of lexical elements in SL. In the sections below on syntax and semantics we provide some examples of constructs in the language. This set of lexical elements includes primitives and derived elements. The primitive elements are sources/sinks (elements that are part of the environment, processes/systems, flows, and stocks. Some of these elements were adopted by Forrester as part of the SD language, namely stocks, flows, sources/sinks. Some are

adapted from Odum's energy language, e.g. processes like producers and consumers (Odum, 2007). SD also models information flows as separate from other flows and its use in regulating flows. In SL we consider message flows (which may or may not communicate information, Mobus & Kalton, 2014, Chapter 7) as in the same way as material and energy flows.

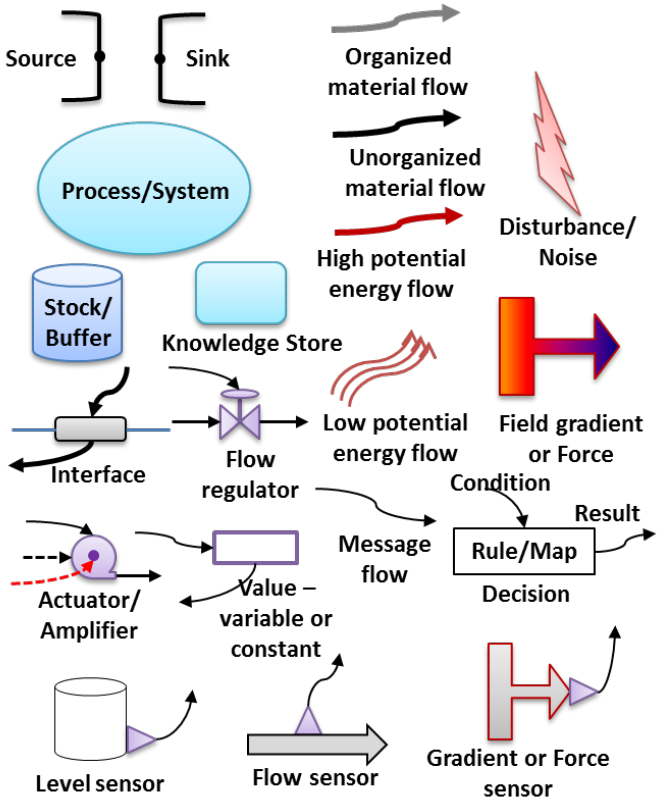

Figure 4. These are a number of iconic representations of elements forming the lexicon of the system language.

As can be seen in figure 4 the lexicon of SL is much richer than in most implementations of system dynamics or the energy language. One of the requirements of SL is that the flows and stocks of "stuff" are differentiated into material, energy, and messages. This is because each of these kinds of stuff obey different rules when it comes to how they move through the system. Material may degrade to garbage but all of the molecules are accounted for throughout the flow (conservation rule). Energy, on the other hand, degrades to a non-usable form, waste heat, as work is done in the various transformations that occur throughout the system (second law of thermodynamics rule). This differentiation figured prominently in the work of Odum (2007, see Semantics below) but is not taken into account in SD or most agent-based languages.

Many of these elements, e.g. an actuator, are derived from a more primitive element (i.e. process) in much the same way that more specialized objects are derived from more general objects in object-oriented languages like C++. Forrester's selection of lexical

elements to support SD models was inspired by his seeking a way to demonstrate physical systems dynamics. Constrained by computer memory sizes and speeds, the idea to model using the most primitive constructs, putting emphasis on the use of difference equations and time steps, allowed for the modelling of reasonably complex systems insofar as their dynamic behavior was concerned. This approach has dominated the field ever since.

The objective of SL is to expand the lexicon to include elements that have special behaviours. Figure 5 shows this derivation for a few of the lexical elements considered. Starting with the same four primitives listed above, we derive special elements for three of them (we do not derive special versions of sources and sinks since these are technically un-modelled[12] and outside the boundary of an SOI).

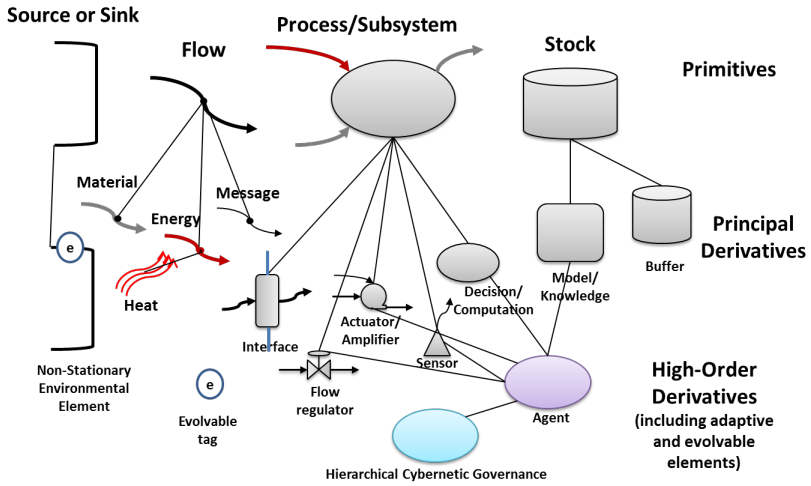

Figure 5. The lexical elements in SL are derived from four basic primitive types, sources/sinks, flows (material, energy, and messages), process, and stocks.

The behaviours of the derived elements are built into the language. For example the behaviour of a flow regulator is given as a process that receives messages that cause the flow to be increased or restricted dependent on the pressure differential between the inlet and outlet sides. Examples are actual valves, variable resistors, and interference RNA molecules (RNAi) that influence the expression of genes by acting on messenger RNA (mRNA). The precise influence of the regulator is contained within its transformation formula (the $t_{ij} \in T$ from above.) The modeller need only designate the use of a regulator object (on a flow) and then specify its transformation formula (also called the transfer function). The base process receives a flow (of some kind) and a message input. Then, according to the formula, the message opens or closes down the flow rate. Both linear and nonlinear transfer functions can be implemented.

[12] By un-modelled we mean that these elements are assumed to be systems/processes in their own rights. However the modeler does not have access to the details of how they work. The only characteristics provided for modeling purposes is the out/in-flows from these objects.

The decision/computation process is a special case of a process that takes multiple message inputs and computes an output message. This can be as simple as a transformation formula, an algorithm, or a complex agent decision requiring neural-like (or statistical) decision processing (Mobus & Kalton, 2014, see chapter 8).

These lexical elements are English terms that represent primitive systemese concepts. In SL provisions are made for these terms to be used as function types and specific public language names to be used in representations. For example a process object is given a name such as "Producer" so that the meaning of the term is understood in the context of the actual system being constructed. The underlying code of "Producer" need not concern the modeller at an abstract level but is implemented as a process with an associated transfer function.

Syntax

The syntax of SL gives rules for the construction of descriptions of both structures and functions. These constructions conform to the formal definition of systems above and are captured in a database, the structure of which, models the definition. The SL support tools should not allow the simulation model to be compiled until these rules are all met.

For example, the language requires that a flow have a source and a sink. These can either be environmental sources/sinks, which are minimally modelled (as explained in footnote 5), or a stock, or other processes. The rules of connectivity ensure that a source for energy is not hooked up to a material interface to a process, a stock, or a sink. Figure 6 shows a first-step model of a system of interest after identification of all of the relevant environmental sources and sinks, along with the flows (or forces) associated with them. Each object in the figure (which corresponds to a graphical frontend for SL) has a K element, an augment table containing names, system id numbers, and other attributes that are relevant to the type of object (e.g. flow rates in appropriate units for the time constant at this level).

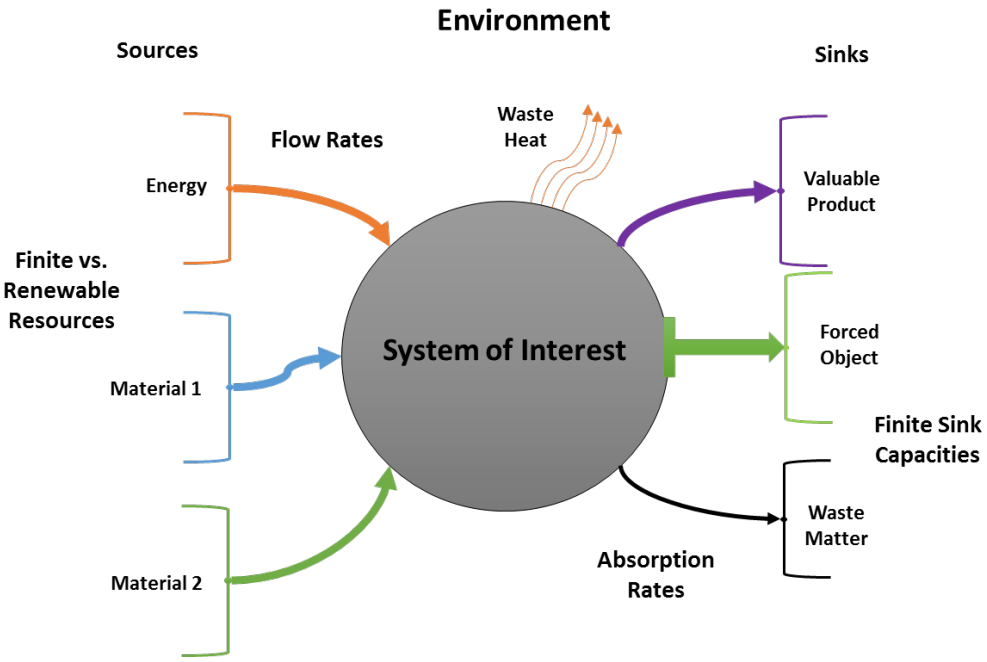

Figure 6. A system is identified by virtue of its inputs and outputs. Here the SOI is opaque as the starting point of a systems analysis using SL.

Even at this level of deconstruction the language compiler will check these attributes to ensure that what goes into the system is accounted for in what comes out (in the long run).

Figure 7 provides a "cartoon" depiction of the next level of deconstruction (when ***l*** goes to 1).

The figure shows the mapping of elements of a system in which several stocks are employed. The work processes are still abstractions, that is they are subsystems that are still opaque and will be deconstructed at the next level down (***l*** = 2). The process identified as a "coordinator" is an information processing and decision making process or agent, described below in the section on semantics.

Again, as above, all of these elements are coded and fit into the definition-based database. Sub-processes are coded with a dotted numbering scheme that provides ready indexing into the database. For example the starting SOI is given an id of S0 (process zero). The five sub-processes shown in the above figure would be numbered S0.1, S0.2, S0.3, etc. Similarly the stocks would be coded S0.Stk1 and S0.Stk2. The same strategy applies to all elements within the transparent view of the SOI.

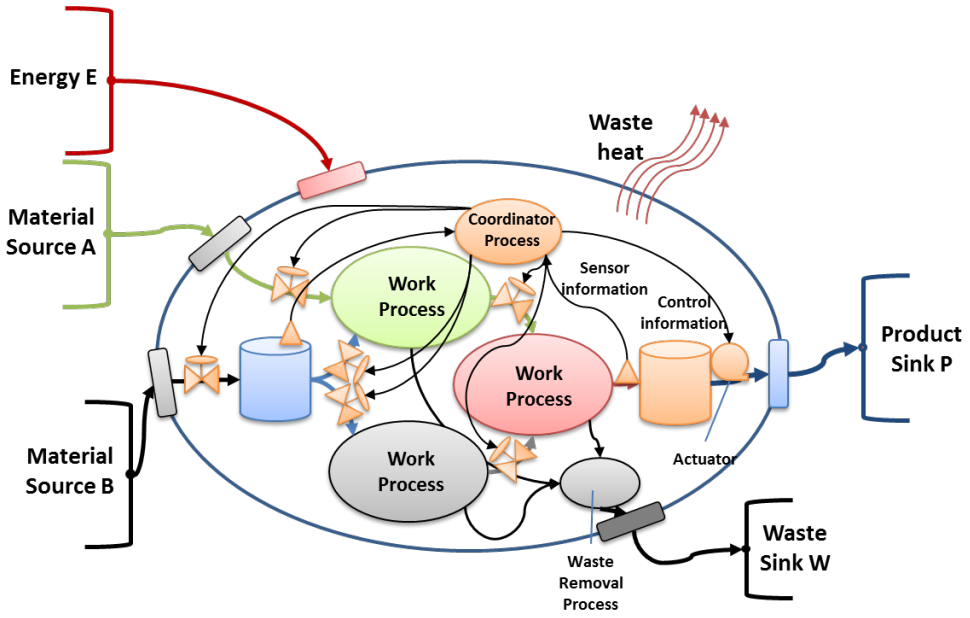

Figure 7. Deconstruction of the SOI in figure 6 (partial) shows how the lexical elements of SL are used to construct the structural description of the internals (transparent view). Note the use of interfaces receiving and emitting flows to/from the SOI. Internal energy distribution is not shown to avoid clutter, however there is a subsystem that captures and distributes energy to all of the work processes, similar to that shown in figure 1.

Representations and Simulation Translation

The graphical representations of a system, as in figure 7, provides a user-friendly interface in which systems analysts and modellers will be able to draw the system. This graphical front-end has two modes. The construction mode allows analysts/modellers to capture the structure and define the functions of elements of a system. The simulation mode provides an animation of the system as it is being simulated by the simulation engine.

The graphic representation captured in the first mode is translated to the database representation (based on the above formal definition) and a related XML markup language representation (for portability between platforms). The simulation engine extracts model information from the database and writes simulation results (dynamics traces and system state snapshots) to XML files that can be further processed for data analysis.

A third representation provides a human readable text file that makes programming-like statements in SL. This file can also be translated to the XML/database format for loading into the simulation engine. Figure 8 shows a sample of statements in SL that establish the environment for a system of interest.

One of us (Anderson) has developed a proof-of-concept simulation engine and XML/database translation model using a subset of SL. We intend to release a codebase in an open source project as soon as the full specification for SL is completed[13].

```
// Example:
 SOI0 mySystemOfInterest :
        ENV myEnvironment :
              SRC Source1 : SRC0.1, MATERIAL, property1, 33.3, 123.3, 0.1, POUNDS, push_material;
              SRC Source2 : SRC0.2, ENERGY, property2, 300, 350, 0.0, BTU, push_energy;

              SNK Sink1 : SNK0.1, MATERIAL, property1, 25.1, 27, 0.0, POUNDS, accept_product;
              SNK Sink2 : SNK0.2, MATERIAL, property2, 8.2, 98, 100.0, POUNDS, accept_waste;
              SNK Sink3 : SNK0.3, ENERGY, GLOBAL.heat_sink, NULL, NULL, NULL, NULL, GLOBAL.accept_heat;
//                            Sink3 is the global heat sink defined in a global file. The properties are set to NULL because the
//                            global properties apply. The modeler may define other global properties in the global file.
```

Figure 8. An example of SL statements shows the definitions of a system environment, capturing the characteristics of various sources and sinks. The list of parameters pertain to the object definitions as related to the formal definition of system given above, and entered into the database.

Semantics

A major difference between SD as an example language and SL is that in the latter the key semantics (see below) form around the concept of a *process*. All systems are processes and vice versa (Mobus & Kalton, 2014). Thus the blue oval in figure 4 acts to contain a system comprised of the other elements in the language. An example is shown in figure 7.

The process semantics is similar to H. T. Odum's energy/material circuit language (Odum, 2007, see Chapter 2). Odum also identified the need to treat energy and material flows separately including the rules pertaining to those flows. Producers were his processes that brought energy and material together to produce a product that would get consumed downstream. Consumers are, of course, also work processes that consume energy in the process. In SL a process is treated as a primitive concept. It is a point at which materials, energies, and messages (which generally convey information) combine or dismantle the inputs to transform them into usable outputs. The details of the transformation are contained in the ***T*** set in Eq. 1. Many additional elements as shown in figure 5 are derived from the process primitive. That is predefined transforms are provided due to their importance in developing descriptions of very complex systems.

The semantics of SL derive from the constructions built out of the lexical elements in syntactically correct form. Process semantics basically say that a system performs internal work on inputs to produce outputs with the language support making sure that all inputs and outputs are accounted for physically (volume, weight, and time). Eventually the internals of a process at the lowest level of deconstruction specify the actual work that

[13] By "full specification" we mean full enough to produce a complete model. We expect SL will be extensible in the same way other programming languages have been. The specification itself is extensible to allow for the fact that we probably have not quite defined a complete language.

gets done on the inputs. For example a process can be a "combiner", it takes several inputs and combines them into a product with the appropriate amount of exergy and heat loss (similar to Odum's producers). Another process is a splitter, taking a complex material input and splitting it into several material outputs, using an appropriate exergy input and heat output (as with Odum's consumers). In addition to work processes (on materials and energies) there are computational-decision making processes that use models (e.g. computer programs or decision trees), take input as data from sensors, and produce messages as outputs to regulators or other processes.

SL has the expressive power to describe any real system since all systems are composed of the structures and functions contained in the lexicon and syntax. We do not claim that SL is yet complete in its current state; there may be found some system elements that we have not yet considered. However, the language has an extensible lexicon and syntax to allow additions (as in figure 5). All languages, formal and natural, evolve over time so it might be expected of SL as well.

What about things that are often treated as systems but are not, at first glance, physical embodiments? For example, one might consider a system of concepts, such as mathematics and claim that such a system is not compatible with the definition or SL. Our position is this. Conceptual systems are still real enough in the sense that their only real existence is in the brains of those who think about them. Mathematics comes to life in the minds of mathematicians, not in the textbooks. The latter are only temporary recordings of the state of mathematical systems for transmission to other minds. The systems themselves are embodied in the neural networks of mathematical thinkers. These networks are every bit as physical as the ordinary notion of systems as we have been discussing. Conceptual systems are real systems in every way that fits our definition. The fact that such systems can be copied into new brains makes them as subject to evolutionary process as any other physical system. The explication of conceptual systems in brain tissues (neocortex) is described in Mobus & Kalton (2014) in asides called "Think Boxes."

Pragmatics

The context of SL is the nature of systems as embodied in a set of principles (Mobus & Kalton, 2014). Table 1 provides a brief of these principles as described in the reference. The formal definition of system given in the section above attempts to incorporate all of these principles. Simple systems may only embody the first several principles. Complex, adaptive and evolvable systems are covered by the full set. Additionally it should be noted that there is no claim that the set is exhaustive. Indeed Mobus and Kalton (2014) have identified several sub-principles in several instances.

Table 1. Principles of Systems Science (from Mobus & Kalton, 2014). Highlighted terms provide reference to the main features of the principle. Note that many principles interrelate.

1. ***Systemness***: Bounded networks of relations among parts constitute a holistic unit. Systems *interact* with other systems, forming yet larger systems. The universe is composed of systems of systems.
2. Systems are ***processes*** organized in *structural and functional hierarchies.*
3. Systems are themselves, and can be represented abstractly as, ***networks of relations*** between components.
4. Systems are ***dynamic*** on multiple time scales.
5. Systems exhibit various kinds and levels of ***complexity***.
6. Systems ***evolve*** to accommodate long-term changes in their environments.
7. Systems encode ***knowledge*** and receive and send ***information***.
8. Systems have ***governance*** subsystems to achieve stability.
9. Systems contain ***models*** of other systems (e.g. simple built-in protocols for interaction with other systems and up to complex anticipatory models).
10. Sufficiently *complex, adaptive* systems can contain ***self models***.
11. Systems can be ***understood*** (a corollary of #9) – Science.
12. Systems can be ***improved*** (a corollary of #6) – Engineering.

Figure 8 shows a relational map of the principles relative to several overarching aspects.

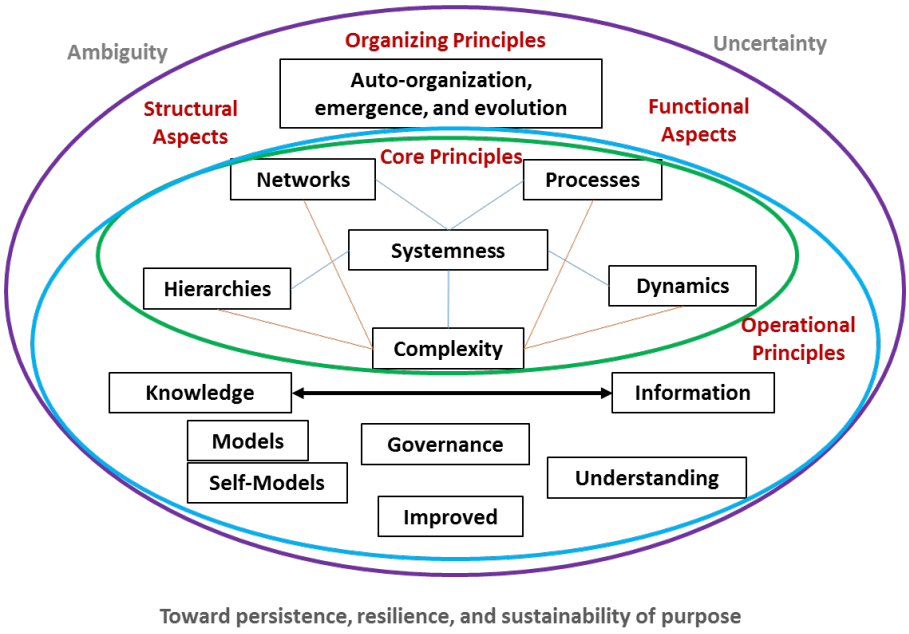

Figure 8. The principles for Table 1 are organized according to overarching aspects.

The green oval in the figure show the core principles that apply to all systems, even simple ones. The blue oval includes those plus operational principles, those that include higher-order patterns that derive from the core principles and lead to systems that have duration and resilience. The outer, purple, oval includes the major organizing principles of auto-organization, emergence, and evolution. The left side of the figure addresses structural aspects of organization while the right side addresses functional aspects. This means that principles, such as knowledge (epistemological aspects), hierarchies, and networks are primarily concerned with how systems are structurally organized. Principles such as dynamics, processes, and information are concerned with the functions of systems, with how they behave and operate.

SL embodies these principles either directly or indirectly. The inner oval, the core principles, are directly embodied in the lexicon and syntax. The operational principles are supported in built-in models, for example of agents. The organizing principles are supported by lexical, syntactic, and semantic aspects but require a working simulation environment to be realized. For example, any lexical element in the model can be "tagged" as "evolvable", meaning that during a simulation run that element's characteristics might be changed according to a probabilistic model, e.g. a flow rate might be increased or decreased within an allowed probability distribution during the simulation run. This applies to both the system of interest internals and to the environmental entities. In other words, the environment may be scheduled to change (be non-stationary) in order to test either adaptability of the system or its evolutionary competence. Currently if a modeller wants to change a model in response to a changed environment it is necessary to reconfigure the model and recompile it for simulation. There is no on-the-fly way to test system responses to changing environments.

DISCUSSION

The purpose of a system language is to provide a way to communicate, meaningfully, the nature of any kind of system regardless of the domain of interest. The system language provides a lexicon and syntax that links to actual objects and actions in any kind of system. The semantics describe systemness in all of these domains.

The approach taken here is to provide a language that captures the formal structure of systemness while allowing ordinary people (non-mathematicians) to express their thoughts (systemese) in a public way. We envision a graphical frontend to the analysis tool which will allow anyone who is familiar with the workings of a system to 'draw' a map of it using lexical icons. This tool will check the syntax on-the-fly so as to make sure the model is syntactically correct. It is still up to the modeller to make sure the system is correct.

If the hypothesis that everything in the Universe is a system, and/or a component in a system is valid then every such system should share common properties in structure and function. The system language is an attempt to capture these commonalities in a highly generalized set of terms and rules of expression. This, in turn, provides a kind of language primitive that might apply to all areas of human knowledge.

Systems biology, chemistry, sociology, etc. represent evolution within those fields toward this idea of a fundamental description of nature in all of these domains. By the end of the twentieth century researchers and administrators of research had become aware of the importance of conducting inquiry on complex phenomena from multiple disciplinary perspectives. An initial response involved interdisciplinary approaches, bringing researchers from multiple disciplines together in hopes of understanding phenomena that involved aspects of these disciplines. The results have been mixed. The notion of bringing disparate disciplinarians together proved somewhat problematic. The more distant the disciplines, or in other words, the more specialized the disciplines, the greater the difficulty the participants have had in understanding each other's specialized language. The reliance on such special languages put up communications barriers. Through the first decade of the twenty-first century attempts have been made to find the right way to mix disciplines in order to tackle the increasingly complex phenomena that desperately need understanding. Interdisciplinarity transformed into multi-disciplinarity, cross-disciplinarity, and other variants, still not achieving the ideal of fluid translation between disciplinary languages.

Transdisciplinarity (Rousseau & Wilby, 2014) is the latest and most promising attempt to provide a means for disciplinary researchers to talk to each other and be understood. In the model presented by Rousseau and Wilby transdisciplinarity is achieved by there existing a common language of description of scientific phenomena which can be spoken by researchers from any discipline with a means of translation to other disciplines. They propose a language based on systems principles as a candidate approach. In this paper we have proposed SL as just such a language.

CONCLUSION and FUTURE WORK

We are entering unexplored territory in framing a language of system in a formal definition. Klir (1969) provided an insightful start toward such a definition. System modellers such as Forrester and Odum developed languages for immediate purposes based on their domains of interest rather than starting explicitly from a general systems theory. Both identified very important system modelling concepts and the union of their lexical sets does indeed cover the basis of an atomic lexicon. But the route by which either came to select the particular elements they did appears largely determined by their individual conceptions of systems in their domains of interest.

With SL (systemese) we are attempting to start from system theory and identify the elements, lexicon, syntax, and semantics based on the pragmatics of systems principles. Those principles, we claim, apply to all systems as shown above (in Table 1 and Figure 8). Based on those principles we have developed the formal structure definition with process semantics, including recursive structure that recognizes the hierarchical nature of systems and their subsystems, and derived additional lexical elements using rules of variation on the atomic elements. This approach is in concert with object-oriented programming languages such as C++ where generalized classes are defined and more

specialized classes are derived from them, adding special functionality to the general functions.

The central claim of this paper is that SL will provide much greater ability to be used in analysis and design of complex systems of all sorts. If we got the ontology right it is a universal language that will allow disciplinarians to speak to one another across domains since any descriptions of models of domain-specific systems can be translated into SL. The language can further be translated into a simulation model.

We invite feedback on the concepts covered in this paper. We are currently planning to advance our proof-of-concept simulation engine to a prototype suitable for open-source development. The visual frontend tool for supporting systems analysis and design, as well as providing a simulation animation tool, needs to be researched and developed. The intermediate representations and database designs follow from the maturity of the engine. Our plan is to work with researchers in specific domains, such as organizational entities and biologists, to develop models of complex systems from those domains to test our platform.

REFERENCES

Chomsky, N. (1965). *Aspects of the Theory of Syntax*, The MIT Press, Cambridge MA.

Deacon, T. (1997). *The Symbolic Species: The Co-evolution of Language and the Brain*, W. W. Norton & Company, New York.

Fodor, J.A. (1975). *The Language of Thought*, Harvard University Press, Cambridge MA.

Klir, G. (1969). *An Approach to General Systems Theory*, Van Nostrand Reinhold, New York.

Mobus, G.E. (in preparation, 1). *A Theory of Sapience: Using Systems Science to Understand the Nature of Wisdom and the Human Mind*, draft available upon request.

Mobus, G.E. (in preparation, 2). *Understanding Systems: Systems Science, Analysis, Modelling, and Design*, Springer (under consideration), New York.

Odum, H.T. (2007). 2007, *Environment, Power and Society for the Twenty-First Century: The Hierarchy of Energy*, with Mark T. Brown, Columbia University Press, New York.

Pinker, S. (1997). *How the Mind Works*, Norton, New York.

Rousseau, D., Billingham, J., Jennifer Wilby, J., and Blachfellner, S. (2016). "In Search of General Systems Theory, *Systema* 4(1) : 76-99, www.systema-journal.org.

Singer, J., Sillitto, H., Bendz, J., Chroust, G., Hybertson, D., Lawson, H., Martin, J., Martin, R., Singer, M. & Takaku, T. (2012), The *Systems Praxis Framework*, *in* 'Systems and Science at Crossroads – Sixteenth IFSR Conversation', SEA-SR-32, Inst. f. Systems Engineering and Automation, Johannes Kepler University, Linz, Austria, pp. 89–90. URL: http://www.ifsr.org/index.php/category/archives/ifsr-conversations

Schneider, S. (2011). *The Language of Thought: A New Philosophical Direction*, The MIT Press, Cambridge MA.

Tomasello, M. (2014). *A Natural History of Human Thinking*, Harvard University Press, Cambridge MA.

Wierzbicka A. & Goddard, C. (2014). *Words and Meanings: Lexical Semantics Across Domains, Languages and Cultures*. Oxford: Oxford University Press.