archetype includes the concept of putting something in, containing for a time, and then taking something out (inflows, stocks, and outflows!).

In sum, then, systemese consists of a set of innate archetype concepts (models) that evolved to reflect the systems (and components of systems) that exist in the environments of animals with which they have to interact. These concepts come with built-in syntactical rules for combination that preserve the semantics and pragmatics of the situations in the world. All animals with brains of any complexity "think" in their version of systemese, and this is not a conscious process. Humans, on the other hand, have an extraordinarily complex environment—complex systems with which to interact—and accordingly have a much more sophisticated set of innate concepts with which to construct more complex thoughts. Moreover, humans possess a supervening concept encoding mechanism that associates specific auditory patterns (speech) with complex constructions, and these become the names given to things, relations, and actions—language.

This is how the human brain/mind builds models and constructs ideas for communications. Our objective now is to replicate this notion of a systemese, archetype concepts and innate syntax, in a formal way so that we can build a communicative language of systems. Chapter 15, which describes a new approach to systems engineering, will revisit the notion of using systemese to construct models, not in the mind but in a machine computable form, which is very much the same process going on in the mind. When a human constructs a spoken sentence in the manner described above, they are, in essence, engineering a design for a thought. We will propose, there, that systems engineering is exactly the same process—or should be. The concepts of language and modeling developed in this chapter will come full circle to provide a method for engineering complex systems. Just as the language of thought gives rise to the public language of communication in order to install a model/thought/idea in another person's head, so too, the same, but explicit, systemese will be shown to give rise to complex system designs.

This program starts with providing a formal definition of system.

## 4.3 A Formal Definition of System

We are now ready to develop a formal definition of a system that, along with the ontological commitments of the last chapter, will be the basis for producing a language for systems. This language will be used to guide analysis of systems, since the definition tells us what we should be looking for, and to build models of systems at various levels of abstraction.[7]

[7] There is a point we need to be clear about. What is being presented here is, itself, a concept about what a system consists of, how it is composed. It is not being represented as *the* "general theory of systems," though it might be a candidate for that title. It is based on having made the ontological commitments from Chap. 3 and following them to their "logical" conclusions. That being said, we are fairly certain that if there is a general systems theory, as posited by von Bertalanffy, for exam-

The definition is given in three complimentary forms: verbal, graphical, and mathematical. All three forms provide views of the system definition that provide access to stakeholders from different backgrounds. The mathematical definition is needed in order to create an abstract representation of the system definition that can be directly applied to creating a language of system (hereafter called SL).

### *4.3.1 Verbal*

The lexicon of SL is taken from the primitive and derived elements in Chap. 3. All verbal descriptions use those lexical elements as the skeleton of meaningful statements. For example, in describing a subsystem that processes material inputs (with energy) to produce a product output we would simply say something like: "Process M takes in materials A and B from sources 1 and 2 along with energy E from source 3 to make product Z with waste product X going to sinks 5 and 6 respectively, at an efficiency of 68%." Additional verbal descriptions would include the rates of flow of materials A and B, energy E as well as those of product Z and waste X. The heat output can be automatically determined given these flows and the fact that the work efficiency of the process is 68%. Another statement that particularizes these would be to identify (name) the materials and products, such as: "Material A is iron." At a still deeper level, we could describe the variations in rates of all inputs and outputs due to various disruptions or things like diurnal cycles: "Material A comes in discrete pulses, one mass of delivery each 24 hours during an interval between 2:00 and 4:00 pm." A complete system description can be given in a paragraph of such statements. Upon deconstruction of the system to find its internal workings, each subsystem and internal flow would generate its own sub-paragraph.

The verbal descriptions start with the lexical elements acting as placeholders for specific items, such as material A is a placeholder for "water" in the above example. With each additional statement, the system description gets refined. As the deconstruction of the system proceeds, sub-paragraphs are added, each describing the subsystem at a lower level of detail.

Figure 4.2 shows a conceptual model of the verbal description above. We show the relations between the main conceptual elements. The sentence demonstrates

---

ple, then it is likely to look something like this. Many theorists have attempted to define systemness in a formal way and suggested that their definition constituted the general theory. Some of them have shared common approaches (e.g., using set theoretical language) yet after nearly six decades, no universal agreement over what exactly "system" means has emerged in the scientific literature. The reader may recall from the discussion in the Preface regarding the way we "talk" about systems science that we regard the latter as more of a meta-science than just another kind of science. Which means that systems science theories are not ordinary theories at all, but metaphysical theories. Ergo, any definition of system must simply accept some ontological (and epistemological) commitments and then get on with it. Only in successes with usage over an extended time will the veracity (or acceptance) of a definition be turned into a meta-theory, i.e., a general theory of systems.

how we can verbally describe a system owing to the semantic and syntactical relations drawn from the underlying systemese.

### 4.3.2 Graphical

Following the age-old dictum that "A picture is worth a thousand words," SL provides a graphical way to express systems. This has actually been done in most modeling languages such as Stella (for system dynamics), and SysML. Various kinds of diagrams can be produced to capture the system as a model. Figure 4.3 provides a graphical version of the system paragraph from above.

A graphical representation such as in Fig. 4.3 can be adorned with the names of the elements and quantitative aspects. It can be used in an animation of the simulation of the model generated from the knowledgebase.

Graphical models of systems help people grasp the essential relations in the system very quickly. In fact, the analysis support tool for using SL in analysis would include a drag-and-drop graphic user interface that would allow them to do the analysis and capture the data in a user-friendly manner.

### 4.3.3 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) used to describe data processing algorithms. These are realized in an actual computer architecture based on the theory of

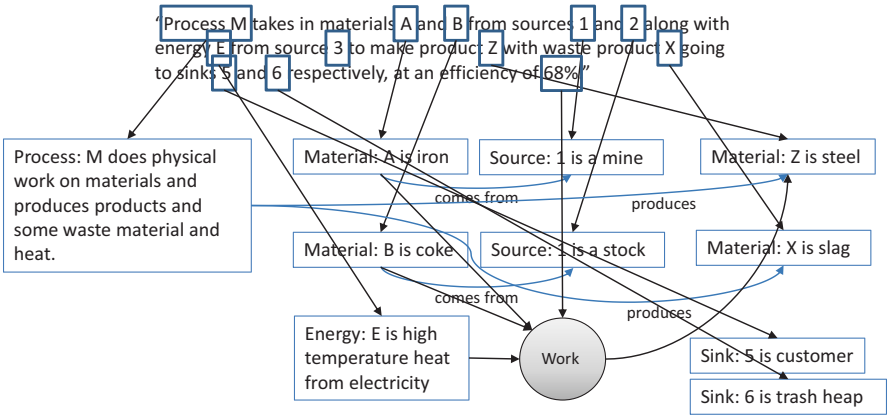


**Fig. 4.2** Verbal descriptions have relations that map to the system language

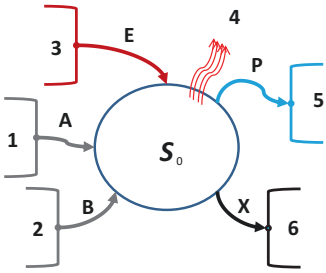


**Fig. 4.3** A graphical representation can be built from the verbal description given above. $S_0$ is the system of interest, lettered arrows represent flows. Numbered entities represent sources or sinks. All of these will be explained presently

computation (e.g., the Turing Machine formalism and the von Neumann architecture).

The following definition is proposed as a starting point for developing a formal definition of system. The development of this approach was inspired originally by Klir (2001) although he, by his own claim, was a radical constructivist, whereas this work is inclined toward a realist interpretation. Another similar approach was that of Wymore (1967). Both of their works were, however, devoted to a purely mathematical approach to defining system and then using the definition and the math to explore the mathematical implications. However, the approach taken in this book is quite different in purpose. We start with a principle-based definition and then apply mathematics as a way to provide a *structure* for "holding" the details of a system description. We will not be *doing* math as much as *using* math.

Deriving a definition of system from the principles and the ontology, a system $\boldsymbol{S}$ is a 7-tuple:

$$S_{i,l} = C, N, G, B, T, H, \Delta t_{i,l} \tag{4.1}$$

where $i$ and $l$ are indexes. The index $i$ is a subsystem index and the index $l$ is the level of organization in the system-subsystem hierarchy.[8] Both are 0 for the initial system of interest, $\boldsymbol{S}_{0,0}$ is the designated SOI. Recall from the previous chapter that the ontological framework specified that the SOI be designated as level 0. This will make sense in this definition as we show how the system definition leads to a natural way to deconstruct the component/interaction level (+1) in the framework. Several later chapters in the book will describe the application of these equations (and the knowledgebase described in Chap. 8) to even more complex adaptive and evolvable systems (CAESs) described in Part III.

[8] As we will see later, we will actually incorporate both indexes into a single coding scheme that will give both the level in the hierarchy and the component index using a 'dotted' numbering scheme. This will be conducive to providing key values for the knowledgebase schema to be covered in Chap. 8.

#### 4.3.3.1 Structural Skeleton

$C$ is a set of components and their "type" along with membership functions in the event the set is fuzzy, that is, the components may have partial inclusion.

$$C_{i,l} = \left\{\left(c_{i.1.l},e_{i,1,l},m_{i.1.l}\right),\left(c_{i.2.l},e_{i,2,l},m_{i.2.l}\right)\cdots\left(c_{i.k,l},e_{i,k,l},m_{i.k.l}\right),\cdots\left(c_{i.n,l},e_{i,n,l},m_{i.n.l}\right)\right\}_l \quad (4.2)$$

is the set of components at level $l$ and $i$ is the component index from the level above (if any). The components of $C_{i,l}$, for example, $(c_{i.k,l},e_{i,k,l},m_{i.k,l})$ 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 $e_{i,k,l}$ are to indicate equivalence classes where applicable. An equivalence class is determined by some set of criteria which are found common among multiple components. For example, in a living cell there are numerous organelles divided into specific types such as ribosomes and mitochondria. All ribosomes share common features and are thus equivalent in those features even if not in terms of locations within the cytoplasm. An alternative approach to Eq. 4.2 would be to use integer numbers to represent, say, the average count of a particular class. 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. Figure 4.4 shows the construction of the set $C_{0,0}$ with three components, one of which is a multiset.

Note that this is not the standard formulation for a fuzzy set. Traditional fuzzy sets are defined as a set and a membership function that applies to all possible members. That is a fuzzy set is defined as a pair, $(C, m)$ where $C$ is the set of components and $m{:}C \rightarrow [0,1]$ is a function mapping a member of $C$ to the interval [0, 1] representing the degree of membership. In the above formulation, each member component has its own membership function.[9] This is called "member autonomy." It allows for members to be individually evaluated for membership based on their particular characteristics. For example, various high-weight proteins and organelles in living cells are prevented from leaving the interior of the cell by the properties of the membrane; they are always members of the cell system. On the other hand, water and various low-weight molecules and ions can transport across the membrane depending on their individual characteristics and the membrane transport mechanisms. Therefore, the high-weight molecules, etc. have a membership function that always returns 1 whereas the membership functions of other molecules depend on their particular properties and those of the membrane.[10]

[9] An alternative approach, one more conducive to some kinds of mathematical treatment, would be to set the membership functions up as a function space or a set of functions that map components with the same index into the system $S_{i,l}$.

[10] Under most circumstances these kinds of components, that is, water molecules, are lumped in various ways, for example, osmotic pressure or concentration for non-water molecules.

Standard fuzzy set theory, along with fuzzy logic, addresses a form of noncertitude regarding the status of objects in a set that appears at odds with the usual notion of probability. Both fuzzy set theory and probability theory map this noncertitude onto the real number-based range. There are long-standing arguments in the mathematical arena on whether fuzziness or probability is the better representation of a lack of certainty about the status of things in real systems. Kosko (1990) puts it this way:

> Fuzziness describes event *ambiguity*. It measures the degree to which an event occurs. Randomness describes the uncertainty of *event occurrence*. An event occurs or not, and you can bet on it. At issue is the nature of the occurring event: whether it itself is uncertain in any way, in particular whether it can be unambiguously distinguished from its opposite. [italics in the original]

Further he distinguishes: "Whether an event occurs is 'random.' To what degree it occurs is fuzzy." The distinction is important. Whether a component is currently in a particular system or not is subject to ambiguity of some kind. In some circumstances it might be best characterized by a probability distribution, for example, $Prob\{c \in S\}$ is given by a classical frequency-based mapping onto [0,1]. But in other cases, whether $c$ is a member of $S$ is not either-or but simply ambiguous. The resolution of this conundrum is going to be dependent on the system and the components. There are two complicating factors that will need to be dealt with. The first is the fact that there are situations in which the component is a member of multiple systems, seemingly simultaneously. This is clearly an ambiguity of the fuzzy kind. We will examine several cases of this condition. For example, a human being seems to be simultaneously a member of a family and a member of an organization (see Chap. 9). This is resolved in recognizing that the component can only do one task at a time and must therefore time multiplex between processes (systems). The second complication comes from trying to describe the collective behaviors of components that have this property. As with the concept of temperature or pressure in gasses, this is resolved by resorting to probabilities. We look at the aggregate likelihood of components being members of one system or another.

Figure 4.4 is a graphical representation of a system with three components in its $\boldsymbol{C}$ set. The two views shown provide different ways to think about systems and their subsystem components. The "tree view" (A) shows the structural aspects of hierarchical organizations. The "map view" emphasizes the nestedness and the topological arrangement of components. It can also show some aspects of heterogeneity among components (e.g., different colors and sizes).

Multisets are allowed. In the figure component $\tilde{c}_{0.2,1}$, with the tilde, represents a set of components with many instances of that type of component. For example, it could represent the water molecules in a cell. Later examples of how this is used will make it clear.

Components of a system that are not multisets or atomic components (as discussed below) may themselves be subsystems, that is, having sufficient complexity to warrant further deconstruction. That is:

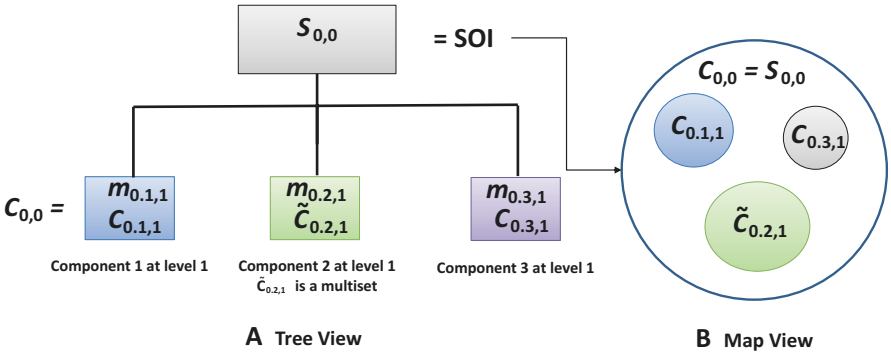


**Fig. 4.4** The structure of a system that is composed of three component subsystems. $c_{0.2,1}$ is a multiset. (**a**) The tree view of the system of interest shows the compositional hierarchy down one level. (**b**) The map view shows the components in $C_{0,0}$ (which is the same as $S_{0,0}$) organized within the boundary of the SOI

$$c_{i.j,l} = \begin{cases} S_{i.j,l+1} & \text{if component is complex} \\ c_a & \text{if component is atomic} \end{cases} \tag{4.3}$$

is the *i*th component treated as a new system of interest at the $l + 1$ level in Eq. 4.3 that describes the recursive structure of system structural hierarchies. The dotted index, *i.j*, is used to maintain the global position of the subsystem component in the original SOI. Later we will drop the leading "0." index from the level 0 SOI since it is the same for all subsystems. Subsequently, the *i* index will designate the level *l* component number. As the number of levels increases (downward) the dotted number index will extend accordingly. For example, the 4th component of the 2nd component at level 2, itself being the 3rd component at level 1 would be designated 3.2.4 (0.3.2.4 truncated).

Equation 4.3 defines a tree structure rooted in the original SOI at level 0. The index *i* designates the branch of the tree, $0 \leq i \leq n$, where *n* is the branching factor. This same scheme continues down the tree where the dotted notation extends the *i* index. Of course, the *l* index is redundant in that the number of dots in the dotted index actually encodes the level in the tree. We include it for the sake of explicitness (to keep the reader from having to count dots!)

The recursion cannot go on forever, obviously. Eventually the tree must have leaf nodes. What stops it? We have identified several stopping conditions, some semi-formal, others a matter of choice by the analysts. As an example of a semi-formal stopping rule, we use the "simplest process rule." This means that a component, $c_{i.l,l}$, $l >> 1$, needs no further deconstruction because it is doing work by 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. Other atomic-level work includes impeding a flow or

propelling a flow. All four of these simple work processes involve the consumption of energy and the loss of energy as waste heat. A fifth simple component is a "raw" stock being used simply as a buffer and without regulating controls. These atomic processes are shown in Fig. 4.5.

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 or ATP molecules do not need further deconstruction as components since their internal structures and specifications for their behaviors are given. Similarly, an organic molecule in a biophysical system need not be further deconstructed. Figure 4.6 depicts a system deconstruction tree resulting from the recursion. The tree view is easier to see the hierarchical relations of system-subsystems-sub-subsystems. A map view becomes unwieldy but is possible to use as a representation. One would simply see ovals inside of ovals. Later, as we present more examples of actual systems, we will see ways to use both representation views as needed. The idea that we can treat any complex component at any level in the hierarchy as a system in its own right, according to Eq. 4.3, supports a way to semi-isolate any component and develop representations for it.

The tree shown in Fig. 4.6 is just the skeleton of the system hierarchy of systems and subsystems. Figure 4.8, will show a more complete tree including subsystems and flows, to be covered next.

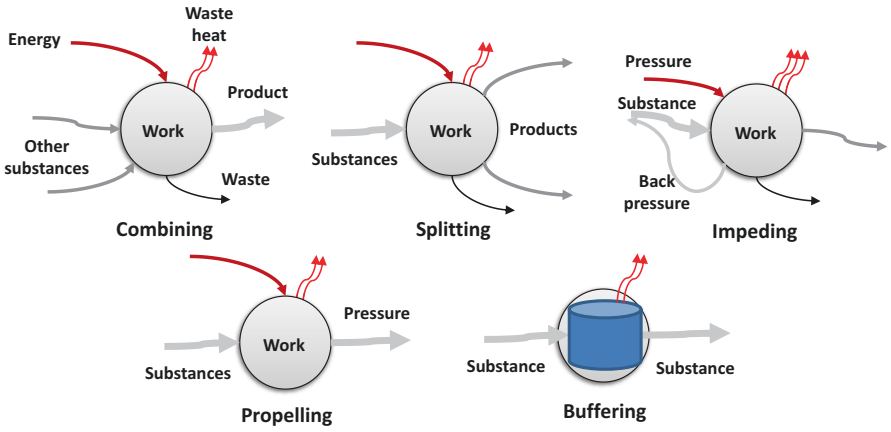


**Fig. 4.5** The formal atomic processes used in the "simplest process rule" for stopping the recursive deconstruction procedure involve either work on the inputs (e.g., combining, splitting, impeding, or propelling), producing output changes (kind or rates), or a passive buffering (stock). The various arrows represent the flows that will be explained below

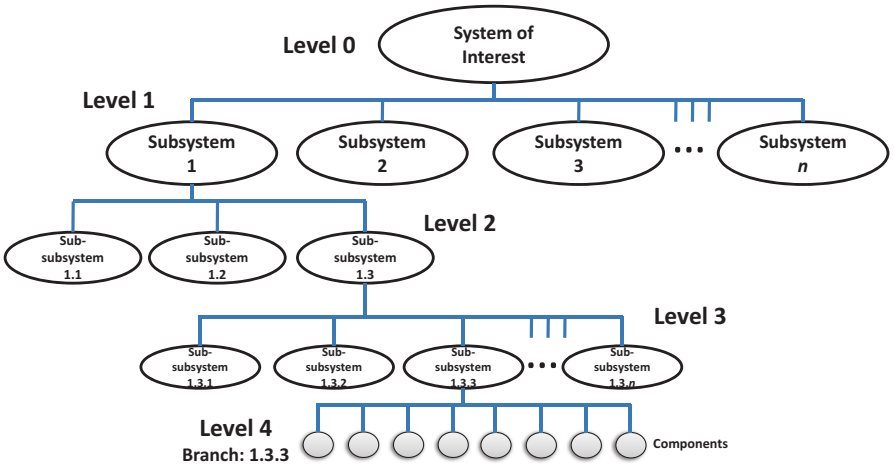


**Fig. 4.6** Equation 4.3 expanded through system deconstruction procedures gives rise to a tree structure. Branch 1.3.3 (where the pre-appended "0." has been removed) ends the recursion with a set of atomic components. Note that the various subsystems are represented in this tree view by ovals rather than boxes as in Fig. 4.4a. Either kind of closed shape can be used

### 4.3.3.2 Interactions

The structural skeleton established thus far has only explicated the organization of subsystems (complex components) and sub-subsystems (complex or atomic components) in a hierarchy of scale. What needs to be established now is the relations between components within the system and between some components within the system and entities outside the system that constitute the system's environment of sources and sinks.

#### 4.3.3.2.1 Between Components Internally

Equation 4.4 is a graph (from graph theory) that defines the interactions between all of the components in $C$.

$$N_{i,l} = C_{i,l}, L_{i,l} \tag{4.4}$$

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

$\boldsymbol{N}$ is generally a flow network through which real substances are moving from one node (component) to the next with causal influence. The term, *cap*, is a function, $cap_{i.k,l}$: $C_{i,l} \times C_{i,l} \rightarrow \mathbf{R}_{\infty}$, giving a capacity describing the flow rates. Rate functions are determined by one or more of the atomic processes from above (as in Fig. 4.3). The actual function will be generally complex in that flows are usually

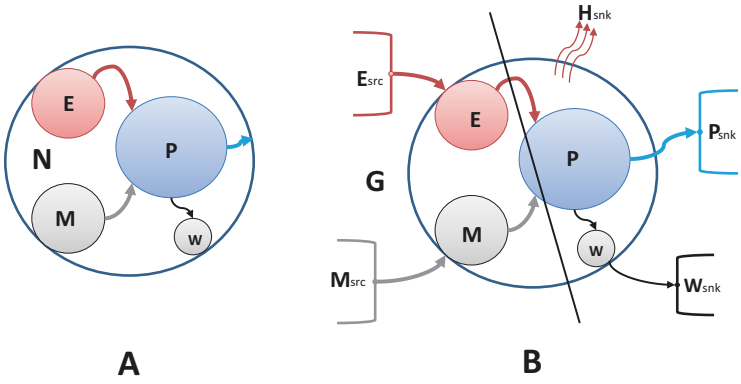


**Fig. 4.7** (**a**) A set of nodes (components within the system) and edges or flows between them. This is the ***N*** graph described in the text. (**b**) The flows from environmental sources, the open rectangles, and to environmental sinks (right side of the figure). This is graph ***G***. The line in (**b**) is the graph partition or "cut." Nodes: *E* energy obtaining process, *M* material obtaining process, *P* production and exporting process, *W* waste removal process. In (**b**): $E_{src}$ energy source, $M_{src}$ material source, $H_{snk}$ heat sink, $P_{snk}$ product sink, $W_{snk}$ waste sink

fluctuating as a function of several different factors. Alternatively $cap_{i.k,l}$ may simply specify the max flow rate possible. The difference will be determined in context.

***N*** captures the internal flows within the system, that is, between subsystem components. Figure 4.4a shows a graphic representation of a four-component system with flows of matter and energy. Note that we treat both flows of actual materials/energy/and messages through channels (e.g., pipes) and phenomena such as application of forces or diffusion (i.e., fields) as generalized flows, where the appropriate equations are used to differentiate in the models.

#### 4.3.3.2.2 Between Environment and Components of S

*G* is a bipartite flow graph defined as:

$$G_{i,l} = \left(C'_{i,l}, Src_{i,l}\right), \left(C''_{i,l}, Snk_{i,l}\right), F_{i,l} \tag{4.5}$$

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 (see Fig. 4.4b and Eq. 4.2). $\boldsymbol{Src_{i,l}}$ and $\boldsymbol{Snk_{i,l}}$ are sets of the nodes situated in the environment. In the figure they are shown as open rectangles (the colors are meant to suggest different categories of sources and sinks). This is because they are unmodeled in terms of their internal workings as elements in the original SOI environment; they are only encountered as members of the

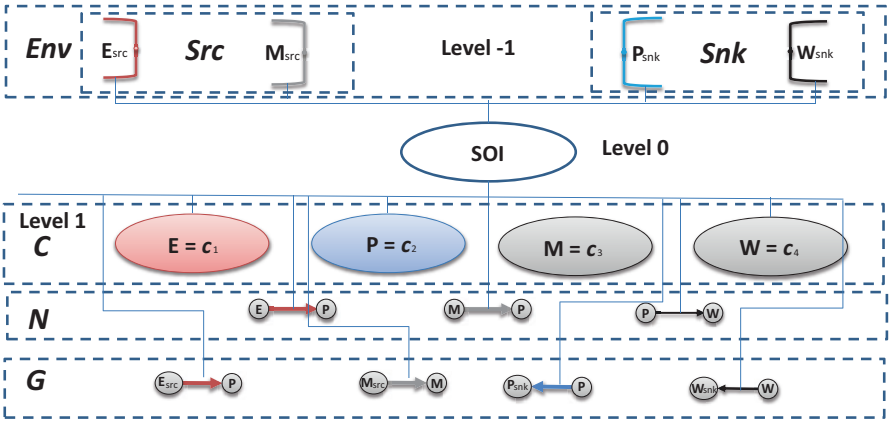


**Fig. 4.8** A more complete tree view of the system depicted in Fig. 4.7 shows the subsystems and both internal and external flows. We also show the system of interest environment and the sources and sinks for the external (***G*** network) flows. Note that flows are edges in a graph and are labeled with the source node and the sink node with the direction of the arrow indicated. Radiated waste heat and the environmental sink are not included in this figure

environment and their internal details cannot be known.[11] Together the sets $\boldsymbol{Src}_{i,l}$ and $\boldsymbol{Snk}_{i,l}$ are a superset, the environment. In certain contexts we will talk about the tuple $E_{i,\,l} = \langle Src_{i,\,l}, Snk_{i,\,l} \rangle$ as being the environment of component $i$ at level $l$.

$\boldsymbol{F}_{i,l}$ 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 $\boldsymbol{c}_{i.o,l} \in \boldsymbol{C}'_{i,l}$, the subset of subsystems in $\boldsymbol{C}$ that are responsible for obtaining inputs from the environmental sources, or $(\boldsymbol{c}_{i.o,l}, \boldsymbol{e}_{i.k,l})$, $\boldsymbol{e}_{i,j,l} \in \boldsymbol{Snk}_{i,l}$ and $\boldsymbol{c}_{i.o,l} \in \boldsymbol{C}''_{i,l}$, the subset of subsystems in $\boldsymbol{C}$ that are responsible for expressing outputs to the environmental sinks. 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.

Figure 4.8 provides a tree view of the entire system, thus far defined. It shows the environment entities (Fig. 4.7b), the internal subsystems (components), and the internal and external flows.

#### 4.3.3.3 Boundary

Up to this point the definition of system should resemble those given in most accounts whether in mathematical form or not. Languages for modeling systems, such as system dynamics (SD) utilize the same sets of elements as described so far but not in this mathematical form. At this point we start to depart from historical

[11] Later we will see this "rule" is modified as we go deeper into the original SOI and deconstruct to lower components.

accounts of system definitions as will be explained along the way. The first departure involves the concept of a boundary. If you recall from the last chapter, we claimed that boundedness is a real element (has first-class ontological status). Boundedness constructs effective boundaries to systems.

This situation is not without controversy. Workers who are primarily interested in modeling systems have maintained that there are no real boundaries but what the modeler chooses (Meadows 2008, pages 95 and 97). This is a reasonable claim when the focus of attention is on the replication of certain dynamics of "parts" of a whole system. In such cases, the boundary of the simulation model, not the system, is being chosen to keep the problem tractable. On the other hand, some systems clearly have identifiable boundaries such as cell membranes, skin, or walls.

Our position is that all systems are kept intact by virtue of some kind of internal binding that produces an "effective boundary." The distinction between what is "inside" a system versus what is outside depends on this effective boundary. Systems that do not have physical boundaries nevertheless have distinct subsystems that interface with the entities in the environment that supply resources or act as sinks for the wastes produced by the system. Another way to think of these kinds of boundaries is that they are a result of the interactions between internally bound special component subsystems, interfaces, to be described shortly, and those external entities. Recall Fig. 3.8 in the last chapter.

For purposes of analysis and design, we will make boundaries explicit elements of a system definition.

The boundary, $\boldsymbol{B}$ in Eq. 4.1, at level $l$, then is a tuple. That is:

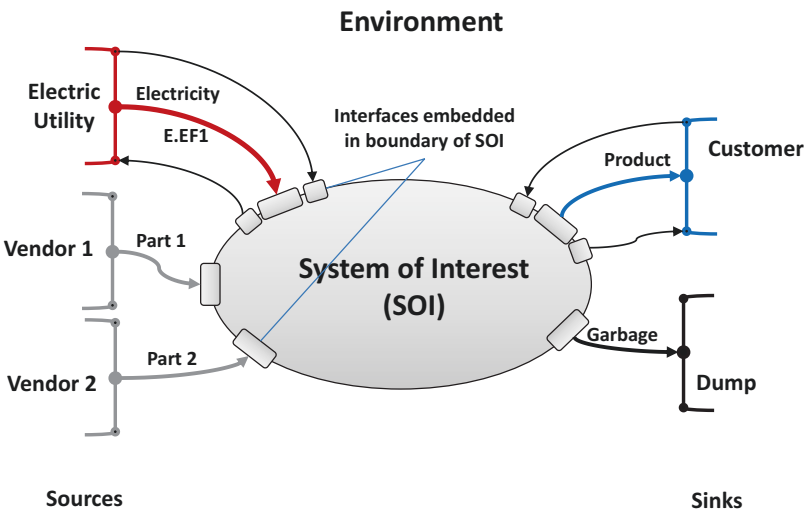


**Fig. 4.9** Interfaces are associated with the boundary of a system. Here they are shown as round-edged rectangles that penetrate the boundary and act as pass-ways for inputs and outputs

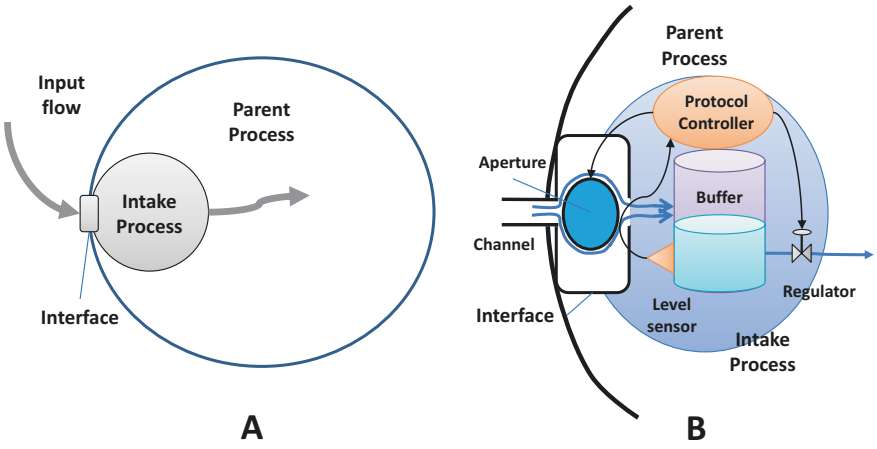


**Fig. 4.10** Interfaces are represented generically (**a**) showing the general form of the interface between the outside world and, in this case, an intake process that then supplies the substance flow to the parent process (or SOI). (**b**) Provides some details of a more complex interface, for example, an active membrane pore on the surface of a cell. The aperture is the main channel control device (here shown as a "ball valve"). Many interfaces are subsystems (processes) in their own rights and so will have additional "equipment" devoted to regulating the flow of the substance. Other elements in the figure (e.g., sensor, buffer, and regulator) will be explained below

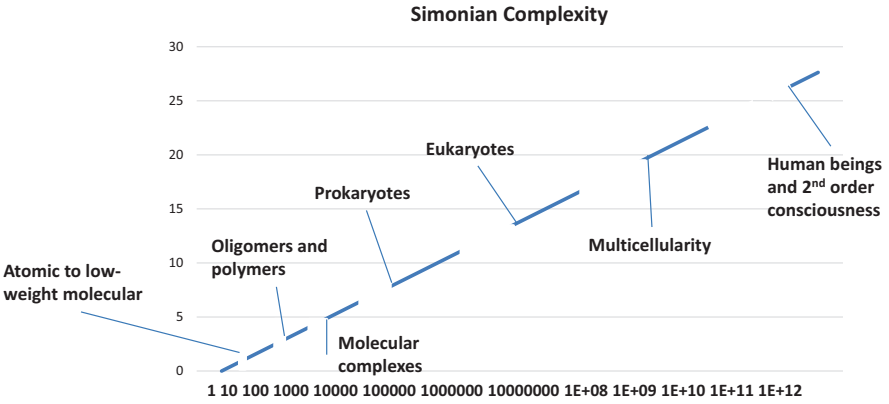


**Fig. 4.11** As Simonian complexity explodes up the levels of organization (and complexity), the logarithm of that value rises linearly and gaps (probably exaggerated) in the measure are seen at the major transitions, where the graph line (blue) appears indicates a range of complexities within the various domains

$$B_{i,l} = P_{i,l}, I_{i,l} \tag{4.6}$$

where $\boldsymbol{P}$ is the set of properties and the second set, $I_{i,l}$, is the set of *interfaces*. The exact form of $\boldsymbol{P}$ is still an object of research. At present it includes such properties

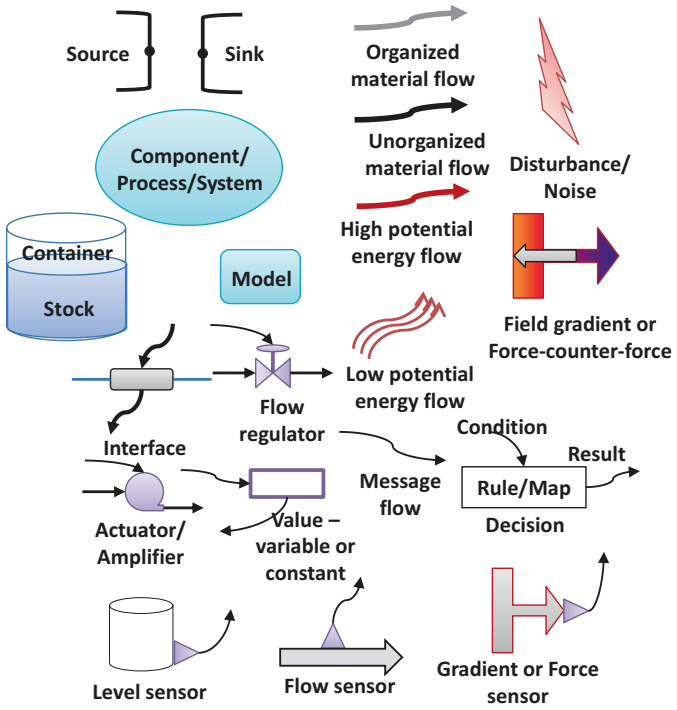


**Fig. 4.12** An error sensing mechanism resulting from the composition of four atomic work processes. The circuit can be used by, for example, a homeostat, a cybernetic feedback control

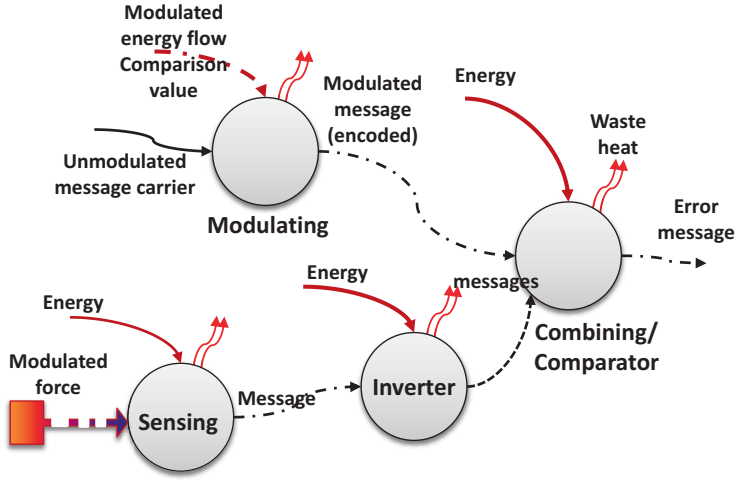


**Fig. 4.13** These are some of the icons used to represent elements of the system language

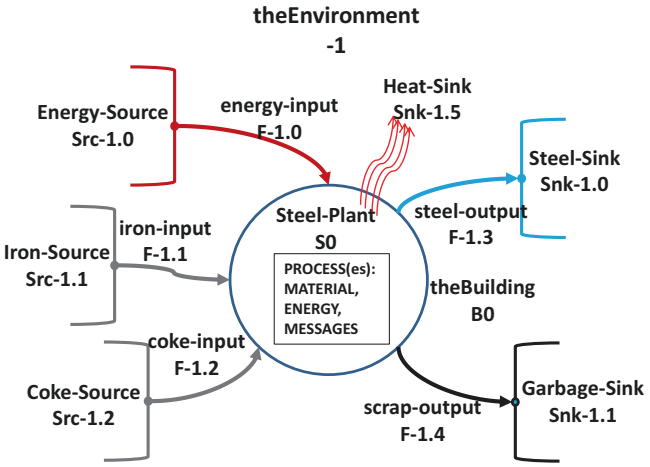


**Fig. 4.14** A simplified example of the use of SL to describe the SOI, a steel manufacturing plant. This figure shows the SOI and its environment. Note that labels (names of objects) are user-defined (using prescribed characters as in most programming and markup languages), but may follow some determined style requirements. As in most programming languages, the strings are case-sensitive. The identification codes, for example, Src-1.0, on the other hand, follow strict conventions that will be explained briefly in the text and discussed more thoroughly in Chap. 6

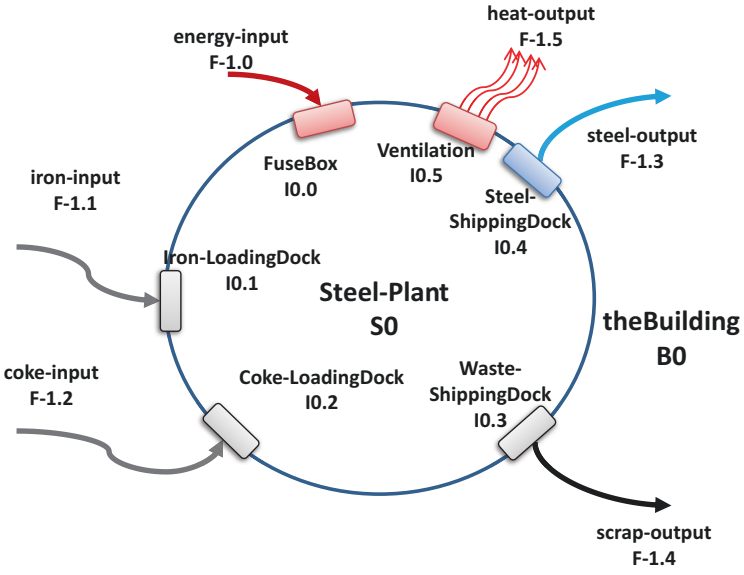


**Fig. 4.15** Analysis of the boundary identifies the main material and energy interfaces for importing resources and exporting products and wastes

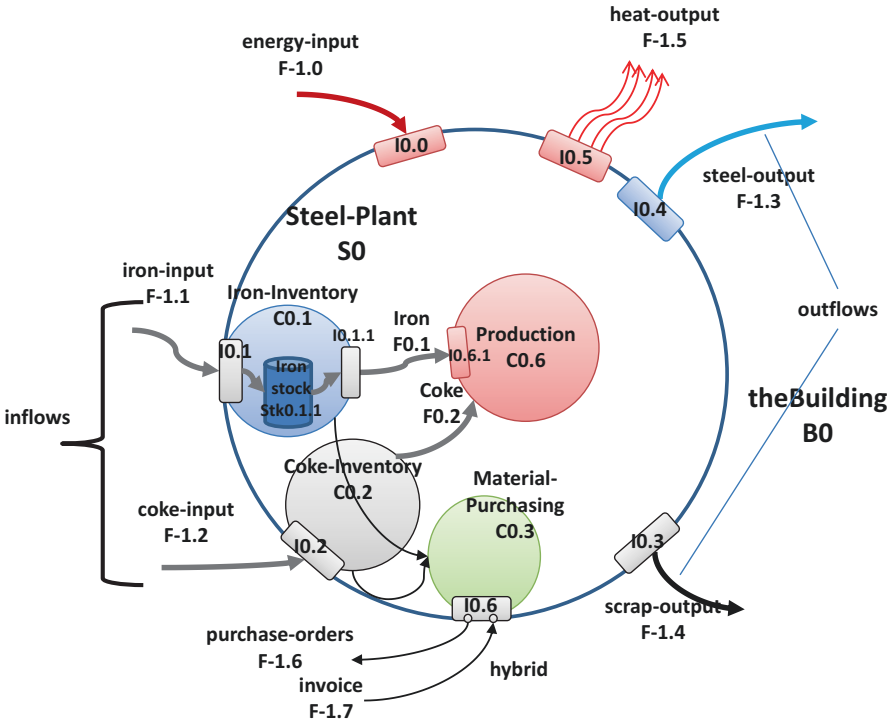


**Fig. 4.16** Further analysis of the SOI as a transparent-box reveals internal components and flows. A new component has been added, the Materials-Purchasing (C0.3) subsystem. Its interface with the supplier demonstrates the situation when a message interface communicates bidirectionally, both receiving and sending. It also indicates that a single environmental entity (the supplier) also sends and receives. Moreover, the supplier is a source and a sink simultaneously

as porosity (0 being completely non-porous) and "perceptive 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 any number greater than 0 but less than 1 being the degree to which a physical phenomenon corresponding to "keeping the insides in and the outsides out."

Boundaries can be hard physical structures such as a cell wall (in plants and bacteria) or a structure resulting from competitive forces such as the phospholipid bilayer membrane of living cells that results from its own unique internal chemistry. The boundary of an ecosystem is very fuzzy and very porous, both perceptually and in the technical sense. Some members of an ecosystem may transit in and out at various times but usually through particular portals (e.g., game trails). Nevertheless, ecologists agree that there are boundary conditions that provide an internal milieu suitable for the species that live there. The climate conditions, mostly regulated by geographical features surrounding that of the ecosystem, provide a supportive home for those species adapted particularly for them.

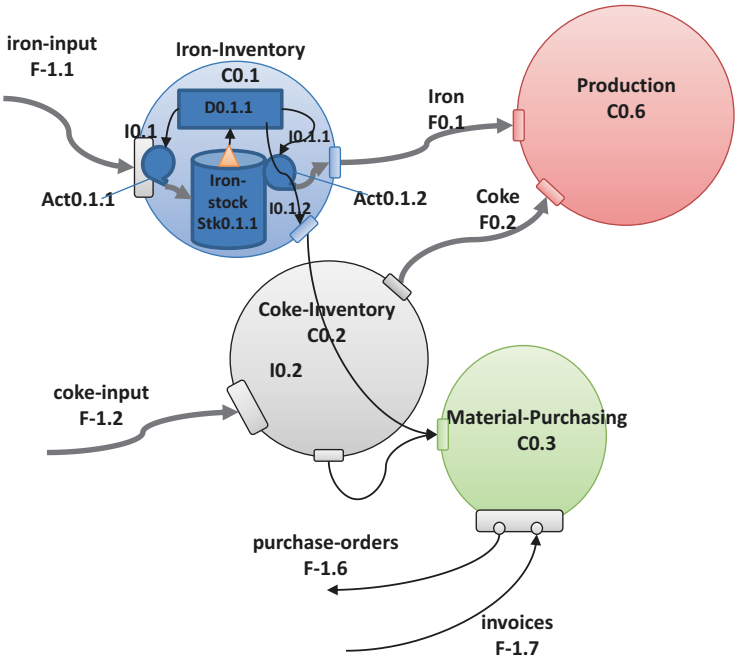


**Fig. 4.17** Further expansion and decomposition applied to the Iron-Inventory shows it to be a more complex process including actuators receiving iron-input and putting it into the Iron-stock (Stk0.1.1). A decision process monitors the level of inventory (orange triangle) and decides when to order more stock (black arrow coming out of I0.1.2 going to Materials-Purchasing)

By analogy with the set $\boldsymbol{C}$ above, the set of interfaces embedded in the boundary are components in the boundary subsystem and are themselves subsystems. That is, every $\boldsymbol{r}_{i,l} \in \boldsymbol{I}_{i,l}$ is an $\boldsymbol{S}$ itself but including what we call a protocol object, $\varphi$. That is:

$$r_{i,l} = \left(S_{i,l+1}, \varphi\right) \tag{4.7}$$

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 substance of the flow, that is, do not transform the substances as a process does to create products.[12] In the case of interfaces as subsystems, the $\varphi$ object is called a protocol which is an algorithm for letting the flow across the boundary in an ordered fashion. The receiving dock of a manufacturing company has a *procedure* that receives packages, checks the manifest, checks the material being received,

[12] With the exception of an interface containing a "filter" protocol. Filters are designed to let the desirable substance through but impeding whatever might be undesirable. Some filters may require periodic replacement if they get clogged up with material substance, or they might be flushed periodically to some waste flow. Other filters such as those used to filter electric flow frequencies dissipate the "waste" continuously.

and sends it on to the inventory subsystem. A neuron's postsynaptic membrane allows ions to pass into or out of the cytosolic compartment through special pores that are activated by the transmission of neurotransmitter. Most interfaces will be found to involve both a mechanical (or electrical) pass-through control and a message processing component that controls the pass-through control.

A generalized model of interfaces on the boundary of an SOI is shown in Fig. 4.9. The flow arrows from sources are shown going into an interface component; arrows coming out of output interfaces go toward the sinks. Interfaces are special kinds of regulators, usually allowing passage in one direction only, and only under the right conditions of the protocol.

Figure 4.10 expands on the nature of interfaces through an example. Each interface provides a connection to the channel of the flow in the environment on its external side. Internally it is interfacing with a subsystem of the SOI that is either a receiver or an exporter process (as in Fig. 4.10b).

The biggest single mistake that system scientists (or scientists exercising systems thinking) make is to ignore or trivialize the concept of an interface (and its protocol). Examples of how this can lead to trouble in adequately accounting for system behavior will be provided in the chapters ahead. Suffice it to say that many mistakes in system analysis and design result from the inattention to interfaces and their protocols.

As a rule of thumb, the interfaces for more complex systems tend to be more complex themselves. A simple opening in the wall of a hut functions as a door, but it allows anyone (or anything) to walk through.[13] An actual hinged door, with door knobs inside and out, and an internal lock mechanism (keyed from the outside) in a modern (and more complex) structure is a bit more complex and has a controlled entry/exit protocol. Entries and exits from airport departure/arrival gates have become exceedingly complex with separate entry and exit paths required by security needs.

#### 4.3.3.4 Transformations

$\boldsymbol{T}$ 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, equation, or algorithm, $\boldsymbol{t}_{i,l}$, 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.

The inputs and outputs are the same flows as represented in the $\boldsymbol{G}$ graph (above). Specifying $\boldsymbol{T}$ for the level 0 SOI will be exceedingly difficult initially since it involves many different inputs and outputs (for complex systems). As we will show in Chap. 6, however, it isn't necessary to make a full specification of the transformation at the start. For purposes of analyzing natural and designed systems both, all

[13] This is an example of a bidirectional interface with a nearly non-existent protocol, unless you consider that if the hole is not very large you might need to duck your head each time you pass through.

that is needed is a rough approximation of the transformation, used as a placeholder that suggests its nature. This is, effectively, an abstract model of the transformation that will be refined as the deconstruction process continues. One of the advantages of following this framework is that the higher-level transformations get refined by discovery of the lower-level ones. Once more is known about the internal transformations of subsystems and their combined effects, the higher-level model can be made more rigorous (Principle 12 in Chap. 2). This recursive improvement tracks the deconstruction all the way down to the leaf nodes in the system tree.

Note that this approach varies from many systems' engineering methods in which an attempt is made to specify the system output (with great specificity) as the first step. This inclination derives from device engineering practices where the function of a device can be a priori specified and the device engineered to produce that result. In systems engineering the final output(s) of the system are generally far too complex to have a perfect specification upfront. When system engineers believe they are obtaining such a specification (e.g., in requirements gathering), they are often faced later with discontinuities between expectations and actualities that are highly disruptive to design projects. With the approach of deep analysis promoted in this book (top-down deconstruction of transformations and bottom-up refinement) the engineering process will be following a formal procedure that they often end up following informally anyway!

#### 4.3.3.5 Memory

***H*** in very complex systems could be 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 microstructures change over time. This is called memory and the current state of ***T*** can be based on all previous states. Some simple systems, like atoms for example, may have a NULL ***H***; that is there is no memory of past states and future states depend only on the current state and current inputs. As just mentioned, on the other hand, brains (and indeed all biological systems) have very rich memories. ***H*** augments ***T*** and all variables associated with elements in ***N*** and ***G*** in that it records traces of the changes in these variables over time. A simple version of ***H*** would be 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 ***H*** would be the human brain, particularly the neocortex, where memories are encoded, stored, and retrieved for use (Mobus 1999).

An example of emulating the way brains develop memories and its possible relation to the system model as captured in the global knowledgebase must wait until we get to Part II. The human brain is the quintessence of a way to build models of systems (as discussed above). There will be numerous comparisons between the use of this mathematical framework and the workings of the brain as ways that understanding is captured.

For our purposes here, however, we will consider more formal methods for capturing history in a usable way. Consider, for example, the use of a recorded time series of state measures. Let $H$ at time $t$ be defined as a set of measures (a list of variables in the system),

$$H_t = \left[ v_1, v_2, v_3, \ldots, v_i, \ldots, v_n \right]_t \tag{4.8}$$

At each time instance the variables of the system are measured with an appropriate instrument and recorded. The time series of $H_t$ sets provide a set of snapshots of the state of $S$ at each time increment. For example, the profit and loss statement of a corporation is an annual snapshot of the corporation's most essential state variables. In the simplest case $H_t$ is record made every $\Delta t$ unit. It is what we call a data stream. However, just having a record of the data is not very useful. Ordinarily, just as with the profit and loss statement, we look for patterns in the data after processing it in some fashion.

#### 4.3.3.6 Time

Finally, the last element in $\boldsymbol{S}$ is $\Delta \boldsymbol{t}$, a time interval relevant to the level of the system of interest. The time interval is familiar to those who work with discrete-time simulations. In general, higher levels in the hierarchy of organization have larger $\Delta \boldsymbol{t}$s; the activities take longer than those at lower levels. $\Delta \boldsymbol{t}$ is generally an integer multiple of the lowest level time constant that is deemed relevant for a particular system. In discrete-time simulation, it is the time step over which the model of that level is computed.

##### 4.3.3.6.1 Time Indexing

For all systems at any level in the structural hierarchy an index of time step, $t$, is used to count the amount of time in $\Delta t$ units for that level between events or state changes. Under ordinary circumstances (e.g., when not trying to model infinite time) there will be an additional parameter used to designate time units in a larger period. Thus $\Delta t$ might be replaced by a tuple, $\langle \Delta t, x \rangle$, where $x$ is the integer count of a single cycle.

##### 4.3.3.6.2 Cyclic Intervals

Real systems are embedded in supra-systems that undergo cyclical behavior. The Earth rotates on a diurnal cycle, tides rise and fall, seasons come and go and come again. Some cycles are regular intervals, the general meaning of the term "periodic."

Others, such as the tides, are "quasiperiodic" meaning that they are irregular but nevertheless repeating, just over varying intervals.

In systems that undergo periodic or quasiperiodic behavior, the element $\Delta t$ can be replaced by a "clock" or "quasi-clock" function that counts $\Delta t$ units until that count reaches a limit and the counter is reset to 0. A quasi-clock has a secondary function that generates the limit number (which is not a constant) according to the phenomenon leading to the quasiperiodicity. Admittedly, this is an unsettled area requiring more research. But models of tides based on the major parameters such as position of the sun and moon have been developed. There are instances when the periodicity of a cycle may be varied in a constrained-random fashion, using, for example, the bounded Monte Carlo method.

#### 4.3.3.7 Considering Very Complex Systems

Complexity is a very difficult concept and the word itself has many different senses. Intuitively most people have the notion that if something has many parts (i.e., subsystems) and many interactions (i.e., flows) between the parts and many interactions with its environment, then it is complex.

There are many different ways to characterize the complexity of a system. Here we investigate two complementary approaches. The first looks at what we can consider "structural" complexity using the above-mentioned intuition. The second approach looks at dynamic or "behavioral" complexity, that is, how are the behaviors of the complex system in relation to their environmental interactions. Both include consideration for the number of states that a system can be in, but behavioral complexity looks only at the states of interactions with the environment.

##### 4.3.3.7.1 Simonian Complexity

We have adopted a definition of structural complexity derived from Herbert Simon's description of a near decomposability of a system (Simon 1998, 209).

Simon's description of systems as hierarchies of modular units (what we called components or subsystems above) gives rise to a metric that can be used to characterize the complexity of real systems. We develop that metric below.

There are two fundamental aspects that go into defining complexity: structural complexity and functional complexity. One deals with the hierarchy of modules and submodules and the other deals with what each module does. The former is exposed by the structural decomposition process outlined in Chap. 6. The latter is much more difficult to estimate. For our purposes here, we will use the concept of a state space to approximate the metric of functional complexity.

Imagine taking a reading on every flow (connection) and every reservoir in a system and all of its subsystems every $\Delta t$ instance. The state, $\sigma_i$, of the system where, $i$, is the index of the set of possible states, $\boldsymbol{S}$, and $\sigma_i$ is an element of that set. The instantaneous measure of all of these dynamical elements at time $t$ defines the

system as being in state $i$ at time $t$, or $i = \sigma_t$. Since by definition a system is an organized set of parts (components and subcomponents) the number of possible states is constrained and so can be represented mathematically, at least in principle, by a Mealy finite-state machine (or automaton). For systems where multiple state transitions may occur from any one state to a successor state and where these are stochastically chosen, the representation is that of a non-deterministic finite-state machine with statistically determined transition probabilities on each node out link. The number of transitions possible in the entire state space is $\boldsymbol{T}$, a list of the pairs of from and to states. The Mealy FSM takes into account the system's interactions with the environment or context. The state transitions are determined by a combination of inputs to the system and the system's current state at time $t$.

As a reasonable approximation of the *size* and complexity of the state space, we can use the number of states, $\boldsymbol{S}$, in the space (number of nodes in the finite automaton) and the number of transitions, $\boldsymbol{T}$.

$$|\mathcal{S}| = f\left(|S| + |T|\right) \tag{4.9}$$

The function, $f$, is as yet unspecified but assumes some kind of scaling factor.

We can then define Simonian complexity measure as a sum of the number of components and subcomponents down to and including leaf nodes (atomic components) along with the sum of interactions in the $\boldsymbol{N}$ networks within each component module, the size of the boundary as a list of interfaces, approximating the number of inputs and outputs, and size of the state machine.

$$\mathbb{C} = \ln\left(\left(\sum_{L}^{l=0}\sum_{I}^{i=1}\left[\left[\left|C_{i,l}\right| + \left|N_{i,l}\right|\right] + \left|B_0\right|\right]\right) + |\mathcal{S}|\right) \tag{4.10}$$

This function sums the number of all components and relations at each level in the decomposition hierarchy and the number of affordances with external entities or number of interfaces on the boundary and takes into account the state space size. The log function compresses the numerical value of the complexity measure.

The working hypothesis is that while $\mathbf{C}$ is a continuous variable, we should see gaps between values for lower complexity systems and higher complexity ones that mark phase transitions. For example, we should see a, perhaps small, gap between elements and molecules and a larger gap between macromolecular assemblies and whole cells. In fact, the hypothesis is that as we ascend the hierarchy of organization, we will see larger and larger gaps between the complexity measure of lower domain systems and higher domain ones as the complexity of higher-domain systems explodes exponentially. Figure 4.11 shows this basic idea.

It is a research challenge to compute the Simonian complexity of various systems as shown in Fig. 4.11. To the degree it might be done, we expect to observe some kind of discontinuity (e.g., the gaps shown) that demarks the transition from a lower complexity domain to a higher one.

#### 4.3.3.7.2 Behavioral Complexity

Behavioral complexity can be far more difficult to characterize compared with structural complexity. The behaviors of a system are related, clearly, to the number of states the system might be in as described above. However, there are possibilities of transitions from internal states that are not easily modeled by finite state machines, even probabilistic machines. Among these possibilities is behavior resulting from internal non-linearities, resulting, for example, in chaotic behaviors. That is, if one captures time series data on the externally displayed states of a system, they describe a chaotic attractor basin in phase space.

### 4.3.3.8 Ontogenesis of Greater Complexity

Ontogenesis, recall from Chap. 3, is the general process that brings components together to form more complex entities. Here we provide a somewhat specific example of how components as "atomic" work processes (from Figs. 3.18 and 3.19) are composed through linkages of outputs to inputs to produce a functional and useful object. Figure 4.12 shows the end result of an ontogenic linkage that results in a device capable of computing an error message, the net value of a difference between a reference signal (e.g., setpoint) and a measured value.

In this example we take four different atomic work processes, sensing, an inverter, a modulator, and a combiner, here being used as a comparator. Refer to Figs. 3.17, 3.18, and 3.19 to the atomic processes. The sensor takes in energy to do the work and transduces a modulated force, such as pressure or temperature producing a modulated message encoding the force parameter. The inverter takes as input a message and does what its name implies; it inverts the sense of the message code (for numerical values it effectively changes the sign of the value). It too uses energy to do this work and output a message. The combining process receives two inputs. In this case, the combining process is for messages, in other words, a computational process. It receives the output message from the inverter and a message from the modulating process which has transduced a modulated energy input onto a message carrier which is output to the other input to the combiner. The latter does the combining, which in the case of messages is a superposition of the two messages resulting in either a null message output or a signed output corresponding to the amount of error.

This whole assembly can now function as an error detector as long as the major inputs, the modulated force and the modulated energy flow, are commensurate. That would mean that the final output of an error message had some veracity in the context of a controller agent that could take a decision and activate whatever power it had to counter the error.

This model is extremely general. It could model the creation of a feedback loop at the molecular level, say in the lead up to the origin of life. Or it could represent the engineering design of an error detection circuit in a nuclear power plant control. In any such case, if the inputs match the outputs and the work processes perform

their functions appropriately, the circuit's use leads to stability and continuation of the larger system in which it is embedded. If not, better luck next time!

In ontogenesis, as described in the previous chapter, luck was a factor in bringing components into coupling proximity and in the right ordering in auto-organization. We saw that a mixing and sorting regimen such as occurs in a convective cycle in a gravitational or electric field increases the chances for, say the sensor and the inverter to couple since their input/output interfaces match and once coupled they are reasonably stable awaiting the proximity of a comparator to the inverter's output interface. There is a kind of "rachet effect" that keeps promising couplings stable for at least a while until the next component joins the group. One explanation for this is that each of these work processes gets an input of high-potential energy that has to be dissipated properly in order for the system to not overheat, as it were. Once two processes are suitably coupled, they create a better pathway for this energy to do so, thus lowering the total energy (specifically the thermal modes) and raising the likelihood that they will be more stable. Add another, or several more processes and this stability ratchet increases.

Of course, the process is purely stochastic for auto-organization, with only slight biases provided by the mixing-sorting regimen. But even for intentional-organization (design) as we will see in Chap. 15 chance still plays some role in getting the right components together in the right way at the right time.

We now turn attention to the problem of how to use our ontological framework, understanding of ontogenesis, and our formal framework for describing systemness to create a way to describe all systems, but particularly able to describe even the most complex systems we encounter. We need a language.

## 4.4 Toward a Language of Systems

Armed with the mathematical structure described in Eq. 4.1 and subsequent equations, along with the ontological commitments in Chap. 3, we are in a position to design a formal language of systems (SL). With such a language we will be able to describe, in principle, any system in any domain, for example, biological or sociological. The language uses generic terms to represent various elements of systemness, for example, nouns like "process" to embody a system that does work and "flow" to embody the movement of materials, energies, and messages. Its syntax is based on allowable patterns or relations, for example, a work process that varies a flow (e.g., a valve or resistor) cannot be put inside a stock; it has to be inserted into the flow.

Ultimately SL's semantics describe the systemness of a specific kind of system, for example, a biological cell. The generic lexicon is translated into domain-specific terms. For example, a vacuole (an organelle that holds stuff) is a kind of reservoir. The semantics of SL (of systemness) provides the fundamental organizing aspects of all systems in any domain.