Chapter 5 – The Process of Deep Systems Analysis

Abstract

The first component discussed in Chapter 4 is the deep analysis of a system of interest using a top-down decomposition procedure. This chapter will provide the guidelines for how this procedure integrates the objectives of a reductionist analysis with retaining the holistic aspects of systemness by using the recursive system definition of Chapter 3 that preserves the interrelations of subsystems at all levels of organization in the system. The procedure is one of ‘deep’ analysis meaning that it is an algorithm for guiding a recursive process exposing increasingly deep details of subsystems and components until we come to stopping conditions, down any leg of the hierarchy based on finding “leaf nodes” representing “atomic” processes. We end the chapter looking at ‘advanced’ concepts of complex systems such as fuzziness, adaptability, and evolvability considerations. These will be revisited in coming chapters.

5.1 What We Seek to Achieve

In this chapter we focus on the analysis phase briefly described in the last section of Chapter 4. This is the part of the process where knowledge of the system is obtained through deep analysis and captured in the knowledgebase. Guidance for the procedures comes from the formal definition of a system given in Chapter 3 along with the language of systems derived from the ontology of Chapter 2 and the formal definition. What we will be describing in this chapter is essentially the procedures to be followed in performing this analysis in the abstract, that is, as they apply to any arbitrary system. In the next chapter we provide examples of how these procedures can be used to analyze three particularly complex specific systems. In Chapter 7 we will cover the nature of the knowledgebase itself – how it stores knowledge for retrieval and use, again as briefly described in Chapter 4.

The procedures we describe are:

- Define the System of Interest (Level 0)
- Environment, Boundary, and Flow Analysis (Level -1)
- Recursive Decomposition of Subsystems (Level 1..*m*)

Within each of these procedures we will cover the methods for collecting data and entering it into the data structures of the knowledgebase. It is the organization of the data in these structures along with the relations between these structures that turn the data into knowledge.

The work to be described in this chapter is not without predecessor thought. Specifically, the works of George Klir (2001) and Peter Checkland (1999) have been instrumental in forming the concepts presented here. Klir was a mathematician who was interested in the abstract representation of systems (as described in Chapter 3) and who developed a rigorous framework for thinking about systems, most particularly of what we now call the ‘hard’ sort. His work, being primarily mathematical in nature, seems to have gotten lost in the general systems literature, at least in the Western world. Our aim is to bring his insights down to earth, so to speak, by making the procedures implied in his abstractions more operational.

Checkland investigated systems that involved human actors (agents) and complex decision making and came to the conclusion that such systems could not be characterized in the same way the so-called hard systems were done. He (with others) developed the concept of soft systems to deal with much less well characterized systems like organizations involving social interactions (which include human emotions and motivations, biases and beliefs). His work and thoughts are pursued today in the frame of organizational systems thinking. Without losing his insights as to what makes a soft system soft, we will attempt to show how more formal constructs can be brought to bear on such systems. Our objective is to show how all systems can be understood in the same framework of systemness and that doing so can give rise equally to scientific understanding of natural phenomena as well as being the basis for generating designs and policy prescriptions. In other words, to unify hard and soft systems under a single notion of systemness.

At the time that both of these thinkers were formulating their approaches there still existed formidable hurdles with respect to characterizing very complex systems, particularly human thinking. And so systems science has taken on a bifurcated set of tracks that cater to hard problems versus soft or wicked problems. Our work will attempt to demonstrate that the deficiencies in characterizing the hard aspects of so-called soft systems are giving way to more rigorous methods (e.g. functional imaging of living brains during perception and conceptualization). Thus, we claim, a general systems understanding methodology applicable to both hard and soft systems is now amenable. We briefly examine the status of Klir’s approach and that of Checkland’s as they represent the dichotomy and then proceed to outline the methodology that we assert will reconcile it such that there is only one concept of systemness to be employed.

5.2 Perspectives on Systems Analysis

In the next two sections we will briefly review Klir and Checkland to provide some framework for arguing the resolution between the hard and soft systems perspectives.

5.2.1 Klir’s General Systems Problem Solver

The idea that a formal approach to understanding concrete and abstract systems through a systems-based methodology was described by George Klir (2001). His approach was very abstract and covered the range of systems problems[1] very generally. He established a basic knowledge (epistemological) framework as a hierarchy of system components and categories, then proceeded to sketch out how the problem solver (GSPS) would work to capture the system knowledge from a specific domain, concrete system.

The subject of this chapter is in line with Klir's concepts. What the chapter describes is the author's view of actual methods to be employed, starting with procedures for capturing what Klir referred to as system knowledge.

5.2.1.1 Epistemological Hierarchy

Klir starts with describing a way to categorize systems based on a hierarchy of forms. For Klir a system could be the actual entity of inquiry, the 'thing' that is a kind of system or what he called a 'source system' or also an 'experimental frame'. Or it could be a higher-level abstraction. The next level up from the source system was a 'data system' in which actual measurements (and their number support) of parameters were to be stored. Figure 5.4, below (section 5.5.1.5), shows how a data system is obtained by measuring a time series of output flows. Figure 5.5 goes on to show a source system fully instrumented and collecting both input and output flows over time. The combination of the source system and the data collected over an appropriate length of time constitute the data system.

In section 5.5.3, below we see the next level up in Klir's hierarchy. This is the construction of a generative system. From the analysis of the input/output data we can estimate what is called a transfer function. Having such a function allows us to compute a semi-unique output from the system for any combination data on the inputs. In other words, we have arrived at a model of the system of interest that allows us to make predictions.

At the highest level in Klir's hierarchy is the 'structure system'. This is the composition of source/data/generative systems that produce higher order systems (what we have been referring to as the higher levels of organization). This is Klir's version of the recursive decomposition of complex systems into sets of simpler systems.

Using this hierarchy, systems can then be classified by combinations of these characteristics. **SE**, **SD**, **SG**, **S^2E**, **S^2D**, **S^2G** represent structures of source, data, and generative systems, including second order structures (**S^2**), structures of structured systems.

---

[1] Mathematicians tend to use the word 'problem' to specify whenever we want to achieve something or figure out why something works the way it does, not just when something doesn't work properly and we need to figure out why. A biologist would describe the need to find out, for example, the details of a particular metabolic pathway as a challenge, but since nothing needs fixing would not consider this a problem. On the other hand, how life got started in the first place is still problematic since we don't have sufficient sources of information to construct a model.

5.2.1.2 Metasystems

Metasystems are categories of similar systems. So, for example, all living cells fit into the general category of 'cell' even though there are significant differences between cell types (their underlying source, data, generative, and structure system characteristics may vary according to their 'thinghood').

The **M** operator represents the categorization operator of systems of the same types as metasystems. Thus, **ME**, **MD**, and **MG** are constructions of the categories of all source systems, all categories of data systems, and all categories of generative systems, respectively. As with structure systems, higher order metasystems are permissible. Moreover, complex combinations, for example, a metasystem of structure system of source system is describable (Klir, 2001, 87).

5.2.1.3 Conceptualization of Systems

Whereas Klir's description of system knowledge and a methodology for obtaining it was an abstract system for classifying kinds of systems and system components and for gaining system knowledge guided by this epistemological hierarchy, he did not provide a broad set of examples of how this was to be done. He also described the GSPS in very abstract terms, a computing system that contained systemhood[2] expertise (he imagined an automated expert system driving the internal operations). As with the epistemological framework he did not provide much detail on its operation and uses (see his block diagram, Klir, 2001, 94). In this chapter we begin to work out some important details of how a real system for system understanding, fulfilling the potential of the GSPS, might be realized and applied. Klir's examples tended to be mathematical or logical and relatively simple. While he did work with fuzzy systems concepts (as described in Chapter 3), the kinds of systems problems called 'wicked' (Checkland, 1999), e.g. social systems involving human decision making, were not represented. The intent of the current work is to bridge the gap between 'hard' systems knowledge and 'soft' systems methodologies and soft systems knowledge (Checkland, 1999). The examples we will demonstrate in chapters 6 and 8 will demonstrate how the method of deep systems analysis can be applied to such soft systems and wicked problems.

Readers are encouraged to explore Klir's work, especially (2001) as it was extremely influential, or certainly inspirational in what follows.

5.2.2 Checkland's Soft Systems Methodology

One of the more influential voices in the systems thinking and its practice in the realm of "human activity systems" was that of Peter Checkland (mentioned above; see especially

---

[2] This is Klir's term which we take to be essentially what we have been calling systemness. He considers the world to be comprised of things that have specific thinghood qualities, e.g. color or size, but also systemhood qualities, e.g. the characteristics of being a system that transcend thingness.

Checkland, 1999) and his conceptualization of what he called "soft systems methodology[3]" (SSM). He recognized what he felt were some fundamental differences between engineered systems such as airplanes, space shuttles, and submarines, and social-based systems such as a corporation or non-profit charity. The former ones are characterized by 'hard' requirements for behavior and engineers need only detail the specifications for performance, costs, and such, to have a basis for implementation. Human activity systems (HAS), on the other hand, have generally messy requirements for what the activity should be, even when the organization has well-articulated goals in mind. Associated with the non-hardness of HASs is the fact that human participants are the agents making decisions and taking action, and humans are notoriously not rational in the same way a mechanical system is rational (Kahneman, 2011). Moreover, humans are subject to extensive noise and distortions in their perceptions as well as suffer from too much influence from ideological beliefs in their decision-making. In other words, it is generally the human factors that make decision processes in HASs messy and wicked[4].

Soft-systems thinking, as Checkland characterizes it, derives from actual experiences of people trying to use general systems thinking to do analysis of wicked problem domains and finding that the engineering approach, hard-systems thinking, could not succeed in dealing with understanding such problems. On the face of it this seems like a reasonable conclusion and there are now two distinct schools of systems analysis in play, one for the so-called hard systems and the other for the soft ones. Figure 5.1 is adapted from Checkland (1999). It shows a process that is followed in SSM.

---

[3] By 'methodology' Checkland means a set of principles applied to a family of related methods developed to address a particular kind of problem domain.

[4] The term 'wicked' is applied to problems that are too complex and are greatly underspecified so that finding solutions is highly problematic. See the Wikipedia article: https://en.wikipedia.org/wiki/Wicked_problem for more background. Accessed 1/28/2018.

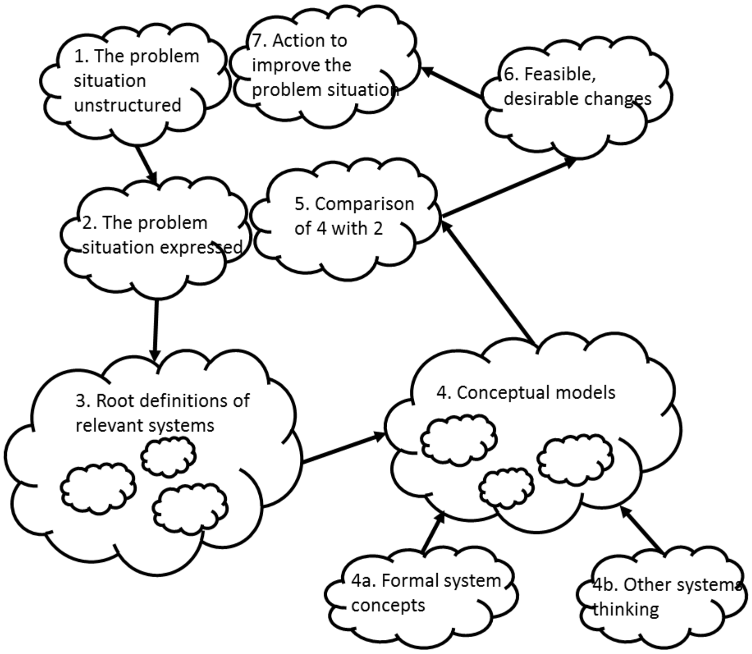

Fig. 5.1. Checkland's general outline of SSM, replicated from Checkland, 1999, Figure 6, Chapter 6, page 163.

5.2.3 Synthesis

What is the actual difference between Klir's hard (and abstract) system and Checkland's soft systems? The framework definition presented in Chapter 3 provides a way to see that the differences might be best characterized not as a schism or dichotomy, but as a matter of degree of complexity and levels of organization. For Klir systems were abstract representations of 'things'; he characterized the systemness as "thinghood." He sought to boil systems down to pure mathematics. Many other systems scientists and engineers have come down on Klir's side but do not take the extreme constructivist position that Klir did. They view systems in the world as real and not just mental constructs. Still they reserve the position that much of the systemness associated with these real systems is a matter of human consciousness. What we tried to do in chapters 2 and 3, culminating with Equation 3.1 and subsequent equations, is show that there is realness to systemhood that exists without the need for human observers. We argued, in fact, that the human brain itself is a computation system that is innately programmed to capture and encode systemness in the world. For Klir a system, S, is just the tuple, <T, R>, where T is the set of things that comprise S and R is the set of relations between things. This is a very sparse definition and has been echoed by many other systems scientists.

Checkland could not reconcile the simple mathematical definition with what was for him the reality of human activity systems – messy "wicked" problems. With humans in the loop, he could not see how one could use the "hard" methods of engineering and mathematics to completely understand these systems. The vagaries of human behaviors made that impossible. The vast majority of systems practitioners today still follow this line of thinking, not without cause. But is it a sufficient cause to warrant the seemingly binary schism between hard and soft systems? We do not think so.

Equation 3.1 and the recursive definition provided in Chapter 3 provides a structure that can capture (at least in principle) even the behaviors of human beings. For example, The *H* object, the history or memory capacity of a system (an example of which is Equation 3.8), provides a very flexible mechanism for including detailed biographies not unlike the data on user visits to web sites analyzed by “big data” algorithms to learn something about that user and make predictions about where they might go next, or what ads they might like to see. The history can be used to track the pattern of a person’s decisions and actions and thereby provide a model with a basis for predicting future behavior. Of course this must be proven in the context of the system of system understanding being proposed in this book. But the existence of big data sets and their uses in modeling human behavior is already being done. There is no reason to believe it could not be expanded and refined in the context of soft system understanding.

Chapter 3 also provided an integration of several modes of communication and did not restrict itself to the mathematical definition alone. We proposed a language of system that can be expressed in, essentially, natural (verbal) language as well as graphical. We think this provides the resolution to treat these so-called soft systems in the same ‘formal’ way one would treat the so-called hard ones. In Chapter 8 we will see this proposal in action, systems analyzing the human society economy (at least partially) which will definitely require that we take human beings into account.

With that we turn to the procedure for deep systems analysis.

5.3 Obtaining Knowledge of the General Systems

5.3.1 Formal Procedures

The point of this chapter is to develop and explain a set of formal procedures for analyzing any system, but particularly complex adaptive and evolvable systems. The complexity of these systems is so great that anything less than a formal method would easily get lost in the process. We know this is the case because it has been demonstrated time and again. Take, for example, the CAES we call ‘the economy’ (see Chapter 12). Anyone who has given the workings of the economy any thought has conceived of it as a ‘system’, but generally not in the way we propose. That is, they realize that there are many complex parts that interact with one another in complex ways but they rarely go much beyond this intuitive stance with respect to trying to better understand how the economy as a whole works. Formal methods as are found, for example, in the study of econometrics have been developed based on the typical modeling approach – best guess the variables and the equations that govern them. But as is becoming abundantly clear, this approach does not produce models that have any kind of valid predictive power. The problem with econometrics has been that the underlying assumptions of neoclassical economics (the academic version) are not based on any kind of reality! We will revisit this issue in Chapter 8 to show how a systems approach to economics produces a very different set of assumptions and, hence, different predictions about how the economy will behave in the future.

The procedures described in this chapter are formal in that they follow the principles of systems science and the definition of system given in Chapter 3.

5.3.2 Representations of the System

In this chapter, we will use several different representations of a system. The basic representation of a system is, of course, in the language of system, systemese as presented in Chapter 3. But there are different ways to structure the system descriptions that achieve different purposes. These are all interrelated and cover the same data, but provide different ways to perceive or use the data. The data itself constitutes the ‘knowledge’ of the system and is the core representation. It is captured and stored in readily retrievable forms in the knowledgebase (see Chapter 7). This is a database system with the schema for relating data elements defined by the formal definition in Chapter 3.

A primary representation insofar as human perception and interpretation will be the various system ‘maps’ or flow diagrams as we have been using. The map employs graphic icons that each represent parts of the system and show how those parts link together explicitly. See Figure 5.11 below to see a system map.

A third representation is the system ‘tree’ diagram. This representation is, essentially, the system map viewed as if from the side with each sub-sub…system drawn at the appropriate depth in the tree. Figure 5.1, below, demonstrates this kind of representation.

The fourth form of representation is the set of equations that describe all of the subsystems’ behaviors. All of the boundaries, flows, and interfaces, etc. are implicitly part of this representation even though not readily visible as such.

5.3.3 A Preview of the Most Complex Systems of Interest

In the Introduction, we introduced a type of system based on complexity and capacity to endure changes in the environment, the Complex, Adaptive, and Evolvable System (CAES. We further elucidated the nature of these systems in relation to the hierarchy of increasingly complex systems that have evolved through ontogenesis in Chapter 2. We have not, however, explicated the nature of a CAES sufficiently to make its existence relevant to the project at hand, namely the analysis of truly complex systems. In this section we will provide a short preview of the nature of CAESs in order to rectify that shortcoming for now. This will be important because in Chapter 8 we will be delving directly into a CAES, namely the human social system’s economy, using the methods described here, so a short preview of the nature of a CAES at this point will be necessary to make progress. For those who don’t mind jumping around, we point out that Chapter 9 and Part 3 will provide a more complete exposition of the nature of the CAES model. What we offer here is just an appetizer!

Put simply, a complex system is one that contains many heterogeneous parts and many levels of organization as covered in Chapter 3. At higher and higher levels of organization the complexity of subsystems includes the ability for those subsystems to obtain the capacity of adaptability or the ability to change internally in order to compensate for changes in the environment that have impact on the functions of the whole system (c.f. Mobus & Kalton, 2015, Chapter 9). For example, a system has the internal capacity to compensate for changes in the external temperature by increasing its internal temperature (warm-blooded animals). The underlying mechanisms are cybernetic subsystems that are involved homeostasis and other response mechanisms.

5.3.3.1 An Adaptive System

Briefly, here, an adaptive system is one that is able to sense a change in a critical environmental parameter (e.g. temperature) and alter its internal operations in order to compensate for that change. The change, itself, must not be radical or outside of adaptable boundaries; the system is pre-designed to accommodate the range of changes but any changes outside that range will be detrimental and result in damage to the system. Homeostasis is an example of a mechanism that provides adaptivity. As long as the homeostatic range is within the preset (phenotypic) capacity of the system, the latter can adjust its internal operations to compensate. Adaptivity depends on a system having the capability to sense the change in the environment that is relevant to its functioning, make an appropriate decision to act to compensate for the change, and have the range of optional actions, what we call “requisite variety,” and the necessary power in action to make the compensation effective. We will develop these ideas more fully in Part 3 of the book.

5.3.3.2 An Evolvable System

Adaptability can make a system resilient in the short-run (assuming that the nature of the changes that warrant adaptive responses are within the ranges of adaptation built into the system). But in the case where changes are trending in a direction that will eventually lead outside the adaptive range that is built into the system, for example, the warming of the global atmosphere or the acidification of the oceans, then an additional mechanism for affording a greater capacity to modify the internal responses to those changes is needed. Evolvability is the ability to make or allow changes to internal mechanisms so that the system can accommodate changes beyond the typical range and it is a further method for achieving long-term adaptability to major changes. For example, in biological species, individuals may “suffer” mutations that do not immediately impact the phenotype under nominal (ordinary) within-range conditions, but under certain stressful conditions (i.e. a change in the environment pushing the limits of the range of pre-adaptive response) can be released so that some individuals exhibit an increased ability to adapt to the stressing changes. In a large population there will be a critical number of individuals with this particular capacity that they will be more fit than their conspecifics and survive the changed conditions thus leading to a new, more fit population. When enough such favorable mutations accumulate the individuals in this population may be so different from related (and historically ancestral populations) that they are effectively incapable (or unwilling) to mate, should they come back into contact.

The key to biological evolution, and evolvability within a species, is the fact that it is successful because of the size of a population that permits a large enough number of non-directing mutations such that at least a few of these will prove advantageous when the change comes about. When population sizes fall below a critical level, there would not be enough individuals with the "right" mutation to constitute a viable subpopulation. The population goes extinct.

Another proviso of this scheme is that the rate of changes must not exceed the rate at which potentially useful mutations can accumulate. For example, the current rate of global warming is extremely high in comparison to prehistoric events of this sort. So, it is very worrisome that many species, especially of higher multicellular organisms, may not be able to evolve at a sufficient rate to ensure viable individuals in any population, no matter how large.

Human beings are actually transitional as evolvable systems. They cannot modify their physiologies to be more adaptive, for example being more heat tolerant. But they can modify their behaviors to achieve, effectively, the same end. This is because the human brain, with the remarkable capacities of the neocortex and, particularly, the prefrontal cortex, are able to act as an evolvable system, learning new concepts and altering behaviors to adjust to changing environments in ways that other animals cannot. Of course, not all human beings are adept at learning new knowledge and changing their concepts and behaviors (Mobus, 2019). Only those with a sufficient capacity for "wisdom" are astute enough to observe the changes in their environments and intentionally alter their mental states leading to adaptive behaviors.

Human social systems, which includes societies, organizations, institutions, and government, to name a few, are the ultimate in evolvable systems in which intentional 'mutations' lead to a long-term sustainable, viable system. In Part 4 of this book we will revisit this aspect of social systems as evolvable based on the concept of intentional-organization and intentional evolution brought about by the nature of human consciousness and individual human evolvability.

5.3.3.3 CAES as an Archetype

A CAES is an archetype model, to be fully explicated in Chapter 9. An archetype model is one that specifies all of the working parts of a whole system of the type. Used in analysis it guides the analyst by asserting what is to be expected to find as the analysis proceeds. It suggests questions that should be asked in the process of discovery. The patterns and sub-patterns presented in the archetype are found within the actual system being decomposed. Alternatively, used in design of a system, the archetype acts as a template for the design. Recursively applied as in analysis, i.e. designing higher order systems with lower order CAESs, extremely complex systems with variations of adaptivity and evolvability can be composed, with appropriate interaction flows, to form the higher order CAES.

The actual origin of the CAES archetype model derived from an amalgamation, integration, and of the works of many previous systems thinkers. A partial list would include: Ashby (1958), Beer (1959, 1966, 1972), Boulding (1956), Checkland (1999), Churchman (1960, 1968a, b), Churchman, Ackoff, & Arnoff, (1957), Forrester (1961), Fuller (1968, 1970, 1982), Klir (2001), Koestler (1967), Miller (1978), Morowitz (1968, 1992, 2002), Odum (1983, 1994, 2007), Prigogine (1984), Rosen (1985, 1991), Shannon & Weaver (1949) Simon (1957, 1991, 1998), von Bertalanffy (1968), Wiener (1950, 1961). Along with the best ideas from these workers, the model incorporates more recent views of governance, agency and agent theory, and the theory of nested economies (i.e. that the metabolism of cells is nested within the physiology of a multicellular being such as a human, and that physiology is nested within the extant social economy which supports life). Perhaps the biggest influence on the author's development of this archetype that is isomorphic across living and supra-living systems was the work of Stafford Beer (1959, 1966, 1972) who developed the Viable System Model (VSM) that includes many of the features found in the CAES model. More will be said of this in Chapter 9.

Since our main interest is in CAESs involving humans and decision processes (humans as agents) we will tend, in these pages, to focus on human social system subsystems. An extensive review of many different kinds of system with the properties of adaptivity and evolvability, including biological species, human beings as learners, human social systems, and ecological systems as subsystems of the Ecos have verified the main points of the CAES subsystem archetypes.

5.3.3.4 CAES Subsystem Archetypes

Chapter 9 will provide an overall description of the whole CAES model. But that model is composed of three sub-models that interrelate with one another, are tightly coupled. Every CAES will have these three sub-models. Chapters 10, 11, and 12 will proceed to treat each of these as a focus of discussion while pointing out how they cannot be handled as completely independent of one another. The sub-models are: agent and agency (how decisions are made and turned into praxis), economy (how work gets accomplished to provide the system with necessary goods and services for its own use but also for export), and governance (how decision types are distributed across the society and economy). The human social system is (or should be) a complete and viable CAES. But it is comprised of sub-social systems, organizations, institutions, and governments, which are themselves CAESs (or should be). In other words, larger CAESs have smaller CAESs within. Each smaller CAES has its own set of subsystems; its own agency, economy, and governance. Moreover, each of these subsystem CAESs may be found to be composed of yet smaller viable CAESs (or should be) such as departments, committees, and so forth. Finally, all of these sub-subsystem CAESs are composed of people (and increasingly AIs) who are obviously agents, but are also CAESs in their own right. Remember, the human brain is capable of evolving new thoughts and behaviors. Each human's economy is what we call its physiology. The brain is the main governance subsystem. The lowest level subsystems having humans as component parts may also have many CAS artifacts (e.g. computers) and many more simply complex or simple systems as tools for accomplishing the purpose of the CAES.

In the following chapter we will explore how the methods described in this chapter apply to all kinds of systems, not just CAESs. But in Chapter 8 we will return to the analysis of human social system and go deep into the analysis of the social economy, in particular we will show how what most people think the economy is, is not at all what a viable CAES economy would be. A fuller explanation will need to await Chapter 12 where we reveal what a viable economy looks like from the standpoint of how a CAES works. But we think the reader will be able to see the main thrusts of the arguments given in Chapter 8.

5.4 "Deep" Analysis

As mentioned in the Introduction and discussed in the last chapter, the term 'systems analysis' has been used in several different contexts since the mid-early 20th century to the present. In engineering the design of a 'hard' system (in Checkland's terminology, 1999, page A16) the term has been applied to the determination of what are called 'requirements' and to an analysis of designs that would fulfill those requirements. What the 'device' or system was supposed to do was a given and so the analysis phase was limited to how it should do it most efficiently and at least cost. In the field of information systems, where the term actually gave rise to a job title, 'systems analyst,' the work of analysis has never been to actually analyze the real system (a 'soft' system in Checkland's terminology), that is the underlying organization to be served by the information SUBsystem[5]. In a vein similar to the engineering of physical artifacts, the purpose of the computational and communications subsystem was assumed a given. A 'needs' analysis, in this field, amounted to little more than the same kind of requirements gathering as done in engineering[6]. The analysts asked the users/stakeholders what their needs were, assuming that the users actually knew what they needed (as distinct from what they wanted). This approach to analysis is at best partial and, depending on the experience and 'wisdom' of the analyst, open to serious vacancies in the completeness of the resulting knowledgebase. Requirements are a pale image of the actual system.

---

[5] An information system is, in fact, just a subsystem of the larger supra-system which it serves. In Chapter 9 we elaborate the role of information systems, or actually the network of message flows and processing that is part of the governance subsystem of a CAES, like a corporation.

[6] The field of software engineering has always had difficulty being just like hardware engineering. Part of the problem stems from the nature of software, which is subject to a vast array of methods for achieving complex functions. Software development is more often like prototyping than product production. In Part Four chapters we will return to this issue and suggest ways in which an overarching systems engineering process could be used to ensure better software development outcomes.

What, then, is *real* analysis? The word 'analysis' has several related definitions and is used in different disciplines in slightly different ways based on the medium of study in the discipline. The number one dictionary definition, however, encapsulates the broad meanings of the term in all of the various disciplines. This definition is from Dictionary.com: "1. the separating of any material or abstract entity into its constituent elements (opposed to synthesis)." The number 2 definition amplifies why separating something into its constituent parts is important. "2. this process as a method of *studying the nature of something* or of determining its essential features and their relations" [emphasis added]. In other words, the process of analysis is taking something apart (carefully) to find out what it is made of, and, hopefully, how it works.

The main process of the sciences is analysis. The philosophical and practical belief that something can be understood by taking it apart, called reductionism, and that the ultimate behavior of the whole thing is nothing more than the collective behaviors of its constituent parts has been the top guiding principle for science since the invention of empirical methodology.

In the last chapter we introduced the analysis of a system and considered the opaque-box/transparent-box procedures. Empiricism requires that as one deconstructs a system one needs to run functional tests on the components. This is not much of a problem for reverse engineering an actual 'box' piece of hardware. The parts of a computer still function properly even when they are not in a whole computer, if you set up the tests appropriately. However, hearts or livers, even when kept artificially alive, may not function at all as they do in the body of a living organism. Possible setups for testing their functions would have to be elaborate and extensive, and, in any case, assumes that the experimenter knows what effects the rest of the body has on those organs in their *in vivo* states.

The methods of systems analysis (as described above) have to date been only partially about taking the system apart. "Requirements gathering" for a physical device or product, if done very carefully, constitutes a kind of virtual analysis, i.e. the analysis of something that does not yet exist except in the minds of the prospective 'users.' In the software world using the same notion of requirements gathering is barely an analytic process. For one thing, the presumptive analysis is not being carried out on the actual system (where the work is done) but on the information subsystem[7]. The relationship between what the 'user' knows about the information requirements and the actual requirements of the work system depend entirely on that user's depth of knowledge of the system itself. All too often that depth is not great. Systems analysts in the software development business would do much better if they analyzed the actual work system (examples to be provided later) to find out what its information requirements are.

---

[7] Systems analysis in the computer-information systems world (or management information system, MIS, as it used to be called) tended to be strictly an exercise in analyzing the computing/communications/reporting aspects of a subsystem that was supposed to serve the work system. Only occasionally did systems analysts ever venture into a deep analysis of the work system to verify that the "user's" requirements matched the decision processes needed to manage the work processes.

In the systems world, analysis has gotten something of a bad rap owing to a misconception about the issue of functionality of the whole versus the functions of the parts. The phenomenon of emergence further muddies the waters leading whole systems thinkers to statements such as: “the whole is greater than the sum of its parts.” Aphorisms like this are useful to remind researchers that when engaged in analysis they cannot lose track of something very important, namely that the parts have relations to one another in the intact system and that it is those relations that give rise to the behavior of the whole (Rosen, 1985, 1991). Deep analysis can include several different ways to observe this dictum. No relations need be injured in the analysis of this system.

5.4.1 Inhibitors

This subject was touched upon in Chapter 4. Here we reemphasize two major inhibitors to gaining deep systems knowledge.

When quarterly profits are in question, these inhibitors will come to the fore to reduce the effort put into deep analysis. As mentioned in the Introduction and Chapter 4, this is ironic since short term profit gains are all too often more than offset by longer term losses due to the project or product failures, or due to the long-term maintenance life-cycle costs that are, in turn, due to poor design (due to inadequate knowledge about the concrete system). The inhibitors to following a principled process such as will be described in this chapter are both real threats, if not carefully considered and poor excuses to minimize the analysis phase in favor of getting right into design.

The profit motive drives most of what we do, even in supposed non-profit operations, like academia and science. The desire to do the most possible with the least drain on capital and labor is just a fact of any society that runs on liberalist ideals. Novelty and innovation – being first to market, for example – and many similar motivations get in the way of doing a thorough analysis. And this is why we too often don’t get things right the first time through. There is an old saying among engineers (chaffing at the insistence of marketing and management to get the thing designed quickly) is quite well understood in that field: “Why do we never have enough time to do it right the first time, but always enough time to do it over?” It captures the dynamic of modern projects in a competitive marketplace.

5.4.1.1 Complexity and Analysis

The degree of difficulty associated with deep analysis depends generally on the complexity measure of the system being analyzed. Recall that complexity in this sense means the logical depth of the hierarchy of organization coupled with the numbers and kinds of components and relations at any given level. It should be obvious that the more levels there are, the more numbers of kinds of components, etc. the longer it will take to accomplish a deep analysis. It will also require much more work in decomposition and data capture. In most current systems analysis projects this is often called the “scope” problem. The larger the boundaries of the system, the more complex it will be and therefore take more time and resources to accomplish.

In order to determine, in advance of expending the effort to do a deep analysis, it is important to get a firm grip on the complexity of the system. But this may create a conundrum. By definition we do not yet have the details of the system including just how complex it might actually be. Fortunately, we do have considerable experience with other similar systems. For example, we have case studies of many projects in the realm of management information systems from which we can obtain a fairly good description of their complexity.

In Chapter 11 – Governance, we will dedicate a section to the management of the systems understanding process. We will return to the issue of complexity and how it affects the overall success of projects, and especially how to handle the scope problem.

5.4.1.2 Cost of Analysis

Complexity plays into another aspect of analysis and that is what appears to be a nonlinear relation to costs. It seems that as complexity rises the costs involved in doing it rise more quickly; perhaps not exponentially, but a higher order quadratic for sure. Indeed some authors have measured complexity in terms of cost estimates (or results). Cost overruns are generally attributed to what some have called “runaway complexity.”

Costs include manpower, capital, equipment, and time. And a substantial portion of project management is given to managing costs to stay in budget. This is the way of capitalism, but it is also the unfortunate cause of failing to do deep analysis before generating designs (as described in Chapter 13 (the unlucky chapter). And, as described in the Introduction, the management constraints on systems analysis generally (all too often) leads to even higher costs later when the built system fails to deliver the real service needed by the stakeholders.

Deep analysis is going to be costly and that should be clearly understood at the outset. Deep analysis means taking your time to discover all of the relevant knowledge of the system so that nothing important is overlooked. From this dictum there is no escape. The problem is with financial management imposing unrealistic budgets on projects to produce products that marketing management has enthusiastically, if ill-advisedly, sold to a customer. There is no solution for this formula and enterprises will continue to suffer failures so long as they keep getting the cart in front of the horse. This subject will be taken up again in Chapter 11 on governance.

Using deep analysis is a strategic approach that is meant to save overall lifecycle costs and thus improve profit performance in the long run. More importantly, for companies, governments, and NGOs thinking about their long-term thrivability, it will ensure customer satisfaction more surely than happy-talking them.

5.4.2 The Objective – Reductionism with Maintenance of Relations

Deep analysis is a form of reductionism in the scientific investigation and understanding sense, but with a critical difference. Because the procedures are dependent on the a priori formal definition of system (and the language) the deconstruction of a system is guaranteed to preserve the functional relations between subsystems, thereby preserving the whole system's behavior.

This is reductionism in the sense that the whole really does depend on the behaviors of the parts, but only in the context of the behaviors of all of the parts taken together. When we get to the generation and use of models of systems, based on what our analysis process produces in the knowledgebase, we will see how this is guaranteed in our analysis methodology. The preservation of relations derives from the analysis based on using the system language and on the formal definition of system from which it comes. By starting with this formal definition we build into the process the fact that the relations are primary characteristics. In effect the analysis could not proceed if we ignored them, even a few of them.

5.4.3 The Three Phases of Systems Analysis

We will briefly review the three phases of analysis as described in the prior chapter and then provide complete descriptions of each phase in the next section. For the present, realize that the descriptions of objects identified in the sets, graphs, and lists comprising the data about a system, $\boldsymbol{S}$, will be collected in forms appropriate to the type of object. Actual examples of these forms will be provided in the section below. And examples of filled out forms will be provided in the next chapter.

Recall from the last chapter that we identified three phases for systems analysis. Two phases are somewhat preliminary, but absolutely necessary in order to succeed in the third phase where the majority of the work will be done. In this section we will briefly review and explain the significance of these three phases. In the next sections we will expand the methods and procedures to be followed. The three phases are: system identification, environmental analysis, and recursive deconstruction.

In describing these phases it may at first seem that they are carried out in a linear fashion, one after the other. However this is not the case and should not be assumed. The entire process should rather be considered as potentially iterative. That is, in conducting any one phase it may become apparent that something important was missed in the prior phase. For example, in doing an environment analysis it may become evident that there is an unaccounted-for flow from a newly identified entity that mandates re-entering the system identification phase and boundary condition analysis. When doing the recursive deconstruction of a system, it is possible that discovery of a subsystem that receives a flow not accounted for in the parent system leads to re-entry of the higher-level organization analysis. The accounting system (tracking the inputs and outputs) provides a trigger that signals such a need for iteration over the higher level. This aspect of the analysis should be kept in mind as we describe what transpires in each phase. In the next chapter we will provide a few examples of this "iteration over recursion" aspect. One way to visualize this is how the science process works. If failure to replicate results occurs in an empirical study, new experiments are designed to check the original findings. Science is thus recognized as a self-correcting process overall and over time.

5.4.3.1 Identification

This phase is tightly coupled with the next phase in the sense that the analysts may need to engage in first one then the other in a piecewise iterative fashion. The phases should not be confused, but there are times when a system boundary is not immediately identifiable until one also starts considering the environment. For example, it may be difficult to identify a boundary and the conditions of flows across that boundary until one has identified candidate sources and sinks.

Nevertheless, the system of interest (SOI) can only be understood when the boundary of the system has been established and the interfaces it has with the environmental entities have been identified. We will work with the formal definition of system and the language of systems from Chapter 3 to do this work.

The first order of business is to analyze the boundary of the system to identify the input and output flows through that boundary by way of the specialized subsystems called "interfaces" (Equations 3.7 and 3.8). At this stage we do not know what inside the SOI is producing the outputs or receiving the inputs, but we can identify the points at which they traverse the boundary[8]. Then, by substitution individual elements of $\boldsymbol{C'}$ for receiving interfaces ($\boldsymbol{r_{i,0}} \in \boldsymbol{I_{0,0}}$) and elements of $\boldsymbol{C''}$ for source interfaces ($\boldsymbol{r_{j,0}} \in \boldsymbol{I_{0,0}}$) from Eq. 3.8 we can then construct the tri-partite graph, $\boldsymbol{G_{0,0}}$, mapping inputs to and outputs from the SOI to environmental entities (see Figure 3.9B).

Once the $\boldsymbol{I_{0,0}}$ set is completed (or as complete as possible for any given iteration) we are now ready to complete the analysis of the boundary object B (Eq. 3.7). This requires a thorough analysis of the boundary's physical properties to be encoded in the set $\boldsymbol{P_{0,0}}$ in Eq. 3.7.

Ultimately the objective of system identification is to specify the 'grand' transfer function of the system process; to be able to say what the outputs are (quantitatively) and how they are arrived at given the inputs (quantitatively)[9]. For this we will be constructing the $\boldsymbol{T}$ set from Eq.

---

[8] We will be explaining the nature of the protocol component of the interface in the next section.

[9] It need not always be the case that systems must be identified with quantitative relations. There are many fuzzy systems for which we are keenly interested in qualitative relations and may not need to necessarily quantify every aspect. However, we maintain that this does not mean such systems are not ultimately quantifiable. For example, psychology has been primarily interested in qualitative descriptions of the mind, such as inferring mental states based on behaviors and first person reports. Modern psychology works hard to find quantitative measures for these aspects. And with the advent of neurobiological imaging technology, the ability to 'measure' directly

3.2 through use of the ***H*** object. We use Equation 3.9 from Chapter 3 to start constructing (at least at this stage) candidate functions for each of the inputs to and outputs from the SOI at time intervals defined by $\boldsymbol{t_{0,0}}$ in Eq. 3.2. As much as is possible from the opaque-box analysis and the instrumenting capabilities of our analytic tools we begin to characterize each output in terms of things like flow rates or volumes per unit time, substance characteristics such as composition, and other relevant factors (e.g. energy content from a fuel processing plant). This is a first pass at resolving questions about how the system processes its inputs to produce its outputs. The final resolution must wait until we get deeper into the third phase, where the details of sub-processes that actually do the work will be discovered. In the below descriptions we will introduce the use of machine learning techniques to do causal analysis of the outputs-given-inputs history that will help us characterize very complex transformation relations between inputs and outputs. Traditionally such transformation formulations have been characterized with sets of differential equations, and for simple systems this is still a legitimate approach. However for very complex systems the transformation functions may be nonlinear and multivariate (with noise also possible) so that a clean set of functions may not be possible. Modern machine learning methods (e.g. what is called ‘Deep learning’) have been used to capture input/output relations that cannot be characterized so neatly.

5.4.3.2 Environmental Analysis

There are a number of components and aspects of the environment of any system that have to be grasped in order to proceed. First is the enumeration and characterization of the various entities and external reservoirs that constitute the sources and sinks for the flows of inputs and outputs (and forces) that impact the system.

This characterization does not go so far as to develop models of the entities themselves. Those are outside the boundary of the SOI. But it does include determining things like the flow rates of substances and messages from/to the entities and external reservoirs. As an example of the latter, consider the carbon dioxide and other greenhouse gasses emitted by the human social system (see Chapter 8) into the atmosphere and indirectly into the hydrosphere. Both of these reservoirs have absorption characteristics that we need to know in order to consider rates of output from human activities. When atmospheric scientists are studying the atmosphere as a system, receiving the CO_2 emissions, they will be concerned with how the gasses impact their SOI (and, of course we are all concerned with the feedback of that impact on things like the climate).

Environmental analysis will involve Equation 3.5 in Chapter 3 where the ***Src*** and ***Snk*** sets and their associated flows will be completed.

---

operations in the brain while subjects are behaving or reporting their thoughts is rapidly changing psychology to a quantitative systems science.

But, how do we go about discovering all of the inputs and outputs for a system when the environment is complex and there are potentially many different kinds of sources and sinks as well as many different individual entities in each kind? Further complicating things is that many entities are not purely sources or sinks. In CASs and CAESs the SOI is actively communicating with those entities and even observing other entities not involved directly in flows of materials or energies. Thus a source of a material resource may also be in two-way communication with the SOI; we will see examples of this in Chapter 8 on the economy subsystem and Chapter 11 on governance. This means the source of material may also be a source of message and a sink of messages. Moreover, a sink for products might also be a source for some other material. The same entity may be both a source and a sink of many different flows. In the examples to be given later in the book we will show how to handle this graphically and in terms of the programmatic treatment. All of these (and other) complications might, at first, seem to make the parsing of the environmental entities and flows extremely difficult. How might we proceed in an efficient procedure?

The answer is to start with a *generic* or *archetype* model of a CAS or CAES (to be elaborated in Chapter 9) in which we represent at least one input of energy, one of matter, one of message and outputs of heat, product/behavior, and wastes. Each of these are top ontological categories that we established in Chapter 2. Complex systems will generally have many inflows and outflows in each of these categories, with subcategories further differentiating them. For example, a system like the economy (see Chapter 8)[10] gets many different forms of energy to do its work. The economy obtains some forms of energy such as fossil fuels, real-time sunlight, hydroelectric power, and so on. The category ‘energy’ can then give rise to (at least) three categories of subcategories of energies. When modeling the economy it might not be necessary to be more specific (i.e. sub-subcategories). All coal mines, for example, could be lumped into one giant mine and the total production of coal (of all kinds) could be aggregated in some appropriate flow rate measure (e.g. tons per day).

Or a complex system might produce many different kinds of products (low entropy material things) or services (performing work on other systems). In section 5.5.1.2 below we recommend the identification of the system of interest starts with analysis of the products (and other outputs). Using the generic model one can then begin to formulate analytic questions about what subcategories of products or wastes there might be, as with the handling of energy above.

This same strategy can be used for all of the generic inputs and outputs. However, a special case might exist for message flows. It turns out, in the general cases for CASs and CAESs, that there will be messages exchanged between all sources and all sinks as mentioned above. Rather than just represent, for example, a flow of material from source A to the SOI, we might as well acknowledge a priori the existence of message channels between A and the SOI even at this

---

[10] See, in particular, figures 8.3 and 8.4 to get a sense of how this decomposition of environmental entities and flows is accomplished by moving down from general categories to more specific ones.

stage of analysis. As noted above we will provide examples of doing this in both chapters 8 and 11.

5.4.3.3 Recursive Deconstruction

Once the SOI has been identified and characterized along with its environment it is time to start the recursive procedure of deconstructing the opaque-box to find out what is inside. The variety of techniques for exposing the internals of any system will depend on the material medium of the system. We will provide several examples in the next chapter. For now we simply refer to the act of discovering subsystems and the flows between them (or, more generally, the interactions between them.)

The procedure may be tackled either using a breadth-first or a depth-first approach. In practice some combination of these will be appropriate. In general, however, a breadth-first approach is the more preferred since going deeper into one subsystem will create a set of unanswered questions about sources and sinks that cannot be resolved until all of the legs of a depth-first approach are completed. At best, depth-first will only expose candidate sub-subsystems. This can be useful as a "look-ahead" procedure to find what potential sub-subsystems will be coming up in the analysis but will not yield a complete mapping of flows and influences that would be needed to provide consistency checks. For our present purposes we will focus on the depth-first approach since it is the most efficient one in terms of keeping track of all relevant flows at each level of organization.

The breadth-first approach means that all of the subsystems of any given level of organization will be determined along with the map of flows between them. Once the subsystems and the flows are determined each subsystem will be similarly deconstructed as if it were the SOI. Assuming a complete analysis has exposed all of the subsystem inputs and outputs (with their boundaries and interfaces) it becomes possible to switch to a depth-first approach local to a specific subsystem. For example if the real questions to be answered only involve a single subsystem, the analysis of all the other systems will act to substantiate the environment of that subsystem as it becomes the focal SOI. We will see an example of an analysis that starts at a very large scope (the whole Earth!) and narrows the analysis down to a sub-subsystem, the economic system of the human social system, in Chapter 8.

5.4.3.4 The Final Outcome

The product of a deep systems analysis is a knowledgebase containing all of the relevant data describing the objects (subsystems and components), flows and influences, and transformation functions (relations). Figure 5.2 shows a hierarchy tree with the relations made explicit. Information captured in the knowledgebase allows for the reconstruction of such a tree. The tree on the left shows a macroscopic view (also called a "collapsed" view) of just the major subsystems hierarchy, down to the level of components. Not shown are additional details of objects that are defined in the formal definition of Chapter 3 such as flows and interfaces. The expansion shown on the right side of the figure provides a more microscopic view that includes some of these objects. A tree object such as this can be visually "browsed" with selected objects expanded for view. The actual data structure containing the detailed data for any object could be shown at a mouse click. Indeed, those same visual access tools could be used to construct the tree in the first place.

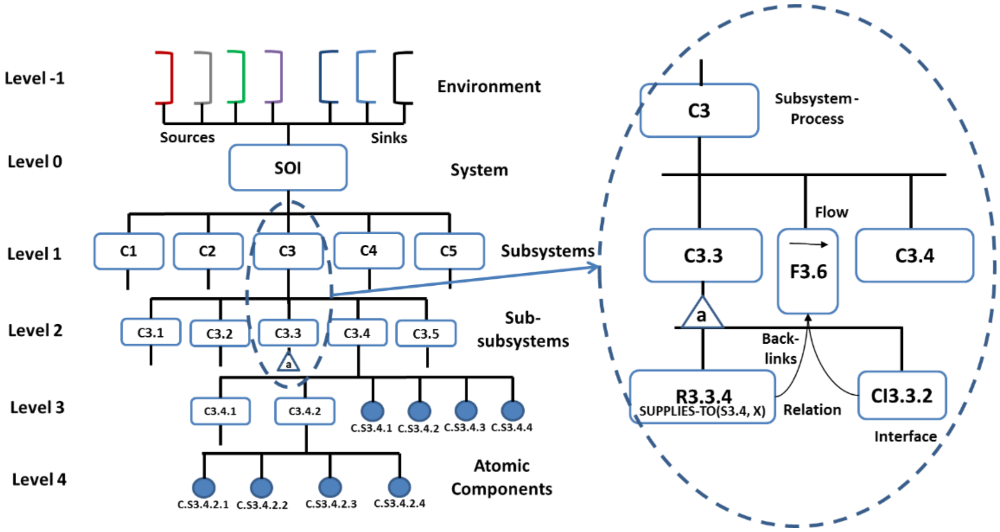

Fig. 5.2. One view of the final outcome of systems analysis is a complete hierarchy tree showing all of the environmental entities/reservoirs at level -1, the SOI at level 0, and the deconstruction tree of subsystems, sub-subsystems…, and components. Subsystem C3 is shown exploded into additional elements such as a flow between C3.3 and C3.4. C3.3 is further decomposed to show details of relations and interfaces. Note that the coding system for subsystems is based on the *C* set in Equation 3.1. Alternatively, by Equation 3.3, the designator C, standing for component, can be replaced by S, standing for system.

Chapter 7 will provide additional understanding of the knowledgebase and how it can be used to manage various other functions of the knowledge understanding process.

Given this brief review/preview it is now time to provide details for how the analysis process is to be accomplished.

5.5 System (of Interest) Identification

The general process of analysis starts with identification of the system of interest, treated usually as an opaque-box (see Figure 5.2 below). At level 0 this involves a complete characterization of the system boundary (especially the list of interfaces), the inputs and outputs, and some preliminary information regarding the sources and sinks. Of ultimate interest is the nature of the transformation of inputs to produce outputs; of raw material and high potential energy going in and products and wastes coming out taking into account appropriate time delays from input to output. The transformation function(s) is characterized as the dynamic behavior of the SOI, meaning that it is characterized as the temporal variations in values of output parameters relative to those of inputs[11]. In this section we walk through the process of system identification as the first phase of the analysis process. The objective will be to populate the formal definition of a system, given in Chapter 3, with actual data objects, such as a flow (of a substance from a source to a sink) with real or design required values of units per time step, frequencies, or other relevant parameters needed to complete the characterization. Here we will be focusing on the general principles and methods for any arbitrary system (the fact of systemness means these will be universal). In the next chapter we provide examples of how these procedures work in practice.

5.5.1 Initialization

How do we get started? In theory we could start with an analysis of the boundary or any arbitrary flow (in or out). This is because we have an a priori model of systemness that is the actual beginning point for all analysis. That is, we have a template of what a system is in general terms; what we are going to call a *presumptive SOI*. We don't necessarily know what the products (or services) are, but we know that the system *must* have them or it would not exist (see discussion in the next section on the nature of 'purpose'). In the engineering sense the products (or services) are givens.

In practice we have developed a structured procedure to be described below. In any case all analyses begin with the consideration of several key questions. By consideration we don't just mean a providing cursory answer. We mean applying hard-nosed reasoning and the use of evidence to back up the answers. By 'key' we mean that these are literally the questions that must be considered for every systems analysis, no matter the complexity or what we 'think' we know in advance.

5.5.1.1 The Questions to be Answered

The most frequent trigger for doing a deep systems analysis is one or more questions coming to the fore regarding a phenomenon that is not understood. Even when the intent is to develop a product or design an organization, the driving force is often a question about how a new product or new organization can perform its functions better. The first key question, then, is: "What is the *purpose* of the system?" This is tantamount to asking: "What does the system do?" And this leads to the follow-on questions: "How does it do what it does?" and "Does it do it well enough?"

In the sciences we often start with the last questions because it may seem as if we have already determined what a system does by historical observations of its behavior. But this can be an illusion! In physics it is easy to identify an interesting system, say a ball rolling down an inclined plain or a pendulum swinging (and undergoing precession). In biology the task is much

---

[11] A form of identification involves only the use of output data, partly described in the Introduction. These methods consider only the output data over time and attempt to derive behavior on this basis alone. Since most real systems are sufficiently complex (and adaptive, if not evolvable) and involve complex internal functions, this approach is reserved for mostly simple physical systems.

less simple. In the case of the ball rolling down an inclined plain, that is not really a system. It does not need to have a 'purpose.' Rather the system is likely to be an experimental setup with measuring tools in place and an observer (the physicist) ready to characterize the action, capture some data, and analyze the later in order to answer some fundamental questions about motion or gravity (forces), etc. So, the purpose of the experiment – the real system – is still to provide information to an outside observer.

Biological systems have their own motive forces and dynamics. The purpose they fulfill is dual. They must stay alive and they must procreate; all living systems are endowed with this programming, even if specific individuals in a population fail to follow through (e.g. worker bees do not reproduce, but the queen passes on their shared genes in their stead.) So with biological systems the questions to be answered are complicated.

Adding to that complexity is the evolutionary problem that in order to carry out the biological mandates, the average organism in a species has to be fit in its environment. That means, it has to be capable of fulfilling its mandates in the average set of situations it finds itself in. It must be generally capable of capturing and using energy to maintain itself (or grow when it is immature) and to either replicate itself (asexual reproduction) or find a mate and procreate. In a sense the fact of the biological mandate makes the 'purpose' question simple. It is the follow-on questions that are much harder to answer.

When we start talking about supra-biological systems, ecological, social, organizational, and so on, the questions of fitness become much more difficult to answer. But this is exactly why we are turning to deep analysis.

In the case of artifact, procedure, or policy designs – of human engineering – the question of fitness is no less important. It is one thing to say that a particular machine should perform this-or-that function. It is quite another to specify the function such that the performance meets the requirements of the environment in which it will operate. Poor fitness in this arena may result in customer dissatisfaction leading to loss of sales in the future. Here, not only may the machine be unfit for duty but the organization that produced it is unfit as well. In this sense fitness means knowing how to understand the requirements and then doing a superior design job to produce the artifact.

Because the overall objective of a deep analysis is to answer these questions, purpose, function, and performance, the logical place to start the analysis is at the output end of the system.

5.5.1.2 Start with the Output

With respect to artifact systems, too often analysts will assume they already know everything about a system's output or behavior and so will tend to focus on the inputs or even the internal processes. And just as often the assumptions about outputs will be wrong, especially with regard to the questions to be answered. For all systems, natural and artifact alike, a good rule-of-thumb in initializing an analysis is to do a thorough study of the outputs relative to the environment. Initially the analyst may have a presumptive model (Fig. 5.3) and have some preliminary ideas about inputs, boundaries, and outputs. This is strictly a conceptual model. It may be a seed for Equation 3.1 at level 0, but the rule to focus first on the output(s) is still in effect.

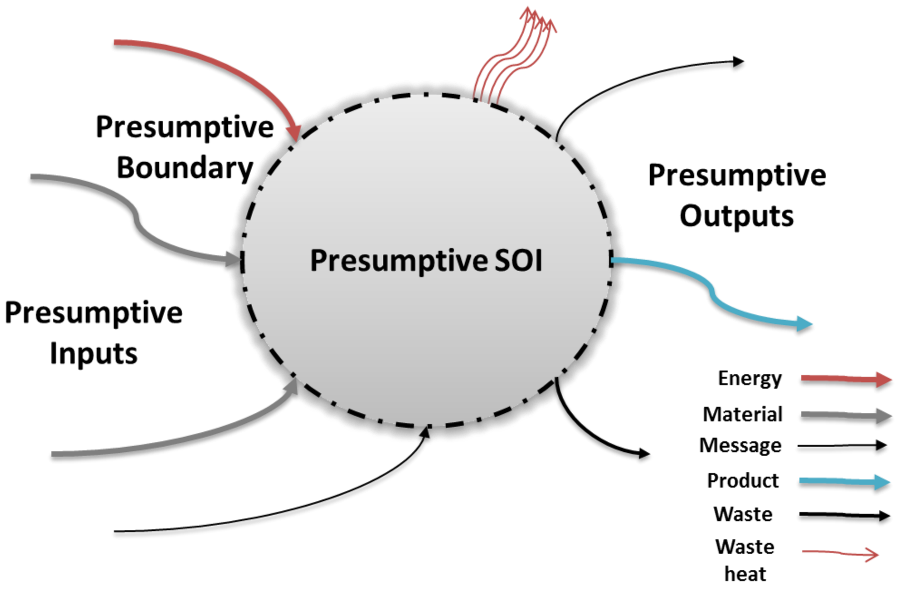

Fig. 5.3. The initial condition for starting a deep analysis may include some presumptions about the SOI, its boundary, its inputs, and its output. Since all concrete systems follow this basic form, this conceptual model is a reasonable place to start.

5.5.1.3 The Presumptive SOI – Level 0

The time has come to begin filling in the details of Equation 3.1, repeated here.

$$S_i^l = \langle C, N, G, B, T, H, \Delta t \rangle_i^l \qquad \text{(Eq. 3.1)}$$

Recall, also, from chapter 3 the elements of Eq. 3.1 at a designated subsystem index, *i*, and level index, *l*, are:

C set of components (or subsystems) of the parent system
N is a graph of components in C and their interconnections, L
G a tripartite graph with Src (set of sources), Snk (set of sinks), nodes in the environment and F, the set of links between sources and sinks and the components of the system
B the boundary object contains a list of interfaces and boundary physical characteristics
T the set of transfer function for each component
H history of components' states

We can now begin defining S_0^0, the system of interest at level $l = 0$. At this point the sets *C*, *N*, *T*, etc. are empty. The time constant, Δt, however must be estimated now as this will guide the selection of sampling rates to be used in measuring input and output flows. To obtain a reasonable estimate for Δt we can use a variety of signal processing methods after collecting the time series data (see more discussion on this below) on the product output[12]. For example we can use Fourier analysis[13] on the time series to identify the range and power distribution of fluctuations (frequencies). At this stage we are still working with rough estimates based on judgments, say for example, what high frequencies represent mere noise and can be ignored (or filtered appropriately). The highest meaningful frequency can then be used with the Nyquist-Shannon Sampling Theorem to derive an estimate of Δt. However, Δt will remain an estimate only until we have many data streams to analyze.

5.5.1.4 Creating the Product Flow and Terminals Objects

A first order of business is to identify the product of the system – the output that gives the system its purpose. For complex systems, of course, there may be a number of products and customers. For example, a corporation has a large number of what we typically call "stakeholders", ranging from shareholders, to nominal product customers, to employees who receive wages. All of these stakeholders are technically customers and receive products (dividends, physical products or services, wages, respectively).

---

[12] As strange as it may sound, even material outputs can be treated as 'signals'. For example, the shipments of products from a manufacturing plant often happen in bursts or pulses (volume or weight) rather than a steady flow stream. The same is true of biological systems. Real concrete systems, unless specifically designed to produce absolutely steady flows, have these fluctuating outputs. This is significant because these signals convey a lot of information about what is going on inside the system itself, the channel through which the flow occurs, and even aspects of the receiving sink. Thus signal processing methodologies, developed for communications engineering, often have relevance to all kinds of material and energy flow situations.

[13] Throughout this description the emphasis will be on the overall procedure and not on specific analytic tools. The latter will be mentioned to provide guidance as to when they may be appropriate to use. The overriding assumption here is that readers who are familiar with the use of these tools will recognize what is meant and need no further guidance as to what to do. More general readers are invited to investigate these tools as they are mentioned, for example in Wikipedia. The author does not presume to tell readers what tools they should use, only what kinds of tools would be appropriate at various points in the general procedures.

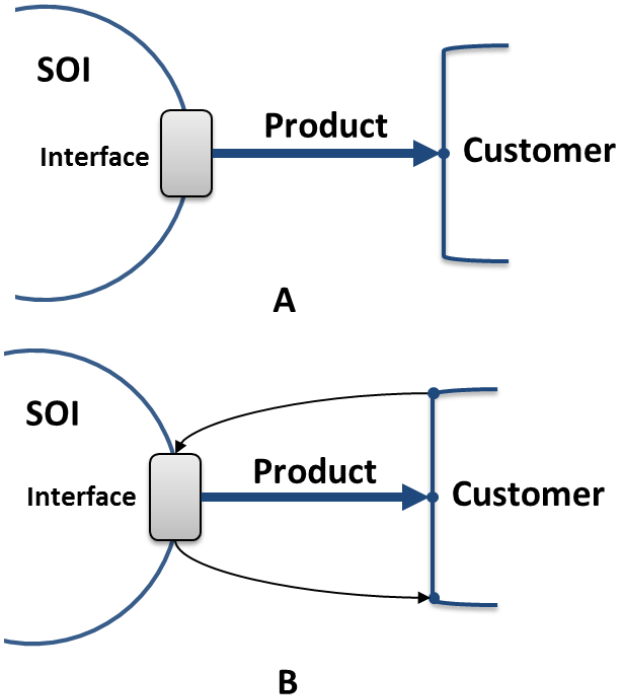

Fig. 5.4. A single product output is identified. A product is any output that is actively accepted or sought by a "customer." A) a customer accepts a product flow in an open loop scenario. B) a customer has established a protocol relation with the SOI wherein the interface (on the boundary) also includes the exchange of messages (thin black arrows) in order to coordinate the flow of the product from the SOI to the customer.

The interface of the SOI is the subsystem on the boundary that exports the product to the customer(s). The simplest case is a customer that accepts all possible output from the SOI and is, therefore, in open-loop relation with the SOI. This situation is probably rare in most concrete systems. As a result the normal situation is depicted in Figure 5.4B where communications between the customer and the SOI are established. This is the situation in which the function of the interface is a bit more complex than that of Fig. 5.4A. We will address this situation in the analysis of the boundary and when it becomes clear that interfaces are special kinds of subsystems. A good example of an interface with communications would be a shipping and receiving department in a manufacturing plant. Shipping products out to customers involves paper (or increasingly digital) flows for invoices, shipping manifests, etc. that work to coordinate the actual product flow.

A product flow object requires the creation of three different objects in the knowledgebase. Recalling Equation 3.7 from Chapter 3,

$$\boldsymbol{B}_{i,l} = \langle P_{i,l}, I_{i,l} \rangle \qquad \text{(Eq. 3.7),}$$

we will be creating an element of the ***I*** set. We will also be creating an element of the ***G*** set,

$$G_{i,l} = \langle (C'_{i,l}, Src_{i,l}), (C''_{i,l}, Snk_{i,l}), F_{i,l} \rangle \qquad \text{(Eq. 3.6),}$$

namely, the element (***C”$_{i,l}$*, *Snk$_{i,l}$***) and the element ***F$_{i,l}$***. Since we are starting at level 0, the index *l* will be zero and all of the *i* indexes will be 1.

In other words, we will create the interface, the flow (of product), and the customer (sink) objects, which will be captured in the knowledgebase as skeleton objects (for the moment). These will then be fleshed out with relevant data. Once completed we will be ready to proceed to a full SOI identification.

5.5.1.4.1 An Interface Object

Because interfaces are special subsystems all $i_{i,j,l} \in I_{i,l}$ are really $c_{i,j,l} \in C_{i,l}$. The *I* set of interfaces are represented in the *C* set of components for system *S*. The difference between *I* objects and other *C* objects is that the former include provisions for a special sub-subsystem called a ‘protocol’. Otherwise the two notations are interchangeable.

The interface in question here is the subsystem of the SOI which is responsible for the ‘management’ of the transport of product across the boundary and transfer of such to the appropriate customer. The latter is achieved by maintaining an existing channel for the flow, on the SOI end.

At this point in analysis we will not be decomposing the interface object. Rather we are just establishing the object in the knowledgebase, reserving its recursive version of Equation 3.1, giving it a coding and inserting it into the *B* object (Equation 3.7). A product interface for the SOI would be item $i_{0,1,0}$ in the $I_{0,0}$ set (if there are a number of products, the items would be sequentially numbered *i* = {2, 3, … n}.) All additional slots in the description form will be left “TBD” awaiting the recursive decomposition phase of the boundary analysis.

So the first interface object $i_{,0,1,0}$ is a data structure that looks like this table:

Table 5.1. An interface object on the boundary of the SOI.

Element	**Data**
Interface:	I 0.1.0
Name:	Main Product Output Interface
Description:	Exports the main product X from the SOI
Type:	Material output
Output Flow Code:	F 0.1.0

The table can, of course, contain additional data items. However let us consider what is shown here.

The first element item, “Interface” establishes this item as belonging to an interface list that is part of a boundary object. The data code, “I 0.1.0” indicates that this is an interface in the 0th list of interfaces (I_i) at level 0 ($I_{i,l}$), and it is number j=1 in the list ($i_{i,j,l}$). For the SOI we might generally expect there to be only one boundary with one interface list. But we maintain the indexing scheme for consistency. At levels greater than 0 we will have to keep track of multiple boundaries and interface lists.

The final element in Table 5.1 indicates a cross reference to a flow object (see below). This item will allow the linking of the flow of product X to the output from the SOI through interface $i_{0,1,0} \in I_{1,0}$. Specifically, flow dynamics are codified in that object.

In Chapter 6 we will revisit these database objects and show more details for specific kinds of systems.

5.5.1.4.2 A (Product) Flow Object

Each flow of material, energy, or messages is represented by a flow object, $f_{i,j,l} \in F_{i,l}$. We create a product flow object. At this point it is possible to identify the nature of the product (i.e. material, energy, or message, what it is specifically), its relevant parameters (e.g. volume or weight, flow characteristics such as pulses or continuous, etc.).

In the section to follow we explore the methods for measuring flows through instrumentation, data collection, and data processing. This is the key to determining the dynamic properties of the flows, and ultimately determining the transformations from inputs to outputs. Here we will indicate the results of such measurement (or specifications) as part of the flow object in the database. Consider the data in Table 5.2.

Table 5.2. A flow object in the database.

Element	**Data**
Flow:	F 0.1.0
Name:	Main Product
Description:	Product X exported to customer Y
Type:	Material, low entropy/high organization
Average wt. per unit time:	15 kilos per Δt
Peak wt.	22 kilos per Δt
Etc.	
Source:	I 0.1.0
Sink:	Snk 0.1.0

A number of other characteristics of the flow would be encoded in this element since flows are the key dynamic elements of any system (the Etc.) Here we are only interested in providing an example of the kind of data element that would be stored in the knowledgebase.

However, the final two items are critical links in that they identify both the source of the flow and the sink for the flow.

It is possible that a single output flow n-furcates between multiple customers. If this is the case n-separate flow objects must be created for each flow. It will then be the responsibility of the software to keep track of the mass flows (i.e. ensure the conservation of mass principle) so that the sum of all the flows equals the total outflow of the product from the interface[14].

5.5.1.4.3 A (Customer) Sink Object

At the initial state of system identification it is only necessary to identify that a sink (customer in this case) exists; create an object in the knowledgebase, and identify it as the sink for the product flow. Table 5.3 shows a customer/sink object.

Table 5.3. A customer is a sink object in the database.

Sink:	Snk 0.1.0
Name:	Customer Y
Description:	Receives outputs from SOI product
Type:	Material – Product X
Ave. wt. per unit time	8 kilos per Δt
Product Received:	F 0.1.0

The important link here is the last item, the Product Received. This establishes, in the knowledgebase, that the flow of F 0.1.0, the product, is received by the sink, Snk 0.1.0 and completes the flow path from source (the interface from the SOI) to the customer (sink). When we reflect on the importance of network theory (Principle 3, Chapter 1) we can see that this embodiment in the database (to be used as the knowledgebase) is a key to our understanding of the system we are studying.

Undeniably, this is a lot of detail to keep track of. However, it is essential to do so if we are to make any sense of what is going on in the system as a whole. Fortunately, with the right software we can readily capture this detail using the visual frontend described earlier. Simply drawing a flow link between the SOI and the presumptive customer calls up the need to fill in the

---

[14] In practice, the author has found it easiest to create separate interface objects for each outflow of the same product. It is an accounting problem, not a systemic problem!

details of what the flow is and its characteristics. Granted the analyst may need to do a great deal of research to fill in the forms, but that is exactly what an analyst is supposed to do!

5.5.1.5 Measuring and Using the Variables, Instrumenting the Flows

This step is fundamental to system identification. We will never be able to characterize a system until we have real data about how it works. Ultimately we will see that system identification involves instrumenting all flows into and out of the system boundary. In Figure 5.5 we see the basic scheme for determining how to characterize the behaviors of the initial elements of the product flow.

What is needed is a set of 'sensors' (as defined in Chapter 3) that measure the various important parameters of the flow. For example if the product is a physical object, we need to measure the number of units through the channel per unit time. There are, other possible parameters that might be important to capture. For example, in the case of a physical product, measures of 'quality' are also important to know. Such measures are generally couched in statistical terms. These parameters are going to become more important when we start deconstructing the SOI as they will also be subject to internal measuring as part of the quality control, management, subsystems.

Sensors appropriate to each relevant parameter of the flow are installed to capture data using the time constant determined previously (Δt), the sampling rate[15]. In Figure 5.4 sensors are placed on the channel in which the flow of product is contained[16]. The figure also shows sensors monitoring the message channels. The selection and use of sensors is a highly technical subject that is best covered in books devoted to, for example, cyber-physical systems[17]. There are several aspects to measurements of flows for the purposes of system identification. In today's world with the advent of digitization of just about everything, we describe these for discrete time methods.

---

[15] The underlying assumption we have been working on is that all data and functions are expressed in discrete form (i.e. digitally). Arguments have already been made that there is an equivalence between discrete and continuous dynamics when the time constants (used for the former) are sufficiently small that the approximation to continuous functions (for example) are sufficiently accurate and precise. Using a discrete value for a time interval, Δt, is thus justified.

[16] Non-channelized flows, for example broadcast messages or diffusion of materials, can also be readily sensed.

[17] The term 'cyber-physical' is relatively new but it applies to systems that employ computers and lots of sensors (both of the external world and the machine's own self) and emulate natural living systems that interact with the world in more complex ways. A dramatic example in the news as of this writing is the self-driving vehicle. See the Wikipedia article: https://en.wikipedia.org/wiki/Cyber-physical_system , for general background. Accessed 8/8/2017. Also (Alur, 2015; Wolf, 2012).

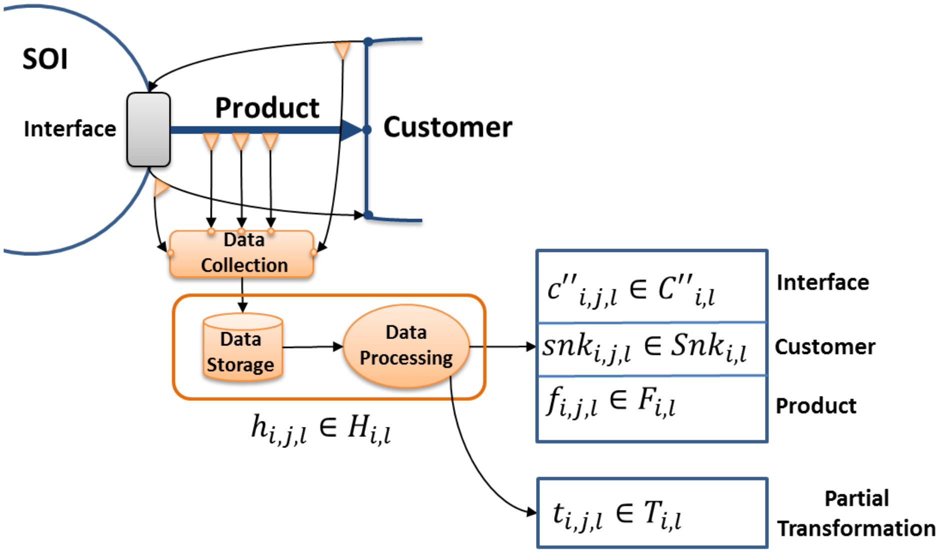

Fig. 5.5. Real-time data are gathered from measuring relevant parameters of the product flow (e.g. volume per unit time). Data on the communications between the interface and the customer are also collected (e.g. number of messages per unit time). The data storage and processing (becoming part of the SOI history object) are used to characterize the interface, customer sink, product flow, and provide partial information regarding the SOI transformation object.

5.5.1.5.1 Parameter Selection

There are several relevant parameters associated with flows that will come into the analysis as it progresses. At the current stage we might be most interested in the flow rate of the product from the interface to the customer sink. Later we will be concerned with quality issues as well. Material products are measured in terms of weight of a unit measure, such as decigrams per unit object, like a manufactured electronic circuit unit. Total mass, however, is the sum of mass measures for each of the individual components in the final unit. These are important since they plus the masses of waste products are used to determine that the full accounting for flows through the system are made ultimately.

Flow rates for something like this kind of product can be measured in units of product exported per unit of time rather than total mass per unit of time – the sensor counts the number of units that pass it and the data collection process in Figure 5.4 keeps this count for each unit of time.

Flows of energy are also measured in a similar fashion. Here the question is how much energy of a certain quality (e.g. amps, current, per second at 120 volts) flows from a source (the interface) to a sink (the customer) per unit of time. The potential difference between source and sink (voltage in the case of electricity) is an important parameter just as the pressure difference between two water reservoirs determines the rate of flow of water in a pipe of given internal diameter.

In both matter and energy flows the key concern is for how much substance flows in the time units used.

In message flows the situation is analogous but the measurements are different. What we would be most interested in in messages is the amount of information that is contained in the message. But this is not something that can be measured directly from the message itself. Information is somewhat analogous to the power factor of energy flow[18] – it is related to the amount of change (work) that will be accomplished at the receiving end. If the receiver's work process is inefficient (wastes more energy than is technically necessary) then the amount of useful energy flowing between the interface and the customer is actually less for the same amount of total energy flowing. For example, an SUV gets fewer miles per gallon of gas. The same number of joules are available in the gallon burned in a compact as in an SUV, but the former will get more work done (moving the same driver and passengers more miles driven per gallon) than the latter. Similarly, the same message may be sent to different receivers, each having different *a priori* knowledge, resulting in different changes in behaviors of each.

However, messages are exchanged between systems for the purpose of sending and receiving information that will make changes in the communicating parties. Therefore, measures of message flow are still relevant at this stage of the analysis. The traditional measure of messages is something like encoded symbols per unit time (e.g. bits per second), which is readily sensed with electronic messaging (both broadcast RF and digital wires)[19]. The actual information flow through message channels can only be determined after we know more about the influence that these messages have on their recipients. In the case of the interface-flow-customer case under current review this means we will need to analyze the 'behavior' of the interface upon receipt of messages from the customer. Since, by definition, we will not be analyzing the customer per se, we can only post hoc estimate the information content of messages from the interface to the customer from changes in customer overt behavior – for example in accepting fewer products per unit time or from the subsequent messages sent back to the interface. We will take a closer look at this problem below in the environment analysis.

The ultimate selection of parameters and the instrumentation of the flows for measurement results in the design of the data collection and the history object shown in Figure 5.4. Given a selection of Δt, emplacement of the sensors and a data capture/storage mechanism in place it is time to start measuring the behavior of this one specific element of the system.

---

[18] A somewhat more useful measure of energy flow is actually something called 'exergy' or the amount of energy in the flow that will accomplish work in the receiving system. But this measure depends on characteristics of the work processes in the receiver and not on the amount of total energy exported by the sender. The same thing is true for receivers of messages. Since information amount in a message is based on what the receiver is expecting the message to be (Mobus & Kalton, 2015, Chapter 7, Quant Box 7.2, page 281) one cannot measure information directly from the flow of the message.

[19] Sensing photonic signals from the outside is problematic (as eavesdroppers fear) but here we are assuming that all parties agree to the instrumentation of all flows so that photonic detectors (sensors) can be inserted into the channel directly.

5.5.1.5.2 Time Scales

The time step, Δ*t*, chosen represents the smallest unit of time in which a 'noticeable' or relevant change in the measurement can occur. The data capture mechanism, shown in Figure 5.4, is set up to take a measurement on all of its sensors for each step. However, patterns of important behavior may actually not show up except in very different time scales. The data collected at the resolution of Δ*t* may contain noise. But more importantly, real world, concrete systems of any complexity at all demonstrate patterns that vary according to different time scales.

The world is fundamentally comprised of systems that operate in multiple time scales simultaneously. Their behaviors during short time intervals (some small discrete multiple of Δ*t'*s) can be, and often are different when viewed over longer time intervals. The world is full of non-homogeneous, non-stationary processes. It is influenced by nonlinear processes characterized by deterministic chaos with inexplicable switches to new strange attractor basins, self-organized critical systems, and catastrophe. None of these phenomena operate on particular time scales. Therefore, as discussed below, the data collection required for very complex systems, for their product outputs, must cover very long time periods in order to capture patterns that would not show up in short samples. Practically speaking, some very complex systems, like ecosystems or human organizations, must be observed for years, perhaps hundreds of years, in order to be certain that all of the relevant behaviors are found.

Consider that the human brain is in the business of observing the world (or its corner of it) continuously for its whole life. It is designed to change its internal representations over time as the various components of the world change on these different time scales. We don't perceive changes in long-time-scale phenomena like sea levels or mountain building. But we do perceive changes in the courses of rivers over years, or the flooding of those rivers over days. The brain is the epitome of a observing system able to operate and modify its representations in multiple time scales.

5.5.1.5.3 Time Series Data

Not yet said explicitly is the notion of what this data stream looks like. What is being collected is a set of time series data points or measurements made in each Δ*t* for a very large number of Δ*t*'s. If the selection of the Δ*t* has been done well, the variances in measures across some interval of time (*x* number of Δ*t*'s) will represent real differences and not noise or measurement errors. In that case the analysis of the data series should yield true dynamics. However the analysis based on a single Δ*t* is not sufficient for real-world systems.

5.5.1.5.4 Multi-Time Scales and Non-stationary Series

In the real world, nothing stays the same forever. Typically time series data are analyzed based on some established time window, some *x* number of Δ*t*'s. This, however, is not necessarily adequate to characterize the dynamics of a flow. To see this, consider the following. Suppose we collect flow rate statistics over $x\ \Delta t$. We compute the average value and the standard deviation or variance. We now have a measure of what the flow dynamics are over that time window. Suppose, then, at 1,000x time units later we collect the same data stream and do the computation again. In most real-life situations we will find that the newly computed mean and standard deviations will be different (slightly) from the previously determined values. As a rule, most analysts will accept this difference as due to measurement error between the two samples. They may be tempted to average the two statistical values and believe they have a more "realistic" estimate of the values. Just to be safe, they may come back and take as third sample set at another 1,000x time units. What if they find yet another set of values? Of course they have recourse in the Law of Large Numbers, i.e. if they took enough windowed samples over very long time scales, say 100,000x, they would feel justified in claiming whatever overall statistics they calculate must be approaching the processes true statistical values.

There are two problems that present themselves. The first is what if the mean values of each window sample is slightly higher each time window? This can indicate a trend in a longer-term change that greatly complicates any analysis they do. What are the characteristics of the trend? Will it go on forever? The second problem occurs when the statistical properties of the sample vary up and down with each window. At first this might look like just a problem with window error, especially if there is any kind of alternating behavior, first up, then down, etc. But a careful further statistical analysis of the window data might reveal large variations in the window means and variance in the within window variance. This could be trouble.

Non-stationarity is a general fact of complex adaptive and evolvable systems. Such systems can demonstrate stable statistical characteristics for long periods of time, and then, suddenly, deviate (sometimes drastically) from their long-term statistical properties.

5.5.1.5.5 Data Analysis

If a time series represents a simple stochastic process, then ordinary statistical properties such as mean and variance (1st and 2nd statistical moments) will adequately characterize the flow for modeling purposes. Unfortunately, most real world systems, and particularly CAESs, do not have simple stochastic behaviors over longer time scales (see the discussion below regarding environment analysis and identification of sources and sinks for more clarification re: signal processing).

In general these behaviors can be characterized as sporadic (that is recurring but not necessarily periodic), episodic (that is lasting for variable amounts of time) and erratic (that is the variance in amplitude within an episode is, itself, variable). In general the statistical properties of these occurrences are non-homogeneous, non-stationary. As such, the analysis of the data must take these possibilities into account. This is no small requirement. Statistical methods for detecting these properties are still in their infancy. But as more work is being done on real-world phenomena from a systems perspective we suspect that more attention will be paid to the characteristics of these phenomena. We have seen this develop in fields such as multivariate statistics where co-correlations and co-variance methods have been developed to reveal causal relations. The same is likely to occur in non-stationary analysis. It will be an interesting area of active research in the near future.

One interesting area where the impact of non-homogeneous non-stationary properties has been felt is in machine learning. It turns out that real neural systems in real living brains deal directly with the problems associated with non-homogeneous non-stationary processes. Synaptic plasticity shows a dynamics that seems to incorporate this phenomenon directly (Mobus, 1999).

Even as we develop more advances in analytic methods such as non-stationary process analysis, it is still incumbent on analysts to understand that these long time-scale issues affect their picture of what systems are doing. We will address how this affects analysis of systems for the present in terms of how systems evolve (Principle 6 in Mobus & Kalton, 2015). Those who take on the analysis of complex systems would do very well to keep in mind that "things change" and that whatever data they collect and analyze currently are likely to change in some future time frame. All solutions are provisional!

5.5.2 Proceed Around the Boundary

Once the primary product flow, its output interface and its customer sink have been at least partially characterized it is time to analyze the entire boundary along with the actual inputs and outputs that traverse it. This entails doing the same kind of analyses for each and every input as well as non-product outputs (e.g. waste products or heat). The process will be to traverse the boundary, finding outflows and inflows and repeating the process of identifying the interfaces. Flow, interface, and source or sink objects can be created and their data forms at least partially completed.

For each flow discovered in the traversal the analysis requires instrumentation and data collection as described above. Eventually, data will be collected on the inflow and outflow parameters setting up history objects for each (Figure 5.5 below).

5.5.2.1 Completeness and Sufficiency of Flow Discovery

The more complex a system is the more difficult it will be to complete a survey and analysis of interfaces and flows around the boundary. Once an initial analysis of the boundary has been done the analysis turns to the second phase, environmental analysis, in which it is possible that new, unaccounted-for flows will be discovered (uncovered) and require reconciliation with existing analysis. The nature of the knowledgebase that is being built up allows additions of new objects as they are discovered. It is always possible to return to an identification analysis when new flows are discovered as a result of analysis of the environment. We will return to this issue below.

Always, the construction of the system identification phase must be considered tentative. It is even possible that once decomposition of the SOI is started we will find flows that had previously been unaccounted for. For example, we might, in analyzing the inputs to an internal subsystem, discover a flow that had not been previously encountered. This is the nature of a discovery process.

Fortunately, the way we have constructed the system definitions of Chapter 3 allows us to recover and add to the structures we have already created. There is never a time when we have to close off the addition of new structures – they are simply added to the sets or lists we have already created. That is, the structures (objects and their relations) are extensible such that they may be easily amended as new information emerges. We will provide some examples of this in the next chapter.

5.5.2.2 Identification of Outputs and Inputs

Once we have identified the major “product” outputs, their flows, interfaces, and sinks we are ready to proceed to analyze the rest of the outputs and then the inputs.

5.5.2.2.1Wastes

Non-product outputs may be classified generally as wastes or waste products. There are two major categories. Energy used up in work processes always exits the system as waste heat, that is energy at a very low potential that cannot be used to do any additional useful work. It might be noted that waste heat from one system (process) might still be useful to another process where the nature of the work does not require high temperature differentials. For example space heating (to keep occupants comfortable) requires very low-grade heat. Technically this is not accomplishing work per se. However it aids in letting the occupants of the space to accomplish other work in comfort. Even so, it will eventually radiate into the environment as completely unusable energy. The total energy of this low-grade heat is the same, but it simply is too diffuse to be usable in driving useful work process.

Material wastes may also ‘diffuse’ into the environment but not in the same way as waste heat. The latter will eventually radiate to deep space from the Ecos. Waste materials, on the other hand, will simple diffuse into the ambient space. For example waste chemicals will end up in diffuse concentrations in the hydrosphere where they may still have chemical or biochemical effects on other aspects of the biosphere.

In natural systems the typical pattern for material wastes is that there are low-order organisms that can recycle them. Or they may be incorporated into sediments to be recycled by geophysical processes. In both cases the final waste products can be inputs to regenerative processes, that is, they can be useful resources for more complex synthesizing processes.

5.5.2.2.2 Inputs

We next proceed around the boundary and consider the inputs to the SOI. These are classified as either resources or disturbances. As with the identification process for product outputs the nature of the analysis is to identify the flows and their dynamic characteristics as well as the input interfaces and their protocols.

5.5.2.2.2.1 Resources

A resource is defined as any input that is used in a work process to accomplish the purpose of the SOI and becomes a component in the products (with some residuals in the waste materials and waste heat).

5.5.2.2.2.2 Disturbances

Disturbances are any impactful encounter with the system that is not a routine flow of a resource. These are the hardest aspects of a system to analyze yet are crucially important in terms of how disruptive to a system's normal function they can be. The analysis of disturbances can be done using traditional risk assessment methods. Risk is defined as the impact (cost of responding and repairing damage, see also Chapter 9 section [Adaptivity]) times the probability of occurrence.

Some kinds of disturbances are readily identified. For example, fluctuations in a flow that exceed the capacity of the system to have an adaptive response. What happens to a car manufacturer plant when the shipments of engines stops, even for a short time? Then other kinds of disturbances are more difficult to predict in order to anticipate their occurrence (and take precautions). Tornados can be very disruptive. Finally, still other kinds of disturbances can seem to come out of left-field, as the saying goes. They are completely unanticipated. Most people have now heard of the "black swan" phenomenon.

Another type of disturbance is more easily understood and analyzed, that is the wear-and-tear or entropic decay of components that lead to subsystem dysfunction. A great deal is known now about how physical things wear down and go into senescence or need repair.

There is just so much an analyst can do in terms of identifying the type and quantifying the risk. But some attempt at doing so is necessary, especially for the first kind that are endemic and just the result of some cause that is not necessarily perceived.

5.5.2.3 Creating the Data Objects

5.5.2.3.1 Flows, Sources, Sinks, and Interfaces

Figure 5.6 shows a schematic representation of an SOI that has been completely instrumented so as to collect data on the inputs and outputs over time (instrumentation of the interfaces has been omitted for simplicity). The SOI is now considered as a "provisional" system in that the identified inputs and outputs and their instrumentation are only assumed to be complete in a first pass around the boundary. It is possible that the environmental analysis (phase two) will reveal additional details or exposure of flows that had not yet been considered and must be incorporated into the model before further analysis can proceed. However, absent such an exposure, we can consider the situation depicted in Figure 5.6 as a complete condition for the opaque-box analysis of the SOI. We are now in a position to produce the first pass identification of the system of interest.

This will entail computing a transformation formula in which the product output(s) are time delayed functions of the resource inputs. Or:

$$O_t = \mathcal{T}(i_1, i_2, i_3, \dots i_n)_{t-n} \qquad \text{Eq. 5.1}$$

the output vector at time *t* is a transformation of the inputs at time *t-n*, *n*, an integer multiple of Δt.

The outputs of a system at time *t* are the result of transformations that occurred to inputs that arrived at a past time. Equation 5.1 oversimplifies this relation since inputs that arrived at different actual discrete times may be involved in a discrete unit of transformation, i.e. a unit of $\boldsymbol{i}_2$ may arrive before a unit of $\boldsymbol{i}_3$. The discrepancies are usually accounted for by the fact that subsystems include internal stocks that mediate the flows of substances. More on this aspect later.

At this point we are now ready to complete the system identification in that we can estimate the transformation function of the provisional SOI. We have collected time-based data (time series) on all of the flows into and out of the SOI. It remains to analyze these data to estimate the transformations that occur within the system to complete an opaque-box analysis. It is important to reiterate that a mere transformation function is not the end goal of a deep analysis. A transformation function would be adequate if we didn't care how a system transforms its inputs to produce outputs. But we have already claimed that that is not our objective. We will want to dissect the system in order to find out how it manages these transformations.

Many system identification exercises end with the determination of a system transformation function because the purpose of the analysis is only to grasp the functions. For example, a controls engineer may only be interested in the identification of a physical plant transformation in order to design a controller to keep the plant within operating specifications. This is perfectly acceptable as far as it goes. It is possible to 'control' a large number of systems processes without a deep understanding of what the system is or how it does what it does. And as a general rule, this is perfectly adequate. But in cases where we have need to understand a system that has broader implications (for example how an economic paradigm affects the well-being of the average citizen) mere grasp of how to control a system is not sufficient. We need to dig deeper.

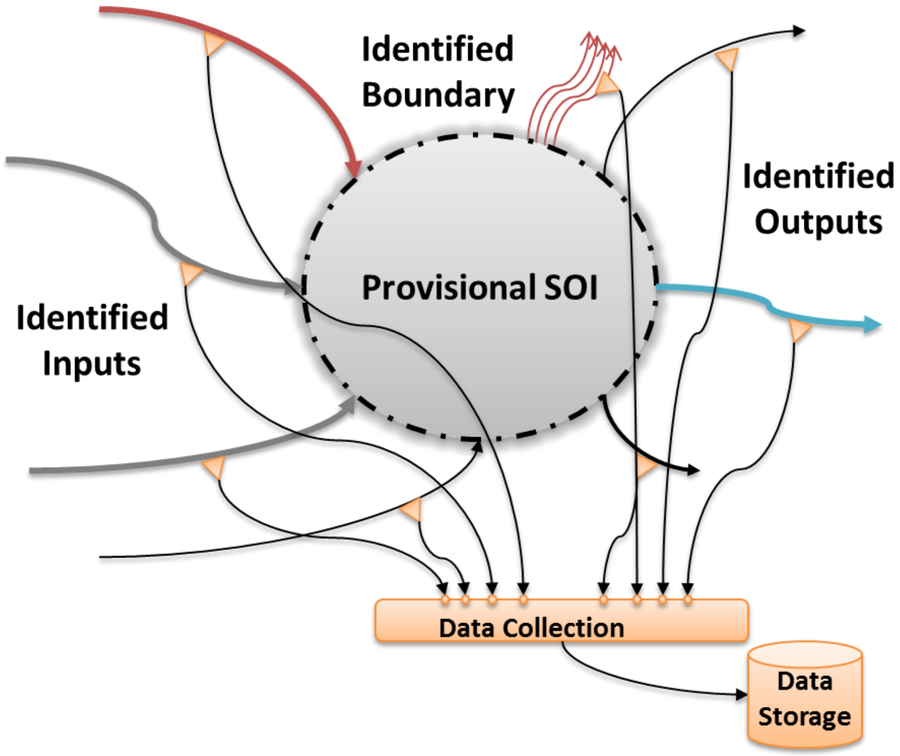

Fig. 5.6. Instrumentation of the entire SOI is used to collect data on the inputs and outputs over time.

5.5.3 Estimate the System Transformation Function

The output vector O_t in Equation 5.1 is given by the transformation function $\mathscr{F}()$. This is, in fact a composite of multiple output functions for all of the output interfaces around the boundary. How these functions are derived is, again, a matter for multiple textbooks, and there are many of those. Here we describe the main points that analysts must consider.

Recall that we are still working on an opaque-box model of the SOI. Recall also that for simple systems this may be completely adequate for practical purposes. But for CASs and CAESs deep knowledge is essential because the system function estimated at this stage cannot anticipate internal adaptations or evolution that will change how the system processes inputs to produce outputs.

5.6 Environmental Analysis

Phase 2 of the analysis involves discovering and characterizing the entities and sources of influence on the SOI.

5.6.1 Limited Models of Entities and Influence Sources

Sources and sinks do not need to be modeled in any substantial form. At most all that is needed is a minimal model of them as ‘naked interfaces’, that is, the points at which flows leave the sources and flows from the SOI enter the sinks. And even there the most that is needed is an analysis of the protocols used. In very many cases the SOI communicates with sources and sinks in order to coordinate the flows of materials and energies. In such cases it is necessary to identify the message flows, as in Figure 5.4 above, and identify the specific message/information relations that affect the flows of substance. The methods for doing so are the same as identifying input and output interfaces for the SOI itself. These will be covered below so this section is a placeholder for how that is to be accomplished with respect to source and sink interfaces.

The relevant information for sources and sinks involves the characteristics of the flows (in or out) without specifying the causal aspects of those flows. Note that if it is essential to capture a causal model of a source or sink, then it is probably the result of having not quite understood the boundary of the SOI properly. The likelihood is that what had been counted as an environmental entity should have been considered as part of the SOI itself. As an example consider the study of a lake system in a mountainous area. Originally the boundary might have been considered as the obvious shores of the lake, with inputs coming from streams arriving from the watershed area. In assessing the input of water through these streams it becomes clear to the analyst that the flow volumes are unaccountable under these assumptions. They decide that the flow volume variations have to be better understood and so expand the SOI definition to include the geography of the watershed. The environmental analysis now also expands so that the input and output entities may become the microclimate conditions around the watershed.

With respect to the internals of the sources and sinks, once the analyst has characterized the long-term flows (i.e. generation from sources and absorption into sinks) it is unnecessary to have further knowledge about how these are obtained.

That said, we will look at the situation when the characterization of the chosen environmental entities still leaves questions unanswered, as in the example of the watershed. It turns out that the beauty of this recursive procedure is that it can be reversed in direction so that what had previously been external entities can be brought into the SOI definition – the boundary can be expanded – invoking the very same analytic methods but in an ‘outward’ direction.

This raises a difficult question. In the recursive decomposition process we have a stopping rule regarding the recognition of when to no longer decompose. When we have the atomic processes (Chapter 3, Figure 3.7) as leaf nodes we know to stop any further decomposition. Going outward has a similar stopping condition. What triggered the inclusion of formerly environmental entities as now part of the SOI was the existence of unreconciled questions about the dynamics of the flows from/to such entities. The stopping condition for reverse recursive analysis is when the analyst cannot detect any such unresolved questions[20].

---

[20] Here we have a possible conundrum that analysts have to be aware of. How far ‘out’ do we continue to include entities and treat them as inside the SOI if we continue to detect unresolved questions? In one real sense such an outward progression could be infinite – there are always questions generated about complex systems! This is more of a ‘practical’ matter than a theoretical one. This is where the judgment of the analyst comes to play. In the

5.6.2 Identifying the Sources and Sinks

The completion of the environmental analysis will be the characterization of source and sink entities, including any presumptive sources of disturbances.

Recall from above that the identification of inflows and outflows proceeds by circumnavigating the boundary looking for flow entry and exit points and their interface objects. In that process we 'named' and coded the source and sink objects but limited our analysis to the metrics of the flows themselves. We now return to the list of sources and sinks, the ***Src*** and ***Snk*** sets in Equation 3.5. Table 5.3, above, provided a limited data object for the product sink or customer. This was a place holder until we undertook a more thorough environment analysis. The only 'measurement' contained in the object was a replication of the flow average obtained from the flow analysis. In the identification of environmental entities we need to go one step further and produce a limited model of the sources' and sinks' behaviors.

5.6.2.1 Measuring the Behavior of Sources and Sinks

In the section, Data Analysis, above, we observed that long-time-scale behaviors of real-world CAS and CAES entities have several levels on uncertainty. These systems are in the general class of processes called 'stochastic' or probabilistic in terms of their behaviors. There are many sources of randomness that affect the workings of CASs and CAESs. These can be from external sources (and sinks), from noisy disturbances from the environment, from internally generated noise, or from nonlinear dynamics and chaos. This gives rise to flows that can be found to be sporadic, episodic, erratic, as well as non-stationary. Figure 5.7 provides a sense of what this might mean.

These discrete time measurements are based on neural firing rates in sensory cortical neurons; each bar is a measurement of a neuron's rate of firing action potentials at a time instance. The time intervals between measurements is a constant, Δt, usually measured in milliseconds (the vertical scale is in number of pulses fired during the sample period). Each neuron is exposed to varying stimuli at varying intervals and for varying amounts of time. This is what is going on in the brain but it reflects the actual encounter of the brain – a system – with actual real-world sources of the signals that stimulate their firing.

---

real world there will always be more questions that cannot be answered without expanding the boundary of analysis! We will attempt to address this 'open question' below.

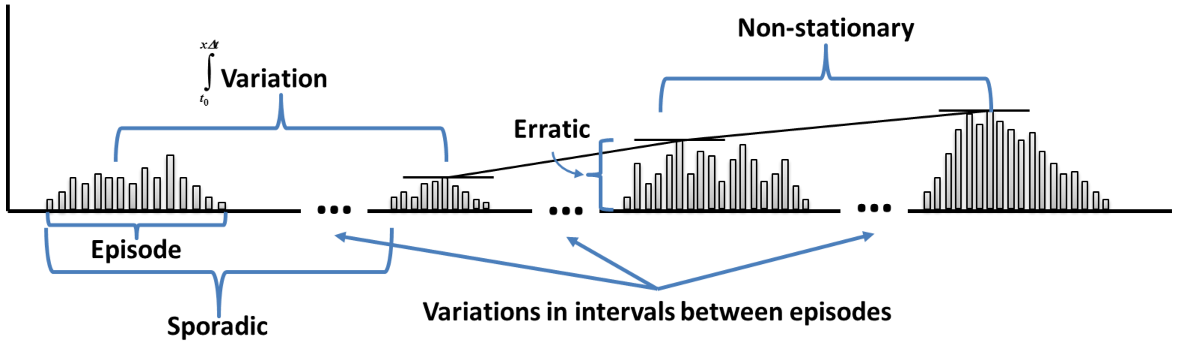

Fig. 5.7. This time series represents examples of neurons in sensory cortex firing whenever they are excited by stimuli in the sensory field. Firing comes in bursts or episodes that generally ramp up and then decline. Episodes are of varying intensity, varying duration, and over the long-time scale can vary in total firing. Some signals can contain noise and be erratic. Finally, firing properties may vary in an indeterminate way over very long-time (and varying) intervals. The vertical scale represents action potentials per 1/10th second. Ten samples on the horizontal scale represent one second of firing. There are usually long periods of quiescence during which very low frequency and random firing may occur. These are represented by the ellipses between bursts.

Even simpler entities that are sources or sinks for matter or energy flows may have seemingly indeterminate behaviors. Parts suppliers, for example, ship varying quantities of parts to a manufacturer, the packages arriving at varying intervals, containing, possibly, varying quantities. The variations may be due to varying needs by the manufacturer or to disruptions in the production facilities of the supplier (unknowable in general). Or a package could have gotten lost in shipment.

As mentioned before, any time series of measurements can be treated as signals and many of the methods of signal processing may be used to analyze the data[21]. It is beyond the scope of this book to go into depth in various approaches to, or the theories of, signal processing. There are a number of very good resources for this, including on-line free texts (Prandoni & Vetterli, 2008; Smith, 1997). The important point to recognize for the systems analyst is that in order to adequately characterize the inflows and outflows, and hence the sufficient identification of sources and sinks it is essential that considerations for the kinds of variability and stochasticity illustrated above be taken into account.

Having an adequate characterization of outflows from an SOI (via corresponding interfaces) will provide additional and essential information about the inner workings of the SOI, particularly the subsystems that export the outputs. However, such information can also be used to characterize the sinks for the outflows. That is, if the variations in the outflows are due not to SOI internal issues, they can be assumed to be caused by something in the sink's internal processes (which remain unknown). Some of this will not be known until the subsystems of the SOI are analyzed, but some of it can be identified by virtue of the analysis of the

---

[21] See, for example, see the Wikipedia article: https://en.wikipedia.org/wiki/Signal_processing. Accessed 8/13/2017.

communications channels (if any) between the SOI interface and the sink (as depicted in Figure 5.5). For example, if a sink is an active agent (customer) that signals a halt to receiving product until further notice, then some portion of the sporadic-ness of the flow may be attributed to the sink rather than the SOI. Communications signaling to modulate flows are found throughout biological examples as well as social systems. The analysis of both input and output interfaces includes an analysis of messages between the interfaces and their source/sink entities. The analysis should identify causal correlations between signals received or sent and subsequent actions by the source/sink entity and the interface. This will be demonstrated below.

5.6.2.2 Modeling Sources and Sinks for Engineered Systems

Thus far we have been discussing the analysis of concrete and existing SOI environments. The strategy has been to instrument the real flows into and out of the system, collect very long time scale data streams and estimate functions that would generate such to refine the knowledgebase for environmental entities and interfaces. What about, however, the case for systems we are going to design? We could also adopt the methods of instrumenting the environment into which this system will be put, and this is probably the best starting place. We identify with which entities in that environment the new system will interact and collect data as before.

There is, however, a fundamental weakness with this approach taken alone that has to be addressed. Namely, the environment will behave differently simply because of the behavior of the new system[22]. It is possible to derive functions for sources and sinks as above, however, in order for them to be useful later when modeling the new system, they will need to be modified to respond to that new system. There is no simple way to approach this problem.

Consider as an example a company wants to introduce a brand-new product – a new technology – into the marketplace. It expects that this new product may compete to some extent with existing products (like Blu-ray competes with DVD) and, at the same time, introduce all kinds of new possibilities (like new features in Blu-ray not possible in DVD). Companies produce ‘pro forma’ statements, which are essentially models of what their revenues, costs, profits, volumes, and other metrics are expected to be into the future. They do this to assure themselves that their profitability profile will be positively affected by the move. Too often companies that are driven by the emotions of what might be called ‘wowy-zowy’ technology make unwarranted assumptions about things like market penetration rates (volume absorption by customer sinks), pricing options, and so on, because they have a wishful bias and want the new

---

[22] An analogous situation would be to consider what will happen in a particular ecosystem when an invasive species is introduced.

introduction to work out. They produce a favorable sounding pro forma so as to attract investors or obtain financing from banks[23].

More thoughtful (and hardnosed business) decision makers will insist on building as accurate a pro forma model as possible. They know that being eager for something to work out doesn't mean it will. How to accomplish this is no easy task and it is never guaranteed to be correct. There will always be guesswork involved in simulating the environmental entities. Even so it behooves managers to do as thorough a job as possible in accurately measuring whatever environmental flows they can (for example it may be possible to retroactively look at the market dynamics of prior new products of similar kinds).

For engineered systems such as information systems to support operations of an organization, the situation is somewhat in between measuring actual environmental entity behaviors and supposing those behaviors in pro forma models. It depends on whether the new system is simply replacing an old system in the same environment, or will be serving a completely new organization function. In the prior case, the information flow channels already exist so it should be possible to observe the existing work processes and use their behaviors as models. In the latter case, the organization is actually in control of the design of the sources and sinks (of message flows) and so the larger supra-system is already known (or better understood) as the environment of the to-be-designed information system.

5.6.2.3 Refining the Knowledgebase for Sources, and Sinks

At this stage we are ready to refine the data objects for sources and sinks (e.g. in Table 5.3). In most instances this will simply involve deriving an equation of function estimated from the transfer functions of the flows completed above. Such functions will take the forms:

$$o_{Src_{i,0},t} = f(q,t) \quad \text{Eq. 5.2}$$

where *o* is the output of a source, q is a measure and *t* is time; and,

$$i_{Snk_{i,0},t} = f(q,t) \quad \text{Eq. 5.3}$$

with *i* being the input to a sink.

These functions should match those in the corresponding interfaces in the boundary object, but with reverse flows, e.g. outputs from sources are inputs to the receiving interface.

---

[23] Unless, of course, you are Apple, Inc. in which case you have your own financing in the form of reserved cash on hand!

5.7 Deconstructing the System

With flows identification and environmental analysis essentially complete[24] it is time to begin converting the opaque-box of the SOI into, first, a grey-box and then a transparent-box. We will decompose the internals of the SOI at the next level down of subsystems. This, recall, is level 1. We will identify all of the subsystems at this level and map the flows of the inputs to the SOI and outputs from it to internal subsystem. We will also map the flows between subsystems such that all of the inputs and outputs are completely accounted for.

5.7.1 From an Opaque Object to a Transparent System

The final phase of the initial analysis is to convert the opaque-box into a transparent-box, to elucidate all of the subsystems and internal flows of the SOI. The outcome will be a map of these objects and flows at level 1.

5.7.2 Finding Subsystems

The logical place to start this analysis is the processes that are associated with the various interfaces that we discovered as part of the boundary analysis. Since our interest is primarily in CASs and CAESs we will assert a rule for their identification. Such processes are generally active. That is they are work processes that use energy to accomplish their functions. Processes (or subsystems, remember) that receive inputs from the environment, through their respective interfaces, are called 'import' processes. The work they do involves not only regulating the interface but also, generally, distributing the inputs through appropriate flow channels to internal processes that do work on the inputs to produce intermediary products and outputs (finally).

Processes that 'push' products and other outputs out into the environment are 'exporters.' They generally consume energy doing work to move the outputs from the interior of the SOI to the environment, to the various sinks. Flows, after all, require a pressure differential between source (in this case the SOI) and the sink (the environmental entities). For example, cells have to expend energy pushing molecules out against a gradient; companies have to expend energy shipping products to customers.

Figure 5.8 depicts the general situation with an SOI interfacing with its environment[25]. It imports resources and exports products and wastes. This is where we start the analysis of the

---

[24] As already indicated we must always consider these analyses as 'provisionally' complete. As we enter the next phase of analysis we may yet find unaccounted for flows that will require a return to the analysis of the environment.

[25] A general, but not absolute, convention for drawing these maps is to put the sources on the left and sinks on the right, i.e., flows go from left to right for environmental elements. However, there are situations (many times in fact) when an entity is both a source and a sink. For example, when there is a communications protocol involved in controlling the flow of material or energy, say from a source, that source will also be a sink for messages sent from the system's interface (see Figure 5.4 above). If a source of, say, substance A is also a sink for substance B, one method for representing this is to have two objects in the map, one on the source side and one on the sink side. The

level 1 (recall Figure 5.2 above) subsystems. Import and export processes are the subsystems that link the environment to the internals of the SOI.

5.7.2.1 Start with Interfaces and Their Associated Processes

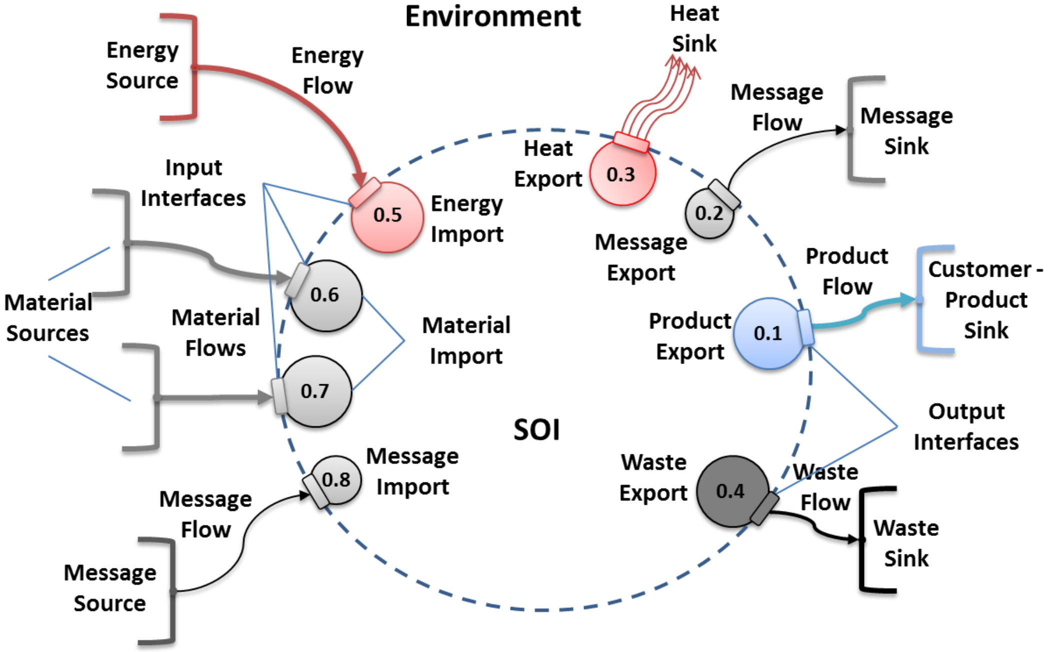

Fig. 5.8. The first step in deconstructing the SOI is to find all of the subsystems associated with the input and output interfaces. For inputs these are import subsystems. For outputs these are export subsystems. Importation processes may also function as distribution subsystems while export processes may function as collection subsystems. Note the process identification numbers. Since the analysis started with the output of the product flow, we have assigned it as level 0, subsystem 1 (0.1). Since all subsystems of level 0 would have a prefix of 0, we will drop this numbering in the future. The prefix, 0, is understood. See below for the actual treatment of the interfaces and subsystems.

These are the first subsystems that we identify. This is because each of these will have outputs (for importers) that get distributed to other internal work processes (as shown below) or collect outputs (e.g. products and wastes) from internal work processes and prepare them for export.

A reasonable way to envision the import and export processes/subsystems is that they 'own' the interfaces, or the interfaces with the external world are essentially subsystems of the import/export processes. It is these latter processes that are responsible for controlling the interface protocols. The interfaces will first be identified in the boundary object (see below) and cross referenced in the component set of the subsystems as they are further analyzed in decomposition.

---

two objects can be given the same 'name' and a logical linkage in the database, but are acting as different entities in their behaviors as sources and sinks.

As an example of the role of an import process, consider the energy importation process. In a plant cell this is the job of the chloroplast. This organelle is responsible for importing light energy at visible wavelengths and exporting carbohydrate molecules that can be used, later, as fuel for respiration (oxidation of the sugar molecules to produce adenosine triphosphate molecules, the energy distribution mechanism for all cells)[26]. The interface that accomplishes the importation of light energy is the molecule chlorophyll; it absorbs photons and transfers electrons to other molecular processes within the chloroplast that then complete the process. A similar function is performed by the electrical interface (through circuit breakers and an electric meter) of a manufacturing company with the electric grid that supplies electrical energy for the machines.

The reason we counted interfaces as part of the boundary of the SOI (originally) is that they occupy a very special role in a whole system. They are both part of the boundary, as previously treated, and part of the import and export processes that are, technically, internal to the SOI. This can cause some confusion for new analysts. Is the interface part of the boundary or part of the internal processes? The answer is 'both.' This will be reflected in the way import and export subsystems are handled in Equation 3.1.

Recall that a boundary object, $\boldsymbol{B}_{i,l} = \langle P_{i,l}, I_{i,l} \rangle$ is a tuple of sets, P, called properties, and I, called interfaces. As each interface is identified, starting as suggested with the product output, it is given an ID code and included in the I set[27]. The actual data object holding the interface information was given in Table 5.1 above. In database terms, the ID code is the primary key for the object in a table of interfaces; the prefix, 0, is required here so as to distinguish the object within the interfaces table. The on-screen identification would be coded I 0.1 (as in Table 5.1), where the I stands for interface. In Figure 5.8 interfaces are assumed to take the same code as their owning subsystem (importers and exporters).

Once the interfaces have been identified we can turn attention to the 'owning' subsystems. These constitute the first items that will be populating the $C_{0,0}$ set from Equation 3.1 and 3.2. At this point we can only identify the existence of the component subsystem and create a data object for it as in Table 5.4 below.

Table 5.4. The first component subsystem identified in the SOI

Component:	C 0.1
Name:	Product Exporter
Description:	Exports the SOI product (X) through interface I 0.1

---

[26] See, for background, the Wikipedia article on photosynthesis: https://en.wikipedia.org/wiki/Photosynthesis. Accessed: 8/155/2017.

[27] By this we mean the new interface is given a unique code, e.g. 0.1 for the product interface, and is added to the set $I_{0,0}$. As the analysis progresses around the boundary, each new interface is similarly added.

Type:	Process
Transfer Function:	$\mathcal{T}(p, t)$
Output(s):	F 0.1.0
Input(s):	TBD
Boundary	P = TBD, I 0.1
Membership:	1

Equation 3.2 provided for components, ($c_{i.j.l}$, $m_{i.j.l}$), to include a membership function in the event that a component of the set had fuzzy properties. In Table 5.4 this is shown as a value of 1, meaning that the component subsystem is always as member of the set. More generally this field could contain a function that returns a 1 only *when* the component is a member of the set C in any Δt or $x\Delta t$ window.

Note that the data object in Table 5.4 indicates a Boundary object with a P set yet to be determined but an I set containing a single member, the interface cross index of the interface identified in the boundary of the whole SOI. This is how the interface to the external environment is handled between the SOI and, in this case, the export subsystem, **C 0.1**, for consistency. The interface needs only be identified initially in the SOI boundary analysis and then cross referenced in the export (or import) component that is discovered.

Once again, the analysis proceeds around the boundary, identifying all of the importer and exporter subsystems associated with the interfaces already identified in the boundary I set. In the next section we will conduct an analysis of these component subsystems as a first step in delineating all of the internal subsystems.

At this point the inputs to the exporter subsystems and the outputs from the importer subsystems (into the interior of the level 0 SOI) are not known per se. We will identify these flows and discover their sources (inflows to the exporters) and sinks (outflows from the importers.

5.7.2.2 Identifying Internal Work Processes and Flows

The final stage for completing the knowledgebase for level 0 of the SOI is to identify the internal subsystems/components, finally filling in the C and N objects in Equation 3.1. Starting with the internal processes that were identified as “owners” of the interfaces with the environment we will treat each as a new problem in system identification. Each import/export process will be given identification codes at level 1 (Figure 5.9).

In order to accomplish this task, we will ‘follow the money’ so to speak. That is we will start with the discovery of the outflows from importers and inflows to exporters through the same kind of boundary analysis we did with the level 0 SOI (see Figures 5.9 and 5.10 which show a sequence of analysis states leading to Figure 5.8 above). In the section below, *Recursive Deconstruction*, we will discuss the formal treatment of deconstruction. Here we describe the first steps in performing this deconstruction more informally.

5.7.2.2.1 Importers

A reasonable place to start is with import processes. Taking each in turn and treating each, now, as an SOI in its own right, we replicate the system identification process, this time looking specifically at its output flows to what we will find to be the internal subsystems. In Figure 5.8 the red oval importing energy (upper left quadrant) through the red interface has a number of output flows of that energy to numerous customers. It is both an importer of the resource and a distributor of that resource to other subsystems. Therefore, the energy importer has another interface associated with its boundary (shown as another interface on the red oval). This one is only associated with the importer. In the case shown in the figure, the interface actually provides for multiple channels through which the energy flows to other internal processes. Energy is generally consumed by every other subsystem, including all of the other importers and exporters, so the figure shows red arrows going from the importer to all of the other processes. In anticipation of the next steps in this stage of analysis the figure also shows two other components, one a material importer, the other the waste heat exporter. Initially, when looking strictly at the energy importer, these would be treated the same as all of the other internal receivers of energy, as sinks (open rectangles as in the system language icons presented in chapter 3). Below the material importer component shown (with an energy receiving interface) we show the second material importer (as in Figure 5.8) as a “ghost” outline underlaying a sink icon. This represents the fact that as we move to a more careful analysis of this object we will associate the sink with an actual importer subsystem that requires energy input to do its work.

Once all of the outputs from the energy importer are accounted for (and entered in the knowledgebase) we will have another system (a level 1 system) identified as an opaque-box, with its own Equation 3.1, but a level, $l = 1$ and index, $i = 5$. This is the first occurrence of the relation in Equation 3.3, $S_{i.j,l+1} = c_{i.j,l}$. Recall from chapter 3 description of this equation, the compound index, *i.j*, is the coding for the tree relation of the new SOI to its all of the other internal nodes in the tree structure. In this case, i = 0, and j = 5.

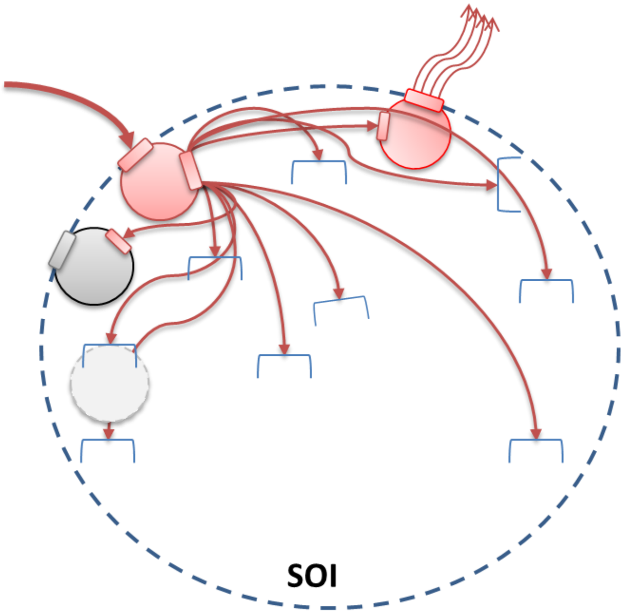

Fig. 5.9. The energy input importer process has been identified and a boundary analysis of its outputs reveals multiple channels of flow to internal sinks. These will be identified later, but at this point the energy importer is treated as an SOI for the purposes of analysis. See text for explanations of the other subsystems.

Figure 5.9 shows the results of a boundary analysis of subsystem, $\boldsymbol{c}_{0,5}$ in which outflows from the energy importer are distributed to multiple internal subsystems within the SOI. This is a preview of the recursive decomposition process that will be covered below.

Raw material resources are imported by the two larger grey ovals in Figure 5.10 (as in Figure 5.9). Presumably one is importing material A and the other material B. Each of these outputs their material types unchanged. Their job was to actively obtain their respective materials and send them on to other processes (the green ovals).

By tracing these various flow channels we discover the internal subsystems that receive them and, presumably, process them. We will return to these entities shortly. For now they can be treated as sinks for the output flows of the importers.

5.7.2.2.2 Exporters

Similarly, the product exporter interface, we can identify the exporter subsystems associated with the interfaces already discovered in the level 0 SOI identification analyses. Once discovered, each of these is also treated as a subsystem SOI and the inputs to it are identified. In Figure 5.10, all of the exporter processes use energy and so receive energy inputs. In this simple model, of course, all of the energy inputs must come from the energy outputs of the energy importer. In more complex real systems, we do not necessarily know where the inputs to the exporters come from – yet.

The darker blue oval in the figure represents the product exporter. What we find when examining its boundary, aside from the energy it needs, is the input to it of the product flow. At this point we do not know the nature of the source entity (the large light blue oval). But we can provisionally code it and reserve data slots for it in the knowledgebase. In the figure, all of the importer and exporter subsystems have been put through the system identification process so that for each of them, their interfaces and input/output flows have been identified.

While at this point we can treat the receiving (from the importers) and sending (from the sources to the exporters) entities as sinks and sources, noted in Figure 5.9, they will not remain un-modeled entities for long.

5.7.2.2.3 Internal Flows

Indeed, by following the flows already established, i.e. outputs from importers and inputs to exporters to their respective sinks and sources, we can account for the mass and energy balances (though we have left most waste heat flows out of the analysis to avoid more clutter, these would also be identified in analyzing subsystem 0.3). At this point we can identify all of the interfaces for all of the importer/exporter subsystems, their respective flows of energy, matter, and messages, and their respective sinks/sources (respectively) as shown in Figure 5.9.

5.7.2.2.4 Internal Subsystems

The clusters of sinks and sources in the interior of the SOI in Figure 5.9 represent internal subsystems that do the work that transforms the inputs to the level 0 SOI into the outputs therefrom. It should be the case that with the identification of outputs from importers and inputs to exporters we have discovered what we might call the ‘next layer’ of subsystems inside the SOI boundary (the first layer being the importers and exporters themselves). These clusters strongly suggest the existence of such subsystems even though we do not at this point know what they are. If objects in our system include, in their descriptions, locations in space, we do have a clue to the linkage among these source/sink objects. When we followed the known flow from/to each importer/exporter subsystem, doing the boundary analysis and identifying each one’s environment, we located those objects and the knowledge that there must be internal work processes in the interior of the level 0 SOI tells us that that is what these clusters represent.

The general procedure, at this point, is to choose a likely candidate cluster of sinks and sources. In Figure 5.10 we have encompassed such a cluster in a red, dashed oval to highlight how the sources and sinks seem to align and suggest the existence of a subsystem.

The role of the analyst, at this point, becomes more detective-like. This is where it is necessary to actually go to that presumptive object and start looking for clues about its boundary and whether or not the various source and sink objects in the cluster are, in fact, part of a contiguous boundary. For the scientist-analyst this may involve setting up observations (instrumenting). For the design-analyst this will involve querying the relevant users/stakeholders or engineers who are directly involved in that piece of the whole[28].

Otherwise, the analyst might provisionally assume that the cluster is a subsystem, assign a boundary to it (as an opaque-box) and insert the various source/sink objects into it as interfaces. That is, we create a trial subsystem and treat what had been a set of sources and sinks (for other previously identified subsystems) as interfaces to the new object.

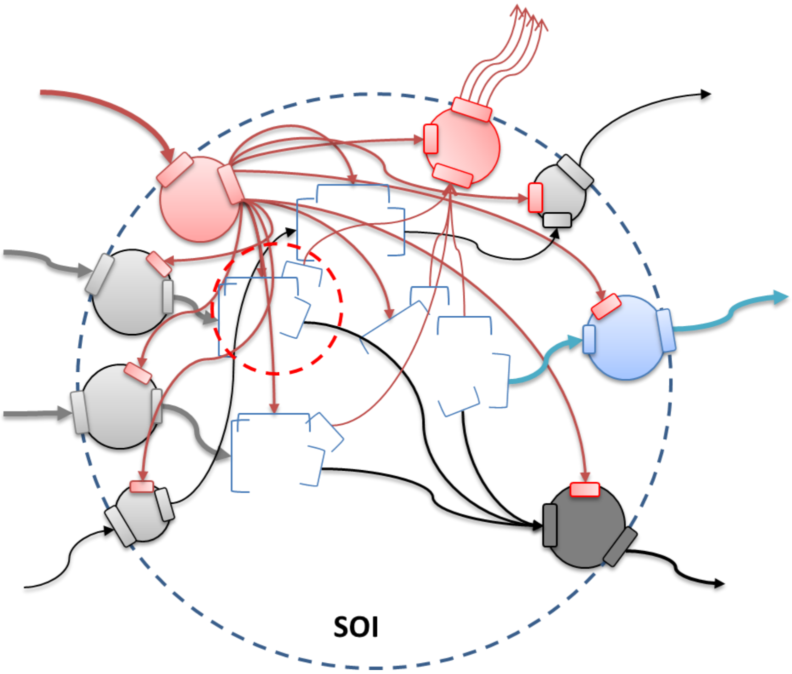

Fig. 5.10. Mapping internal work processes begins by mapping flows from inputs to candidate processes and outputs from those to the export processes already identified. Here the import of energy, matter, and messages is shown distributed to various internal work processes, including the importer/exporter processes, that convert material inputs to intermediate and then final products for export. Note the clusters of internal sources and sinks established through the environment analysis of each importer and exporter. Also note the red dashed oval encompassing one cluster of sinks and sources. That is the starting point for discovery and opaque-box analysis of the internal work processes.

What ensues is analyzing each new interface in exactly the same manner as we did with the original level 0 SOI. This time, however, we start with the inputs, since their characteristics are already identified. Generally speaking, a good place to start is with the energy inputs, since all work processes use energy and we already have identified the energy flow from the energy importer, we know what the interface should be like. This can be verified through instrumentation and analysis of the flow dynamics, generally correlated with heat output from the process – Figure 5.10 shows a few heat flows from the yet-to-be identified process shown, to the heat exporter process as a single thin red arrow with open head. The correlated flows of

---

[28] This is essentially what systems engineers do. They pose questions to the domain engineers who know what that particular piece of the overall design is.

energy in and low-grade energy out will provide important clues later regarding the amount of work being done by this presumptive process.

In a similar fashion, we can consider material inputs. One good rule-of-thumb in choosing which material input to consider next is to look at the embedded energy in the material. For example, a simple ingot of steel has only the embedded energy of making the steel from ores and pouring it into molds. A shaped steel part, like a sheet or beam, on the other hand has more embedded energy since more work was done on the original ingot to form it. We can think of the highest embedded energy input as the ‘main’ input in many cases. Another example is the importing of food to a biological entity. Animals eat plants and other animals, the tissues of which represent very high amounts of embedded energy.

When it comes to messages as input the strategy is similar but very hard to implement directly. For message quality the relevant measure is, of course, information content. But this can only be assessed by the effect it has on the SOI receiving it. This is not at all easy to assess, especially when so many other variables (inputs) may be changing dynamically. For this reason, it is best to leave message analysis to last. Interfaces for messages in and messages out can certainly be identified as best possible and given a place in the boundary object. Indeed, sometimes it is possible, as in the case of human/machine communications systems, to analyze the interface to see what kinds of messages transfer through and how their protocols are done. From such analysis it is possible to make educated guesses about the quality of the messages, but these must be revisited after a more thorough understanding of the other parties to the communications are known, and one has an opportunity to observe behavior changes that result from the receipt of information.

Next, we perform an opaque-box analysis on the red, dashed oval cluster assuming the oval will be replaced by an actual subsystem boundary. That result is shown in Figure 5.11 below. Taking all of the previously identified sources and sinks in the cluster, analyzing them as interfaces, and inserting them into a presumptive boundary we create a new internal subsystem. And we create a provisional knowledge base object (Equation 3.1) for it. Further analysis of this new object identifies a new interface and flow, an intermediate product (green arrow in the figure), and creation of yet another sink object. The latter appears to be part of the cluster of sources and sinks in the center of the SOI. As we did with the opaque-box analysis of all of the other subsystems, the importers and exporters, we complete it for this new object and confirm that inputs and outputs correlate through the transformation function.

We now do the same operation for the other clusters.

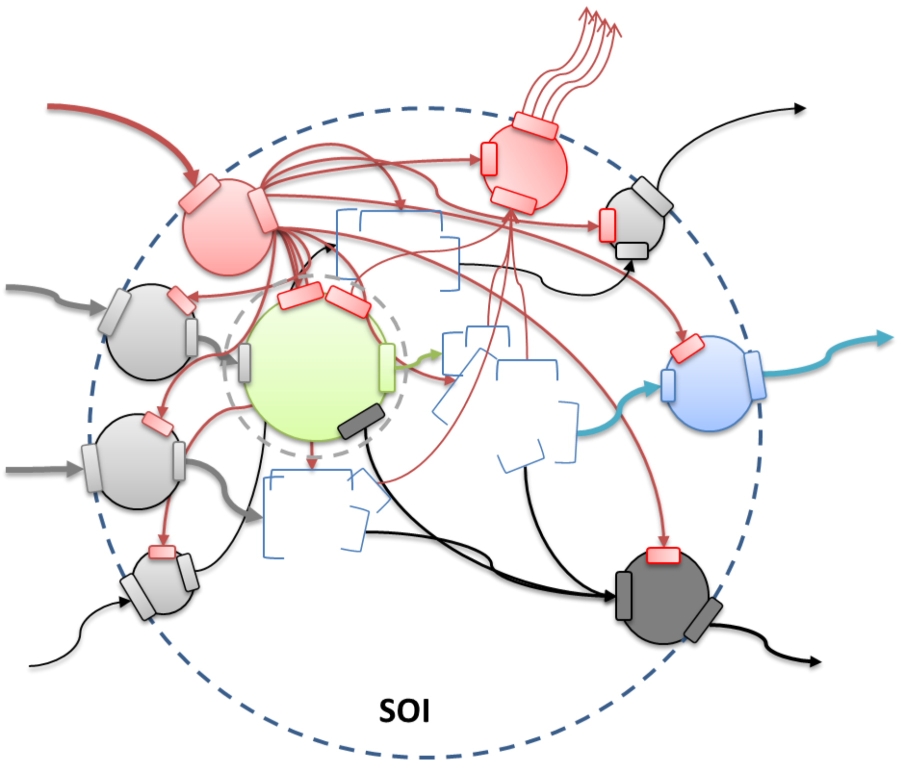

Fig. 5.11. The analysis of the first cluster of sources and sinks shown in Figure 5.9 has identified the interfaces and its environment. All of the input flows and the heat and waste flows have been resolved because the sources and sinks previously noted had already been identified as import and export objects. A new output (the green arrow) and its sink were identified. Note how the sink identifies with the cluster of sources and sinks already noted.

The result of following this discovery and mapping process is shown in Figure 5.12. A careful bookkeeping analysis of all of the flows will verify the total mass and energy balances for the system. With this information, we can refine the transformation function for the entire level 0 SOI because we have more detailed knowledge about what goes on inside it!

The new processes discovered are assigned Equation 3.1 objects with the number codes as shown in the figure. At this point we have a level 0/level 1 tree with all the components, flows and relations determined and mapped.

However, we still need to refine our understanding of the message flows in the system. In Figure 5.12 you will see an orange oval labeled “Governance Process”. The only message flow in the figure is shown coming into the system through import process C 0.8. The output from this process goes to the governance process, now numbered C 0.12. Finally, we see a message output from that object to an export process, C 0.2. We don’t know much about the sink that receives that message but may have to reiterate the environment analysis of the level 0 SOI to have a better idea of it.

What, in fact, is the case in all CASs and CAESs is that they all have some kind of internal regulation and management functions that process messages for information and make decisions that impact the operations of the other internal processes. As will be argued and demonstrated in

Chapter 11 on Governance, this is a universal capacity of all such systems. Simple systems, such as heating and air conditioning systems, may have very simple cybernetic-based control systems, like thermostats. But all adaptive systems need multiple levels of regulation and management in order to maintain their overall behavior and fulfill the system's global purpose (see Mobus, 2017 for a discussion regarding the requirement for a governance system within a CAS/CAES in order to achieve and maintain sustainability).

The vast majority of communications within and between systems is for the purpose of governance. In Mobus & Kalton (2015), chapters 7, 8, and 9 we give considerable coverage to the subjects of information, knowledge, how the latter is produced by computation in special information processes, and how it is all used in cybernetic frameworks (including a brief introduction to the hierarchical cybernetic governance system to be covered in Chapter 11 of this volume).

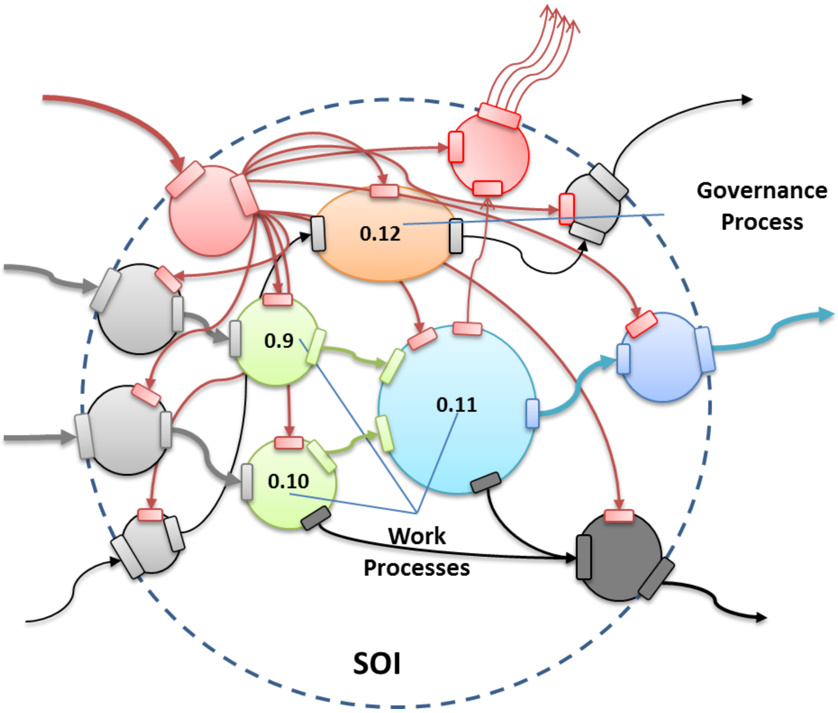

Fig. 5.12. The analysis of the internal flows and processes results in a complete decomposition of level 0, the SOI.

Accordingly, we will not expand much on the flow of messages in this chapter. The governance process shown in the figure is just a kind of placeholder. In reality, there will be multiple communications channels between all of the processes and with a "central" governance process (if one exists). For example, the flows between interfaces are generally modulated (regulated) by message flows between protocols in the interfaces (recall Figure 5.3B). If process C 0.11 in Figure 5.11 were a final assembly manufacturing process the flows of intermediate products (e.g. subassemblies) from C 0.9 and C 0.10 would be controlled by messages from the former to each of the latter requesting parts to be supplied for the assembly process (sometimes called a 'parts kit'). If one of the subassembly processes is running behind they would send a message back to the final assembly process telling it so. Of course, then, a decision maker who is actually embedded in the final assembly process might not take kindly to that fact!

Which introduces the idea that will be further elaborated in Chapter 11, that each sufficiently complex subsystem will have its own internal governance process. Governance, in most real (and successful) CASs and CAESs is distributed among all of various subsystems and even within their next level down subsystems. The nature of this distribution of decision making and authority will be covered in Chapter 11. It is mentioned here just so you will understand what you see next when we now begin the recursive deconstruction to the next level down and beyond.

5.7.3 Recursive Deconstruction

5.7.3.1 Going to Level 2 and Beyond

There are two basic strategies for doing the recursive deconstruction of a system. Both of these strategies mirror computer science ‘tree search’ algorithms[29]. A tree is a kind of data structure, essentially like that in Figure 5.1A but without the level -1 objects. The algorithms for traversing a tree structure such as this start at the ‘root’ node, level 0, and progress down the tree structure, visiting nodes at deeper levels (depth in CS lingo). One method is to traverse the tree down a path all the way to the terminal (called leaf) nodes and then backtrack up to the last parent node, finding the next path to deeper nodes, until all nodes at the deepest level have been visited. It then backtracks back up a level and repeats the traversal on the next pathway (if it exists). This is called ‘depth-first’ traversal.

The other strategy for traversal of a tree is called ‘breadth-first’. In this strategy each node in a tree at the next lower level is visited (discovered and characterized). After all of the nodes at that level have been visited, the algorithm goes back to the first discovered node and repeats the process, going down to its children nodes, then exploring each node at the parent level in turn.

The system-subsystem-sub-subsystem organization is, effectively, a tree form of a graph. Therefore, we apply the same strategic exploration strategies as guides to the decomposition process.

5.7.3.1.1 Depth-First Deconstruction

One strategy is called “depth-first” decomposition. Figure 5.10 (above) suggest this possibility. Once any subsystem at any level has been identified and characterized as an opaque-box system (flows, interfaces, etc.) it is possible to go to the next level down inside that subsystem. For example, in Figure 5.10 subsystem C 0.9 has been identified and characterized while the other internal subsystems at level 0 remain “mysterious.” It is possible, even before

---

[29] For a better description of the computer science concepts of tree search see the Wikipedia article: https://en.wikipedia.org/wiki/Tree_traversal for more details. Accessed: 8/30/2017.

determining the other subsystems at this level, to focus on C 0.9 (in this example) as the new SOI immediately and begin decomposition of this system as shown in Figure 5.13.

The completion of the opaque-box analysis of C 0.9 allows for at least the identification of the in and out flows (along with placeholders for the relevant interfaces). This means that we have all the information needed to proceed with the deconstruction of C 0.9 in the same way we did with the original SOI. We can do a transparent-box analysis of C 0.9 to discover all of the components at level 1 (C 9.1, C 9.2, … C 9.5 in the figure; recall that the prefix, “0.”, has been dropped). But, alternatively, we can choose one interface on C 0.9 and proceed to discover and characterize the importer subsystem, in this example, C 9.1.

In other words, we do not need to deconstruct all of the SOI, necessarily, in order to delve deeper into the system’s sub and sub-subsystems. Once C 0.9 was discovered and characterized we could ignore the other subsystems in level 1 and focus on decomposition of C 0.9 alone. Of course, once we identified C 9.1 as the first sub-subsystem (presumably the product output from C 0.9) we could similarly ignore the other sub-subsystems at level 2 and proceed to deconstruct C 9.1, and so on down the levels as indicated by the red path lines in Figure 5.13.

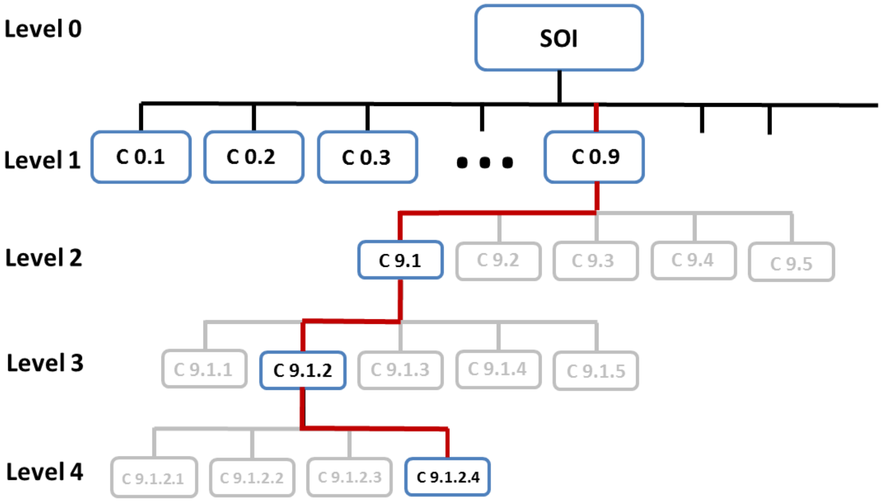

Fig. 5.13. The red path represents a potential depth-first decomposition of a system. The analyst may choose to pursue the decomposition of subsystems (and sub-subsystems) based on criteria mentioned in the text. The pursuit leads, in this example, to a sub-sub-sub-subsystem (level 4) before any other sub-…systems are identified and decomposed. The red path identifies a depth-first decomposition. The greyed components at each level exist but have not yet been identified or characterized and thus are not included in the analysis.

The reasons for doing this involve the control of the scope of an analysis, and that involves considerable foresight with respect to the nature of the level 0 SOI. For example, suppose in the analysis of an organization it has been *a priori* determined that a “problem” exists in a particular department deep in the organization (i.e. component C 9.1.2.4 in the figure). A depth-first decomposition analysis could be used to rapidly focus in on the workings of that department while “ignoring” the rest of the organization. However, that does not mean the beginning of the analysis is the department itself. There is no shortcut to rapidly circumscribing the problem area which is what a depth-first decomposition accomplishes. A principled and structured analysis of the system still requires starting at a sufficiently high level in the organization in order for the analysis to ascertain the complete context of the sub-…system where the problem exists. The number one mistake made by systems analysts (especially in organizational contexts) is to too rapidly vector in on the subsystem without adequate consideration for the context of how that subsystem works. A depth-first decomposition from the top level of an organization is used to establish that context and helps assure that all of the relevant factors (that is flows in and out of the system of interest) have been accounted for.

Backtracking as needed

5.7.3.1.2 Breadth-First Deconstruction

The second approach is to complete the analysis at each level before going down to the next level. For example, in Figure 5.12 all of the importers and exporters as well as all of the level 0 internal subsystems and flow channels have been completely identified. Then, choosing those subsystems, one after another, the process of transparent-box analysis is repeated for each. In other words, we do a breadth-first analysis, exposing all of the sub-subsystems at the next level.

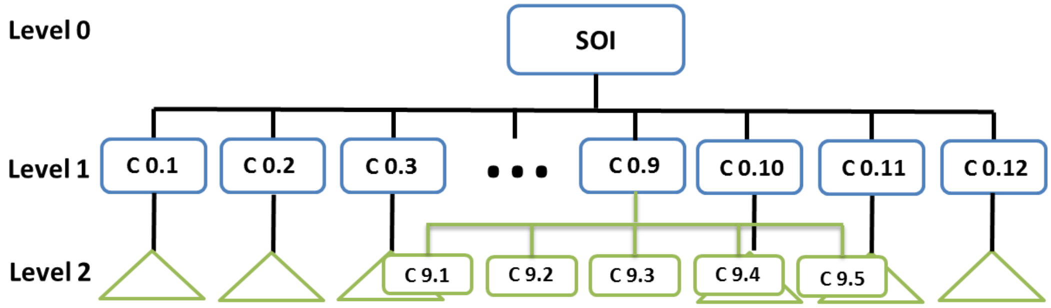

Fig. 5.14. A breadth-first decomposition completes the deconstruction of each of the subsystems at a given level (here level 0) before proceeding to the next level. All of the subsystems of the SOI constitute level 1. All are identified before any one of them is subsequently deconstructed. The green triangles represent the decomposition of each subsystem at level 1, with C 0.9 deconstruction shown explicitly as a set of sub-components, as in Figure 5.12.

5.7.3.1.3 Choosing an Approach

There is no simple rule for choosing a decomposition approach. Much depends on the needs of the analysis process. For example, in the desire to understand as completely as possible a 'new' system – to gain deep understanding – the breadth-first method would be most appropriate. On the other hand, as mentioned above, if the analysis is of an existing system such as an organization, in order to identify a 'problem' area, the depth-first method might be more appropriate. This is a decision that needs to be made by the analyst in charge.

If we observe the big picture of the process of Science (with a capital S) we will see that it has been a general process that alternates in efforts between these two approaches. Sometimes we delve deeply into the mechanisms of a system – the reductive approach or depth-first.

Sometimes the sciences proceed in a breadth-first manner trying to resolve questions of how various mechanisms work at a singular level before delving deeper.

As examples, consider the works done in natural biology versus molecular biology. In the former the science of ecology seeks relations between species, life histories, food webs, etc. seeking to obtain a broad vision of how different subsystems relate to one another, before delving into the details of, for example, trophic energy flows. In the latter, the emphasis has been on discovering, for example, how mitochondria work and then discovering how mitochondrial DNA contributes to this mechanism.

The sciences have been unguided by any coordinated decision as to which approach would work best. And this has worked quite well so far. Science has been spectacular at doing systems analysis without benefit of a structured, principled approach because it has been guided by the scientific process that is the search for objective truth. Quite fortunately the methodologies and overall science process has guided the accumulation of both deep and broad knowledge. But, it has also revealed the problem of not having a more disciplined approach. We are now on the verge of seeing how the real complexity of reality cannot be managed by ad hoc searches for truth. We have come full face with why the systems approach is mandatory for future progress in understanding reality.

From now on, it will be necessary to consciously pursue breadth-first and depth-first pursuits of understanding according to the needs of that pursuit. In the next chapters in this section we will provide some examples of this pursuit based on conscious decisions grounded in systems science rather than blind search.

5.7.4 Reverse Deconstruction

Up to this point we have probably left the impression that analysis is strictly a top-down process. This is generally the case, but not exclusively so. Suppose during the decomposition process we find anomalies that cannot be resolved by further decomposition. We will, from time to time, encounter new questions that cannot be answered with the information already available. What do we do then?

For the analysis of any SOI it can happen that the environmental analysis did not adequately capture a complete understanding of the sources and sinks originally identified. This generally is the result of leaving something out of the original SOI boundary analysis. In other words, the opaque-box analysis failed, originally, to identify an important input (or sometimes an output).

A frequent symptom of this is a failure of the mass or energy balance of transformation functions. An analysis of an internal subsystem indicates that there is an input that wasn’t caught in the original opaque-box analysis of the system. Let’s face it, real CASs and CAESs are not easy to analyze! Almost invariably something is going to get overlooked in the initial analysis. This is why it is necessary to iterate back to earlier stages when we run into anomalies.

Quite often the problem is that there was an input to the SOI (at any level of decomposition) that was not accounted for in the initial analysis. In these cases we are faced with several approaches to re-analysis. We can simply add a new source and input, looking for the relevant interfaces on the SOI boundary and the importer subsystem. Or we can consider the possibility that we have overlooked an essential component of the SOI that should actually be included. That is, we may need to 'refactor' the system to include previously unrecognized inputs, or we may need to expand the original boundary such that an external source becomes a subsystem of the SOI. More often we do both.

Figure 5.15 provides a view of a system in which the analysis exposed an energy importer and distributer subsystem. The interface analysis along with the environment analysis produced a mapping of the flows of energy and messages that allowed the importer to obtain needed energy from the energy source as shown. This model indicates that a single kind of energy is used internally (let's say electricity).

However, let us say that during subsequent analysis of energy uses within the SOI we discover that there exists a second kind of energy usage (e.g. gasoline for company cars) that had not shown up in the original analysis. This means that something was overlooked in the original boundary and environment analyses.

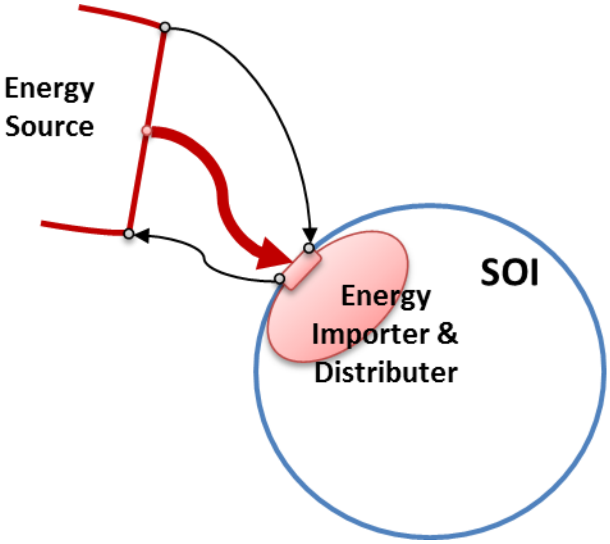

Fig. 5.15. The initial analysis of a system resolved an energy importer and its interface (with protocols) with an energy source in the environment.

From the principles of systemness, in particular network structures in hierarchies, we can reverse the direction of analysis. We start by re-analyzing the importer subsystem and discover a second energy flow and interface. We thereby discover a second energy flow/interface and then follow the second input flow back to its interface. A reanalysis of the energy sources and flows reveals that the source of energy inputs to the original SOI was actually a previously unmentioned (unrevealed) subsystem that properly belongs to the SOI as the "ultimate" importer. The restructured situation is shown in Figure 5.16.

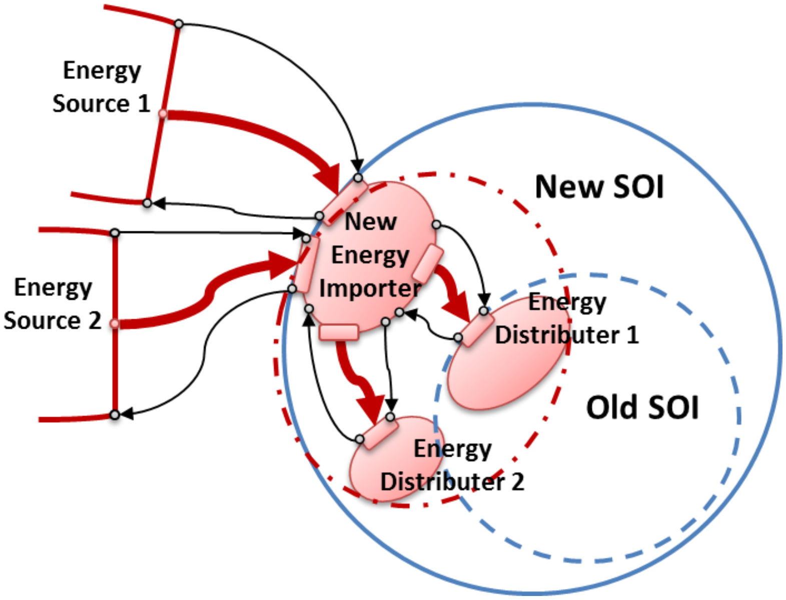

Fig. 5.16. The new SOI results from two operations that recognize a new importer subsystem and refactor the work of importation and distribution into three separate subsystems recognizing the distinctions in the two different energy types.

Note that the revised boundary analysis requires the addition of a new interface for importing the second kind of energy. The environment analysis also reveals the second source.

The red dashed oval in the figure indicates how the original analysis gave a single energy importer as in Figure 5.15. Since by the definition of a system this single subsystem is already considered to be decomposable into yet finer sub-subsystems, the discovery of another energy flow simply means leaving the original importer/distributer as coded and occupying the knowledgebase (tree) and inserting three new subsystems below it. In Figure 5.8 the energy importer was coded C 0.5. These three new subsystems would thus be C 5.1, C 5.2, and C 5.3 respectively. The new energy source can simply be added to the environment structure just as the interfaces can be added to the boundary structure.

5.7.4.1 Expanding Boundaries

There are many reasons for this operation. Generally, it is because an external source or sink were originally thought to be outside the ‘control’ of the SOI beyond the protocols of requests and notifications that accompany any regulated flow. Those protocols (e.g. purchase orders and receipts) are in the domain of cooperative cybernetics. This will be more fully explained in Chapter 11, but for now note that cooperation is a volunteer action. Neither agent/actor in a mutual procedure is compelled to follow through except to the extent there is some kind of reward for doing so (like the sender will get paid for doing so.)

In other cases, the external source or sink might actually be under more compelling forces emanating from the original SOI. For example, many big-box retail chains exert more than just cooperative influence on their suppliers, dictating, for example, prices that they can charge. At first glance the chain and the supplier seem like independent systems in their own rights. But in reality the supplier has become a captive of the chain and is, in effect, a de facto subsidiary. Thus, the true analysis of the SOI (chain)-supplier relation should probably include the supplier within the boundary and also include a decomposition of the supplier as an essential component of the ‘real’ system.

In very typical system dynamics modeling it is not uncommon to realize that an important variable that had a non-trivial impact on the dynamical results had been left out of the original model and needs to be included. This is an example of expanding the boundary of an abstract system, which seems to many as an arbitrary exercise. However, it is not much different from the above big-box chain-supplier example.

5.7.4.2 Refactoring

At times the expansion of boundaries, or even just the discovery of a previously left-out subsystem, will alter some aspects of the component coding that had been done to that point. In the example above of the second energy flow and interface, the boundary list of interfaces can be expanded but the ordering of the interfaces already discovered and coded makes it awkward to track the new objects. For example, the original interface for energy input was coded I5, corresponding with the single importer subsystem C 0.5. The next interface was to be found in the importer C 0.6 and numbered I6 accordingly. Now it turns out that there are actually two interfaces associated with C 0.5. Since these interfaces are not only subsystems of the importers but also associated with the boundary of the SOI this can create a bit of confusion. We could simply add the new interface to the end of the boundary list, numbering it I9, and make a special note that indicates it really belongs to C 0.5.

A more elegant approach is to refactor the coding scheme so that I9 is numbered I6 and all thereafter are renumbered (incremented). There is no necessary condition that would require the interfaces in the importers’ boundary lists to have the same number as the subsystem itself, especially if, as in this case, there are more than one interface associated with a more complex importer. At the same time, it would be useful to keep the interfaces in the SOI boundary list in sequence order of discovery.

The same conditions can be invoked in the case of discovering a previously un-discovered subsystem. This generally comes about as in the energy flow example above by finding anomalies in the internal mapping of subsystems and their flows. If, for example, the sum of the inflows to subsystems does not match the sum of the outflows of other subsystems then there is a clear indication of a gap in the analysis. Upon reanalysis the missing pieces can be added to the map and refactoring of codes as needed can be done.

Refactoring a complex system by hand would be a monumental task. But with the appropriate software tools for doing the analysis (to be discussed in the next chapter) and filling in the slots in the knowledgebase, the chore becomes trivial.

5.8 Iterative Analysis

Real systems analysis of extremely complex systems is in no way easy! But, for deep understanding it is essential. With the process being guided and done by human decision makers it would likely be impossible not to make mistakes and leave essential details out inadvertently. Fortunately, the procedures of analysis we have outlined in this chapter provide principled ways to recover from such mistakes. Or, alternatively, when the problem is a lack of analytical tools with which to ferret out answers, a portion of the system must necessarily remain an opaque-box with only a best guess system identification as a placeholder for more definitive knowledge. The analysis must await the invention of the appropriate tools[30].

By providing both pathways down from the top toward the revealing of the smallest components and from the components up to the very top (or as shown above to expand the scope of the original SOI) we have a way to iterate as needed over the space of the subsystems to be discovered and characterized. Whenever an anomaly appears at any level of the system model we can backtrack upward to discover what we missed at the parent level, or reiterate the downward recursive decomposition using a depth-first approach to get to the problem.

But for the lack of specific deconstruction tools, we have no excuse to not understand complex systems.

5.9 Considering Advanced Concepts

In the following four sections, we will briefly discuss how to deal with the more advanced concepts of complex systems that involve issues of fuzziness, adaptivity (CAS), evolvability (CAES), and memory recorded in the *H* object. All of these are factors found in every CAS or CAES and so must be considered when analyzing or modeling such systems.

5.9.1 Dealing with Fuzzy Systems

A fuzzy system is one in which components may be in the system during some period of time and outside that system (and in another system) at other times. A complete specification of the component in the set of objects that comprise the system (recall Equation 3.2 repeated below) requires a temporal membership function, the $m_{i,j,l}$ items in the equation.

---

[30] Not unlike how the better analysis of the brain had to await higher resolution EEG, single neuron unit recording methods, and fMRI tools in order to turn the opaque-box into at least a grey-box.

$$C_{i,l} = \{(c_{i.1.l}, m_{i.1.l}), (c_{i.2.l}, m_{i.2.l}), (c_{i.3,l}, m_{i.3.l}), \cdots (c_{i.k,l}, m_{i.k.l}), \cdots (c_{i.n,l}, m_{i.n.l})\}_l \quad \text{(Eq. 3.2)}$$

$$m_{i,j,l} = \begin{cases} 1, if\ component\ is\ always\ a\ member \\ f_{i,j,l}(\tau_i, p), otherwise \end{cases} \quad \text{(Eq.5.1)}$$

where, τ_i represents a time period in which the component can be found as a member of system i, and p is a probability factor as when a component is “likely” to be a member of system i during period τ_i. Recall from Chapter 3 that most systems have periodic or quasiperiodic behavior.

The idea that a component is a member of multiple systems but at different times and with different probabilities means that the component must be accounted for in those multiple systems. For example, let us consider a component $c_{i,j} \in C_i = \text{A}$, (the level index has been dropped for simplicity). Suppose the membership function associated with this component in A is,

$$f_{A,j} = \begin{cases} 1, if\ t \leq \tau_A \text{ with probability } p_A \\ 0, otherwise \end{cases} \quad \text{(Eq. 5.2)}$$

The problem is how to represent the very same component in a second or more other systems. Figure 5.17 depicts the situation in which a single component has membership in three different systems at different times and with different probabilities. The figure shows a period of $\tau\Delta t$ for the supra-system containing subsystems A, B, and C (Figure 5.17). The period is divided into three sub-periods in which a single component, $c_{i,j}$, will be found based on the count of t. If $t \leq \tau_A$, a constant, for example, then with probability p_A the component will have the identity $c_{A,j}$.

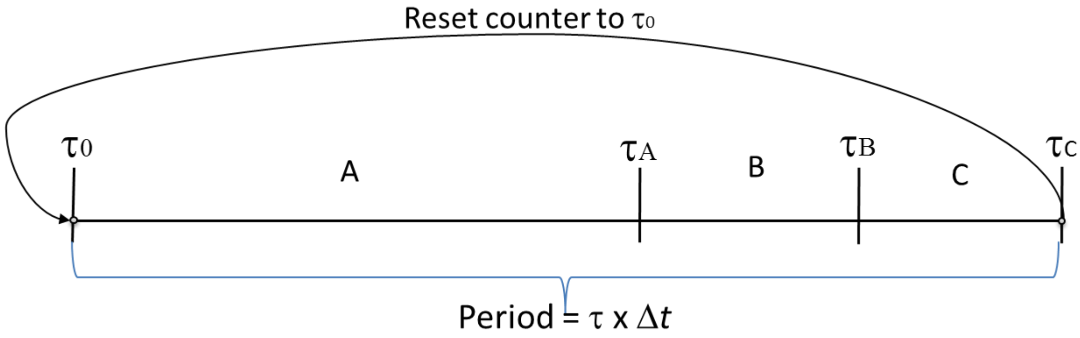

Fig. 5.17. This depicts a periodic cycle during which a component may have membership in any of three systems depending on the value of the counter, t.

Figure 5.18 shows the situation when the counter t is less than or equal to the constant τ_A, the component is located within system A.

Consider a human being who has a day job (system A), an evening gig playing guitar at a local pub (system B), and goes home afterward to eat and sleep (system C). This is not a hard scenario to imagine. The human is the component in question. Membership functions for the human component's presence in any of these three systems would be relatively easy to specify. The total period is a day and the sub-periods are the portions during which the human moves from one system to the next. This is a routine schedule, so is easily represented. As another example, one a little more exotic, consider the membership of an electron in a valence orbital in one or the other atom sharing that orbital in a covalent bond. The membership function would probably look suspiciously like the Schrödinger equation!

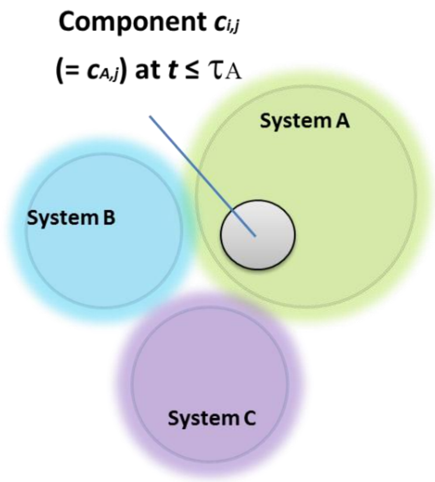

Fig. 5.18. A component can be a member of any of three systems (subsystems of a larger supra-system) but at times specified by the sub-period as in Figure 5.17.

There are probably many ways in which a system can be fuzzy beyond the membership of a component subsystem. For example, a variation on the fuzziness described above would consist of membership conditions on interfaces, which are, after all, just special kinds of subsystems associated with the boundary of the embedding system. We call this a 'fuzzy boundary' in which, for example, it is difficult to locate a 'solid' boundary in space and time. Fuzzy boundaries have been a difficult subject conceptually and have even led some theorists to reject the idea that there are real boundaries at all, asserting that boundaries are always an arbitrary choice of the analysts. Indeed, there are arguments to justify skepticism with respect to pinpointing a physical boundary. We already saw how it is sometimes necessary to 'expand' the boundary to include inside the system an environmental entity that had been previously (inadvertently) not included. This kind of exercise can lead one to conclude that boundaries are arbitrary and subject to the analyst's whims.

But, just because a boundary is fuzzy does not mean it is arbitrary. It simply means that it will take more information to specify it.

5.9.2 Dealing with Adaptivity

Adaptivity refers to the capacity of a system to alter or modulate an internal function in response to external changes in the environment. All CASs have to adapt because they operate in variable conditions. Simple or merely complex systems may be capable of operating normally in a wide domain of environmental variables without altering internal parameters. Adaptive systems, however, include mechanisms that can change their operations in response to changes in demands made on them. An airplane autopilot can respond to a wide domain of factors like wind speed and direction to stay on course. But there is a hard limit (not counting on over-engineering) to the domain size for these variables. So, from this perspective we could count an airplane with an autopilot as being in the category of CAS.

CASs have built-in capacities in the form of transformation functions of various (generally of specific) subsystems that allow them to continue operations in light of changes in normal environmental conditions. For example, if the input flow of a vital resource is reduced below the optimal level for proper functioning of the system’s overall performance. CASs include backup subsystems or ways to substitute other resources for those inputs being restricted.

An adaptive transformation function involves having a limited range of outputs related to a larger domain of input variations (many-to-few, see Figure 5.19). The underlying process that produces the transformation undergoes internal changes or shifts in operations to handle input conditions that are outside an “ideal” domain.

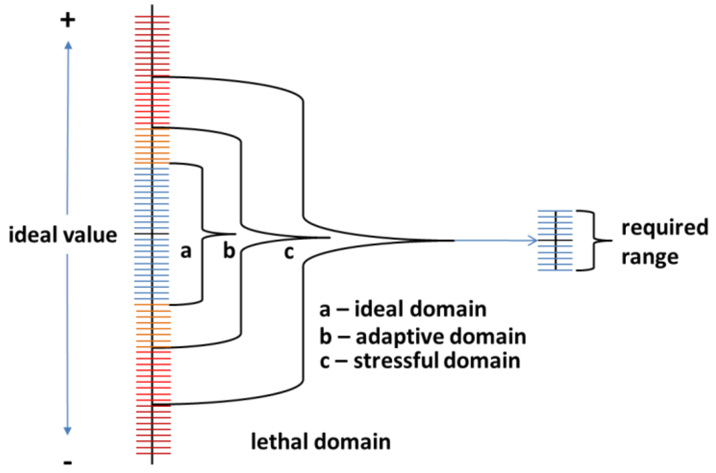

Fig. 5.19. An adaptive function has to be able to map, generally, a larger domain of input values to a limited range of output values (neither domains nor ranges need necessarily be symmetric about the ideal as shown here). The domain is divided into a set of subdomains around a central point or ideal value. Subdomain **a** is the small domain of ideal values or values which can readily be compensated for into the range. Subdomain **b** are values plus or minus the limits of subdomain **a** that require the system to modify its internal functioning in order to continue mapping the actual input value to the required range. Subdomain **c** is the values above and below subdomain **b** that require more work to accomplish the mapping than can be sustained by the system for longer periods; the effort is stressful. Finally, values outside of domain **c** are lethal and there is no mechanism that can accomplish the mapping.

Below we discuss several examples from the biological world.

The need for adaptivity arises from operating in a stochastic world in which the relevant variables cover a wide domain. Fortunately, in our world the dynamics of these variations are not random or dominated by noise. They are generally smooth functions over time, although occasionally a catastrophe or step-function-like variation can occur. But since the world isn't in constant turmoil life has been able to evolve limited capacities (in individual entities) to react to changes and compensate for conditions that get out of the 'friendly' domain.

5.9.2.1 Homeostasis

Homeostasis is a well understood feedback control function that is used to maintain a critical variable such as internal temperature or blood pH. The mapping function employs a mechanism that counters the input variable when it tends to get further away from the ideal value of the domain. For example, with body temperature, the input domain is the external temperature, which can vary considerably diurnally or seasonally. To counter the heating (+) or cooling (-) effects of ambient temperature on an animal's core, a mammal (a warm-blooded animal) may shiver, its muscles thereby generating extra heat, when its core temperature starts to drop (the range for core temperature is extremely small). Or it may pant or sweat when the temperature starts to rise.

Homeostasis is found employed widely in every biological process, including what are called demand-driven processes, where a depletion of a critical component signals the restocking operation to return levels to nominal. It can operate normally within the ideal domain. It can operate with some additional work and consumption of more than usual amounts of resources such as materials and energy reserves when the input value is in the adaptive domain, but not for very long (without the next described mechanism kicking in). And if the domain is in the stressful values the mechanism may break down physically. Both hyper- and hypothermia occur in the stressful regions of the domain and can cause irreparable damage to tissues if the temperatures remain above or below the merely adaptive domain respectively for too long.

Of course, a wall thermostat that controls heating and cooling in a building is an example of homeostatic device, even if not substantially complex.

5.9.2.2 Autopoiesis

A slightly more complex form of adaptive function is found in autopoiesis or self-constructing/maintaining mechanisms. As noted above, demand-driven processes are used to signal the need not just for components, but for the construction and/or repair of more complex subsystems, generally that supply those components. For example, when mitochondria, the energy supply subsystem in cells, are decommissioned other mitochondria are put under a stress load and respond by reproducing. Both the internal genes and those nuclear genes associated with mitochondrial construction are put into play to produce whole new mitochondria to relieve the load. The same mechanism kicks into play when long-term demand for muscular work (as when someone does strength training with weights). Not only does each muscle fiber require more energy reserve, but all of the other support systems need to be 'bulked' up as well. More fibers are needed to adapt to the stresses as well as more blood supply and lung capacity.

Autopoiesis is called into play to repair damage to other subsystems. For example, when homeostasis alone is not sufficient to counter a domain value in the stressful region and damage to that mechanism itself occurs, then the repair (once conditions return to the ideal region) of the mechanism can be accomplished. But it uses materials and energy out of reserves and incurs a cost above normal operating costs in those substances.

5.9.2.3 Adaptive Response

In general, biological systems (at the single individual level) have various capacities for adaptability in response to changes in relevant parameters in their ambient supra-systems. These capacities evolved over time when the parameter (or condition) tended to vary from a previous norm (the ideal domain) slowly enough that genetic mutations giving rise to changes in the mapping function could pursue those external trends. We will return to this below in dealing with evolvability.

The basic adaptive capacity follows a pattern. Minor variations within the ideal domain can be handled through the existing mechanism (homeostasis). Short term variations outside of this domain may be handled by recruiting extra resources for a period corresponding to the demand variation. A general rule of biology and the condition of mechanisms and tissues is, “only maintain readiness to the degree that the average demand requires.” If for short periods of time, the conditions go outside of that mapping, be prepared to recruit help, but once the situation normalizes, go back to the average state of preparedness.

If, however, conditions persist over extended time frames, cells and tissues have the ability to build their homeostatic capabilities up through autopoietic mechanisms as described above. That is, the capacity is adapting to higher levels of demand and the average level of preparedness may rise accordingly. The weight training, muscle building example above is in this category. If a person just starts weight training the strength capacity does not adapt immediately. It takes time and repetition to reinforce the signals that there will be higher demand occurring as a “new” norm. Over time, the muscles respond by building bulk and strength capacity. The person’s body has adapted to new conditions and the level of preparedness for dealing with heavier lifting has risen.

Of course, if that person, after adapting to heavier weights, loses interest in looking buff and stops training, over time the muscles will lose strength and preparedness, returning to their pre-training status, or at least nearly so. The loss of capacity seems to take longer than the gaining. This phenomenon is called memory and it fades at a rate much longer than the initial learning.

Mobus (1999) provides a detailed analysis of the phenomenon as the basis for neural synaptic plasticity and learning in biological neuronal networks. The paper includes analysis of various ‘costs’ associated with ordinary operations (homeostasis), construction of new capacity (autopoiesis), and repair of damage by not having an increased capacity for response to stressing stimuli. The purpose of adaptive response is to minimize these latter costs by possessing a greater capacity of preparedness[31].

5.9.3 Dealing with Evolvability

This is the most difficult aspect of a system (a CAES) to analyze and characterize. Evolvability is a capacity for a system to go beyond mere adaptivity and change the adaptive function entirely. Usually this happens in a speculative fashion, as in genetic mutations in populations that give rise to new capabilities that are then tested for efficacy by the selection function of the environment. This strategy works well when there is a large population of evolvable elements as is the case in genes in a population of a species. To be clear, the evolvable system in that case is the species, not the individuals even though the mutation affects genes in various individuals (as, for example, a cause of cancer). For genetic mutations to affect the population or species it has to be in the germ line cells (i.e., sperm or ova). They are then heritable.

The general principle may be called “copy error with inheritance.” A mutation in a gene in a germ cell is such an instance in the biological world.

Human organizations are not a population of similar entities in the same sense that a species is. Even so organizations operate on codified knowledge like genes and chromosomes in biological entities. For example, most organizations have bylaws, policies, and procedures. The later are like genes in prescribing how some subsystem is to behave. And organizations work at persisting in time. The encoded knowledge has to be actualized (performed) by humans. They are analogous to the enzymes that encode the knowledge in the gene’s DNA into messenger RNA and, in turn, like ribosomes that produce the proteins from the mRNA messages.

From time to time a human can misinterpret a procedure (or a policy) and implement something different from what was intended in the original code. If the changed implementation is effective it can be repeated over time leading to what amounts to a genetic mutation. At some later time, someone may notice that the original code is not being followed, but the new ‘practice’ is working better than the original would have. The procedure may be rewritten to reflect the newer practice and so the mutation is, in this sense, heritable.

Yet another example of an evolvable system is the mammalian brain with its neocortex. In the case of humans, anyway, it seems to be able to encode an infinite variety of concepts, both from actual experience and from a creative process we call imagination[32]. The later are the interesting case because it resembles mutation-like change in concept space that can lead to

---

[31] In Mobus (1999) the exploitation of adaptive response in neurons produces a capacity for anticipatory behavior that is shown to reduce overall costs of responding to higher demands since an animal, if it can predict the onset of an excessive demand, can act to avoid it.

[32] There is a growing body of evidence that many other species, including birds, generate some novel representations that lead to adaptive behavior.

altered or new behavior. Once again, if that change leads to success for the actor, then the originating concept is reinforced and inherited into the future time frames.

Evolvability is sometimes described as a species having the ability to allow increased mutation in certain genes when under environmental stresses. However, other writers simply require enhanced variability potential. All agree that genetic mutations that occur to create that variability must have a relatively high frequency of viability (most random genetic mutations would presumably be detrimental or neutral)[33].

For other kinds of systems, e.g. ecosystems, economies, or organizations, evolvability might be a combination of chance change (like mutation) in a particular subsystem and intentional modification in anticipation of future conditions. Organizations that do strategic planning (successfully) and human beings who 'think ahead' are examples of intentional modification of subsystems in anticipation of the future.

The formal definition of system given in Chapter 3, especially Equation 3.1, does not directly make provisions for handling evolvability in a system. The concept implicitly involves considering very long histories of the system (or class of systems). Thus, we will briefly outline an approach to dealing with evolvability as part of dealing with the system history. Evolvable systems, especially in the realm of model building, is, itself, an evolving area. In other words, more research is needed!

5.9.4 Dealing with Memory – And the *H* Object

By definition an adaptive system must retain a 'memory' of its past experiences. It behaves differently at a later time than it did at an earlier time for the same set of conditions because the past conditions were recorded in some manner that influences the condition/action mapping. Thus an adaptive system's history impinges on its future.

As described in Chapter 3, the history object is, or can be, fairly complex and varies from system to system. For example, a computer-based system, or subsystem of a larger work system, maintains exact recordings of states and events in digital coded form. These recordings are stored in databases or, by extension, on archive media. If records are ever deleted they are gone. Alternatively, the human brain (or any animal brain for that matter) does not store explicit representations of states and events in the same way a computer does. Memory in the human brain works on completely different principles that are only now beginning to be understood (Squire & Kandel, 2009). However, some broad comments will help in understanding why memories are difficult subjects.

The word itself has different meanings based on context and field of study. In general, however, it refers to the incorporation into the system of a lasting impression that has impact on

---

[33] See the Wikipedia article: https://en.wikipedia.org/wiki/Evolvability for a general background discussion of the concept related to biological systems. Accessed 9/3/2017.

a system's future behavior. As covered in Chapter 2, this what we generally mean by 'knowledge', except we differentiate between knowledge that is modifiable based on on-going experience versus the knowledge that is a priori built into the structure of the system that allows it to interact with other systems. The first kind is 'learned' knowledge and is a characteristic of an adaptive system.

A system that retains a memory of experiences that it has had behaves according to both its current situation and the memory of prior situations for which the memory is relevant.

There are fundamentally two kinds of memories resulting from experiences[34]. The first is the exact recording of states of the world (conditions) and of the system itself. In human psychology these are called episodic memories, as when you remember specific incidents in your life. Such memories, in the case of humans, are not accomplished the same way a digital camera captures every pixel and fixes them in its memory. And human episodic memories are notoriously prone to modifications.

Episodic memory is one example of a more general ability to record specific concepts (e.g. facts) and have ready recall of those memories for use in declarations. This is called explicit or declarative memory. We do not have a clear idea as to how much explicit memory capabilities other animals possess. We do know that many mammals and birds are able to recognize other individuals, for example, but we do not know if their internal memory of, for example faces, can be brought into consciousness in the same way we can.

The second major category of memory is called 'implicit' or 'tacit'. One commonly experienced form of implicit memory is called 'procedural' memory, as when you just know how to ride a bicycle without having to consciously recall each action you need to take to do so. Implicit memories appear to be formed from repeated experiences of successful actions in response to contextualized situations. A major form of this kind of memory for humans is what we call intuition.

Whereas in episodic memory the concepts are explicitly recorded and recalled, even if only fuzzily, in tacit memory experiences appear to be used to develop and reinforce a situation/action mapping that works for all similar contexts.

This is a mechanism that seems to apply to all adaptive systems (Alkon, 1987; Mobus, 1999; Squire & Kandel, 2009; Sutton & Barto, 2017)[35]. Basically, the memory trace is a kind of time weighted average of all of the experiences accumulated to date. The Adaptrode, which models

---

[34] The following discussion as applied to human memory in particular is based largely on the works of Squire & Kandel (2009).

[35] This is certainly the case for living systems. But it is also the case for a number of manmade systems that are considered adaptive, e.g. Kalman filters (see the Wikipedia article: https://en.wikipedia.org/wiki/Kalman_filter for background).

the dynamics of biological synapses (Mobus, 1994) uses the following formula for updating a synaptic weight:

$$w_t^0 = w_{t-1}^0 + \alpha x_t^0 - \delta w_{t-1}^1 \qquad \text{(Eq. 5.3)}$$

w is the synaptic efficacy weight. The superscript refers to a time domain; 0 is the "real-time" domain, whereas 1 is the time average window. The variable $\boldsymbol{w^1}$ is a slowly moving "basement" value. α and δ are rate constants, generally much less than 1. And $\boldsymbol{x^0}$ is the real-time input (an action potential or rate of action potential arrivals at the synapse). The Adaptrode is maintaining a time averaged value that ranges between 0 and 1 that represents the history of activation. Long periods of quiescence results in a weight near zero. On the other hand, a high frequency of stimulations pushes the value toward 1. In (Mobus, 1994) it is shown how this mechanism operates to form a memory trace in a neural network (see also, Alkon, 1987 for the biological basis for this model).

Thus, memory traces in the brain are based on time averaged experiences and are the basis for tacit memory formation. This is thought to be the basis for all memory encoding in the brain. Explicit memories appear to be based on a new trick that the neocortex can do, using implicit encoding but putting the memories in effectively isolated parts of the cortex (especially more frontal areas).

The *H* object in a system knowledgebase may be instantiated by some form of network-based encoding scheme such as this. In Chapter 11, on governance, we will discuss in greater depth the meta-system called a 'decision agent.' We will see that this is a special case of an adaptive (and evolvable) system that relies on having a storehouse of experientially-based, implicit knowledge. That knowledge is acquired by learning. Thus, it is probably necessary, in order to capture knowledge of such systems, that the *H* object be, in effect, a brain-like mechanism (or, in other words a human emulating AI!) Clearly, a great deal of research is needed in this arena.

5.10 Conclusion

In this chapter we have given a rough outline of the process of systems analysis that leads to deep understanding, or the complete deconstruction of complex systems, turning all opaque-boxes at all levels in the decomposition hierarchy into transparent-boxes, and complete with transformation functions for each.

Systems analysis is the heart of the entire system understanding process. It is the process that extracts information about the system and collects that information into a systemic data structure (the knowledgebase to be described in Chapter 7) given by the formal definition of a system given in Chapter 3.

The procedures describe in this chapter provide a maximal flexibility in divining the details of systems and how they work. The analyst has a maximum amount of flexibility in pursuing system details in either a top-down or bottom-up fashion (iterating as necessary). Assuming the necessary instrumentation is available to turn an opaque-box into a transparent one, the procedures provide an algorithmic approach to obtaining knowledge of the system.

In the next chapter we will describe a set of specific tools that would be needed to support the activities of a full systems analysis and then a series of examples of how the process works with a few representative CASs and CAESs.

5.11 References

Alkon, DL (1987) *Memory Traces in the Brain*, Cambridge University Press, Cambridge UK.

Alur, R. (2015) *Principles of Cyber-Physical Systems*, The MIT Press, Cambridge MA

Checkland, P. (1999) *Systems Thinking, Systems Practice, Includes a 30-Year Retrospective*, Wiley, New York

Kahneman, D (2011) *Thinking, Fast and Slow*, Farrar, Straus and Giroux, New York

Klir, GJ (2001) *Facets of Systems Science*, Second Edition, Kluwer Academic/Plenum Publishers, New York

Mobus, GE (1999) “Foraging Search: Prototypical Intelligence”, *The Third International Conference on Computing Anticipatory Systems*, Liege, Belgium, August, 1999 and The Journal for Computing Anticipatory Systems, Vol. x, pp yyy-zzz,

Prandoni, P. & Vetterli, M. (2008) *Signal Processing for Communications*, EPFL Press, Lausanne, Switzerland. Available in .pdf at no cost: http://www.sp4comm.org/. Accessed: 8/13/2017.

Smith, SW (1997) *The Scientist and Engineer's Guide to Digital Signal Processing*, California Technical Pub. California. Free, on-line version: http://www.dspguide.com/. Accessed 8/13/2017.

Squire, LR & Kandel, ER (2009) Memory: From Mind to Molecules (Second Edition), Roberts & Company, Greenwood Village, CO

Sutton RS & Barto, AG (2017) Reinforcement Learning: An Introduction (2nd Ed. In Draft), The MIT Press, Cambridge MA

Wolf, M. (2012) *Computers as Components: Principles of Embedded Computing System Design*, Elsevier, New York