Paper #117

System Frameworks

Joseph J. Simpson
System Concepts
6400 32nd Avenue N.W., #9
Seattle, WA 98107
jjs-sbw@eskimo.com

Abstract

The activities associated with systems engineering are closely associated with problem solving and complexity reduction. Many claims are made about the benefits of systems engineering, both for complexity reduction and complexity management.

This paper will explore the relationship between systems, meta-systems, complexity types, complexity measures and complexity reduction. The primary thesis of this paper relates to systems engineering as a method to reduce complexity. A system "abstraction frame" concept is developed and used as a device to support the analysis and measurement of complexity associated with systems and systems engineering. Meta-systems concepts are explored for use as a mechanism to further describe systems engineering activities and also as a basis for the development of a systems definition framework.

Introduction

The ability to create and deploy systems has two concurrent impacts, 1) development of new conceptual and scientific tools and 2) increasing environmental (the system context) variety and complexity. These concurrent impacts must be separated and evaluated in terms of a given system environmental context and system abstraction frame. The abstraction frame is viewed as a "snapshot" in time with the upper bound of human understanding and design capability labeled as a "complexity limit." As our abilities and understanding evolve, activities that were complex in a past abstraction frame become simple activities in the current abstraction frame. Similarly, human activities that are complex in the current "abstraction frame" may become simple in the next abstraction frame. Systems engineering research must focus on identifying the practices, processes and mechanisms that achieve this type of complexity reduction. The complexity reducing aspects then need to be quantified and incorporated into the general practice of systems engineering.

System complexity metrics are also developed and used for the measurement of "system complexity." Alternative views of complexity and systems development complexity are then bounded in a conceptual framework that has provisions for complexity measurement, complexity reduction and complexity management.

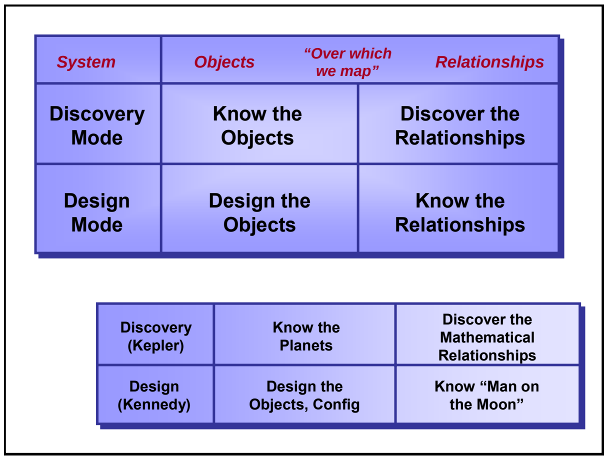
The rate of systems development and deployment is increasing. This constant rate increase can no longer be ignored and techniques to deal with these dynamic contexts must be developed. This paper proposes a series of mechanisms to manage the dynamic context change, including: system context abstraction frames, a systems engineering language, a systems development framework and standard metrics to measure the change in the system context.

System and Meta-system Definitions

Several standard sources of systems engineering literature were reviewed and analyzed to produce the following definitions for use in this paper. Two general types of definitions are presented for a system, one - a "construction rule" type and the other - a functional type. The "construction rule" definition for a system is "a non-empty set of objects and a non-empty set of relationships mapped over these objects and their attributes." The functional definition of a system is "a constraint on variety," wherein constraint identifies and defines the system.

Meta-systems are also given different types of definitions. The first definition of a meta-system comes from A.D Hall, who describes a meta-system as "a set of value sentences which describe the wanted physical system, and which imply or actually comprise the parts and relationships of the meta-system" (Hall, 1989). Palmer provides two definitions of a meta-system each of which is focused on the relationship between a system and the system's environment. These definitions are (1) "the meta-system is normally a set of integrated complementarities of complementarities that defines the environment or ecosystem that the system finds itself within, and inhabits"(Palmer, 2000); and (2) "the meta-system is all the possible sequences through all the possible gestalts in an environment considering all the systems in that environment" (Palmer, 2002). Heylighen defines a meta-system as "a constrained variation of (a) system(s), i.e., a constrained variation of constrained variet(y)(ies)" (Heylighen, 1994). These definitions of a meta-system are related in that (1) all value is determined in context and (2) the concept of a system is used in different modes.

In practice, a system is viewed in two different modes: system discovery mode and system design mode (see Figure 1). In the system discovery mode, the objects are known and the controlling relationships are in the process of being determined. In the design mode, the controlling relationships are known and the objects are in the process of being designed. In either operational mode, a system is defined by discovering or specifying the system's boundary and function. System definition proceeds by developing greater levels of detail in an incremental and recursive fashion. Standard systems engineering processes are used to manage and control large distributed system design activities. System discovery activities are controlled by the general scientific problem solving process. (Simpson and Simpson, 2003)



System	Objects	"Over which we map" Relationships
Discovery Mode	Know the Objects	Discover the Relationships
Design Mode	Design the Objects	Know the Relationships

Discovery (Kepler)	Know the Planets	Discover the Mathematical Relationships
Design (Kennedy)	Design the Objects, Config	Know "Man on the Moon"

Figure 1. Discovery and Design Modes.

Each system exists in a complex context that has a direct relationship to the system. One of the most important aspects of the system context is the environmental "problem" the system was designed to solve. At any point in time, an engineered system exists to solve some problem in the general environmental context (Goody and Machol, 1957). However, the general environmental context can change in a manner that reduces the value of the system. The environmental problem can change, another system can solve the problem more effectively or the system components can change. System design and development tasks become more complex as more engineered systems are introduced into the environment and each engineered system contains a high percentage of software. These existing complex systems can be combined to quickly produce even more complex systems (Warfield, 2002).

Systematic, structured process is one of the fundamental tools used to control and cope with increasing complexity in system design activities. These processes often take the form of a meta-system. CMMi is an example of this type of controlling/coping mechanism. The CMMI processes are designed to constrain the variability in the product systems that are produced. There is also an increasing emphasis on system abstract architecture including software, functional and physical views. Abstract architectural designs will change at a much lower rate than the physical representation of these abstract functional and operational architectures (Levis and Wagenhals, 2000). The Department of Defense Architectural FrameworK and the Zackman Framework are examples of these types of system architectural description activities. The U.S. Air Force has established an Integrated Information Environment (IIE) as a standard mechanism for exchanging information among systems.

System Complexity Measures

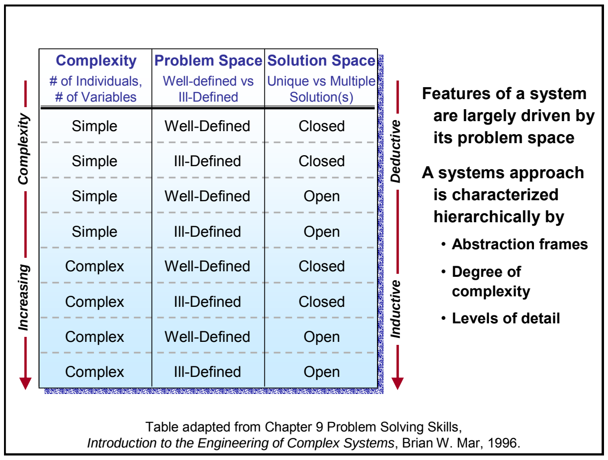
System engineering processes and practices are presented as a mechanism to reduce the complexity associated with system design, deployment and operation. However, only a few scattered system and process complexity metrics are found in the systems engineering literature. Clear system complexity metrics must be developed to facilitate the measurement of system complexity. System complexity reduction or increase will only be consistently controllable after these metrics have been developed and applied in a structured fashion.

Given the previous fundamental definition of systems and meta-systems, the following eight primary system complexity metrics were developed:

- Number of objects
- Number of relationships
- Number of different types of objects
- Number of different types of relationships
- Rate of change of objects
- Rate of change of relationships
- Rate of change of the environment
- Range of variability

The definition of system complexity has been restricted to a combination or a subset of these system characteristics by some authors. One such metric for complexity states: “Complexity is a measure of the inherent difficulty to achieve a desired understanding. Simply stated, the complexity of a system is the amount of information necessary to describe it” (Bar-Yam, 1997). This metric is a general addition of all the above factors. Shell limits the definition of system complexity to “an implemented system solution that, at a particular level of design abstraction, is perceived to contain many (possibly interconnected) component parts.” The behavioural and functional relationship complexity aspects are defined as “system complication” by Shell. (Shell, 2003)

The complexity of a system can also be associated with the complexity of the problem the system was created to solve. As shown in Figure 2, the number of individuals and/or variables, the definition of the problem space, and the solution space itself will impact its overall complexity. This problem space topology provides a graphic outline of this type of complexity mapping. It is clear the “problem space” and “solution space” metrics are strongly associated with the meta-system and weakly associated with the system solution.



Complexity # of Individuals, # of Variables	Problem Space Well-defined vs Ill-Defined	Solution Space Unique vs Multiple Solution(s)
Simple	Well-Defined	Closed
Simple	Ill-Defined	Closed
Simple	Well-Defined	Open
Simple	Ill-Defined	Open
Complex	Well-Defined	Closed
Complex	Ill-Defined	Closed
Complex	Well-Defined	Open
Complex	Ill-Defined	Open

Increasing Complexity
Deductive
Inductive
Features of a system are largely driven by its problem space
A systems approach is characterized hierarchically by
• Abstraction frames
• Degree of complexity
• Levels of detail

Table adapted from Chapter 9 Problem Solving Skills, *Introduction to the Engineering of Complex Systems*, Brian W. Mar, 1996.

Figure 2. Problem Space Topology.

This observation emphasizes the need to establish a clear context and environment for the system or systems of interest. A well-developed system context provides supporting information about the system and is also considered an integral part of the meta-system by some authors (Hall 1989, Palmer 2000, 2002).

System Abstraction Frame

Logical rules for system description and operation are excellent tools for complexity reduction because they reduce the volume of text required to describe the system as well as providing a clear set of operations that are associated with the system. Very complex ideas and systems can be effectively described with a small set of logical rules (Grady 1995, Mar 1997). System abstraction frames are used as a mechanism to further organize logical aspects of system development.

System abstraction frames encapsulate the system discovery mode and the system design mode activities associated with system development. Knowledge developed from the system discovery mode is captured in a usable format and applied to the current system design activities. Operational systems generated in one abstraction frame are available for use in the next system abstraction frame. System abstraction frame sequencing and content relationships are shown in Figure 3 and Figure 4.

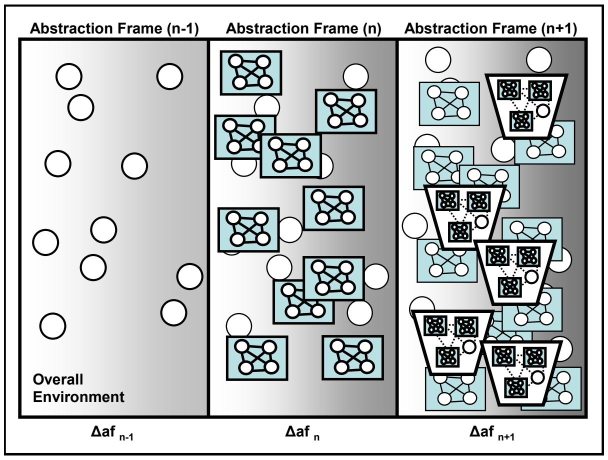


Figure 3 Abstraction Frame Sequencing

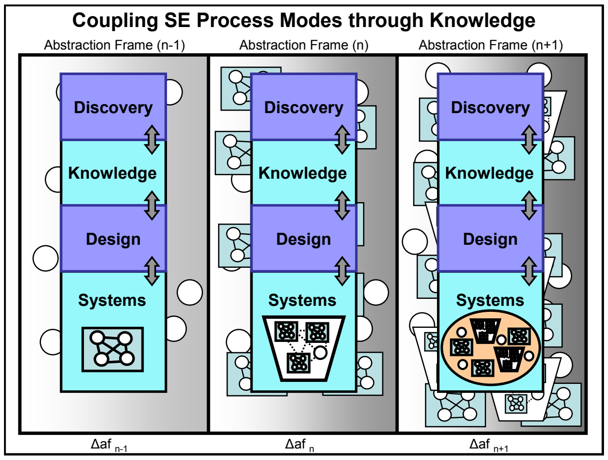


Figure 4 Frame Context Relationships

These frames will now be applied to system design using a modified FRAT process, a simplified system engineering process (Mar and Morias 2002). The modified FRAT process is named CCFRAT, after the six views of any system required by this method. The CCFRAT model was developed by adding two additional holistic views to the classic FRAT model. The new holistic views are the context view and the concept view. These two new views focus on the value set constraining the total system context and the controlling concept set associated with the relationships contained in the core design process. System abstraction frames and meta-system concepts are strongly associated with these views. The first step in the CCFRAT process is the establishment of the system context, including a definition of a current system abstraction frame. The system context is set by the environment system and the system production system. Some authors consider the system context the "meta-system" which is the source of all system application values and the primary force for change associated with the current system (Hall 1989, Palmer 2000, 2002, Simpson 2002).

Every system has six views used to identify the logical relationships important to the system at each level of abstraction and development. These six views are: the context view (defines the system context and meta-system), the concept view (defines conceptual relationships inside the system), the functional view (defines what the system does), the requirements view (defines how well the functions are done), the answer view (defines how the function is performed), and the test view (defines how you know that the answer does the function as well as the requirement states). See Figure 5.

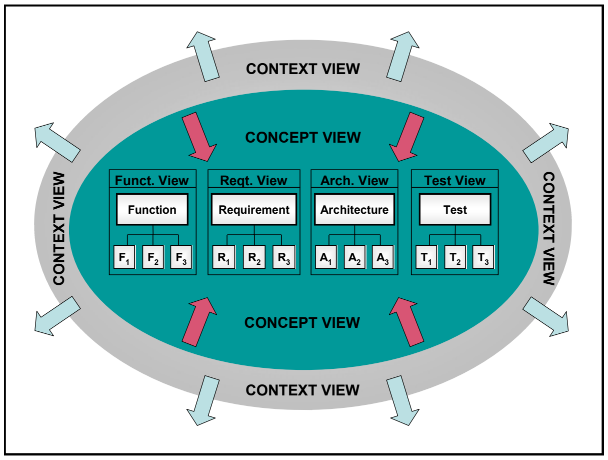


Figure 5. CCFRAT.

As CCFRAT concepts and processes are used to recursively model and design the system of interest, system models and descriptions are developed at lower and lower levels of detail. At the same time, the number of unique system solutions are reduced and the design space is further constrained. At each abstraction level the system is modelled and evaluated against the given concept and value constraint context. The well-defined logic of the CCFRAT process provides a mechanism to directly relate each level of system decomposition with the levels above and the levels below the current level of interest as well as to define value relationships that have network-type non-hierarchical connections to other systems in the environmental context. When the FRAT design engine is used at the core of the CCFRAT activities, these value connections and relationships appear as constraints, goals, objectives, value-structures and decision criteria. These aspects of system design are recorded, modelled and evaluated in the context and concept views of the system as shown in Figure 6.

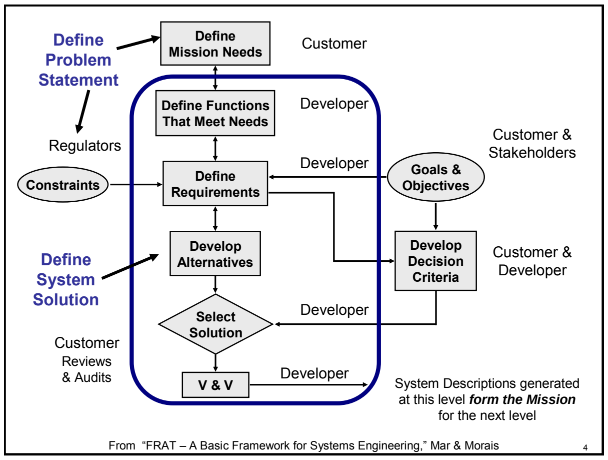


Figure 6. The FRAT Design Engine.

The system concept view details the relationships between the system functions and the architectural design as well as specifying the proper level of abstraction for system modelling. The inward focus of the concept view is combined with the outward focus of the context view to emphasize the system boundary and to provide a common abstraction layer for the complete system under evaluation. At high levels of abstraction the system may be modelled with only the context view or a combination of the context view and the concept view. Using these two holistic views of a system, network models, value models, functional constraint models and other types of non-hierarchical system interactions can be more readily evaluated without resorting to a detailed functional analysis or architectural analysis of each component in the proposed system. Parametric values and performance budget values can be assigned in these views of the system. When alternative functional and architectural system solutions are prepared these candidate solutions may then be evaluated against these parametric performance constraints. Figure 7 presents an example graphic of a system context network.

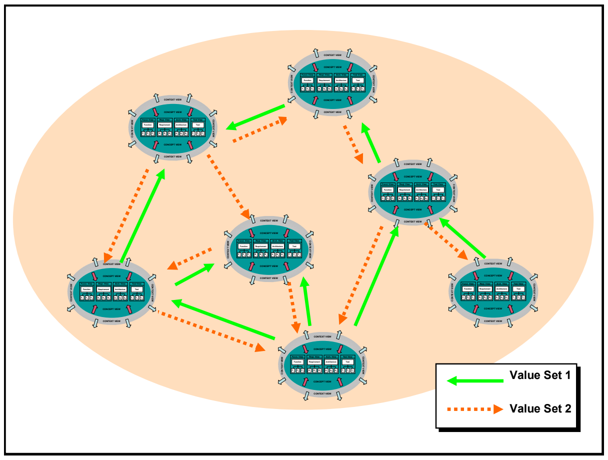


Figure 7. Value Network in the Context View.

These system context views can then be combined with a standard systems approach to create a robust set of system views and abstractions. The classical FRAT process requires that the four core system views need to be developed for each system abstraction level before moving on to design the rest of the system (Carson, 2000). The context and concept views provide a mechanism for creating layered system frameworks that can be evaluated in a parametric manner. If the system is modelled as a hierarchy, then the evaluation of alternative component connections is facilitated using these two holistic system views.

Figure 8 provides a graphical outline of the CCFRAT approach as it applies to system phases, content and hierarchies. The core meta-process is a generalized problem solving process that is recursively applied across any specific problem space (Simpson and Simpson 2001). The system evaluation hierarchies can take many forms that range from real component decomposition hierarchies to abstract conceptual hierarchies. The meta-system levels are an example of abstract conceptual hierarchies that apply to the systems engineering domain.

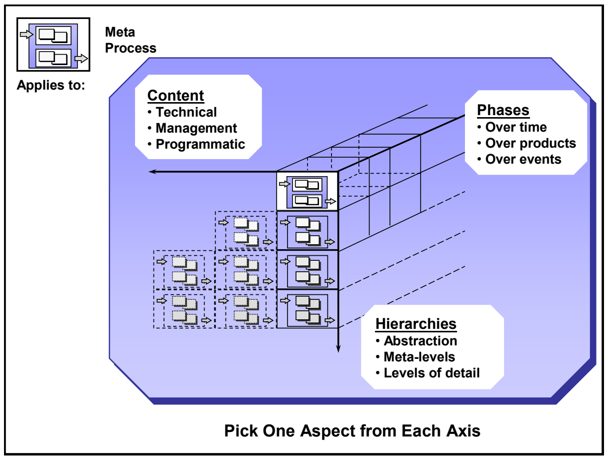
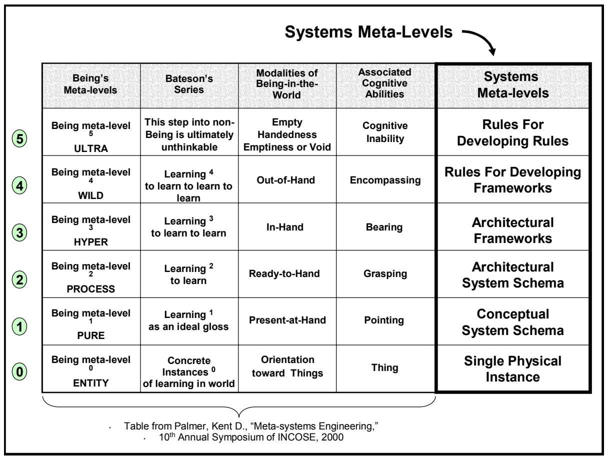


Figure 8. CCFRAT Approach: Phases, Hierarchies, Content.

These system views can then be evaluated and analyzed using the system meta-levels shown in Figure 9. This robust set of system meta-levels provides the system designer or system engineer with a clear set of conceptual transforms for use in the reduction of system complexity. The set of system meta-levels and meta-level transforms must be formalized in a structured fashion to support the development of a systems engineering language. Using a combination of the CCFRAT methods with well-defined system meta-levels and heuristics provides a strong basis upon which a formal system engineering language can be developed. A formal systems engineering language will greatly reduce complexity in the practice of systems engineering, as well as facilitate the solution of even more complex problems.

n.



Systems Meta-Levels

	Being's Meta-levels	Bateson's Series	Modalities of Being-in-the-World	Associated Cognitive Abilities	Systems Meta-levels
5	Being meta-level $_5$ ULTRA	This step into non-Being is ultimately unthinkable	Empty Handedness Emptiness or Void	Cognitive Inability	Rules For Developing Rules
4	Being meta-level $_4$ WILD	Learning ^{4} to learn to learn to learn	Out-of-Hand	Encompassing	Rules For Developing Frameworks
3	Being meta-level $_3$ HYPER	Learning ^{3} to learn to learn	In-Hand	Bearing	Architectural Frameworks
2	Being meta-level $_2$ PROCESS	Learning ^{2} to learn	Ready-to-Hand	Grasping	Architectural System Schema
1	Being meta-level $_1$ PURE	Learning ^{1} as an ideal gloss	Present-at-Hand	Pointing	Conceptual System Schema
0	Being meta-level $_0$ ENTITY	Concrete Instances ^{0} of learning in world	Orientation toward Things	Thing	Single Physical Instance

- Table from Palmer, Kent D., "Meta-systems Engineering,"
- 10th Annual Symposium of INCOSE, 2000

Figure 9. Meta-Levels.

References

Reference list.

Bar-Yam, Yaneer, *Dynamics of Complex Systems*. Perseus Books, Reading, MA, 1997.

Carson R. S., "Global System Architecture Optimization: Quantifying System.Complexity." *Proceedings of the Tenth Annual International Symposium of the International Council on Systems Engineering*. INCOSE-2000, Minneapolis, MN, USA

Goode H., Machol R., *Systems Engineering, An Introduction to the Design of Large-scale Systems*. McGraw-Hill Book Company, New York, 1957.

Grady J. O, (1995), "The Necessity of Logical Continuity." *Proceedings of the Fifth Annual International Symposium of the National Council on Systems Engineering*. NCOSE-95, St. Louis, MO.

Hall, Arthur D., *Metasystems Methodology, A New Synthesis and Unification.* Pergamon Press, New York, NY, 1989.

Heylighen, Francis, "(Meta)Systems as Constraints on Variation – A classification and natural history of metasystem transformations." *Free University of Brussels Research Paper sponsored by Belgian National Fund for Scientific Research (NFWO)*, Brussels, Belgium, 1994.

Levis A. H., Wagenhals L. W., "C4ISR Architectures: I. Developing a Process for C4ISR Architecture Design." *Systems Engineering: The Journal of the International Council on Systems Engineering*. Vol. 3 Num. 4. 2000.

Mar B. W., Morais B. G., "FRAT – A Basic Framework For Systems Engineering." *Proceedings of the Eleventh Annual International Symposium of the International Council on Systems Engineering*. INCOSE-2002, Las Vegas, Nevada.

Mar B. W., "Back to Basics Again, A Scientific Definition of Systems Engineering." *Proceedings of the Seventh Annual International Symposium of the International Council on Systems Engineering*. INCOSE-97, Los Angeles, CA, 229-236.

Mar B. W., Morais B. G., "Systems Engineering Heuristics – Is there a need to innovate, integrate and invigorate?" *Proceedings of the Tenth Annual International Symposium of the International Council on Systems Engineering.* INCOSE-2001, Melbourne, Au.

Palmer K. D., “Vajra Logic and Mathematical Meta-models for Meta-systems Engineering.” *Proceedings of the Eleventh Annual International Symposium of the International Council on Systems Engineering*. INCOSE-2002, Las Vegas, Nevada.

Palmer K. D., “Meta-systems Engineering.” *Proceedings of the Tenthth Annual International Symposium of the International Council on Systems Engineering*. INCOSE-2002, Las Vegas, Nevada.

Simpson J. J., “Innovation and Technology Management.” *Proceedings of the Eleventh Annual International Symposium of the International Council on Systems Engineering.* INCOSE-2002, Las Vegas, Nevada.

Simpson J. J., Simpson M. J., “Systems and Objects.” *Proceedings of the Twelfth Annual International Symposium of the International Council on Systems Engineering.* INCOSE-2003, Washington, DC.

Warfield, John N., *Understanding Complexity: Thought and Behavior*. Ajar Publishing Company, Palm Harbor, Florida, 2002.

Biography

Joseph J. Simpson’s interests are centered in the area of complex systems including system description, design, control and management. Joseph has professional experience in several domain areas including environmental restoration, commercial aerospace and information systems. At Westinghouse Hanford Company, Joseph was responsible for the conceptual and preliminary design of a requirements management and assured compliance system. While working in the internet domain, Joseph developed and deployed test-bed software essential to a major web-based commercial product. In the aerospace domain, Joseph has participated in a number of system development activities including; satellite-based IP network design and deployment, real-time synchronous computing network test and evaluation, as well as future combat systems communications network design.

Joseph Simpson has a BSCE and MSCE from the University of Washington, is a member of INCOSE and IEEE, and has obtained various information and computer system certifications.

Currently Joseph is enrolled in a system engineering management masters program at the University of Missouri at Rolla.