

System Integration Frameworks

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

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

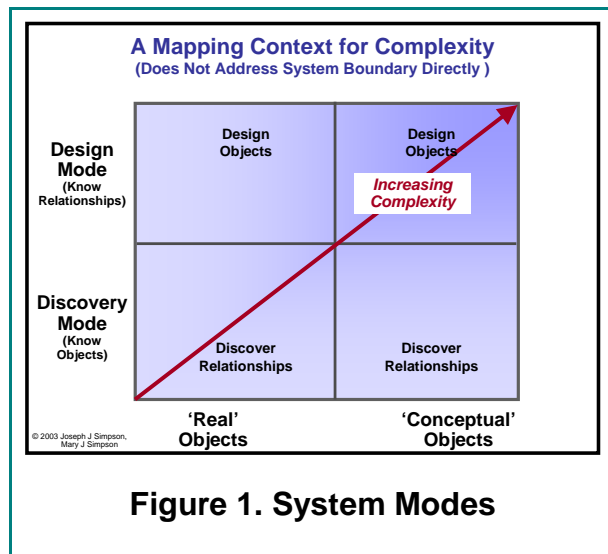
Abstract. An organizations ability to effectively create, deploy and use system integration frameworks is a discriminating technical capability which will continue to grow in importance for the design, development and deployment of future engineered systems. Engineering activities utilized in large-scale systems integration processes are strongly associated with organized problem solving and complexity reduction. This paper outlines the relationships between systems, meta-systems, and integration frameworks as well as complexity measures, reduction and management. The primary thesis of this paper relates to the design and use of integration frameworks as a fundamental means of complexity reduction. A system “abstraction frame” concept is presented and related to classical systems engineering mathematics and practices. A proposal for a systems engineering language completes the content of this paper. 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.

INTRODUCTION

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.

System modes. In practice, the concept of a system is central to two different modes of human behavior: system discovery mode and system design mode (see Figure 1). In the system discovery mode (utilizing the construction rule for systems), the objects are known and the main activity is the process of determining the controlling relationships among the objects. In the design mode, the controlling relationships are known and the main activity is the process of designing the objects that create these relationships. In either operational mode, a system is defined by discovering or specifying the system's boundary and function. System creation 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 that are essential to the development of modern large scale systems. While system discovery activities are controlled by the general scientific problem solving process. (Simpson and Simpson, 2003)



Each system exists in an environmental 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 engineered 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 and/or coping mechanism used by an organization that produces large complex systems. The CMMI processes are designed to constrain the variability in the product systems that are produced by constraining the variation in the system production process. An increasing emphasis is currently being placed on system abstract architecture formulations that include software, functional and physical system views. Abstract architectural constructs will change at a much lower rate than the physical representations 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 Open Systems Interconnection (OSI) model describes a standard architectural framework for the establishment of networked communication systems. While the all of these architectural frameworks qualify as a meta-system when applied in the production of real physical systems, each has its own area of application and interconnections between and among these frameworks are not clear.

SYSTEM COMPLEXITY MEASURES

Complexity. The concepts closely associated with system complexity populate the discussion and literature of system engineering and system theory. Two general views of system complexity can be identified; the system view (Type 1) that allocates the complexity characteristic to the system of interest as an intrinsic property of that system and the second relationship view (Type 2) that assigns the complexity characteristic to the relationship that exists between and among systems.

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 and management 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 as well as the two types of complexity, the following eight primary system complexity metrics were developed for Type 1 complexity to be applied to construction-rule based systems concepts:

- 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 of, 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 behavioral 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 is created to solve. The number of individuals and/or variables, the definition of the problem space, and the solution space itself will impact its overall complexity. This type of complexity is most often associated with systems that are defined using the constraint on variation of the problem or solution space. Systems in the early conceptual stages often use the parametric constraints to begin the definition and description of the solution system. These constraints and parametric boundaries are part of the system context and are transformed into specific system constraints using the controlling systems concepts. This problem space topology provides a graphic outline of this type of complexity mapping (see Figures 2 and 3). It is clear the “problem space” and “solution space” metrics are strongly associated with the meta-system and weakly associated with the system solution.

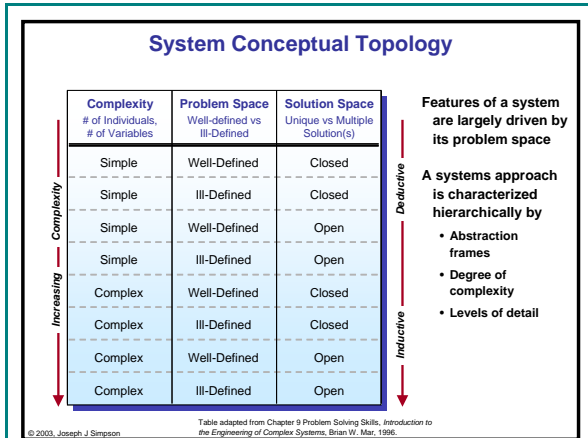


Figure 2. System Conceptual Topology.

Complexity measures for Type 2 system complexity are associated with the relative complexity in the relationships between systems, usually the system of interest and the system observer. This observation emphasizes the need to establish a clear context and environment for the system or systems of interest as well as the relationship between the system of interest and the system observer (see Figure 4) (Casti, Warfield). 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)

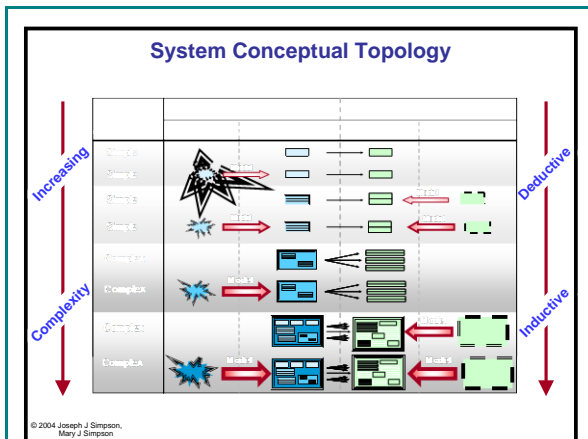


Figure 3. System Conceptual Topology.

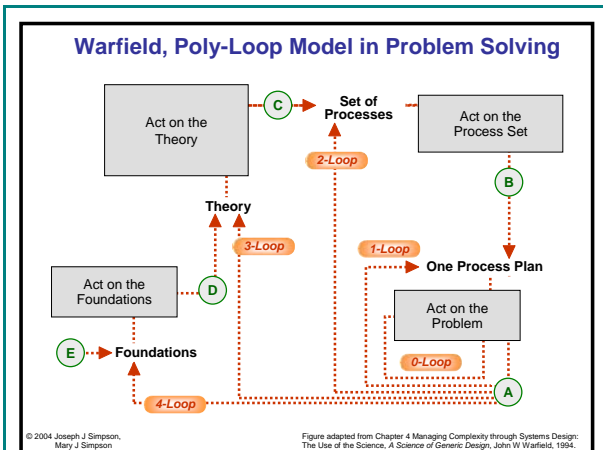


Figure 4. Relationship between Systems.

SYSTEM ABSTRACTION FRAMES

System Frameworks. 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 and provide the context for Type 2 system complexity evaluation.

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 5 and Figure 6.

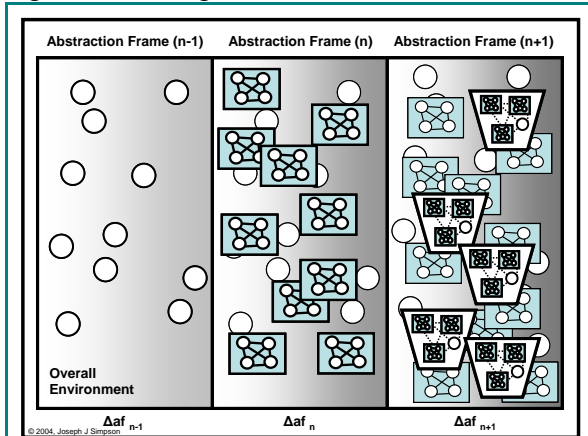


Figure 5. Abstraction Frame Sequencing.

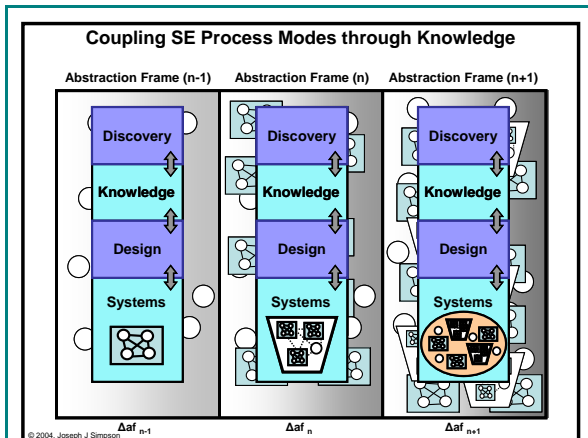


Figure 6. Frame Context Relationships

These frames will now be applied to system design using a modified Function, Requirement, Answer and Test (FRAT) process, a simplified system engineering process (Mar and Morias 2002). The modified FRAT process is named Context, Concept, FRAT (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. These two views provide a natural place to define a system using the constraint on variation definition for a system. 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 that are 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), functional view (defines what the system does), 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 7.

As CCFRAT methods 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 is reduced and the design space is further constrained. At each abstraction level the system is modeled 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. This logic also is used 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, modeled and evaluated in the context and concept views of the system. See Figure 8.

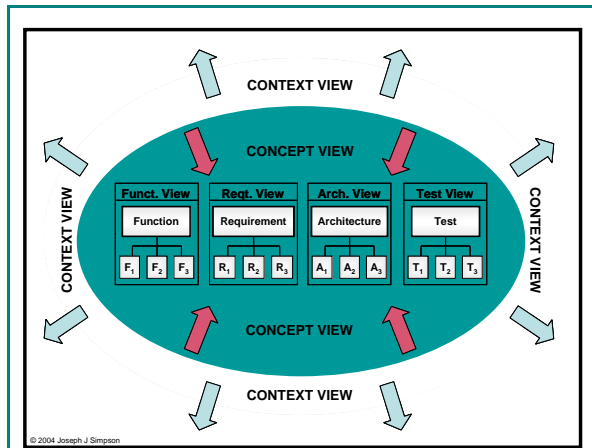


Figure 7. CCFRAT.

The system concept view details the relationships between the system functions and architectural design as well as specifying the proper level of abstraction for system modeling. The inward focus of the concept view is combined with the outward focus of the context view to emphasize the system boundary and provide a common abstraction layer for the complete system under evaluation. At high levels of abstraction the system may be modeled with only the context view or a combination of the context view and the concept view.

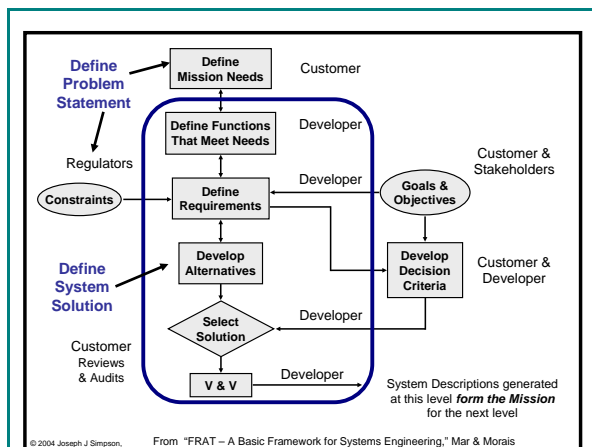


Figure 8. The FRAT Design Engine.

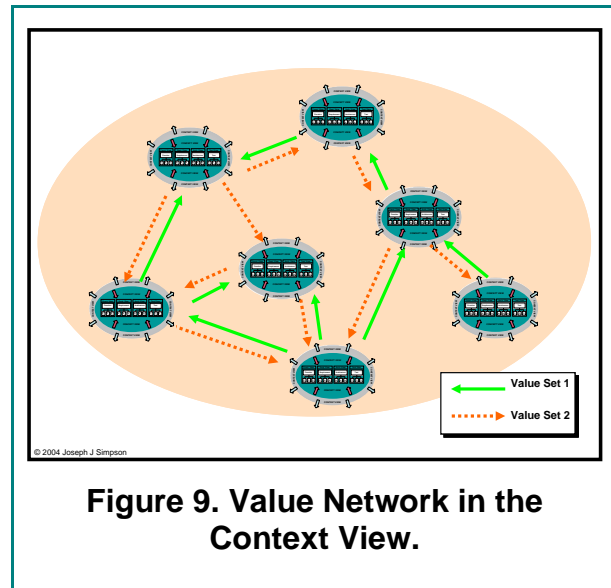


Figure 9. Value Network in the Context View.

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 9 presents an example graphic of a system context network.

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 must 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 to create layered system frameworks that can be evaluated in a parametric manner. If the system is modeled as a hierarchy, then the evaluation of alternative component connections is facilitated using these two holistic system views.

Figure 10 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.

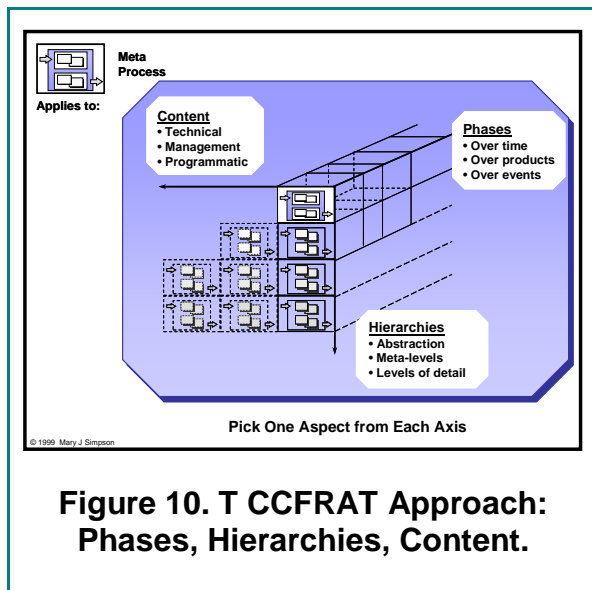


Figure 10. T CCFRAT Approach: Phases, Hierarchies, Content.

These system views can then be evaluated and analyzed using the system meta-levels shown in Figure 11. This robust set of system meta-levels provides the system designer or system engineer 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.

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	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
① Being meta-level 0 ENTITY	Concrete Instances 0 of learning in world	Orientation toward Things	Thing	Single Physical Instance

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

© 2004 Joseph J Simpson

Figure 11. Meta-Levels.

References

- 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, 2000.
- Casti, John L, *Alternate Realities: Mathematical Models of Nature and Man*. John Wiley & Sons, New York, NY, 1989.
- 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., "The Necessity of Logical Continuity" *Proceedings of the Fifth Annual International Symposium of the International Council on Systems Engineering (INCOSE-1995)*. St Louis, MO, 1995.
- 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., 2000, "C4ISR Architectures: I. Developing a Process for C4ISR Architecture Design." *Systems Engineering: The Journal of the International Council on Systems Engineering*, 3(4), 249-288.
- 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., and Morais, B.G., "FRAT – A Basic Framework for Systems Engineering." *Proceedings of the Twelfth Annual International Symposium of the International Council on Systems Engineering*, Las Vegas NV, 2002.
- Palmer K. D., "Meta-systems Engineering." *Proceedings of the Tenth Annual International Symposium of the International Council on Systems Engineering*. Minneapolis, MN, 2000.
- Palmer K. D., "Vajra Logic and Mathematical Meta-models for Meta-systems Engineering." *Proceedings of the Twelfth Annual International Symposium of the International Council on Systems Engineering*. Las Vegas, Nevada, 2002.
- Shell, Tony, 2003, "The Synthesis of Optimal Systems Design Solutions." *Systems Engineering: The Journal of the International Council on Systems Engineering*, 6(2), 92-105.
- Simpson, J.J, and Simpson, M.J., "U.S. DoD Legacy SE & Implications for Future SE Implementation," *Proceedings of the Eleventh Annual International Symposium of the International Council on Systems Engineering (INCOSE 2001)*, Melbourne, Australia, 2001..
- Simpson J. J., "Innovation and Technology Management." *Proceedings of the Twelfth Annual International Symposium of the International Council on Systems Engineering (INCOSE-2002)*. Las Vegas, Nevada, 2002.

Simpson J. J., Simpson M. J., "Systems and Objects." *Proceedings of the Thirteenth Annual International Symposium of the International Council on Systems Engineering (INCOSE 2003)*. Washington, DC, 2003.

Warfield, John N., *A Science of Generic Design*. Iowa State University Press, Ames IO, 1990.

Warfield, John N., *Understanding Complexity: Thought and Behavior*. Ajar Publishing Company, Palm Harbor F, 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. 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, an MSSE from the University of Missouri-Rolla, is a senior member of IEEE, and a member of INCOSE, and ACM. Currently Joseph is enrolled in a system engineering doctorate program at the University of Missouri-Rolla.

Mary J. Simpson's experience and interests focus on cognitive support and decision-making systems, integration and complexity reduction, threat and vulnerability analyses, systems sciences and systems engineering. Mary applies systems solutions to complex problems encountered in organizations, processes, and systems interactions ranging from strategic to tactical levels in fields including aerospace, security, information systems, energy, tank waste, public involvement, emergency planning, and biological and chemical systems. Mary is currently applying her leadership and integration capabilities to multiple opportunities in science and technology, homeland security, cross-threat risk assessment and systems sciences at Pacific Northwest National Laboratory which is operated by Battelle Memorial Institute.