Paper #106

Formal Systems Concepts

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
mjs-sbw@eskimo.com

Abstract

A common conceptual thread that runs through a large portion of systems, systems engineering, and software engineering literature is explored by this paper. This conceptual thread is based on the formal concept of a Moore type of sequential machine (Moore 1956) as well as the definition of a system and a meta-system. Practical approaches and applications generated by this common thread are outlined and discussed in terms of complexity reduction, design efficiency and communication enhancement. Hartmanis and Stearns utilized the abstract definition of a sequential state machine as a mathematical model of discrete, deterministic computing devices with finite memory. These machines have a finite set of inputs, outputs and internal configurations. Wymore expanded on this work while developing his model-based systems engineering approach. In each case, Moore and Wymore, the system concept has the same form. The system content; specific inputs, outputs types and transition functions, are different. In the software engineering arena, "abstract state machines" and "requirements state machines" have been developed as formal approaches closely related to Moore state machines. This paper presents an expanded systems and meta-systems framework that extends the formal system concepts. System abstraction stacks are introduced in this paper.

Introduction

Similar to formal logic, formal system concepts are concerned with the form of the system and not the system content. The concepts associated with systems and meta-systems have a long and varied history in scientific and engineering literature. A.D. Hall defined a meta-systems methodology as an applied science to be used to solve complex problems (Hall 1989). Warfield developed the science of generic design as a primary method of system complexity management and control. These comprehensive approaches to formal system definition provide a well developed context to address more focused formal systems definitions like sequential state machines, discrete systems, abstract state machines, system abstraction frames and system abstraction stacks. The clear definition and distinction between a system and a meta-system is a powerful tool in the communication of complex systems concepts. A clear definition of system abstraction levels works closely with the system and meta-system distinctions to provide a comprehensive context for system design and development analysis and communication. In a similar manner, the distinction between the problem to be solved and the solution system that solves the problem is a fundamental concept in system engineering and design (Goode 1957).

Formal systems concepts are developed to facilitate the organization and communication of these basic systems engineering activities. These basic practices are critical to the successful application of structured systems design and engineering practice (Mar 1997).

Systems and Meta-Systems

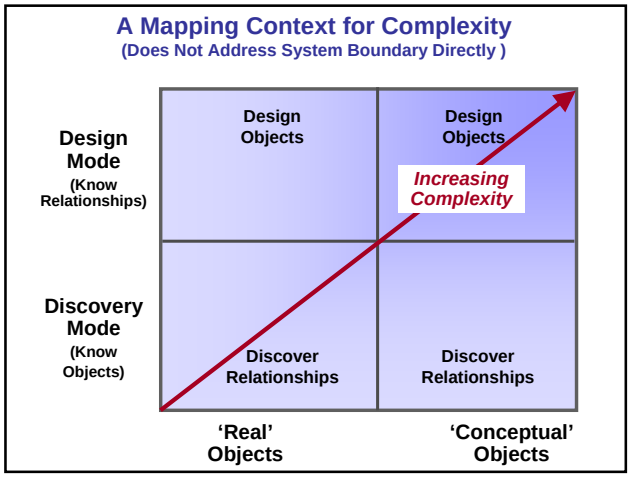

Figure 1. System Modes

Two definitions are presented for a system, one - a "functional rule" type and the other – a "construction rule" type. The functional rule definition of a system is "a constraint on variety," wherein constraint identifies and defines the system of interest. 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 system boundary, a primary component of any system, is implied in both of the system definitions. Systems are used in one of two modes (see Figure 1), system discovery and system design. Using the construction rule definition of a system in the discovery mode we know the objects in the system and we are in the process of determining the relationships. In the design mode we know the relationships and need to discover and/or design the objects that provide the needed relationship (Simpson and Simpson 2003).

The concept of a meta-system has also given rise to different definitions. 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). Another 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). 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. All of the definitions are valuable conceptual tools and will be used as necessary in this paper.

Meta-Systems Methodology

A.D. Hall, III outlined an "available common systems methodology, in terms of which we can conceive of, plan, design, construct, and use systems of any arbitrary type in the service of mankind." This systems methodology is an applied science that focuses on the total system-environment interaction. Hall presents a formal cycle for the scientific method and model building, see Figure 2. Theory and concepts are built upon natural observations and

measurement data. Induction is combined with the detailed data to form the theory or general principles. Then deduction is used with the theory to identify specific instances of the natural system. This approach to systems development provides a clear formal set of system concepts that can be used in the production and/or the discovery of systems.

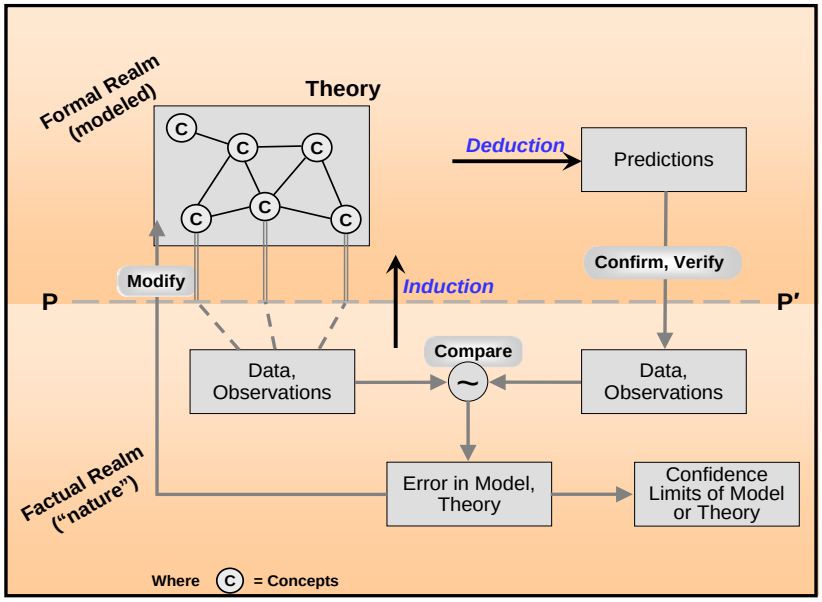

Figure 2. A.D. Hall – Cycle for Scientific Method & Model Building

Science of Generic Design

The science of generic design was developed by John N. Warfield as the primary method for system complexity management. Warfield developed a set of formal concepts, rules, and methods for use in the systems and systems engineering domain areas. The Poly-Loop model developed by Warfield sets the formal context for the science of generic design and its practice.

The Poly-Loop model starts by addressing and acting upon the problem at hand. If the problem can not be addressed effectively in the current process context then the model directs effort toward the development and refinement of the process set. If the current system theory does not support the required process actions then the theory must be addressed. If the scientific foundations do not support the theory then the scientific foundations must be addressed. Figure 3 shows the Poly-Loop model. Each of these four distinct areas can be viewed in terms of system and meta-system pairs. The scientific foundations are the meta-system for the theory, the theory is the meta-system for the process set and the process set is the meta-system for the current problem. Conceptual framing and modelling provides a structured tool set that helps focus the organization that is addressing the problem solution. Successive

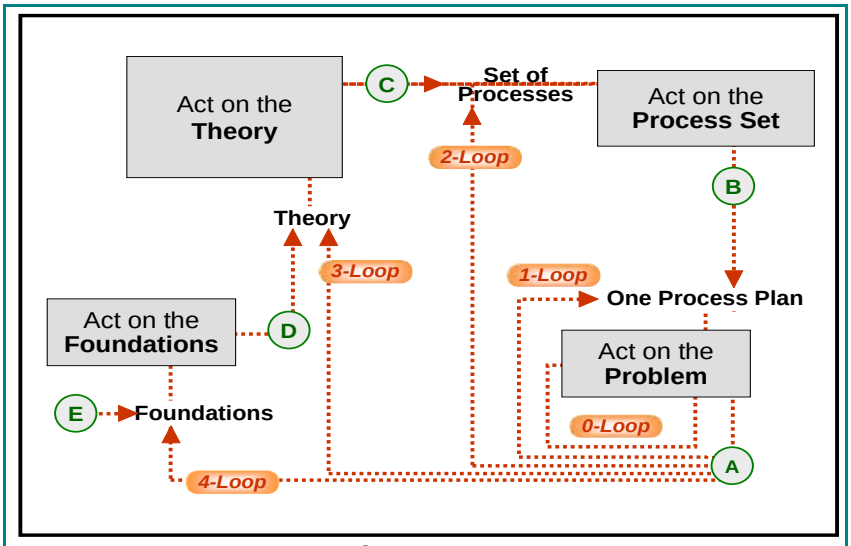

Figure 3. Warfield – Poly-Loop Model

reframing and remodelling, using the Poly-Loop Model, helps identify, organize and control the complexity associated with large-scale system design (Warfield 1990).

Sequential State Machines

The concept of a sequential machine presents a formal, mathematical model for a deterministic, discrete computing device with finite memory. This formal sequential machine concept is very closely related to the concept of a system with both having inputs, outputs and internal operations that transform inputs to outputs. The Moore sequential machine is defined as a quintuple: M = (S, I, O, *b, l*); where

- S is a finite nonempty set of states
- I is a finite nonempty set of inputs
- O is a finite nonempty set of outputs
- *b:* (S x I) ➔ S is called the transition function
- *l:* S ➔ O is called the output function.

A Mealy machine is defined in a similar form (Hartmanis and Stearns 1966). This formal definition appears to be the basis of the formal system definition used by Wymore in his model-based systems engineering approach. When the machine inputs and outputs are an alphabet, then the machine can be viewed as a single system. However, if the system form stays the same and the input and output data types are expanded to accept vectors of systems or state machines, then there is a hierarchy of systems that can be viewed as a meta-system, system pair. While the form of the system stays the same the system behaviour and content can vary over a wide range.

Model-Based Systems Engineering

Wymore proposed a set of formal systems concepts, methods and a system design language in his model-based systems engineering work. The formal concepts found in this approach are a combination of the high level formal concepts outlined by Hall and Warfield as well as the formal concept of a sequential machine or sequential state machine. Wymore defines a system as a quintuple:

Z = (SZ, IZ, OZ, NZ, RZ) where:

- Z = system name
- SZ = set of discrete system states
- IZ = set of discrete system inputs
- OZ = set of discrete system outputs
- NZ = next state function of the discrete system
- RZ = readout function of the discrete system

There is the same form as the Moore sequential machine. However, the discrete formal system defined by Wymore has completely different content. Wymore adds a definition of a system time scale as well as definitions for open and closed systems. Given this formal system definition, Wymore goes on to define an approach to define a connectable vector of systems (VSCR) and the system connectivity (CSCR) for the system vector. A system coupling recipe

(SCR) is composed of a VSCR and a CSCR. There are three primary categories of coupling recipe categories; pure feedback, feed-forward (cascade, conjunctive, singular), mixed (feed-forward and feedback). This system model allows a system to have other systems as inputs and outputs. Therefore, the system that accepts other systems as inputs can be considered a meta-system from the input systems context or point of view. There could be many layers of meta-system, system pairs.

Systems engineering activities are focused on developing the system design problem statement. The system design problem is completely defined by six system requirements. These requirements are:

- Input/output requirement
- Technology requirement
- Performance requirement
- Cost requirement
- Trade-off requirement
- System test requirement

Each of these requirements can be clearly and formally specified using the system theoretic constructs that are part of the model-based systems engineering approach. The input/output requirement for a system is composed of the following specifications:

- The set of inputs to be accepted by the system to be designed
- The set of input trajectories (input variation over time) which the system might experience
- The set of outputs producible by the system to be designed
- The set of output trajectories (output variation over time) that are possible for the system to be designed
- The eligibility function that matches outputs and inputs or input trajectories and eligible output trajectories. The eligibility function specifies (by limitation) the behaviour of the system to be designed
- The operational life of the system to be designed, which sets the time scale for all input and output trajectories.

The input/output requirement determines the set of functional system designs. The input/output requirement is satisfied when the system design specifies the same inputs and outputs as the input/output requirement. The technology requirement is a set of specifications of the components that are available to build the system under design. The technology requirement is used to clearly specify components that must be used or must not be used in the system construction. All available components for a system are not enumerated unless the technology choice is very restricted. The performance requirement for a system to be designed is an algorithm that is used to consistently compare a given functional system design to the input/output requirement. An example algorithm is a list of key performance parameters and the relationship between each key parameter. Performance figures of merit are used to define the system performance requirement. The cost requirement for a system under design is an algorithm that consistently compares any build-able system design against the cost figures of merit. Cost figures of merit are used to define the cost performance requirement. The trade-off requirement for a system under design is an algorithm that consistently compares any implement-able system against the trade-off function between performance and cost requirements. Trade-off figures of merit are used to define the trade-off requirement. The trade-

off algorithm can only be used on systems that have agreement on the relationship between cost and performance or systems that have equivalent performance and cost relationships.

The system test requirement specifies for each possible implement-able test item:

- Observance: How the real system shall be observed, sampled, measured, stimulated, and simulated, in order to estimate a value for each performance, cost, and trade-off figure of merit. The product of this activity is called test data.
- Conformance: How it shall be proved, in terms of test data; that the real system is in conformance with the implement-able system design from which it was built.
- Compliance: How it shall be proved, in terms of test data; that the real system is in compliance with the system requirements.
- Acceptance: How it shall be proved, in terms of test data; that the real system is acceptable to the customer.

This set of systems language concepts can be used to specify systems, to determine if systems are equivalent, to evaluate system modes and to model system behaviour. These language components are part of a system design language developed by Wymore for use by systems engineers in the practice of systems engineering (Wymore 1993).

Systems and Meta-Systems Frameworks

Frameworks are finding increasing use in the organization of complex bodies of knowledge and processes. The Department of Defence Architectural Framework, The Zachman Framework and the Open Systems Interconnection Framework and software frameworks are examples of applying frameworks in the solution of complex problems. System frameworks can provide the foundation for the development and application of domain specific languages and pattern languages (Levis 2000). These languages are excellent tools for complexity reduction because they provide a clear predefined set of concepts and operations for a specific system design domain. Using a small set of well-defined rules enables the effective description of very complex systems (Mar and Morais 2002).

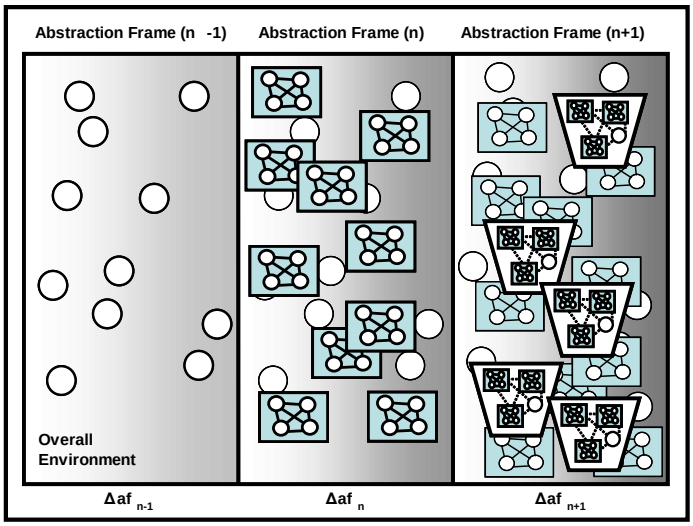

Figure 4. Abstraction Frame Sequencing.

System abstraction frames and system abstraction stacks have been introduced to address two fundamental aspects of system production: 1) activity or context sequencing and 2) system or concept abstraction levels. Figure 4 shows the abstraction frame sequencing. System abstraction levels can be related directly to a hierarchy of meta-systems with each higher-level system or meta-system setting the rules and context for the next lower level system or meta-system (Simpson and Simpson 2003).

The Context, Concept, Function, Requirement, Architecture and Test (CCFRAT) approach was developed as a conceptual framework to support the clear definition of a system boundary and function (Simpson and Simpson 2005). Figure 5

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 highly abstract conceptual structures. Each form of evaluation and system representation adds value to the systems engineers and designers tool set. However, these hierarchies need to be closely coordinated with the design context and system development phases. As a system advances through each abstraction frame, the content and values associated with the content and phases change. These meta-system levels are an example of abstract conceptual hierarchies that apply to the systems engineering domain.

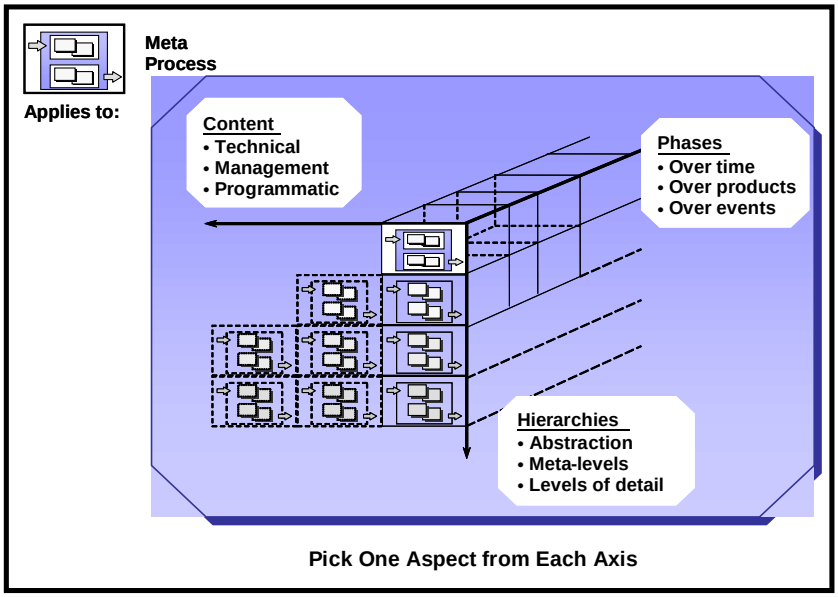

Figure 5. CCFRAT Approach: Phases, Hierarchies, Content.

System abstraction stacks are shown in Figure 6. Abstraction stacks provide ordering to the system and meta-system context that allow domain experts the ability to write specific rules to control the transition between each level in the abstraction stack. A robust set of system abstraction stacks provide the system designer or engineer a clear set of conceptual transforms that can be used to reduce system complexity.

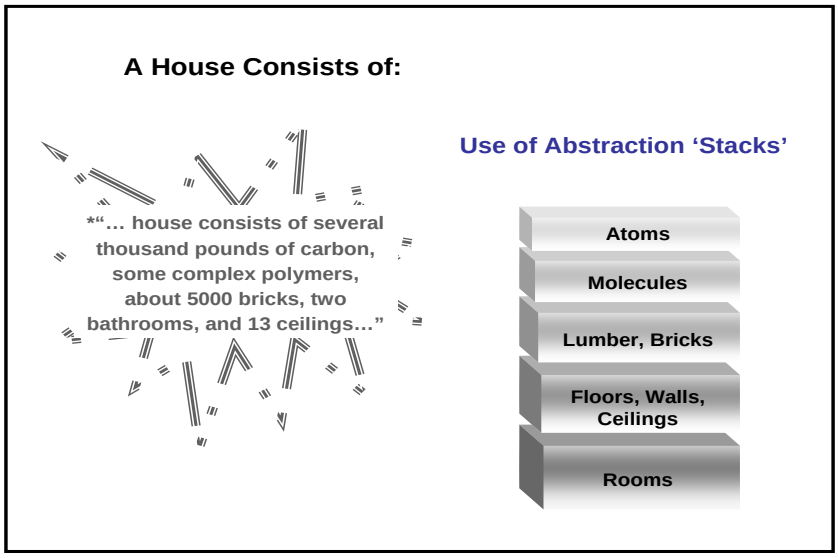

Figure 6. Abstraction Stacks

Domain specific stacks are grouped in a specific system context. For example, the OSI (Open System Interconnection) stack is a well known domain specific abstraction stack. Using a combination of the abstraction

frames and well-defined system abstraction stacks, the context for system coupling recipes development is well bounded (see Figure 7). Each of these bounded system contexts can then be used to control the application of system value sets and other relationships. The systems design language, proposed by Wymore, is mathematically based and developed to enhance communication among systems engineers. However, many systems engineers find the application of this system design language to be time and resource intensive. An executable systems engineering language that sets the context for and includes aspects of Wymore's system design language will greatly reduce complexity in the practice of systems engineering, as well as facilitate the solution of even more complex problems. By bounding the complete system problem and setting the proper context for each system development domain, the use of an executable system engineering language will enable large distributed teams in identifying the proper context, value sets and concepts for each specific system development activity. Support for formal system design activities should be embedded in the executable systems engineering language. As shown in Figure 7, the proposed executable systems engineering language has a rule based component for operation between system development meta-levels as well as standard procedural components for application at each level of the system development abstraction stack.

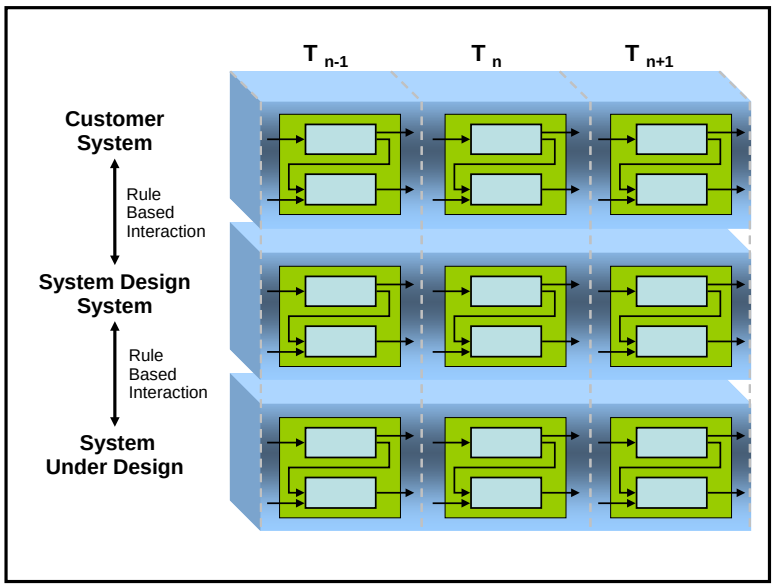

Figure 7. Systems Engineering Conceptual Context

Summary and Conclusions

Sequential state machines provide a powerful conceptual pattern for system description and design. Wymore's system model has a strong similarity to classical state machines. However, the use of other systems as vector inputs into the Wymore system model creates the need to clearly discuss and quantify systems, meta-systems and system abstraction stacks. In the software domain, abstract state machines are used in a similar fashion to address differing levels of system abstraction. A combination of meta-systems, system abstraction frames and system abstraction stacks provides the necessary context for the development of an executable systems engineering and design language. This executable systems engineering language would allow large, distributed teams the ability to clearly and quickly normalize their concepts and approaches using the rule based components of the language. Detailed systems design activities would then be bounded by the given system abstraction level and abstraction frame and can be addressed by the procedural components of the language. More research is needed to develop this executable systems engineering language and tie it to the current state of knowledge representation and artificial intelligence. Future research should focus on the development of the abstract data types, including abstraction frames and abstraction stacks, as well as the required rule based components of the language.

References

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

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

Hartmanis, J. and Stearns, R.E., *Algebraic Structure Theory of Sequential Machines*. Prentice-Hall, Inc, Englewood Cliffs, NJ, 1966.

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.

Moore, E. F., *Gedanken-experiments on Sequential Machines*, pp 129 – 153, Automata Studies, Annals of Mathematical Studies, no. 34, Princeton University Press, Princeton, N. J., 1956.

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.

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., 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.

Simpson, J.J. and Simpson, M.J., "System Integration Frameworks." *Proceedings of the 15th Annual International Symposium of the International Council on Systems Engineering* (Rochester, NY, July, 2005).

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

Wymore, A. Wayne, *Model-Based Systems Engineering*. CRC Press, Inc., Boca Raton, FL 1993.

Biography

Joseph J. Simpson's interests are focused 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, biological and chemical systems, and defense systems. Mary is currently applying her leadership and integration capabilities to multiple opportunities in science and technology, homeland security, quality risk assessments, defense systems and systems sciences.