Systems and Objects

Joseph J. Simpson
The Boeing Company

Mary J. Simpson
System Concepts

Abstract. This paper discusses basic system concepts including structure, content, semantics, scale and context. These basic concepts are reviewed and discussed in the light of two different human activity modes: system discovery mode and system design mode. A system meta-model is outlined and discussed in terms of system behavior and context. A generic systems engineering model and process is reviewed and used as a basis to construct a system object model that facilitates the representation, description and detailed communication of system relationships and characteristics. The model is presented at the conclusion of this paper.

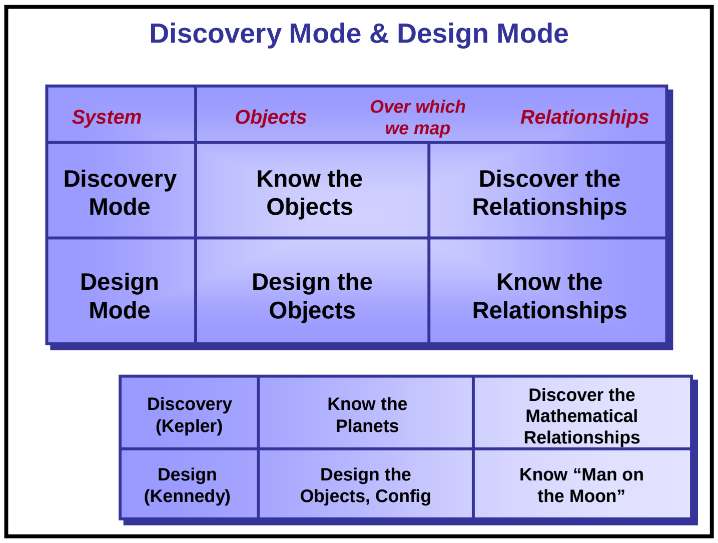
The System Meta-Model Definition

The primary focus of Systems Engineering is the creation of systems. However, there is no common formal definition of a system. Several "system definitions" are presented next to provide a basis for the following discussion.

The system science community provides the following common-sense definition for a system; "A system is a set of things (objects or components) and a relation among the things." (Klir, 1991) While the general systems community offers the following definition for a system. "A system is a set of objects together with relationships between the objects and between their attributes." (Hall and Fagen, 1956) Further, the semiotics community (Noth, 1995) presents the following commentary on the idea of a system. "A system most generally implies the idea of elements forming an ordered whole, with the relations among these elements forming the structure of the system. The systemic character appears only as the elements function within the system."

For further discussion, a system is defined as a non-empty set of objects and a non-empty set of relationships mapped over these objects and their attributes. This system definition establishes a basis for describing the system object model, and presents two basic system components: objects and relationships. Typically, the system concept is applied in one of two basic modes that relate to these fundamental components - discovery and design mode.

Discovery mode and design mode are depicted in Figure 1. In discovery mode, the things or objects are known and the primary activity is the determination of the relationships between the objects. For example, Kepler analyzed the data collected from the observation of the planets (things) to determine the relationship (planetary laws) between the things in our solar system. In the design mode, the relationship (function) is known and the objects and the configuration required to provide that relationship are the subject of study. An example of this mode can be seen in John F Kennedy's decision to put a man on the moon. The function – put a man on the moon – was known. The objects and object configuration needed to accomplish this function were not known.


Discovery Mode & Design Mode

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. Objects and Relationships over Discovery and Design Modes

System and Context Interaction

Current human society is the result of the application of system and scientific thought and processes to the understanding, modification and adaptation of the physical environment. Large-scale systems, such as interstate highways, national armies, international satellite communications systems, are pervasive in the current environmental context with significant impact on and benefit to humans. The wide variety of systems and system components provide an opportunity for system scale change by linking or interlinking existing systems and system components. This pervasive opportunity for system connectivity and system-scale change drives system complexity and potential for failure. (Warfield, 1994)

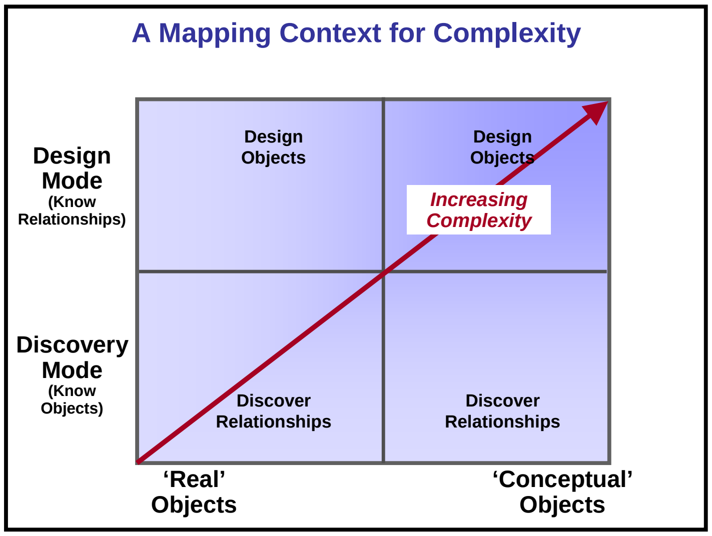


Figure 2. Mapping Context for Complexity

The ability to create and deploy systems has two concurrent impacts, 1) development of new conceptual and scientific tools and, 2) increasing environmental variety and complexity. These concurrent impacts must be separated and evaluated in terms of a given environment or “abstraction framework.”

Further, this abstraction framework is viewed as a “snapshot” in time with the upper bound of human understanding and design activity labeled as a complexity limit (See Figure 2.). As human beings evolve and participate in each of these sequential “abstraction frames”, 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. In order to apply these complex system concepts to systems design and engineering, system complexity metrics must be developed and integrated into the systems analysis process. 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 definition of complexity provides a clear connection between the expressive power of a given language and the complexity limit associated with the "abstraction framework" that employs that language.

The Systems Approach: Phases, Hierarchies, and Content

The discipline of systems engineering (SE) facilitates the solution of complex problems using and focusing a wide range of activities, tools and people in order to provide a system - or a 'system of systems' - which deliver fulfilled needs to customers. The system design process is domain specific with similar phases (described over time, products and events) and decision processes appearing as a common thread in all of these design activities. The design process is implemented using system-specific and customer-specific procedures, methods, and tools.

To establish a common context for this paper, a highly simplified view of a SE process is presented in Figures 3a and 3b.

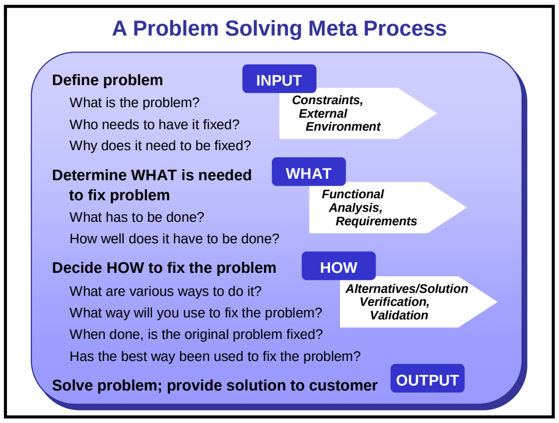

A Problem Solving Meta Process
Define problem
INPUT
What is the problem?
Who needs to have it fixed?
Why does it need to be fixed?
Constraints, External Environment
Determine WHAT is needed to fix problem
WHAT
What has to be done?
How well does it have to be done?
Functional Analysis, Requirements
Decide HOW to fix the problem
HOW
What are various ways to do it?
What way will you use to fix the problem?
When done, is the original problem fixed?
Has the best way been used to fix the problem?
Alternatives/Solution Verification, Validation
Solve problem; provide solution to customer
OUTPUT

Figure 3a. Problem Solving Meta Process

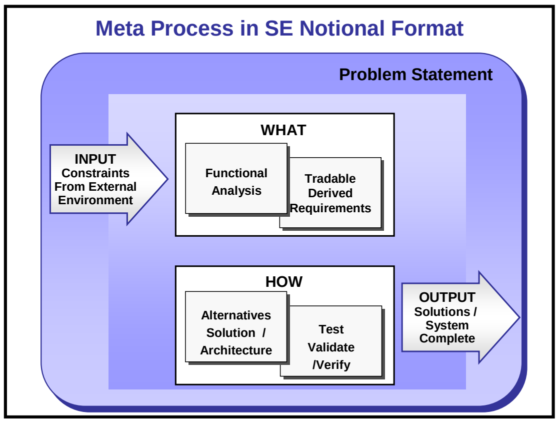

Meta Process in SE Notional Format
Problem Statement
INPUT Constraints From External Environment
WHAT
Functional Analysis
Tradable Derived Requirements
HOW
Alternatives Solution / Architecture
Test Validate /Verify
OUTPUT Solutions / System Complete

Figure 3b. Meta Process in SE Notional Format

This SE process builds on the previous definition of a system by asserting that anything can be viewed as a system. This simplified view establishes a common basis for the discussion of an SE activity and its relationship to the system under design, the organization designing the system and the general environment in which they both exist. It is natural that a given knowledge domain would choose to adapt and/or tailor a more elaborate SE process, and would choose to name or word it differently than that chosen here.

This process establishes a system boundary, an input from outside that boundary, an iterative examination of what is needed for the system and how to deliver a solution to that need, and an output from the system to the outside environment. One assumption made herein is that most SE processes are applied in an iterative fashion over the time-phased duration of the system, in its knowledge domains, and with its own specific application of logic. This standard phased approach to system development provides some great advantages in the design, development and deployment of complex systems. However, there are certain types of systems that are too complex even for this type of development.

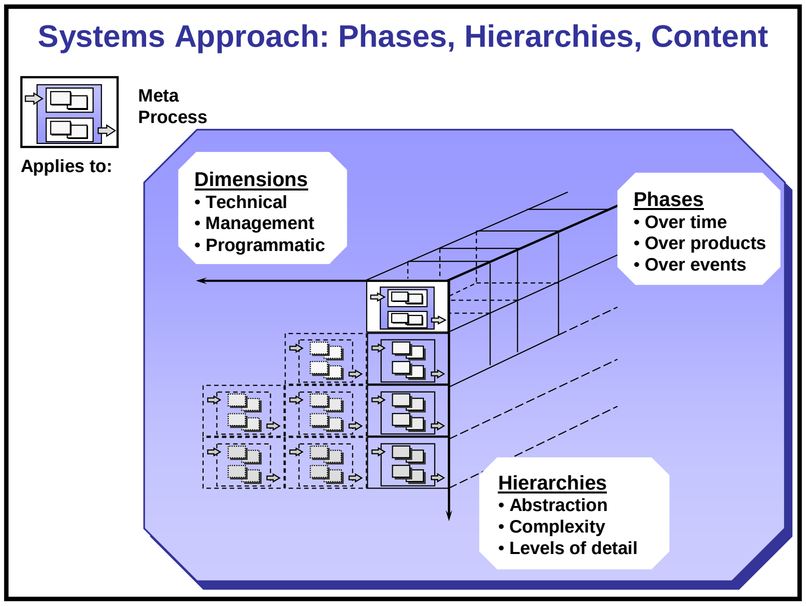


Figure 4. Systems Approach: Phases, Hierarchies, Content

Figure 4 is a notional depiction of a SE process being applied repeatedly through a full spectrum of knowledge domains, logical hierarchies, and over time. To focus on a system's environment - rather than on the way people are organized to deal with that environment - it is further assumed that the question of where program and project management ends and systems engineering begins is irrelevant. As a matter of interest, one general aspect or measure that can be used to separate SE from program management is the level of technical performance required and the level of uncertainty associated with delivering the needed technical performance. If a high-level of technical performance is required and a high degree of uncertainty is associated with providing that technical performance, then the activity is more likely to be in the SE domain. For the purposes of this paper, the critical design information needed for successful implementation of a system is essential to the success of both program and project management and systems engineering.

Environmental Context

Each system design and development task takes place in a unique environmental context with dynamic trends and forces shaping the system owners desires and the system supplier's ability to provide a viable product to the customer. In major system development activities with long life cycles, of 20 or more years, this dynamic nature of the environment may be one of the most important sources of technology solutions as well as a source of great technological uncertainty. Figures 5a and 5b list some of the general considerations that appear from this aspect of the system environment.

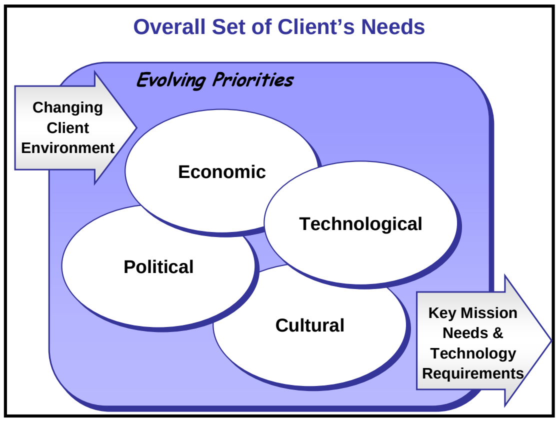
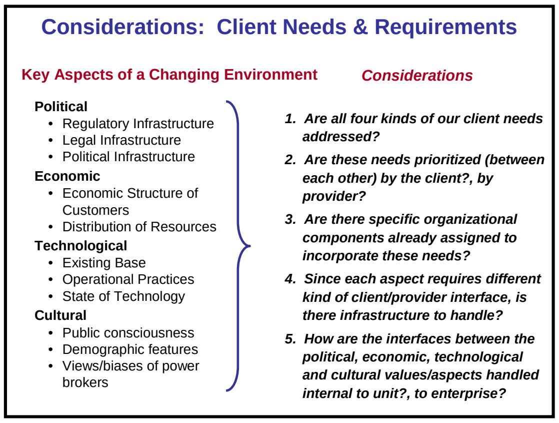

Overall Set of Client's Needs
Evolving Priorities
Changing Client Environment
Economic
Technological
Political
Cultural
Key Mission Needs & Technology Requirements

Figure 5a. Overall Set of Client's Needs



Considerations: Client Needs & Requirements

Key Aspects of a Changing Environment

Political
- Regulatory Infrastructure
- Legal Infrastructure
- Political Infrastructure

Economic
- Economic Structure of Customers
- Distribution of Resources

Technological
- Existing Base
- Operational Practices
- State of Technology

Cultural
- Public consciousness
- Demographic features
- Views/biases of power brokers

Considerations

1. ***Are all four kinds of our client needs addressed?***
2. ***Are these needs prioritized (between each other) by the client?, by provider?***
3. ***Are there specific organizational components already assigned to incorporate these needs?***
4. ***Since each aspect requires different kind of client/provider interface, is there infrastructure to handle?***
5. ***How are the interfaces between the political, economic, technological and cultural values/aspects handled internal to unit?, to enterprise?***

Figure 5b. Considerations: Client Needs & Requirements

One key aspect and characteristic of the SE activity is the description and documentation of system requirements, design decisions and architectural solutions associated with the system under design. Many of these classical SE techniques were developed during the evolutions of the US Military Systems Engineering Management approach. However, the current trends in the computing industry provide computational power and resources at an ever-increasing rate. This availability of computing power, if applied correctly, can greatly reduce the efforts associated with the SE system life-cycle tasks. However, the availability of computing power alone is not a guarantee of successful systems deployment. When George Friedman was Northrop's Chief Technical Officer he studied the failures associated with systems that failed in their final test phases. The most prevalent cause of failure was, "The models the engineers used to predict performance were incomplete; many of the interactions were omitted, despite the presence of massive computer resources." (Warfield, 2002)

Therefore, any SE management activity should include a capability to track the "completeness" of any system performance metric, parameter and/or behavior. This idea of "completeness" requires that a system interaction model be developed at all relevant scales of system behavior.

System Engineering Meta-Model Definition

Managing the technical design and uncertainty associated with SE activities requires the application of a wide range of engineering expertise and the management of a vast quantity of technical data and information. A SE meta-model is used to organize the production, management and communication of this system technical data and information. Based on the general system definition using the design mode, the FRAT approach specifies that anything can be viewed as a system and that a minimum of three basic systems must be defined: environmental system, product system and process/organizational system (See Figure 6a).

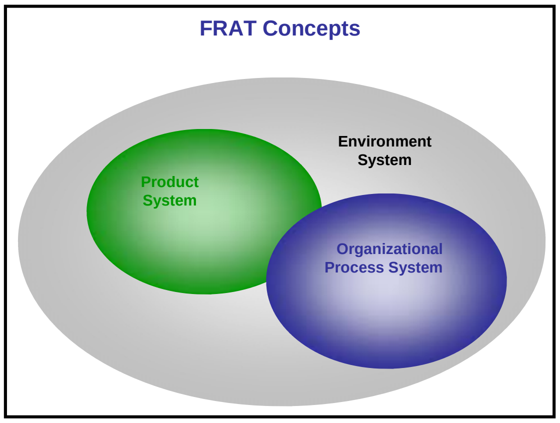


Figure 6a. FRAT Concepts

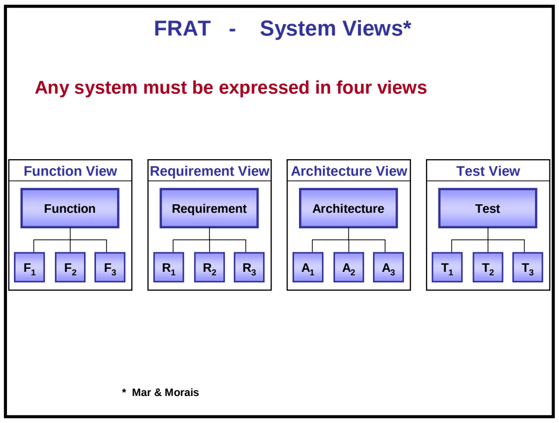


Figure 6b. FRAT - System Views

Further, each system must be described by four basic views: functions, requirements, architectures and tests (See Figure 6b). (Mar, 1994, 2000)

FRAT views are applied recursively using the FRAT Design Engine (See Figure 7a.) over the phases of the system design life cycle (See Figure 7b.). This means that FRAT is defined at the 'mission-level' of a system, then is redefined with increasing amount of detail at each new phase of system design, including the conceptual, preliminary and initial design levels - less commonly extending into the engineering prototype phase and below, depending on the particular application or domain space.

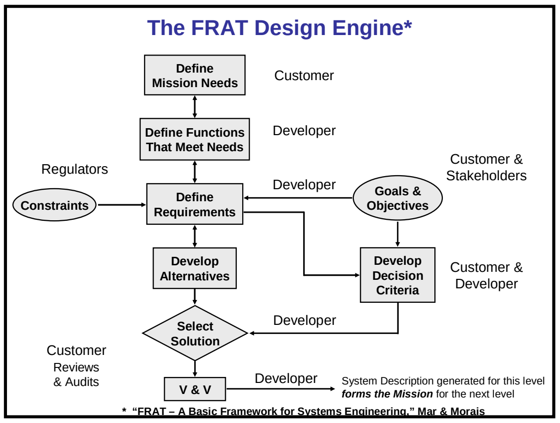

The FRAT Design Engine*
Define Mission Needs
Customer
Define Functions That Meet Needs
Developer
Regulators
Customer & Stakeholders
Constraints
Define Requirements
Developer
Goals & Objectives
Develop Alternatives
Develop Decision Criteria
Customer & Developer
Select Solution
Developer
Customer Reviews & Audits
V & V
Developer
System Description generated for this level *forms the Mission* for the next level
* "FRAT – A Basic Framework for Systems Engineering," Mar & Morais

Figure 7a. FRAT Design Engine

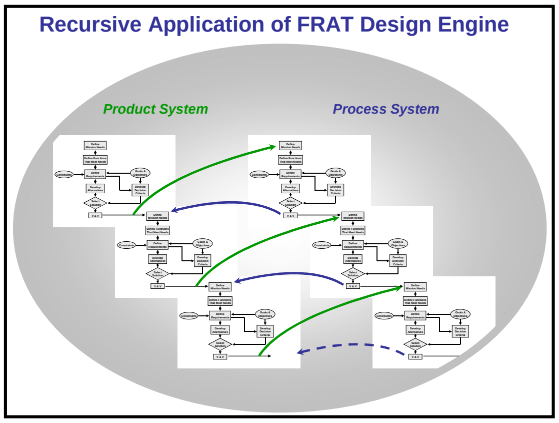


Figure 7b. Recursive Application

During the application of the FRAT process to a problem domain, it is important to maintain equal "scales" or level of detail for each of the four FRAT components. Further, the amount of information at each level should be limited in order to facilitate the understanding of the complete system at that level of detail or scale.

The four FRAT views provide a complete system description at any level of detail or abstraction. The functional view describes what the system must do. The requirements view describes how well the functions must be done. The architecture view details the mechanism that will perform the assigned function. The test view outlines the methods used to ensure that the selected architecture performs the assigned functions to the level stated by the requirements. In the context of these system models, system or mission objectives are achieved by controlling the interfaces and interactions among the base systems and their subsystems.

System Object Model

A proposed common set of system objects for use in object-oriented system design tools and object-oriented system analysis is presented next. The primary purpose of this "system object model" is to provide the basic components of a computer executable "language" that can be used to track the system component actions and interactions in a "complete context" representation. This will provide a foundation for text-based system information processing as well as analytical computational representations of the current system development state.

The basic system object model is composed of the following abstract classes:

- General system class
- Environment system class
- Product system class
- Organization system class
- Function class
- Requirement class
- Architecture class
- Test class
- Complexity class

The general system class (GSC) is the top-level system class from which all other system classes are derived. The GSC has the following class attributes and methods:

- System identification number
- Complexity record
- Calculate complexity measures
- Compile system state record

The environmental system class (ESC) is derived from the GSC and contains the following class attributes and methods:

- Function vector (collection of environmental functions)
- Requirement vector (collection of environmental requirements)

- Architecture vector (collection of environmental architectures)
- Test vector (collection of environmental tests)
- Normalize scale
- Compile active system list

The product system class (PSC) is derived from the GSC and contains the following class attributes and methods:

- Function vector (collection of product functions)
- Requirement vector (collection of product requirements)
- Architecture vector (collection of product architectures)
- Test vector (collection of product tests)
- Normalize scale
- Compile development state record

The organization system class (OSC) is derived from the GSC and contains the following class attributes and methods

- Function vector (collection of organizational functions)
- Requirement vector (collection of organizational requirements)
- Architecture vector (collection of organizational architectures)
- Test vector (collection of organizational tests)
- Process vector (collection of organizational processes)
- Value vector (collection of organizational values)
- Decision vector (collection of organizational decisions)
- Normalize scale
- Compile decision state record
- Process decision

When developed and deployed in a high-level language like Java, these objects will be able to interact with multiple data formats including XML, ASCII, and others. A properly balanced program will support analytical system design as well as publishing text reports and summaries of the design process state.

References

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

Klir, George J., *Facets of System Science.* Plenum Press, New York, 1991

Koth, Winfried, *Handbook of Semiotics.* Indiana University Press, Bloomington IN, 1990

Mar, Brian, "Systems Engineering Basics." The Journal of NCOSE Vol. 1 Num. 1 1994

Mar, Brian W. and Morais, Bernard G., 'FRAT – A Basic Framework for System Engineering,' 12th Annual International Symposium of INCOSE, Las Vegas NV, 2002.

Simpson, Joseph J., "Innovation and Technology Management," 12th Annual Symposium of INCOSE, Las Vegas, Nevada, 2002

Simpson, Joseph J., Simpson Mary J., "U.S. DoD Systems Engineering and Implications for SE Implementation in Other Domains," 2nd European Systems Engineering Conference, Munich, Germany, 2000.

Simpson, Joseph J., Simpson Mary J., "U.S. DoD Legacy SE & Implications for Future SE

Implementation," 11th Annual Symposium of INCOSE, Melbourne, Australia, 2001.
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. 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.

Mary J. Simpson. Mary Simpson's experience and interests focus on engineering systems solutions to complex problems encountered in organizations, processes, and systems interactions. As the principal author of the DOE's Strategic Plan (1996) at Hanford Site, and the principal author of DOE's Mission Direction Document - which became a formal part of their performance-based contracting mechanism, Mary successfully applied strategic planning, domain analysis, and high-level modeling in the evaluation and integration of complex systems. Mary Simpson applied strategic engineering and process management analysis to her work with executive management at Boeing Commercial Airplanes as well as with their major integration and re-engineering initiatives. Currently, Mary is employed as a Chief Systems Engineer at Battelle Memorial Institute, with the Pacific Northwest National Laboratory.