SYSTEMS ENGINEERING INFORMATION MODELS

by

JOSEPH JAMES SIMPSON

A THESIS

Presented to the Faculty of the Graduate School of the

UNIVERSITY OF MISSOURI-ROLLA

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

2004

Approved by

_____	_____
Dr. Ann Miller, Advisor	Dr. Cihan Dagli

_____

Dr. Scott Grasman

ABSTRACT

This thesis addresses information models for systems engineering activities. Systems engineering tasks are a central set of technical management tasks used by large distributed groups of individuals to design, develop, produce, and operate large, engineered systems. In this work, standard systems engineering methods and activities are analyzed and evaluated in the context of current engineering practice. Specific methods and models used for the capture, encoding and storage of systems engineering information and artifacts are given special attention during the evaluation and analysis phase. A generic systems engineering meta-model is then developed and used as a basis for the systems engineering information models that are developed and presented in this work.

The generic systems engineering meta-model developed in this work is founded on the scientific problem solving process. A unique process named CCFRAT (for the Concept, Context, Function, Requirement, Architecture and Test views that are the core system views used in this process) is developed in this work. The meta-model and process are then used as a foundation for the standard relational data modeling activities required to produce the complete logical information models that are developed in this thesis. Relational database tables and processes are developed and presented to support the capture, storage and management of systems engineering program data. Open-source, freely-available computing components were used in the development of this thesis.

ACKNOWLEDGMENTS

Many people have contributed to the development and completion of this thesis. I would like to thank Dr. Brian Mar from the University of Washington for introducing me to the profession of systems engineering. I would further like to thank Dr. Ann Miller, Dr. Cihan Dagli and Dr. Scott Grasman for their help, encouragement, and insights into the practice of systems engineering. I wish to acknowledge the Boeing Learning Together Program and the Boeing Systems Engineering Program for their administrative and financial support during the development of this thesis and the completion of my systems engineering masters program.

Many thanks go to my wife, Mary Simpson, for her support, understanding and shared interest in the field of systems engineering. I am further grateful to all my family members for their understanding and support as I took time away from the family to work through this program.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure Page

1. INTRODUCTION

1.1. DEFINITION OF SYSTEMS ENGINEERING

Systems engineering is a structured technical design and management process used in the design, development, production and operation of large-scale complex systems. The first development and practice of systems engineering is generally attributed to the United States Military during the First or Second World War.[1] However, some authors trace the general practice of systems engineering back much further to Military Engineering Schools in Europe in the 1700's or even before that time to the Roman aqueducts or Egyptian pyramids.[2,3]

Technology brings many benefits to our daily lives. Technology development and deployment also brings a constant increase in complexity associated with the deployment of new technology. Systems engineering methods were developed in-part to deal with this ever increasing level of technical and operational complexity. Systems engineering methods are an extension of basic scientific problem solving methods that have been directly applied to designing systems to solve existing problems in the environment. [4]

1.2. SYSTEMS ENGINEERING PRACTICE

Prior to the mid 1980's most systems engineering techniques were based on the production of written (textual) problem and solution statements as well as analytical (graphical and computational) techniques used to explore various aspects of the problem at hand. Since the mid 1980's and the explosion in the availability of computing resources, many systems engineering methods and processes have been translated to

computer-based tools to facilitate the production, management and distribution of system design information and methods. However, much of the systems engineering material and many of the system artifacts produced on major system programs are based on “pre-computer” information types. Systems engineers commonly refer to specification documents, contract documents, and requirements linking. These are all document-based systems engineering products that are static and non-executable.

The systems documents for a specific program are usually arranged in a “specification tree.” The specification tree is useful to illustrate the order and precedence of each specific document. The task of managing these engineering documents and the requirements links is very important. If done correctly, the task output creates great benefit for the systems development program. However, the task of managing the document tree is time consuming, cognitively demanding, and very sensitive to small errors in logic and content semantics.

1.3. SYSTEMS ENGINEERING MODEL DEVELOPMENT

The systems engineering documents are artifacts of the engineering organizations that participate in the system specification, acquisition, design, development and deployment. In many cases these are legal and contractual documents that create a binding contract between the system owner and the system builder. A rich history of organizational process and development practices that produce both the system product and the system product documentation has been recorded in numerous books and professional publications. The systems engineering methods were first encoded into organizational structures, roles and desk instructions during the 1960’s.

The specific processes and techniques used to develop systems engineering products were enforced and controlled using company specific command media, procedures and processes.

During the late 1980's and early 1990's significant efforts were made to transfer these methods and operations to computer-based systems. In the same time frame, the International Council On Systems Engineering (INCOSE) was established to provide a forum for the further development and publication of systems engineering practices and literature. An INCOSE technical working group, the Tools Database Working Group (TDWG), has developed a list of typical systems engineering tools. These typical systems engineering tools have been mapped to the areas of the standard systems engineering process where they are normally used.[5] Figure 1.3-1 shows a typical three-tier computing system layout and some typical distributions of operational computing components across the three tiers.

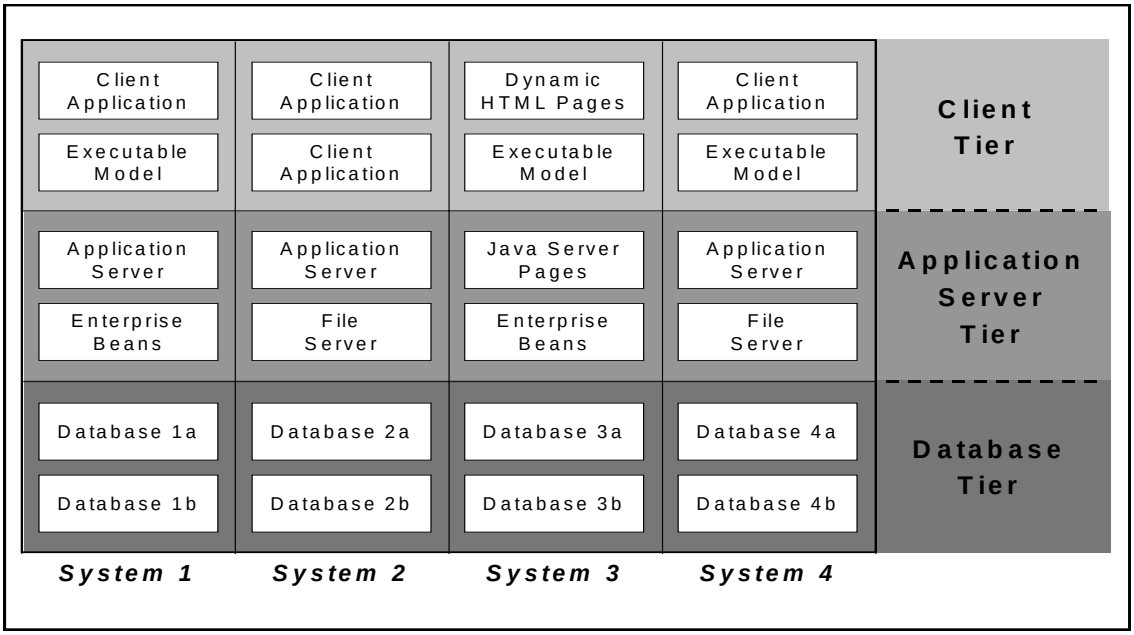

Figure 1.3-1 Three-Tier Computing Systems

Systems engineering computer-based tools have a wide range of application, structure and content. In many cases, companies that create and support specialized systems engineering tools have gone out of business leaving large programs with the data encapsulated in an information format that can no longer accessed. Similarly, other companies have modified integrated office suites to support specific systems engineering tasks, only to have these office suites become obsolete and unsupportable over the long term. This thesis develops a basic systems engineering information model and maps this model to a standards compliant open source relational database system. These types of computing systems will be available from numerous sources and supportable over the long term.

1.4. THESIS APPROACH AND ORGANIZATION

The systems development process literature has been searched to identify systems development models and process approaches. These standard processes are then analyzed to develop a generic systems engineering process that is used as a basis for the design of the systems engineering information model developed in this work. The literature review section covers the history of systems engineering, common systems engineering practice, and current systems engineering standards, tools and approaches.

Given the generic systems engineering process, a systems engineering conceptual model was first developed. Next, the logical model was developed from the conceptual model and used in a standard fashion to develop relational database tables. Open source relational database management systems are then analyzed in to select the

best candidate software package for this work. The PostgreSQL relational database management system (RDMS) was then selected as the deployment platform for the database tables, processes and other artifacts.

The systems engineering global conceptual model was the developed from the information contained in the literature review. The global systems engineering conceptual model was developed as a meta-model that is adaptable to all types of systems engineering development activities. A systems logical data model and relational database tables were then developed based on the high level meta-model. The thesis is completed with a discussion of general considerations in deployment and utilization of a system that is based on the systems engineering models and techniques developed in this thesis.

2. REVIEW OF LITERATURE

2.1. SCOPE OF LITERATURE REVIEW

The literature review covered a wide range of material, including books, professional journals, web sites, and conference proceedings. Given the ubiquitous nature of the term “systems engineering”, only publications and sources that focused on systems engineering as the creation of large, complex system solutions were used as reference. Other interpretations of systems engineering were not considered valid for inclusion in the production of this thesis.

Many definitions are used for the systems engineering professional discipline. The following definitions will be used in this work: Systems engineering is the professional discipline that controls the ordered, systematic design and development of complex systems which require the services of a wide range of domain engineering, specialty engineering and other technical disciplines to successfully deploy a viable system. From this definition we can see that the systems engineer has technical responsibility for the complete system, system segment or subsystem, depending on that engineer’s current assignment and role in the system production process.

2.2. SYSTEMS ENGINEERING HISTORY

Systems engineering activities have been traced to recent large large-scale human activities associated with the First and Second World Wars. However, some authors trace systems engineering activities back much further to the Roman and Egyptian times. In each of these early instances of systems engineering activities, the “systems engineer” or “systems architect” used organizational control to sequence the

activities and guide the systematic steps in the production of the product system. By the time of the Second World War, the tools for systematic design and control of large-scale systems had grown from organizational control to include technical and planning methods like; industrial production planning, distributed real-time communications systems and operational analysis and optimization.

The United States Air Force was the first government activity to have a published systems engineering procedure manual. Air Force Systems Command Manual (AFSCM) 375-5 was published in 1964.[6] The Department of Defense (DoD) published MIL-STD-499, Systems Engineering Management, in 1969 as a mechanism to standardize the content of systems engineering management plans used on DoD programs. MIL-STD-499 was updated to MIL-STD-499A, which was published as guidance for defense contractors working on large defense programs.[5][2] By the middle 1980's MIL-STD-499A and the Army Field Manual (FM) 770-78, "Systems Engineering" were viewed as the basis for the practice of engineering management in the DoD community. The Defense Systems Management College (DSMC) published the first "Systems Engineering Management Guide" in 1983 based on the material contained in MIL-STD-499A and the Army FM 770-78. [6] [7]

A change in the DoD's standards policy, away from military standards and moving toward commercial standards, started in the early 1990's. This fundamental change stopped MIL-STD-499B from being adopted and was the primary driver for the development of two commercial system engineering standards; Electronic Industries Association (EIA) 632, "Processes for Engineering a System", and Institute of Electrical and Electronics Engineers (IEEE) Standard 1220, " Standard for Application and

Management of the Systems Engineering Process."[2] Both of these commercial standards are still in use today addressing the coordination and process integration issues associated with the production and deployment of large complex systems by large complex organizations.

The International Organization for Standardization (IOS) and IEC published ISO/IEC 15288, "Systems Engineering – System Life Cycle Processes", in late 2002. This standard focuses on standardizing the process steps used during the acquisition of a system.

2.3. SYSTEMS ENGINEERING CURRENT PRACTICE

The activities of modern man are facilitated by and generate large-scale complex systems in a seemly endless process. Engineering techniques, engineering systems and systems engineering management processes developed during the 1950's, 1960's and 1970's have been continually adapted and applied to an ever increasing variety of complex systems that are built upon the previous layer of complex systems developed and deployed a few years earlier. [1]

The current practice of systems engineering can be divided into two general classes for discussion and analysis: government systems engineering, and commercial systems engineering. The primary differences between these types of systems engineering activities are customer expectations and the type of system support products that need to be delivered with each product system. These differences are also associated with the methods used to identify the need for new system. [8]

2.4. SYSTEMS ENGINEERING STANDARD PROCESSES

The Electronic Industries Association (EIA) 632 and The Institute of Electrical and Electronic Engineers (IEEE) 1220 standard publications outline two approaches to the processes of engineering complex systems. Each of these documents is valuable in providing a standard framework, vocabulary and approach to systems development that reduces the misunderstandings, improves communications and reduces the time required to produce a complex system product.

In addition to EIA 632 and IEEE 1220, EIA published a companion standard EIA 731, “Systems Engineering Capability Model”, this standard is used by an organization to perform a structured evaluation of its ability to accomplish effective systems engineering tasks and process steps. [9][10]

2.5. PROCESSES FOR ENGINEERING A SYSTEM

The EIA 632 standard established a systematic approach to the processes needed to engineer a system. EIA 632 incorporated best practices gathered from over fifty years of professional experience designing, developing and deploying large complex systems. The approach in EIA 632 is based on three assumptions:

- A system is one or more end products and sets of related enabling products that meet stakeholder needs and expectations.
- Stakeholder defined requirements are fulfilled by products that are composed of a hierarchical set of integrated elements.

- The application of well-defined processes by a skilled, knowledgeable multidisciplinary team to each hierarchical element in the product system will produce the required integrated system.

Based on these assumptions, the standard is intended to assist system developers in:

- Discovering and managing a consistent, complete set of system requirements.
- Enabling
- Enabling the delivery of feasible, cost-effective product systems
- Delivering a product system within cost, schedule, and technical risk constraints.
- Satisfying stakeholder life cycle system requirements at every stage of the product system lifecycle.

The processes defined by the EIA 632 standard are divided into five logical groups which are them further decomposed into sub-processes under each major process. The five main process groups and sub-processes are:

- Acquisition and Supply
    - Supply Process
    - Acquisition Process
- Technical Management
    - Planning Process
    - Assessment Process
    - Control Process
- System Design

    - Requirements Definition Process
    - Solution Definition Process
- Product Realization
    - Implementation Process
    - Transition to Use Process
- Technical Evaluation
    - Systems Analysis Process
    - Requirements Validation Process
    - System Verification Process
    - End Product Validation Process

The specific steps associated with these five fundamental process areas can be used to develop all types of product systems. The specific context of the systems acquisition and development activity may impact the sub-process definition and specific sub-process tasks and methods. However, the high level fundamental processes will be the same in all cases of system acquisition. [9]

2.6. STANDARD SYSTEMS ENGINEERING PROCESS, IEEE 1220

The IEEE 1220 standard was released in 1998 as a full standard that addresses a lower level of system development processes than those addressed by EIA 632. IEEE 1220 is focused more on the technical process activities and tasks of system production and does not address the complete contracting and acquisition process elements that are more fully addressed by EIA 632.

IEEE 1220 process sections are:

- Requirements Analysis
- Requirements Validation
- Functional Analysis
- Systems Analysis
    - Requirements Trade Studies and Assessment
    - Functional Trade Studies and Assessments
    - Design Trade Studies and Assessments
- Functional Verification
- Synthesis
- Physical Verification
- Control Process
    - Configuration Management
    - Data Management
    - Interface Management
    - Performance Based Progress Measurements
    - Risk Management

Both the EIA 632 and the IEEE 1220 standards present block diagrams that locate the product system in the context of a system hierarchy. However, these system representations are quite different. [5][11]

2.7. SYSTEMS ENGINEERING CAPABILITY MODEL, EIA 731

The Systems Engineering Capability Model, EIA 731, is intended to support an organizations effort to improve its capability to consistently design, develop, and

produce high quality systems within schedule and budget constraints. EIA 731 is a companion standard to the EIA 632 and IEEE 1220 standards, and augments the deployment and utilization of these standards by creating a structured set of methods to evaluate an organizations ability to perform systems engineering tasks. Successful application of the practices contained in EIA 731 will allow an engineering organization to:

- Decrease the time required to produce a product system
- Create product systems that meet all stakeholder requirements
- Reduce system life cycle costs
- Control and decrease the frequency of engineering changes
- Increase product system quality
- Enhance the technical communication ability of engineering teams
- Create more adaptable and sustainable product systems
- Decrease product system development risks

EIA 731 is composed of two parts, EIA 731.1 “Systems Engineering Capability Model” (SECM) and EIA 731.2 “Systems Engineering Capability Model Appraisal Method” (SECM AP). The SECM is a logical tool designed to support an organizations efforts to access, improve and develop systems engineering capabilities. The model is developed around a set of focus areas that cover the complete span of systems engineering activities in an organization. These focus areas are grouped into three categories: the Technical Group, the Management Group, and the Environment Group. Practices in the Technical Group represent and correspond to practices and definitions found in EIA 632 and EIA 1220. Practices in the Management Group represent, and

correspond to, industry-wide best practices used to increase efficiency and promote cost-effectiveness in the implementation of the systems engineering process. Practices in the Environmental Group align technology development practices with business objectives to enable the sustained execution of systems engineering practices across the total organization. The Technical Group is made up of the following focus areas:

- Define Stakeholder and System Level Requirements
- Define Technical Problem
- Define Solution
- Assess and Select
- Integrate System
- Verify System
- Validate System

The Management Group contains the following focus areas:

- Plan and Organize
- Monitor and Control
- Integrate Disciplines
- Coordinate with Suppliers
- Manage Risk
- Manage Data
- Manage Configurations
- Ensure Quality

The Environment Group completes the list by adding the following areas:

- Define and Improve the Systems Engineering Process

- Manage Competency
- Manage Technology
- Manage Systems Engineering Support Environment

Taken together these nineteen focus areas cover all major organizational activities associated with the practice of systems engineering. [9]

The Systems Engineering Capability Model Appraisal Method, EIA 731.2, is a formal, structured audit and appraisal method used to evaluate an organization's ability to perform systems engineering tasks in an effective and efficient manner. The raw data collected during the appraisal is used evaluate and rank the organizations systems engineering capability. The categories used to rank organizational capability are defined below:

- Initial; No system engineering practices are performed. Limited or no effective system engineering processes are used.
- Performed; Some limited systems engineering practices are performed by specific individuals in the organization. Activities are ad hoc with little formal process control.
- Managed; Systems engineering activities are planned and tracked, with work products conforming to written standards. Established metrics are used to measure activity compliance. The organizations activities are managed based on this measured performance.
- Defined; Standard well-defined processes are used to perform activities. An organization-wide standard planning process is used to manage and improve the defined processes.

- Measured; Organizational and program metrics are used to track task performance in a quantitative manner.
- Optimizing; Quantitative performances goals are used as a basis for efficiently managing processes. Processes are under constant, continuous improvement based on the data collected from the process measurement activities.

This audit method allows an organization the ability to focus improvement activities in the areas that will provide the maximum benefit for the least effort.

2.8. DEFENSE SYSTEM SOFTWARE DEVELOPMENT, DOD-STD-2167A

The Defense System Software Development standard is reviewed and discussed from the systems requirements traceability viewpoint. The document-centric and hierarchal nature of software development activities created a need to trace systems requirements from the body of one requirements document to the body text of other requirements documents. The lower-level requirements documents detailed the mechanism for the fulfillment of the requirements contained in higher-level requirements document. Requirements traceability throughout the total system lifecycle was mandated by DoD-STD-2167A. This mandate formalizes the process of linking or relating the document sections that describe a need, and the document sections that describe how that stated need will be satisfied. [12] [13]

Even though the practice of traceability was mandated for use on projects that developed software systems for the DoD, no common model or specific set of practices were specified to guide the implementation of traceability practices. The Naval

Postgraduate School conducted a series of studies on requirements traceability practices associated with government software projects. A primary outcome of these studies was a set of conceptual information models that address the basic areas of software systems development. Four conceptual models were developed during this work, they are: Requirements Management Model (Figure 2.8-1), Design Allocation Model (Figure 2.8-2), Design/Implementation Decision Making Model (Figure 2.8-3), and Compliance Verification Model (Figure 2.8-4). The requirements traceability model is composed of three out of four of these models: the Requirements Management, Design/Implementation and Allocation, and the Compliance Verification. The fourth model, Design/Implementation Decision Making is used only for the decision process during design and allocation.[13][14] These models all focus on the types of requirements traceability links that should be used tofulfill the mandated requirements linking activity in an intelligent and useful manner.

In the Naval Postgraduate studies two types of traceability uses were identified, “low end users” and “high end users.” The “low end users” had a much more limited traceability model than the “high end users.”[13] Therefore, only the “high end users” traceability models are considered in this thesis. Much of the semantic and conceptual structure necessary to effectively use these traceability models must be imposed by the organization creating and using each specific requirements document. This organizational structure also includes the form and relationships of the program specification tree. These information models are solely designed to link text based system descriptions, and do not directly address executable system models. Executable

models and graphical system representations are absent from this set of information models

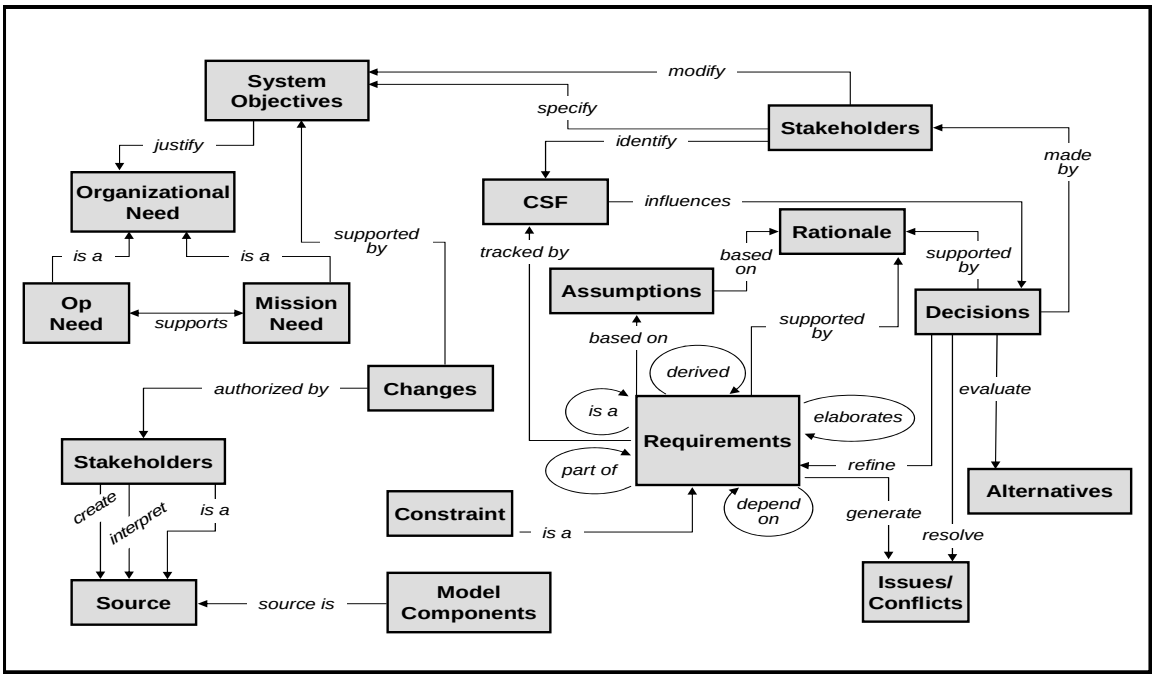

Figure 2.8-1 Requirements Management Model

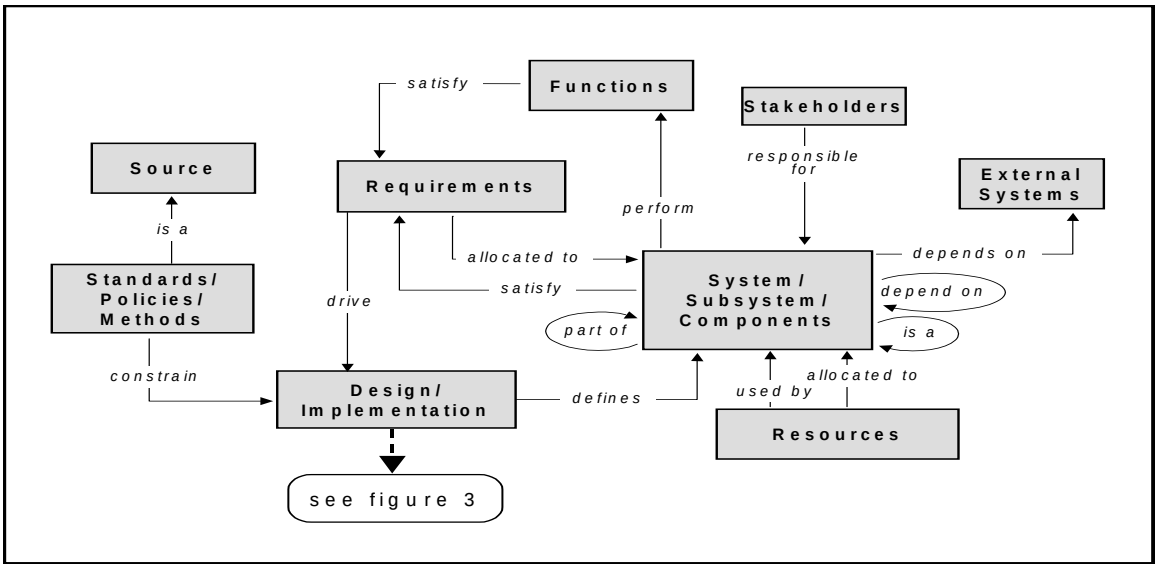

Figure 2.8-2 Design Allocation Model

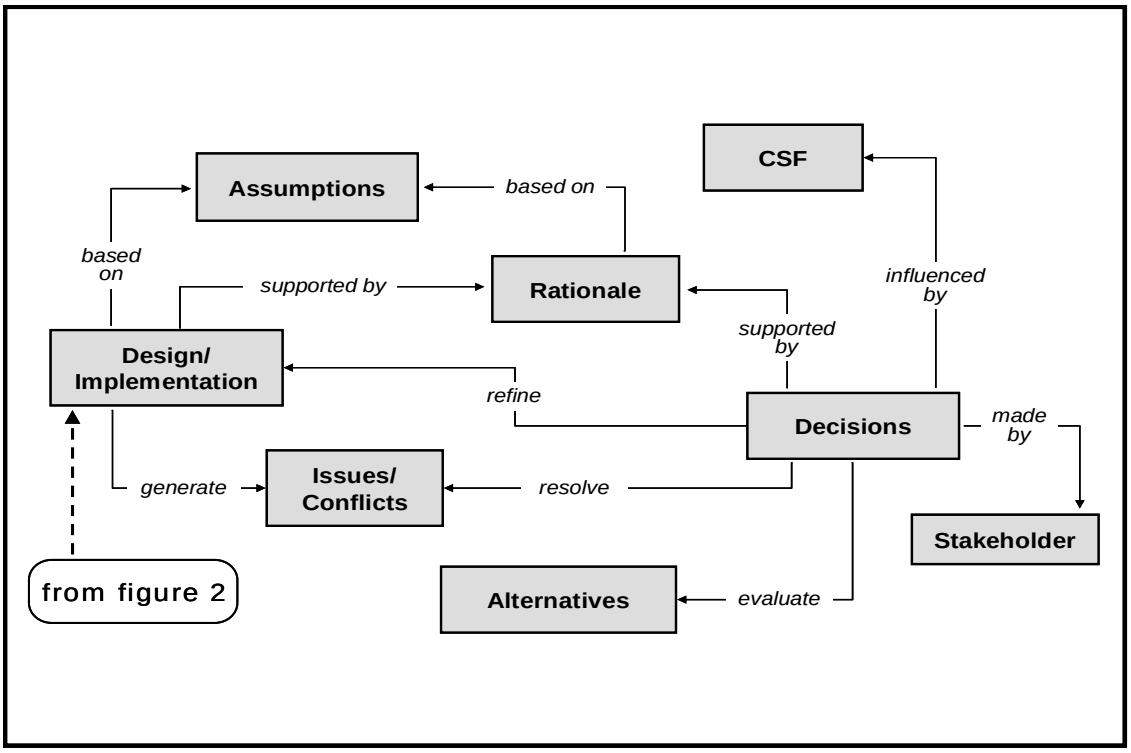

Figure 2.8-3 Design/Implementation Decision Making Model

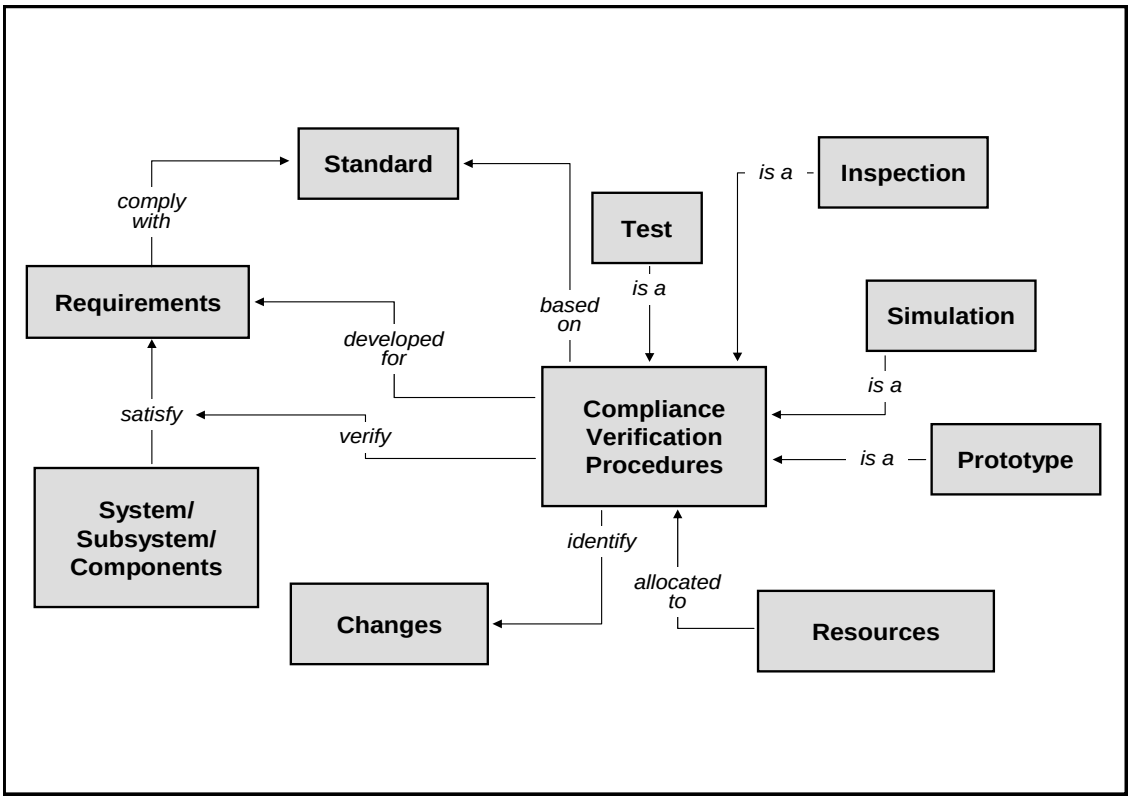

Figure 2.8-4 Compliance Verification Model

2.9. SYSTEMS ENGINEERING PROCESS IMPLEMENTATION

Almost all systems engineering activities are associated with a controlled, time-phased process that integrates the necessary multiple specialty engineering disciplines required to design, develop and deploy large-scale complex systems. Chase outlined a scientific approach to systems design and engineering which linked the steps in the systems design process to the steps in basic scientific problem solving. Figure 2.9-1 shows these relationships. [15]

Steps to Apply Scientific Method to Problem Solving	Early Man Developing Cultural Patterns	Basic Research	Operations Research	System Design
1 Recognize problem	Unsatisfied Physiological Need	Identify gap in body of scientific knowledge	Identify operational objective to be achieved	Describe mission or use requirements
2 Describe problem	Discover alternative ways to increase satisfaction of need	Develop theory of probable cause and effect relationships	Define situation & resources which can be used to attain objectives	Define req'd operation and logistic functions to attain use objectives
3 Select hypothesis for solving problem	Select favored way of satisfying need	Select hypothesis for investigation	Describe tailorable variables to achieve desired objectives	Specify the system performance / design requirements
4 Develop model for testing hypothesis	Devise implements & techniques to practice favored way	Describe experimental model to test hypothesis	Construct statistical model to interrelate variable conditions	Accomplish detail design & qualification testing of components
5 Conduct tests under controlled conditions	Use selected techniques for some period of time	Conduct controlled lab/field investigation to obtain data	Perform computation to obtain statistical values	Build, assemble, test complete prototype system
6 Analyze and evaluate test data	Decide if techniques result in tolerable satisfaction of need	Analyze and evaluate collected data	Analyze and evaluate summary statistical data	Analyze and evaluate test data
7 Derive conclusions to confirm, deny, modify hypothesis	Transmit techniques to others & establish cultural pattern	Derive conclusions to confirm, deny, modify hypothesis	Recommend actions to achieve desired objectives	Recommend modifications for production system

Figure 2.9-1 Scientific Approach to Problem Solving

Differences between systems and software processes notations and design approaches create the need for specialized documentation, integration and communications activities to successfully complete these large-scale tasks.[16] [17] The relationships among systems engineering and software engineering tasks and discipline

interfaces must be evaluated in terms of the complete system design task. The total set of system behavior, objectives and functions must be evaluated in a manner that is understood by all engineering participants.[18] The functions, requirements, architecture, and test (FRAT) approach developed by Mar is one type of conceptual framework that can be used to address the semantic confusion found in the systems engineering discipline.[19] The FRAT approach has been found to be a useful tool in many large scale projects.

Systems engineering is an enterprise activity whose processes are described in organizational process and procedure manuals. The specific operational steps are sequenced and enforced by organizational controls. In large, complex system creation activities systems engineers can lose sight of the total system and the role they play in the system creation process. [19] However, some systems engineers have developed and applied Model Based Systems Engineering to the problem of large distributed systems designs.[20]

Oliver provides an overview of the application of Model Based Systems Engineering to the systems engineering process. The approach proposed by Oliver creates an engineering process and context model with six layers. These six layers are: the systems engineering process layer, the information representation layer, the tool layer, the changes layer, the staffing layer and the external visibility and review layer.[21] The systems engineering process layer contains a core set of seven process steps that map very well to the seven problem solving steps outlined by Chase. The seven systems engineering core process steps are: assess available information, define effectiveness measures, create the “what” model, create the “how” model, perform

trade-off analysis, create sequential build and test plan. A functional flow block diagram (FFBD)of the process flow is shown in Figure 2.9-2.

A core set of processes and information models for systems engineering are developed and discussed by Oliver in a book and other publications.[22][23] The processes are presented in a FFBD graphical notation while the

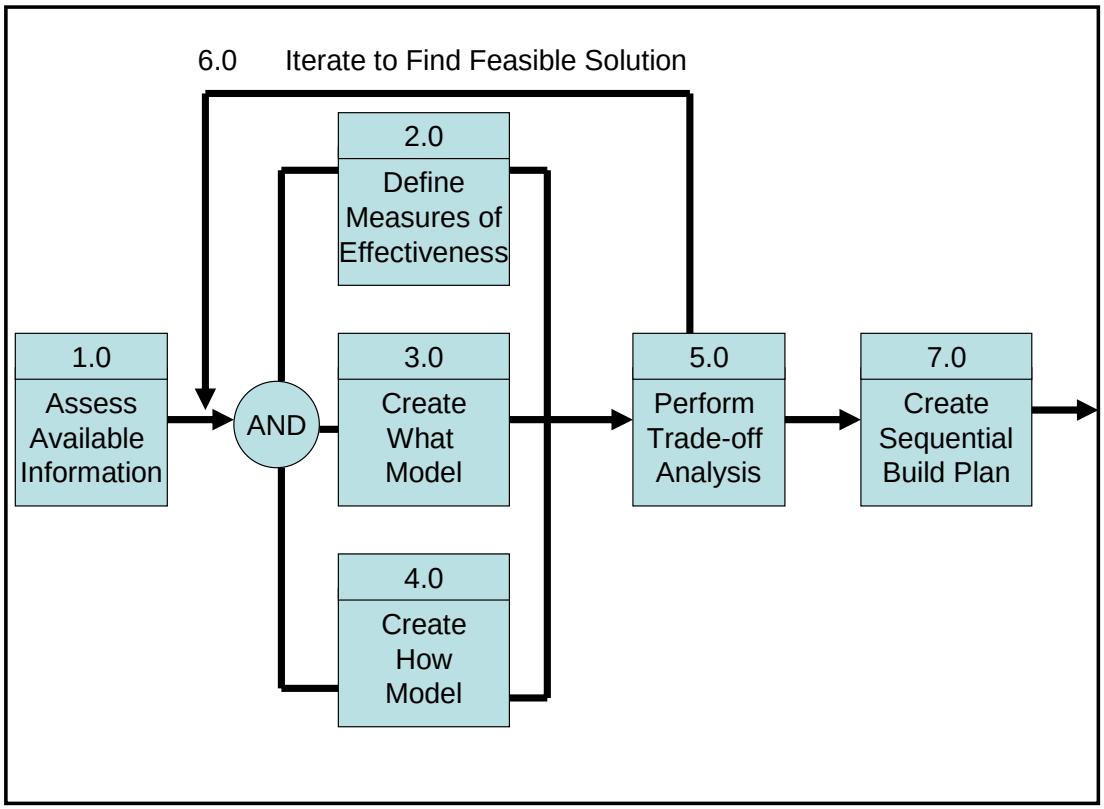

Figure 2.9-2 Core Process Flow Step

information models are presented using a subset of the Object Modeling Technique (OMT) notation. Using these notation sets, a series of information models are generated for the following application domains:

- System behavior
- System input and output

- System structure and behavior
- System requirements
- Effectiveness measure creation
- Text requirements, behavior and content
- Build and test plan

These information models were analyzed to identify each of the classes, or entities, and the relationships between these classes. A list of the classes and their corresponding relationships for the system behavior model are shown in Table B-1 in appendix B. Table B-2 lists the classes and relationships for the system input and output model. This system input and output model contains many different types of objects (functional, physical, state) and many different object modes and states (storage, access, condition). The classes and relationships associated with the system behavior and structure information model are shown in Table B-3 in appendix B. The classes and relationships associated with the system requirements model are presented in Table B-4.

The classes and relationships that make up the system effectiveness measures (EM) creation information model are presented in Table B-5. In the information and systems modeling approach used by Oliver, some of the classes in the information model are color coded to distinguish the type of information provided by the object. Three general information types are marked: text information, behavior information, and context information. This is an interesting addition to the standard information model and provides a richer semantic and deeper understanding of the information models content and meaning. Table B-6, in appendix B, lists the classes and relationships associated with the information model for text requirements, behavior, and context.

Each object class name is marked to indicate the information category, (t) for text, (b) for behavior, and (c) for context. The next table, Table B-7, shows the content of the information model for text requirements, structure and context. Each class in the model and the relationship between these classes is listed. This model is focused on the system structure, which details "how the system is built", while the previous model focused on the system behavior, which is "what the system does."

Systems engineering activities are used to develop alternative designs as well as select the preferred candidate solution from the pool of acceptable alternatives. This set of systems engineering activities is commonly called tradeoff analysis. An information model designed to represent tradeoff analysis activities is presented in Table B-8. Consistent with the functional flow block diagrams the information models listed in these tables are associated with a core systems engineering process that has six functions;

- Assess available information
- Define effectiveness measures
- Create behavior model
- Create structure model
- Perform tradeoff analysis
- Create sequential build and test plan

Table B-9, in appendix B, lists the classes and relationships associated with the information model for the sequential build and test plan (SBTP). The information models that have been presented here are based on the work of Oliver. [19][20][21][22] These information models are an object-oriented type of model that is basically different

than the “entity relationship” diagram. However, they do provide one perspective about the types of objects and relationships found in the systems engineering domain. The systems engineering objects and relationships become better defined when they are applied in a specific system development context in support of a specific set of tasks.

2.10. SYSTEMS ENGINEERING INFORMATION VIEWS

Key products of the systems engineering process are large, complex systems with long life cycles and multiple levels of components, interactions and processes. One method that is used to manage the complexity associated with these types of systems is the development and application of standard processes and interfaces. The United States military has developed a set of global, high-level standards. The primary purpose of these standards is to constrain systems solutions in a manner that encourages system interoperability. Associated sets of rules establish three basic views of a technical architecture: functional view, physical view, and technical view. While these high level standards and rule sets are not static, they will change at a much slower rate than the physical components and subsystems that comprise the final physical instance of any given large scale complex system. [24]

Given the high level constraints placed on the design solution space, common design processes are viewed as a positive benefit that encourages the production of large scale interoperable systems. Executable systems models are very important in verifying the dynamic systems interactions. A standard structured systems analysis process was developed to facilitate the communications, command, control, computer, intelligence, surveillance, and reconnaissance (C4ISR) systems design including, the executable

models necessary to verify and validate each candidate system solution. [25] In a similar fashion, an object oriented approach was developed to create candidate systems solutions in a structured, systematic manner. [26]

Dividing a system into various views is a fairly common idea. Mar developed the functions, requirements and architecture (FRA) views of a system.[19] Later on Mar added a test view to the basic set of views creating the functions, requirements, architecture and tests (FRAT) views of systems engineering as well as the FRAT process for the engineering of large complex systems.[27] The FRAT approach also included a strict definition of the term function and the term requirement. Defining the terms provided a logical, operational semantic tie between the two terms. Functions are defined as what the system must do, while a requirement is defined as how well the function must be done. These two definitions are used to combine the function-requirement pair into a problem statement that details what must be done and how well it must be done. In the same manner, the architecture and test terms are grouped together to provide the solution and verification for the problem statement. The FRAT approach not only uses four different views of a system to organize system development activity, but also reduces confusion relating to the word function and the word requirement. While this approach does not remove all of the confusion in this area, it does help to greatly reduce the uncertainty associated with these terms when they are applied in a specific development context.

An alternative set of five system views for large, complex, computer-based systems, was proposed by Karangelen and Hoang. These five views are the environmental capture view, informational capture view, functional capture view,

behavioral view and the implementation capture view. These five views were presented as a basis for constructing a robust, linked, systems-design methodology that has clearly defined interfaces, semantics and defined systems views.[28] This approach has been used as the basis of a computer aided design tool, however it is not directly applicable to systems of a general nature.

Another application of system views for system complexity control is presented by Frank.[29] This application focuses on views of the system architecture as a system design aid. One cohesive aspect of the system under design is formulated into a system view that describes what that aspect is and does. Eight system architecture views are presented: mission application view, mission scenario view, data view, user view, hardware view, state and mode view, infrastructure view, and facilities view. These eight views are perceived to be a complete set of necessary architecture views one must consider when building a large, complex, software-intensive system or set of systems.

Systems views also play an important role in systems architecting. Maier discusses the importance of systems views during the process of architecting hardware and software systems.[30] Creating and architecting the hardware-software interface present a unique challenge. This challenge was also discussed in detail by Alford.[17][18]. Maier suggests that the conceptual mismatch between hardware systems and software systems design can be addressed using five system views. These views are the logical view, process view, physical view, development view, and scenario view. These views are used to organize the system requirements and facilitate a structured bottom-up type of development process. Percivall details the application of a systems architecture standard to a satellite development project. In this particular

activity five system views were selected to represent the system under development: functional view, data management view, communications view, security view and systems management view.[31] While a satellite is a complex system combining both hardware and software, the system views chosen were quite different than those proposed by Maier.

2.11. SYSTEMS ENGINEERING REQUIREMENTS MODELS

Systems engineering has two basic foundations: a systems engineering process and requirements management activities. Just as there are many "standard" systems engineering processes there are many different interpretations for specific requirements management activities. Grady discussed the connection between poorly written systems engineering standards and confusion associated with the sequence of requirements analysis and functional analysis.[32] The key area of confusion, according to Grady, is the relationship between the current level of system definition activity and the higher-level system need or function that produced the lower-level system requirements activity. This fundamental semantic relationship between specified need and specified solution, as it appears in the project lifecycle, is documented by the following authors; Forsberg and Mooz, Mar and Morias, as well as Simpson and Simpson. [16][33][34]

Most requirements management practices are based on managing a collection of text documents and sets of linkages between and among the documents in this collection. The arrangement and relationships between the documents in the collection are usually program and project specific, with many of the documents carrying a contractual connection. Document hierarchies usually begin with high level need

statements from the customer, and flow-down to system and system segment specification documents that detail the types of systems solutions that will be built to fulfill the stated need. [35][36][37][38] The differences between software, hardware, and organizational specifications, as well as other systems development activities, add to the uncertainty associated with the requirements sources, priority and level of system interaction. In different systems development and design projects the "natural systems development" order and information flow varies across the individual aspects of the system design, development and production processes. These differences in the development order create activity and requirements gaps that must be evaluated and addressed by the systems engineering process. [39]

Systems engineering requirements models have been used to organize the various aspects of the systems engineering phases, tasks and activities. It is clear that both textual and executable types of data and models are required to effectively address the numerous types and forms of requirements information encountered in a systems development process. The approach used in each specific case must match the available resources and skill level of the individuals using the systems engineering requirements models and support tools. [40] One of the primary characteristics of a requirement is a unit of measure that is used in the testing process to verify and validate the requirement. One of the primary functions of a requirement is supporting design decisions during a trade study. Systems engineering requirements models must support both of these types of information capture and processing. Decision support, and requirements verification and validation, are key aspects of any systems engineering requirements model. [41][42]

Standard decision metrics can be coupled with matrix processes to evaluate the stakeholder requirements at each level and phase of system development. [43]

Systems engineering requirements models have been applied to many aspects of system development. Sometimes these requirements models are just applied to the linkages between textual data sets. Other times they are applied to both the system production process and the system being produced. The large number of ways that a major systems development project can be (and have been) described in requirements models adds to the complexity of the practice of systems engineering. Executable requirements models have been developed to assist in the evaluation of requirements flow and utilization in a systems development process.[44] Tools and specialty notations have been developed to support the design of executable systems requirements models. RDD-100 and IDEF0 are both examples of this type of tool.[45] Custom tools have also been built to support a single aspect of requirements management, like verification or concept development requirements traceability.[46][47]

In other cases, systems engineering requirements models are complicated by the amount of software used in the final product. The level of reliability and fault tolerance needed in the final product system also adds complexity to a project. These aspects add further variation to each specific type of systems engineering requirements data capture, management and storage task. Software, systems and hardware requirements have common terms, but in many cases the meaning of these terms are context dependent. The mixed nature of many systems requirements models does not specifically account for these variations in context, concept and meaning. [48] [49][50][51]

2.12. SYSTEMS ENGINEERING MODELS

Systems engineering models are used in the development of complex systems that require the coordination of large groups of individuals, organizations, facilities and other resources. The nature and type of models used are directly related to the systems engineering methods practiced by the organization or specified by the customer. Military and government customers focus on the product system or the system they are purchasing as the primary subject for their systems engineering efforts. However, a growing number of authors are advocating the application of systems engineering modeling concepts and practices to all relevant systems associated with the systems development activities.[19][32][33][52][53] Modeling all primary systems relevant to the current systems design and development process provides a uniform method of describing what is not in the product system, identifying and recording important aspects of the system boundary and presents a mechanism to record the systems development process details. Modeling of all of the primary systems also raises some fundamental questions about the nature of a system, what aspects of a system should be modeled and what techniques are best for executing the systems modeling activity.[54][55][56] These fundamental questions need to be addressed in a common uniform manner to better understand, develop, document and operate the current set of systems under development. The interaction and boundary between man-made systems and natural systems, as well as the level of effort and resources required to create the primary systems, will drive any set of systems modeling activities.[57][58] The current set of systems engineering models cluster around two main areas: systems engineering models

that describe the system under design and systems engineering models that describe the information and computing resources required to support system design.

Model-driven system design is a systems engineering design approach that incorporates executable models and design notation as well as the traditional text-based approach to the description of systems and systems requirements. Model driven design activities are organized around the established systems engineering process, and are used in conjunction with specialized tools and notations to create models of complex systems and systems architectures.[59][60] The general approach to systems modeling using systems architecting techniques is basically the same as systems engineering modeling. However, the final product is not a system model but a system architecture. The systems architecture is a “higher level” description of the system to be produced. [61][62][63] Systems architecture models are useful for a number of reasons, including abstract information communication, data transfer and design documentation. Global system architectures can be modeled, or a subset of the architecture can be modeled, to address the stated design needs. Global architecture analysis, used to evaluate and select optimized systems architecture, provides insights into various aspects of system complexity and design completeness.[64] Other specialized systems models focus on the conceptual system description phase or specific system development tasks like test and validation. These models are useful tools during the systems development phase.[65][66] In each case, a common systems engineering process model and systems performance metrics are required to evaluate the effectiveness of the current systems design activity. [67][68][69]

Systems engineering, by its very nature, requires a suite of supporting computing and information management resources. Vast amounts of data are created during a systems development process and this data must be properly managed to extract the greatest value from the data and information.[70][71] Models for systems engineering computing resources have been developed from various points of view, including, business systems re-engineering and systems engineering automation.[72][73] Systems engineering models that are based on the functional view and the solution view of systems design as well as system architectural informational content have also been proposed. [74][75]

Integration and application of systems engineering computing resources requires the definition and modeling of the phases of the systems engineering process as well as well-defined interfaces between systems engineering computer based tools and information repositories. These integrated tools and processes are the basis for the computing resource models required to support systems engineering activities. [76] [77] [78] Detailed process algorithms have been developed to support the development and application of computing resource models. [79]

2.13. SYSTEMS ENGINEERING INFORMATION MODELS

Systems engineering data is stored in a repository for project or program systems engineering data. The form of this data repository is most often a relational or object-oriented database. For this database to be useful it must be an integrated, multi-user database that provides a repository for all information associated with the systems engineering process and activities. This repository will include all data, schema,

models, tools, technical management decisions, process analysis, requirements changes, process and product metrics as well as trade studies and trade-off analysis. Two groups of relational database tables are proposed to store and track systems engineering data. One group of tables contains data that tracks management activities. The other group tracks the systems architecture evolution.

Four tables are defined to track management data: systems engineering management plan (SEMP) table, systems engineering master schedule (SEMS) table, transactions table and the status table. The SEMP table contains the following attributes: paragraph, title, help text, detail text, figure, supports SEMS ID. The SEMS table contains the following attributes; SEMS ID, project phase, major event, title, description, completion criteria, deliverables, system breakdown structure (SBS) ID impacted. The transactions table contains the following attributes: transaction ID, SBS ID impacted, transaction type, problem statement, analysis summary, recommended solution, detailed analysis reference, agreed by, responsible, open date, and close date. The status table contains the following attributes: status ID, SEMS event, percent complete (technical), exception report, remarks, SBS entity, systems engineering metrics 1, systems engineering metrics 2, systems engineering metrics n, and status date.

The four tables defined to track the systems architecture evolution are: requirements table, functions table, SBS table and test table. The requirements table contains the following attributes: requirement ID, parent, title, summary text, detailed text, classification, test method, analysis summary, derived requirement, functional allocation, and version. The functions table contains the following attributes: function ID, relationship, title, summary text, detail text, test method, analysis summary, SBS

allocation, percent evolution, and version. The SBS table contains the following attributes; SBS ID, title, type, summary description, detail description, test reference, analysis summary, statement of work, risk analysis, responsible, cost breakdown, and version. The test table contains the following attributes: test ID, paragraph, title, summary statement, detail statement, remarks, function tested, and version. A database constructed from these two groups of tables, management tables and architecture tables, will provide a system wide information resource for design and management activities.[80]

Systems engineering information models have also been developed to specifically support the technical communications and systems development data needs of integrated process teams when they are in the process of creating a new set of systems. These information models have been divided into two groups: management information, and design and verification information. These information models are presented in a “class-relationship-class” type of diagram. Table B-10, in appendix B, shows the management information model. The design and verification information classes and relationships are presented in Table B-11.

These two information models are used to develop formal databases to support and integrate product teams and structured design processes.[81] The two previous systems engineering information models were based on a top-down analysis of the organization processes and systems development processes. Another activity is taking more of a bottom-up approach to the development of information models for systems engineering. This effort started out as an activity to develop a basis for the exchange of information between systems engineering design tools and has expanded to cover the

complete system design space. A conceptual information model has been developed and is presented in Table B-12 of appendix B.[82][83] The classes and relationships presented in the conceptual information model are developed at a high level of abstraction.

Other views of systems engineering information models have been developed. These information models focus on managing the complexity of the systems development activity by providing a well-designed, process-based context. In this way, a structured systems information model is developed and maintained. One such information model has been developed to support the systematic management of requirements. This information model has five main classes: top-level specification, relationship from top-level specification to system specification, system specification at intermediate decomposition level, relationship from system specification to equipment specification and equipment specification. These main classes, combined with a standard process and general main class attributes, form the foundation of this approach to systems engineering information modeling.[84]

Another approach to the development of an information architecture for systems engineering uses three primary elements: an information architecture, an information structure optimized by domain, and an enterprise-wide, decision-making and technology-planning process. The information architecture is developed to relate system requirements, system models and system decisions in a general enterprise context. To support this architecture, information structures are developed in a modular fashion for business data, user and customer data, product development data, project management data, miscellaneous data, link data and schema definition data. Each of these specific

data modules contains data that is necessary to support requirements management and decision analysis in the larger enterprise.[85]

Systems engineering information models and architectures have also been designed to control the increasing complexity found in the systems engineering and development domains. This complexity reduction approach is based on the assumption that mastering the "natural order" of the information in the system development domain will reduced the complexity associated with systems engineering tasks and activities. Systems engineering activity complexity is reduced by recognizing and using the underlying semantic structure and patterns associated with the systems engineering domain. The basic patterns in these information structures are associated with the product architecture and the process architecture. An incremental development approach to transform from legacy information systems to "smart information architectures" was developed using four phases. These four phases of information architecture development are: (1) modeling and analysis of legacy architecture, (2) identification of enterprise specific objectives and boundary conditions, (3) synthesis and optimization of reference architecture, and (4) identification of upgrade paths. This basic approach creates a flexible information management system that is used to manage enterprise information and knowledge.[86] These types of adaptable information and knowledge management systems are necessary to effectively manage the vast amounts of data associated with current systems engineering activities.[87]

There are other examples of systems engineering activities that have been facilitated using custom data structures for each step in the standard process as well as the use of concept mapping techniques to communicate detailed technical

information.[88][89] In these specific cases, structured data models and concept models are developed to enhance the communication among systems organizations and individuals involved in the development of large systems programs.

3. RELATIONAL DATABASE MANAGEMENT SYSTEMS

3.1. GENERAL APPLICATION

Systems engineering is an engineering activity that is dispersed over a large group of technical disciplines and engineering personnel. A systems engineering information management model is developed using the concepts previously developed. To facilitate the widest possible deployment, only open source relational database management systems are considered as possible candidate deployment targets for use in this activity. Open source systems have some general advantages over other proprietary or closed source database management systems. These advantages include: access to application programming interfaces (API), high quality, low cost and robust operational features.

In addition to advantages provided by open source characteristics, each database management system will provide further advantages and benefits by closely adhering to well-known, publicly-available open standards. One such standard is the Structured Query Language (SQL) standard. SQL was first standardized in 1986 with a major revision in 1992 (SQL-92). The latest version, SQL-99, contains object-oriented features.[90] The combination of open source advantages, and detailed standards compliance, provides a strong foundation for wide-spread, industrial-quality information systems deployments that are flexible, adaptable and that provide long term functional capability.

3.2. MYSQL DATABASE MANAGEMENT SYSTEM

MySQL is an open source relational database management system that claims to be the "World's most popular Open Source database." While MySQL is quite popular and very fast, especially of web-based transactions, it does not implement very many of the standard SQL processes.[91] Only entry-level SQL-92 functions are supported at this time, with a goal of supporting the full SQL-99 standard in the future.[92] The lack of foreign key support, triggers and stored procedures in MySQL removes this product from further analysis for the current system development activity.

3.3. HSQL DATABASE ENGINE

HSQL is a lightweight 100 per cent Java SQL database engine. This small, less than 160k, database engine supports a large subset of SQL-92 as well as foreign keys, triggers, stored procedures and constraints. [93] Even though HSQL supports more of the SQL-92 standard than MySQL does, it does not support database replication and provides weak support for multi-user client server types of applications. [94] [95]

3.3.1. POSTGRESQL

PostgreSQL is the most advanced open-source database and supports Atomicity, Consistency of Preservation, Isolation, and Durability (ACID) process actions. In addition to these features, PostgreSQL implements most of the SQL-92 standard as well as much of the SQL-99 standard set. Further advantages of PostgreSQL include its support of: (1) many of the object-relational concepts in SGL-99, (2) a wide range of built-in complex data types, collections and sequences, table structure inheritance,

multiple computer language types, and (3) is available for use on many different types of operating systems. [96]

PostgreSQL has been chosen to support the work covered by this thesis. PostgreSQL contains all of the features required to support the implementation of the database constructs derived from the system engineering conceptual level data models.

4. MODEL DEVELOPMENT

4.1. GENERIC INFORMATION MODEL DESIGN CRITERIA

As highlighted by the previous literature review, information models used to support activities in the systems engineering domain are designed for many purposes. These information models range from structured concept maps, used to help individuals communicate, to object oriented systems models used to facilitate the storage and manipulation of data in complex data stores. Any generic systems engineering information model must be able to support this wide range of application. In essence, the generic information model must be based on fundamental systems engineering patterns that reduce domain complexity for the human users. At the same time, the generic model must provide a powerful, extensible construct upon which the design of large, distributed database systems can be effectively based. The primary design goal is the reduction of system complexity, both in system production and system use.

The meta-model must be based on the “natural information flow” found in the systems engineering domain. The concept of “natural information flow” is a very powerful construct when combined with the concepts of hierarchy and semantics in context. From the engineering aspect, the model must address the activity of problem solving. At the same time, from the systems aspect, the model must address basic systems concepts. The most important systems concepts that must be addressed are system components, system boundaries and the system universe or environment. The most important engineering concepts that must be addressed by the meta-model are problem solving and the application of systems concepts to the development of systems solutions for complex problems.

Based on fundamental systems concepts, each system exists in an environment populated with other systems. From a systems engineering point of view the most import systems are the “product system” or the system under design, the “production system” or the system that produces the product system as well as the environment system that contains the production system and provides the context in which the product system will operate. The environment system also provides resources from which the product system is constructed and competitive forces that operate on the production system value set. Most “production systems” will be modeled by a systems engineering process; a small set of standard systems engineering processes were discussed previously.

The primary components of the systems engineering meta-model are a set of three generic systems: the environment system, the product system and the process system. These three systems are adopted from systems science and the FRAT model developed by Mar and Morias. Each of these three systems is abstracted into six views, context view, concept view, functions view, requirements view, architecture view and test view. Four of these views, the function view, requirement view, architecture view, and test view are adopted directly from the FRAT model. The four basic FRAT views are encapsulated in a system concept view and a system context view.

The system concept and context view are added to the meta-model to provide a mechanism where the system designers can detail specific global aspects of the current system design and how these aspects relate to other systems in the environment. The context view is used to detail relationships that are important between the current system and other systems existing in the environment. The concept view provides a mechanism to detail controlling concepts that directly relate to the current system design. In any

specific system design, the design or systems solution approach can be decomposed and described in a number of ways. These ways include functional decomposition, “part of” decomposition, “configuration item” type of decomposition or a combination of all of these approaches. . The concept view is used to detail the controlling system abstractions and processes in each specific case. As the system design process develops, more and more systems in the environment can be identified and the product systems components and system boundary become increasingly well defined.

4.2. GENERIC INFORMATION MODEL DESIGN

Relational database system and data model design is a well-defined activity with an established set of general practices and methods that are used to control the systematic domain evaluation, parameter analysis and component design. One popular set of design steps starts with high level domain analysis that produces a conceptual data model. After the conceptual data model is developed, then data base rules are applied to transform the conceptual data model into a logical data model that can be supported using the standard record structures associated with the relational data base model. Figure 4.2-1 outlines the general development cycle. [97]

The generic systems engineering model is based on the activities associated with systems engineering as well as the meta-model outlined previously in this document. The conceptual model context will be very general, identifying primarily functional roles, general relationships, and generic processes associated with each of the systems in the context of interest. In this manner, the generic model will support specific, detailed system definition as the data and relationships are either discovered or designed.

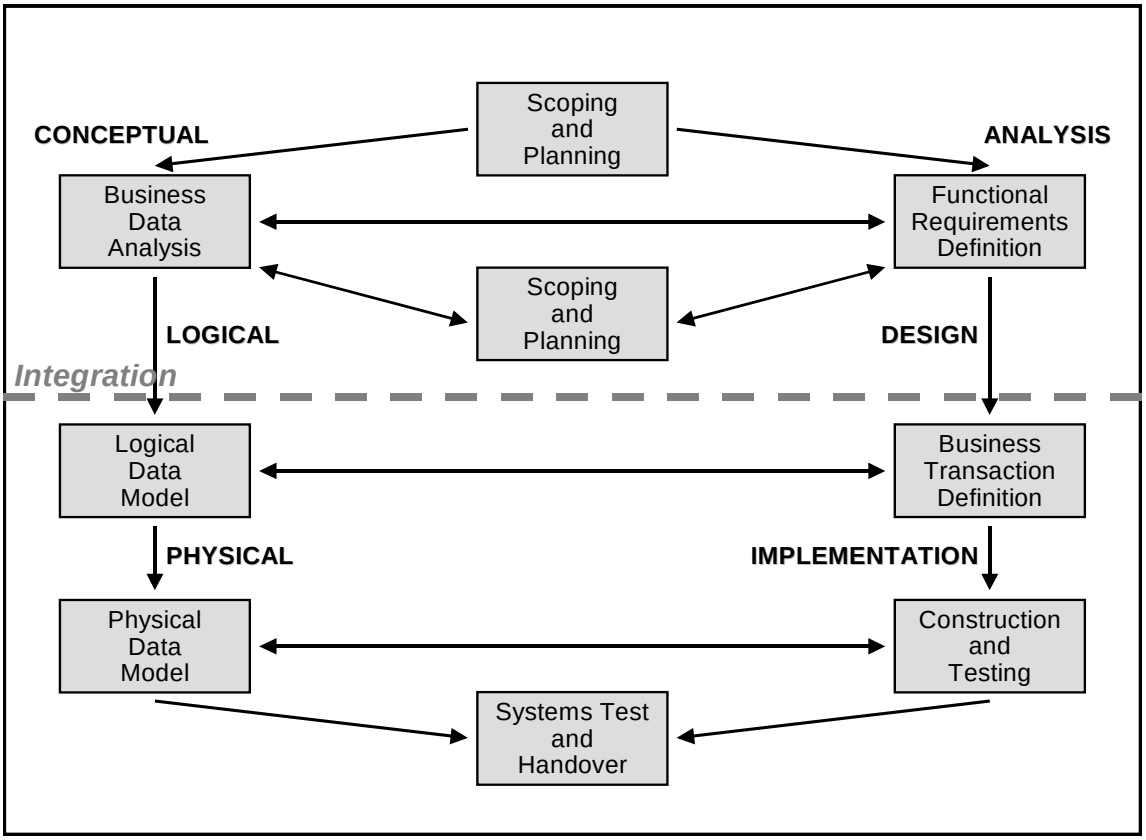

Figure 4.2-1 Database Systems Development Cycle

The generic systems engineering model approach will start with the system engineering meta-model as a base. This basic set of three systems and six views of each of these systems will be expanded to provide the information and data required to define the current system of interest to the required level of detail. Another design goal of the generic model is to create a loose coupling between individual systems and the various levels of detail associated with each specific system description. In other words, a specific system will be able to be described and modeled at various levels of system abstraction, decomposition or detail. A change of data or information in any one of the system models at any level of detail should not cause uncontrolled change or unacceptable amounts of change in the other views that represent the same system.

The generic or global system engineering model components consist of the environmental system, the product system and the process system as a minimum. Each of these systems is represented by six distinct system views that were detailed earlier in this document. During the conceptual model design phase these basic systems will be decomposed one or two levels to further quantify the system components and relationships. The global model components provide the high level global data model upon which the further data analysis and relational database design will be based.

4.3. CONCEPTUAL MODEL DESIGN

The standard process of conceptual model design that is associated with relational database development is a set of analytical steps that further refine the data and information associated with the systems engineering domain. The conceptual model will identify all entities within the systems engineering domain, relationships between these entities, clarify the entities degree and type as well as provide a complete definition of the entities attributes.

The base system entities are the environmental system, the product system and the process system. Each of these systems can be further decomposed in a “system of systems” type of hierarchal decomposition. However, for the purposes of this document some very general assumptions about the three base systems will be made to facilitate the further development of the systems engineering conceptual business model for a relational database application.

The environmental system will be represented as a series of five streams of change. These five streams are:

- Science stream or the development of new science

- Technology stream or the development of new technology
- Application stream or the development of new applications
- Product stream or the development of new products
- Organization stream or the development of new organizations [53]

These five fundamental streams of change are shown in Figure 4.3-1. The environment system is populated with the customer organization, subcontractor organizations, the organization that implements the system production process and other organizations. For our current purposes, it is assumed that these organizations are all large government or government contractor organizations with the necessary resources and capability to produce large scale engineered product systems.

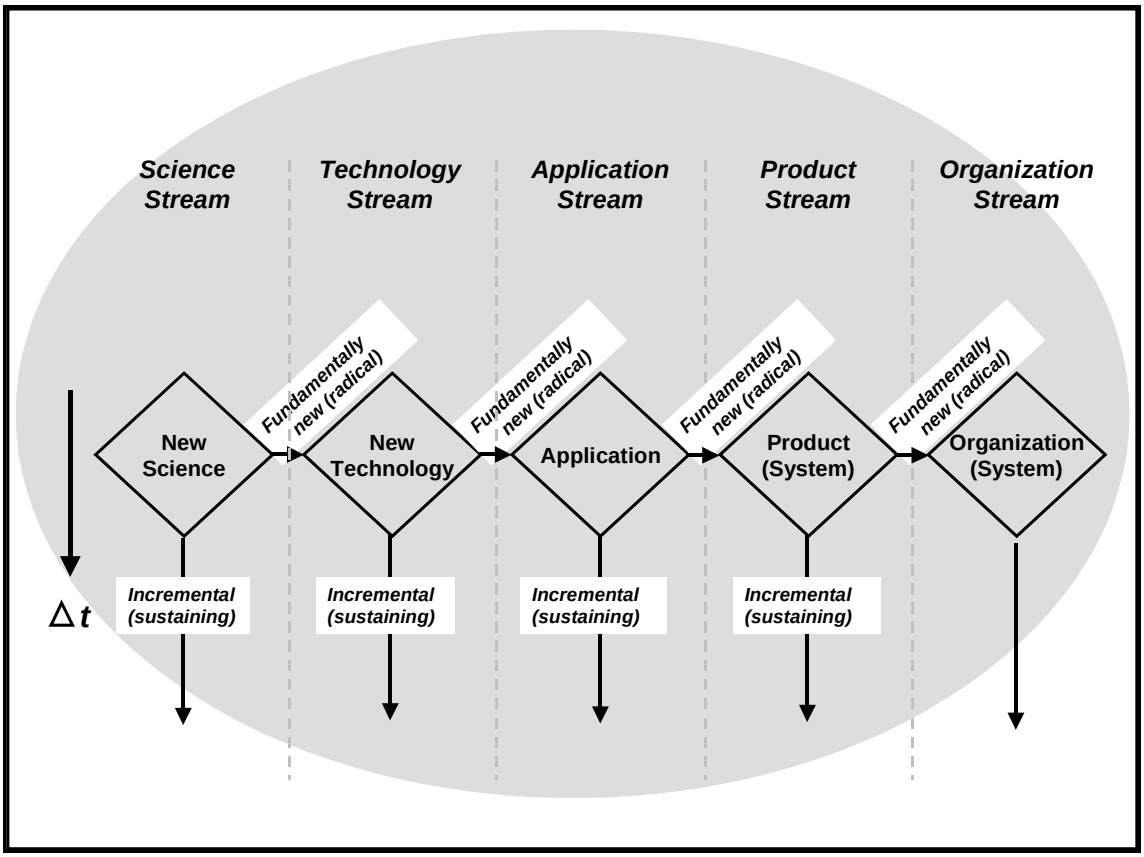

Figure 4.3-1 Streams of Change

The process system represents the organization and associated processes, methods and techniques that are used to produce the product system. This process system is viewed as a member of the organizational stream located in the environmental system. The process system is represented by three basic elements, organization structure, process structure, and an operational rule set. Any large-scale process system will have some type of organizational structure and the following organizational architecture, shown in Figure 4.3-2 is used for the current activity. [98]

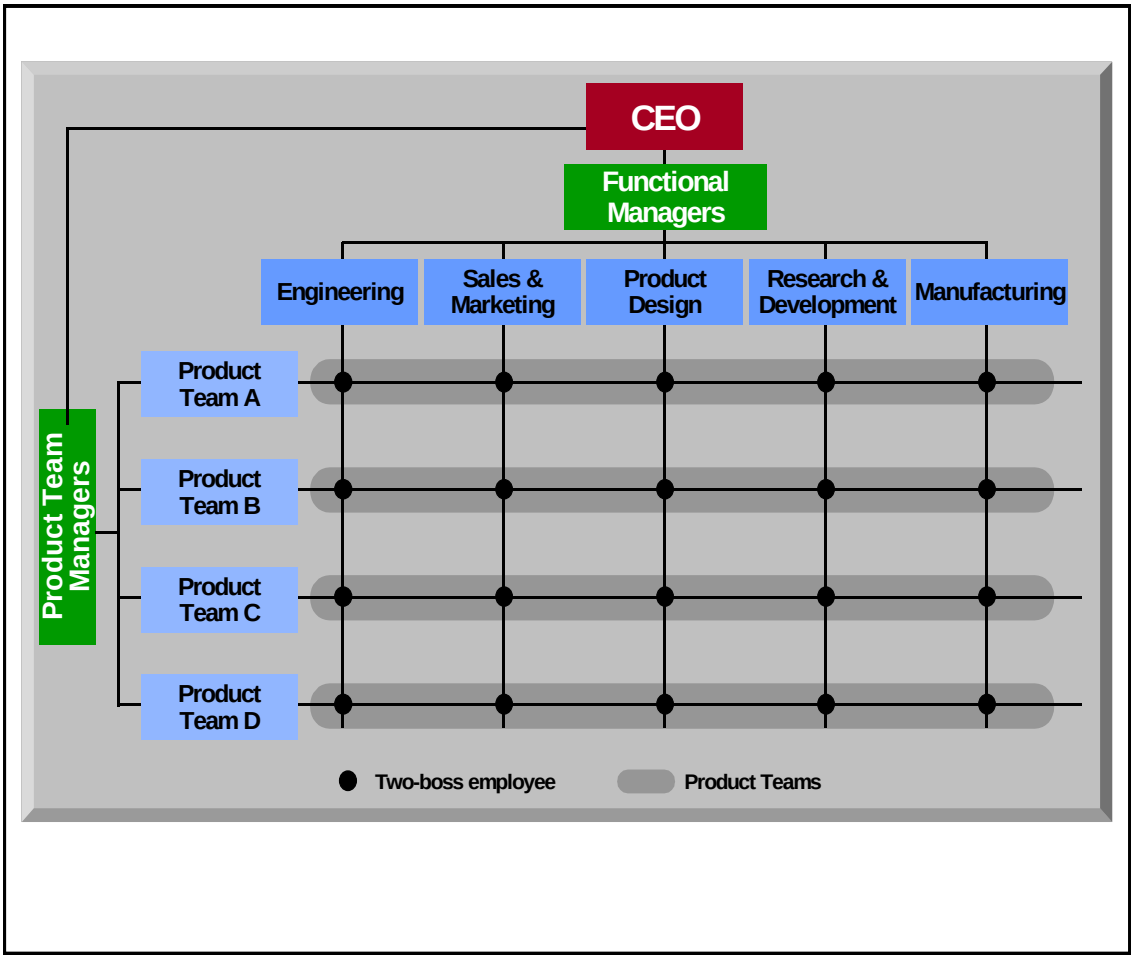

Figure 4.3-2 Organizational Architecture

In the Process System, the process structure will be represented by two general items or artifacts; the Integrated Master Plan (IMP) and the Integrated Master Schedule (IMS). The IMP is a list of process tasks arranged in a logical order that are linked in a

logical, functional network. The IMS includes all the IMP tasks; these tasks have been associated with time and calendar elements. The IMP has three basic types of elements associated with any given logical node in the planning network. These three elements are events, accomplishments and accomplishment criteria. For an event to be complete all accomplishments that are a part of that event must be completed. To determine if an individual accomplishment has been completed all of the accomplishment criteria must be evaluated. When all of the accomplishment criteria associated with a specific accomplishment are satisfied then that accomplishment is complete.

The Process System is completed by assigning an operational rule set to use in developing and deploying the Product System. In this case we will use the set of rules contained in EIA 632 as the governing Process System rule set.

The Product System considered here is described in very general terms, contains a high percentage of software, and is considered a part of the product stream associated with the environment system. The Department of Defense Architecture Framework (DODAF) and the Zachman Architecture Framework will be used as a general guides to determine the categories and types of information that will be associated with the Product System. The high level architecture of the product system will then be represented by a series of system models that are all based on the CCFRAT concepts. Product system specifications and requirements documents as well as other written data will be modeled as a standard program document tree, all of which is generated from the system database. Other patterns of system documentation and description may also be supported by the database depending on the controlling set of system concepts.

4.4. BASE SYSTEMS CONCEPTUAL MODELS

The conceptual data models of the base systems will now be developed. The conceptual data model is a high-level model and is the first step in the data analysis process. The purpose of data analysis is the determination of a data architecture that can be used to represent and store all of the data associated with any organization's activities of interest. The data analysis activity consists of three major tasks: identifying major "objects or things" (entities) with which data will be associated, determining the rules that are associated with the lifecycle or existence of these "objects or things" and how these things interact, and identifying the major macro processes associated with the interaction of the three base systems in the environment.

4.5. DATA ANALYSIS NOTATION

A modified version of the standard graphical notation for Entity Relationship (ER) diagrams will be used in this work. The standard unmodified notation supplies graphical notation for the following concepts:

- Entity
- Weak entity
- Relationship
- Attribute
    - Identifier (key)
    - Descriptor (non-key)
    - Multi-valued descriptor
    - Complex attribute

Figure 4.5-1 shows the standard graphic constructs for reference. [99] Entities, relationships and attributes also include the concepts of degree, connectivity and existence. The graphical notation for these concepts is shown in Figure 4.5-2. These

concepts are designed to support the development, analysis and modeling of complex data models.

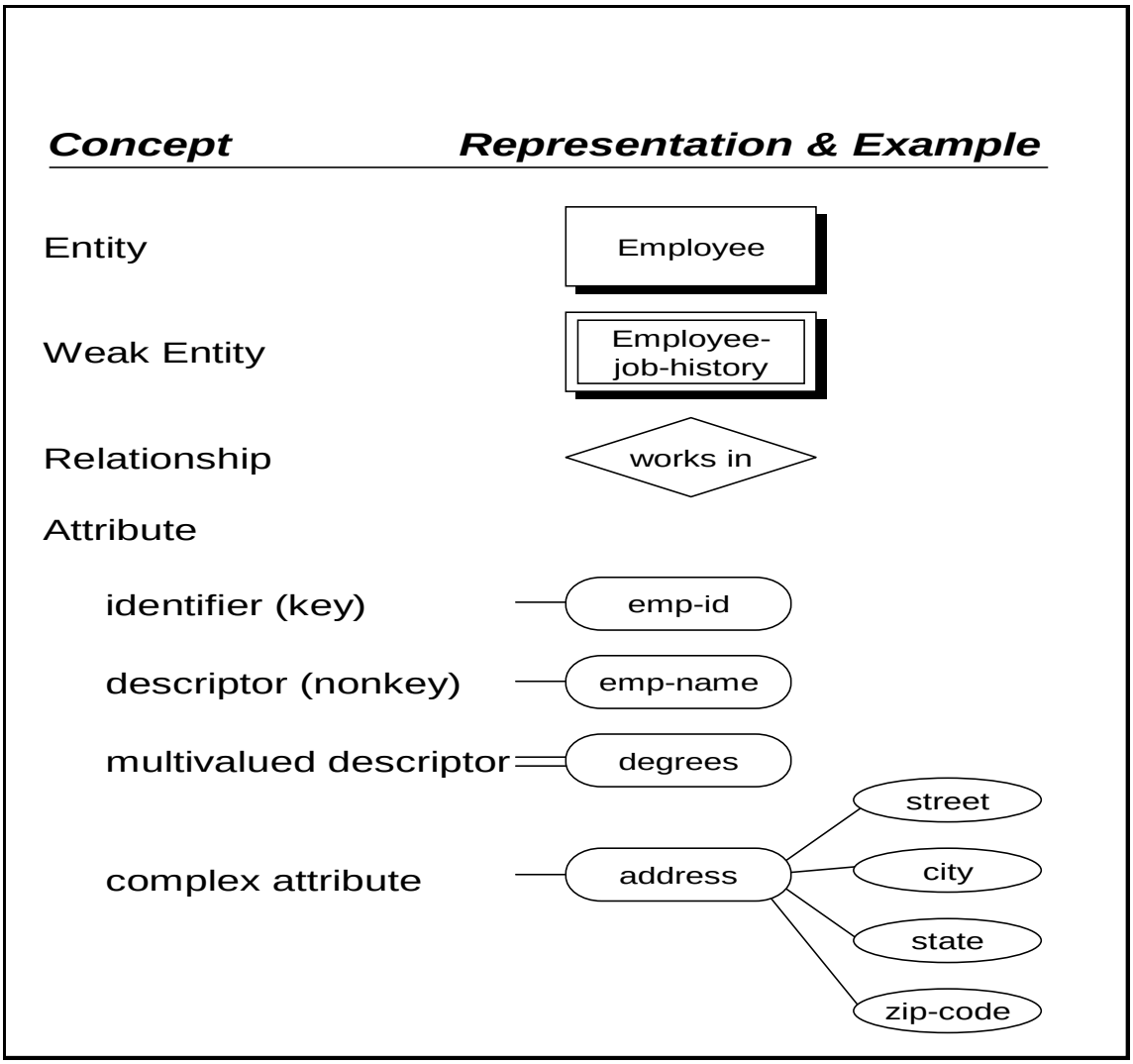

Figure 4.5-1 Basic Entity Relationship Constructs

This standard notation has been modified for use in this thesis. This modified notation uses only some of the graphical markings in a specific model diagram. This technique allows a specific aspect of the data model to be highlighted and reduces the number of symbols on the diagram. Only the most important aspects of the domain will

be detailed in the model. These aspects will be developed in further detail as the model becomes more fully developed.

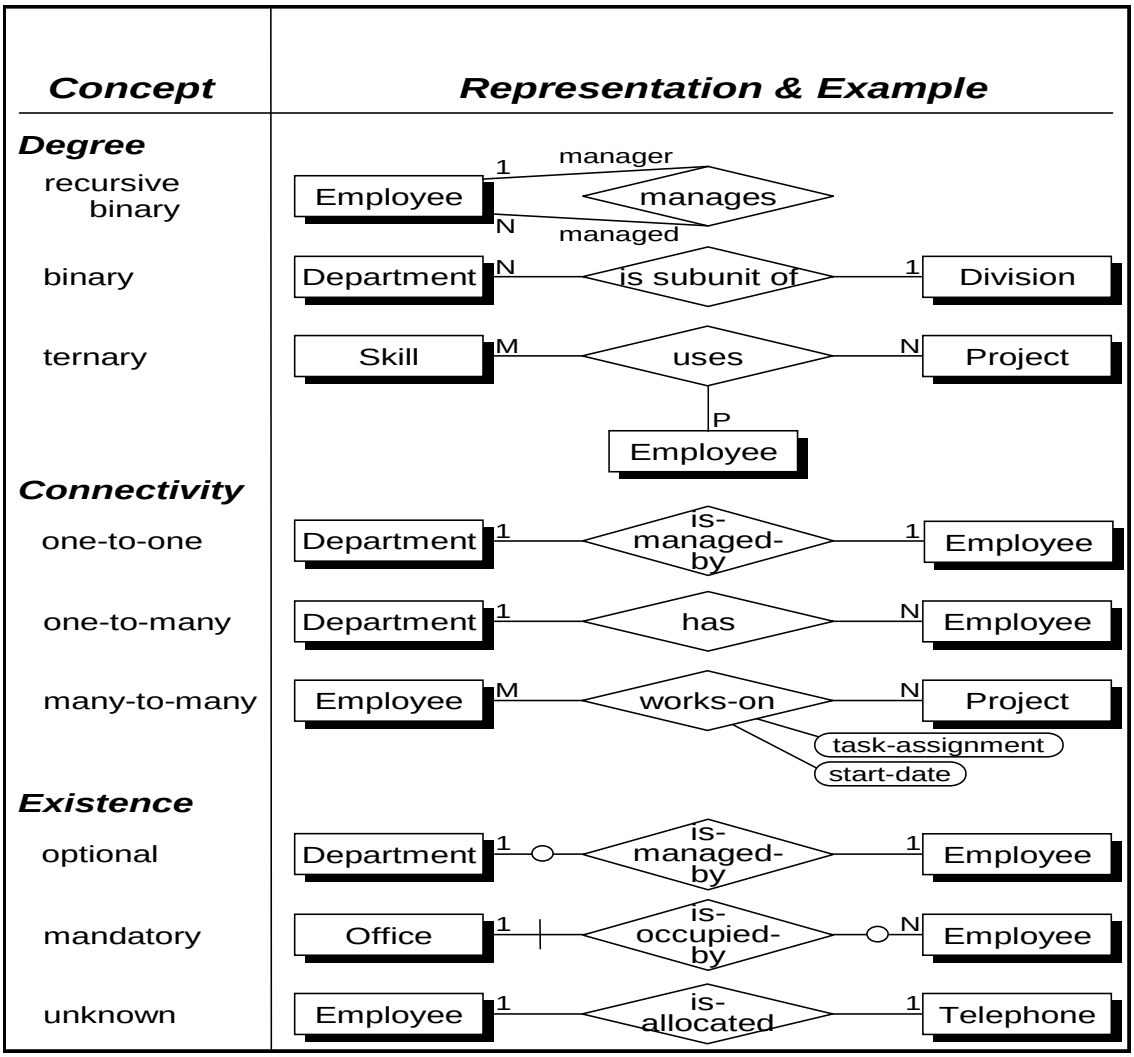

Figure 4.5-2 Entity Relationship Concepts

The three primary systems - environment, process and product - define the global context for the system engineering activity. Each of these primary systems will be modeled using the context, concept, functions, requirements, architecture and test views. These primary system views are then decomposed using a “top down” approach that

establishes a complete set of model views at each level of system abstraction or decomposition.

The driving organizational logic for establishing product model boundaries is comprised of the Process System IMP phases. Each major phase of the IMP logic is considered to produce a complete Product System at a given abstraction level. A complete system model is developed for that level of abstraction and detail. The Process System decomposition is matched to the Product System model levels of abstraction to provide a complementary set of "process and product" models for use during the design of the Product System.

The Environmental System contents are developed to the level of detail necessary to support the current state of the "process and product" model development. The Environmental System contains the system customers, competitors, regulations, sources of constraints, and many other factors and systems. It is important to develop the environmental material to the level of detail necessary to support the system development and modeling tasks. The rationale supporting the level of detail in the current development activity is contained in the concept view of the system description.

Once the first set of primary system models have been developed, the systems engineering process proceeds in a recursive, stepwise fashion to design, model and develop systems at lower and lower levels of detail until the system has been fully described and modeled. After the design, modeling, and component production is complete, the product assembly and test starts. Each lower level system component is integrated and tested, starting with the lowest level components being assembled and tested. Integration and test continues until the total system is assembled and tested.

4.6. SYSTEM CONTEXT DATA MODEL

The system context view is focused on the external relationships and connections associated with the system of interest. This view records the driving need for the system as well as the system value network, external stakeholders and controlling value sets. Figure 4.6-1 presents the conceptual data model for the system context view, from high level needs statements to detailed design activities. The context view becomes more detailed and specific as the design process matures from high-level needs statements to detailed design activities.

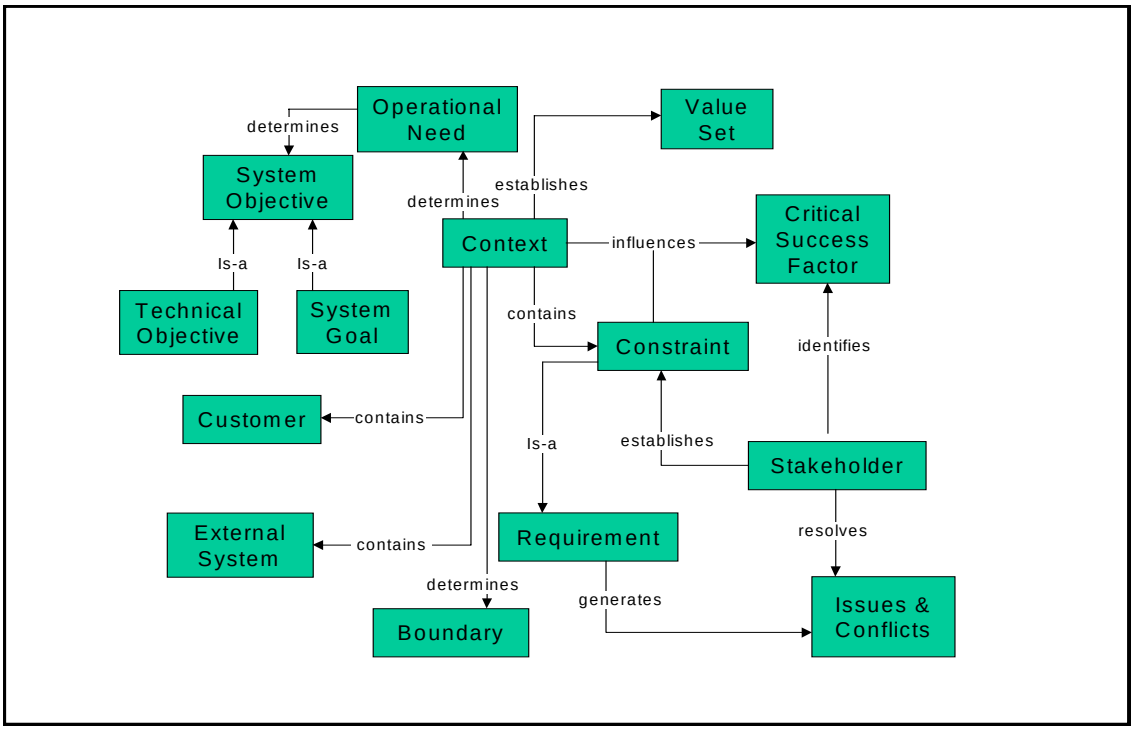

Figure 4.6-1 System Context View Model

4.7. SYSTEM CONCEPT VIEW DATA MODEL

The system concept view is focused on the system of interest, its current level of abstraction, level of decomposition and abstraction boundary for this specific set of

system concepts. The core four system views, function view, requirements view, architecture view and test view can be evaluated, analyzed, documented and modeled in a number of ways. The concept view establishes a well-coordinated and executable set of rules for system definition and analysis. The concept view details the semantics, processes, and approach used to evaluate and analyze the system represented in the current context. Figure 4.7-1 shows the System Concept View Data Model.

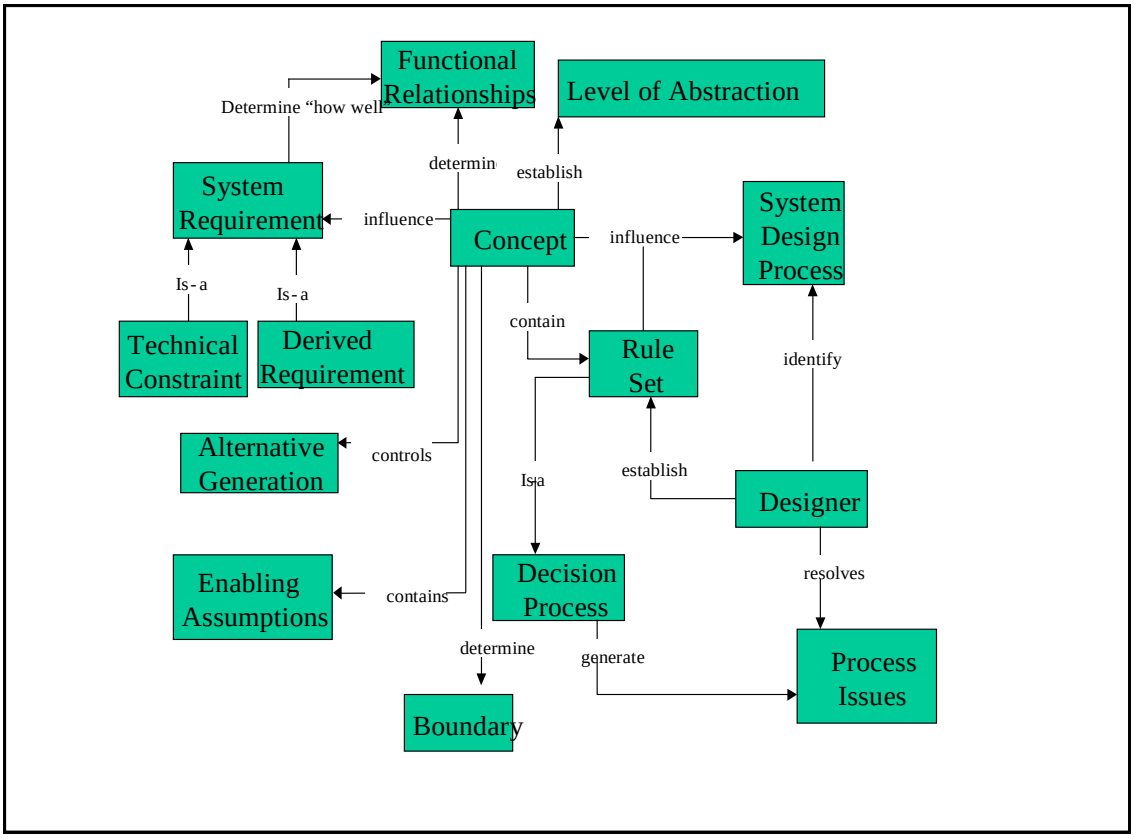

Figure 4.7-1 System Concept View Data Model

4.8. SYSTEM FUNCTIONAL VIEW DATA MODEL

The system functional view details the required system behavior at the current level of system abstraction. The operational approach to define and model the system

behavior is found in the rule set contained in the system concept view. The primary system functions and the functional alternative analysis are covered in this system view, shown in Figure 4.8-1. This system view provides for a structured functional development from “higher level system” requirement to the current system functional need. Functional modeling and design documentation are also covered in this part of the data model. This data model supports the generation of system design data for both text descriptions and executable model inputs. The specific application of the systems functional design data is covered in the concept view associated with the current functional view.

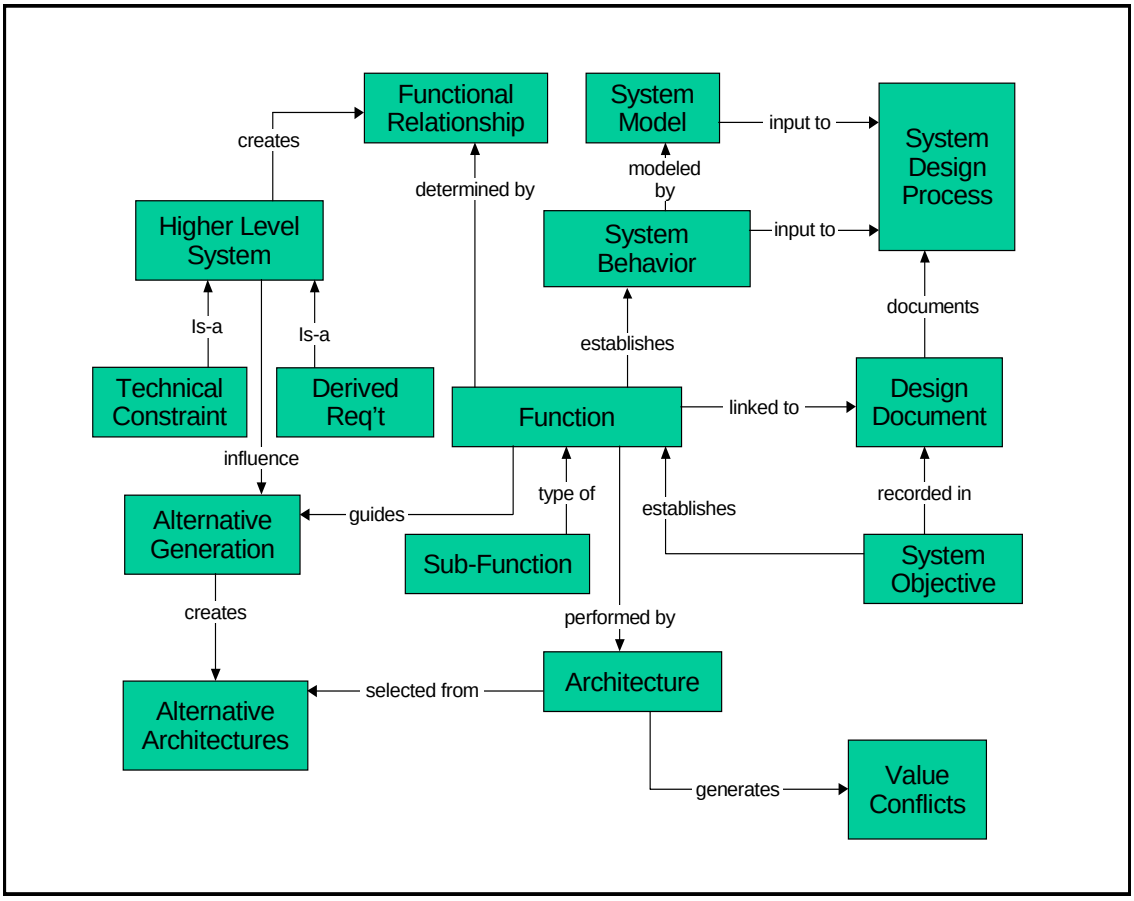

Figure 4.8-1 System Functional View Data Model

4.9. SYSTEM REQUIREMENT VIEW DATA MODEL

The system requirement view has a direct relationship to at least one system functional view. The system concept view will outline how the function view and the requirement view will relate and interact. In general the requirement view determines how well the system function must be performed. This creates a direct logical relationship between the function view and the requirement view. Further, a requirement must be under the design control of the design engineer, so the design engineer can trade-off any specific requirement against other functions and requirements to balance the system design. If the requirement cannot be traded-off against other requirements it is labeled as a constraint and treated as a fixed value that must be achieved. Process and design risk are also considered a type of system requirement. System functions and objectives are traded-off against risk in the requirements management and system design process.

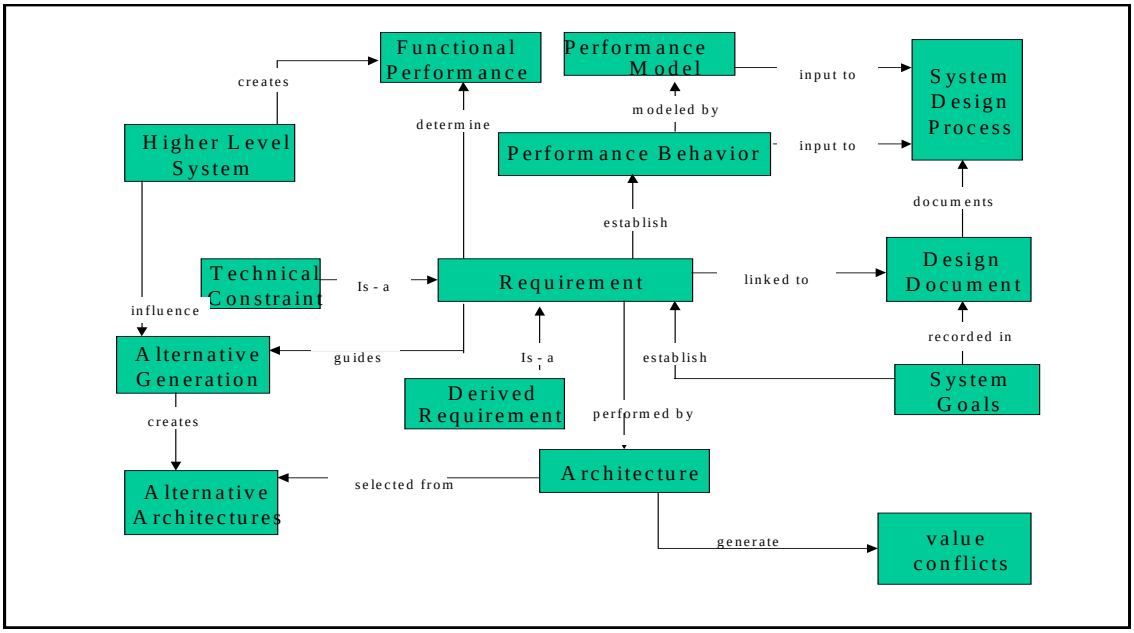

Figure 4.9-1 System Requirement View Data Model

4.10. SYSTEM ARCHITECTURE DATA MODEL

The system architecture view details the selected system architecture and all candidate architectures that were developed for consideration using the process and approach outlined in the system concept view. The selected architecture is directly linked to the system design documents and dynamic system models. These design artifacts help the system designer track the system architecture design process in a structured and coherent fashion. The system decomposition hierarchy that is documented in the higher level context and concept views of the product system and the process system are the decomposition patterns that are used to complete the system design process. In general these patterns can change but each specific change must be documented in the proper system context and concept view.

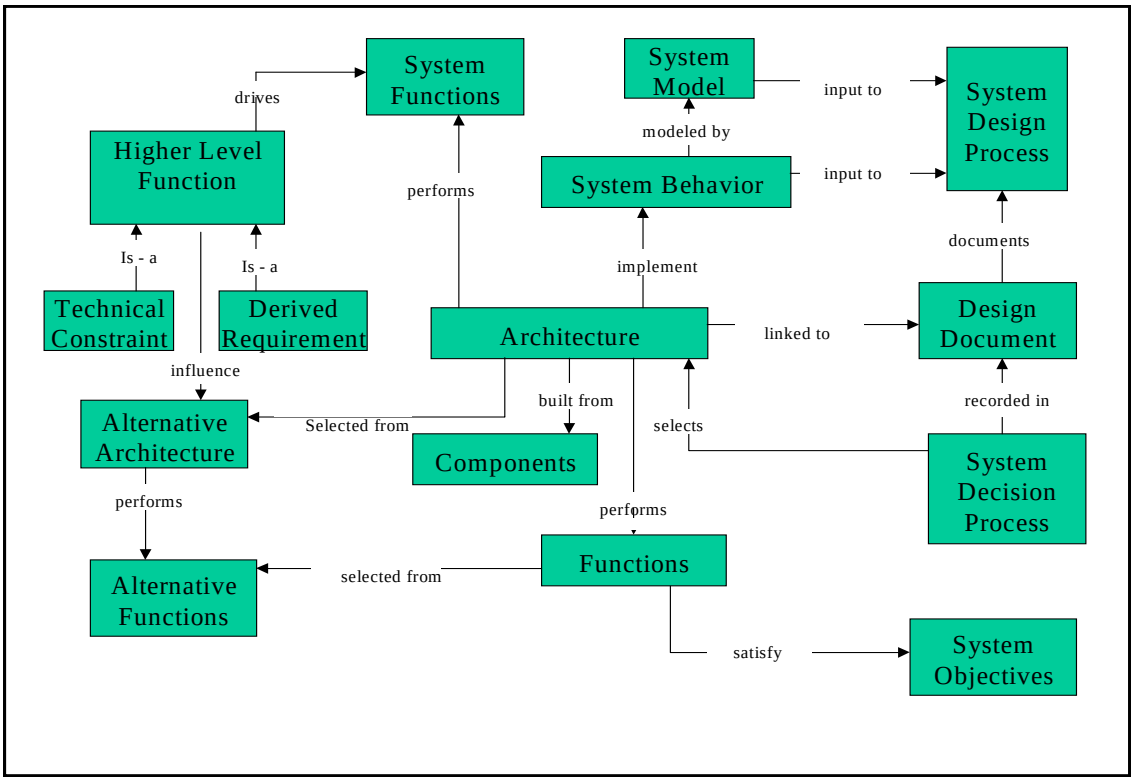

Figure 4.10-1 System Architecture View Data Model

A primary goal of this data model is the documentation of these “architectural design patterns” in a global and structured manner. The CCFRAT approach provides the conceptual mechanisms necessary to capture many types of systems engineering design patterns in a reusable fashion.

4.11. SYSTEM TEST VIEW DATA MODEL

The system test view is established, modeled and documented in increasing levels of detail just like the other system views. The system test view , shown in Figure 4.11-1, details the process activities necessary to determine that the selected system architecture will perform the required functions to the specified performance levels within an acceptable risk margin.

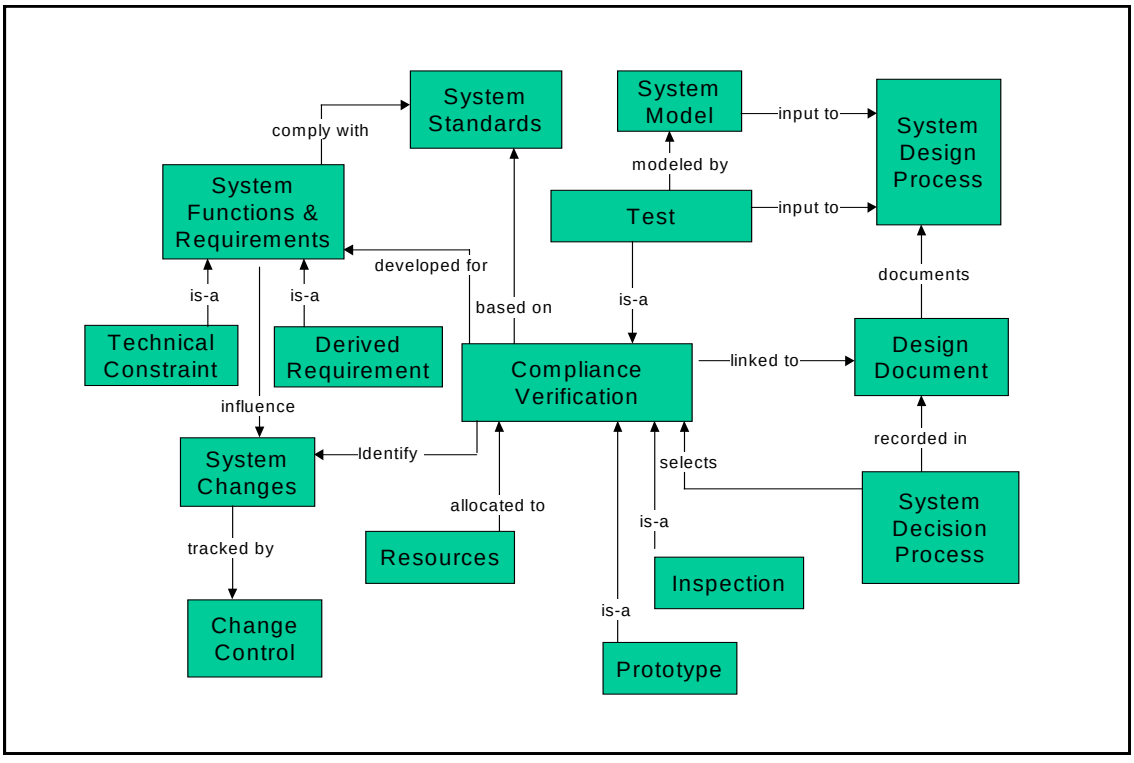

Figure 4.11-1 System Test View Data Model

The test view is a very import view that may drive many system design considerations. For example all functions and requirements must be measurable and testable. If these aspects cannot be tested then they should not be tracked in the design process.

5. LOGICAL DATA MODEL

5.1. LOGICAL MODELING PROCESS

Once the high level conceptual data modeling is complete, the data modeling process continues with the development of logical data models that will be the basis for the construction of relational database tables. During the data development of the relational database tables the data models are normalized to support efficient data storage and recall. When complete the logical data model will be in a third normal form.

The logical data modeling process starts by identifying all of the entities from the conceptual data models and establishing relationships between the entities as well as the relationship degree, type and attributes. The logical data modeling process is completed by creating a relational data model in third normal form, creating a complete definition of logical record structures, and supporting all relationships with the proper foreign keys.

The logical models will be organized around the environmental system, the process system and the product system contained in the CCFRAT approach. In general, the environmental system contains everything of interest to the current system design problem. The process system is contained in the environmental system and is the system that produces the product system. The product system is designed and built by the process system. After the product system is complete it is operated by the customer system in the environment system. Each of the basic CCFRAT systems will be analyzed and a basic logical entity relationship (ER) model will be designed for each base system. Design decisions, tests, functions and requirements as well as other entities found in the systems engineering process will be developed to support the complete process.

5.2. ENVIRONMENTAL SYSTEM LOGICAL MODEL ER DIAGRAM

The environmental system logical model ER diagram is shown in Figure 5.2-1. The environmental system is the macro context for the current system design activity. Because there is nothing outside of the environmental system, the environmental system becomes the context view in this model. The environmental system contains all other systems and resources available for system design. The key entities and the degree of their relationships are outlined to further communicate the organization of the CCFRAT information model.

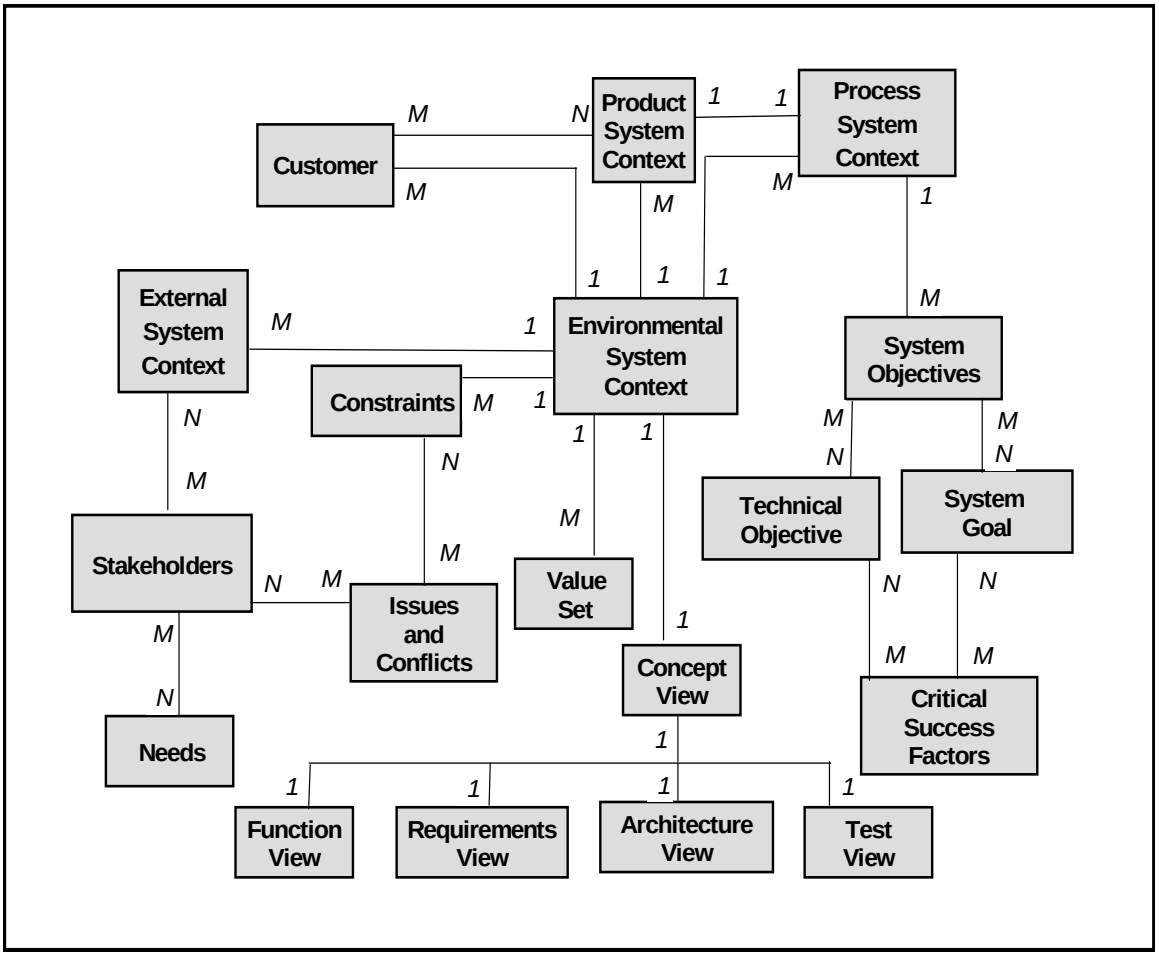

Figure 5.2-1 Environmental System ER Model

5.3. PRODUCT SYSTEM LOGICAL MODEL ER DIAGRAM

The product system ER model is shown in Figure 5.3-1. The product system model contains a context view that details the product system external relationships and connections. The product ER model indicates that only one product system is being considered in a product system model. The environmental ER model could contain many product systems, process systems and external systems. However, the product model is constrained to one product system only. If other systems from the environmental system are incorporated into the product system then these systems will be recorded in the architecture view and incorporated in a manner described in the concept view.

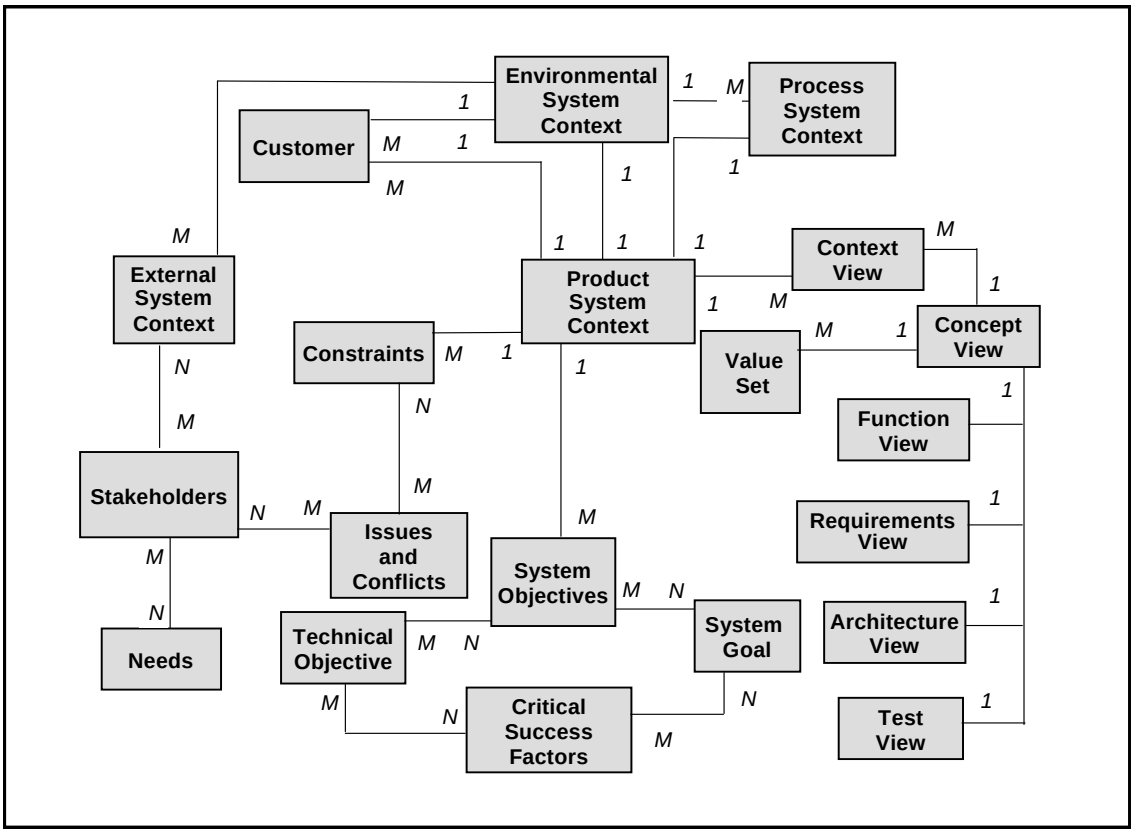

Figure 5.3-1 Product System ER Model

5.4. PROCESS SYSTEM LOGICAL MODEL ER DIAGRAM

The process system ER model is shown in Figure 5.4-1. The process system model contains context and concept views. The concept view is the most important view in the process model because it is directly tied to the design and decision process. However, the context view is also important because it is one of the primary connections between the product system and the process that is creating the product. Design phases and value sets are also key entities in the process system ER model.

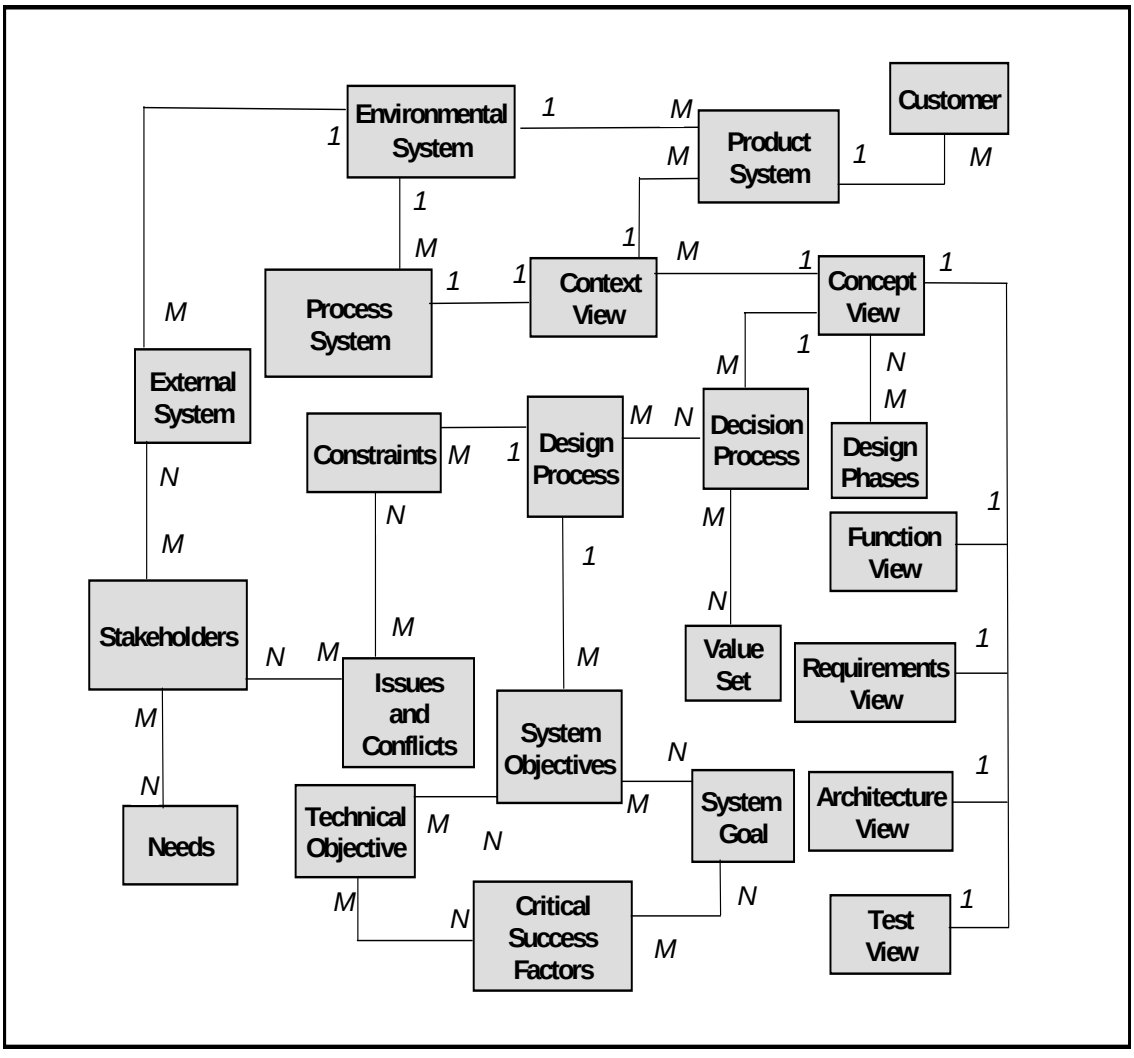

Figure 5.4-1 Process System ER Model

5.5. LOGICAL MODEL RELATION TABLE DEVELOPMENT

Database tables for all initial entities are presented next. The project database will contain, as a minimum, the following main database tables:

- System table
- Context table
- Concept table
- Function table
- Requirements table
- Architecture table
- Test table
- Decision table
- Type table
- Document table
- Model table
- Database model table
- Fr_link_table
- Fa_link_table
- At_link_table

More tables can be added to support a specific type of system or activity. The detailed composition of these tables is presented next.

The system table contains a row for each system in the model. The minimum number of system table entries would be three, one for the environment system, one for the process system and one for the product system. A data store for a real project would

have ten's if not hundreds of entries in the system table. Figure 5.5-1 presents the system ER diagram. The system database table consists of the following attributes:

- System_id – serial – int4
- System_name – char - 64
- System_acronym – char – 64
- System_description – text
- System_type – type_id (FK)
- System_context – context_id (FK)
- System_concept – concept_id (FK)
- System_model – model_id (FK)
- System_db_model –db_model_id (FK)
- System_decision – decision_id (FK)
- System_document – document_id (FK)

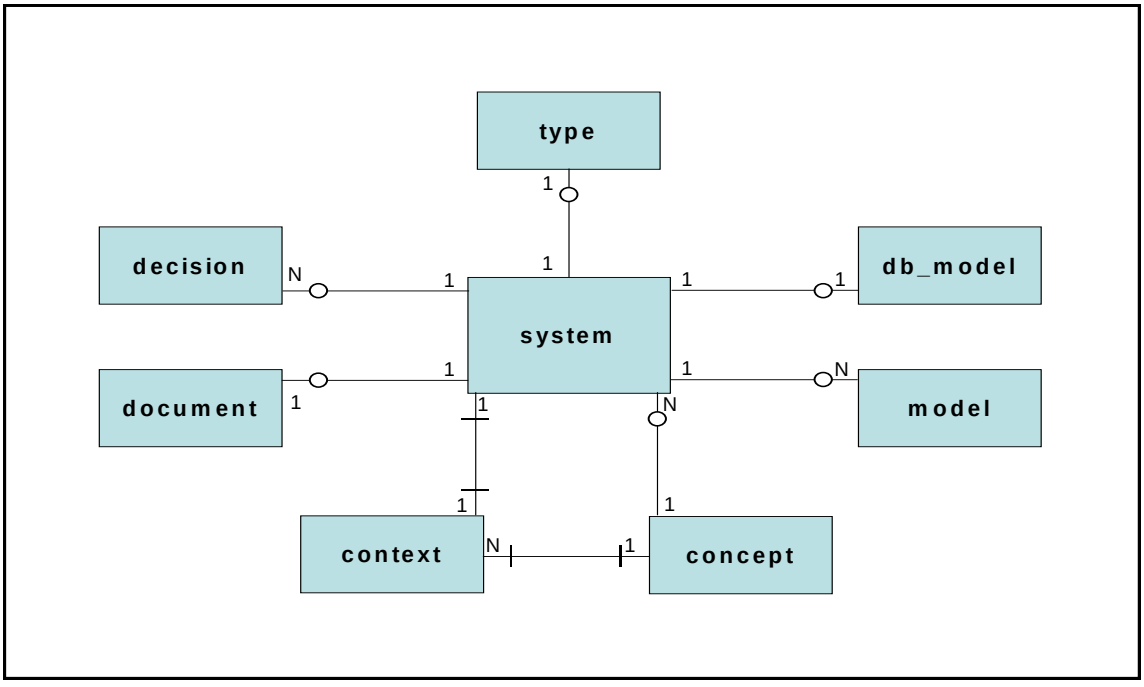

Figure 5.5-1 System Logical ER Model

The context table contains a row for each system context view in the system model. The context view is focused on the outward looking view from the system of interest and details the important connections and interactions in the system environment. Systems can reside inside of other systems, so, there are three general types of connections that can be made by starting at the context boundary and traveling outward. These general connection types are shown in Figure 5.5-2. The first general type of outward connection is a "context A to context B" connection. In this first case, both systems (A and B) reside in the same context plane at the same level of abstraction. The second general type of outward connection is a "context A to concept C" connection. In this second general connection type system A resides inside system B. The third general connection type is a connection inward from the context boundary to the systems own concept boundary.

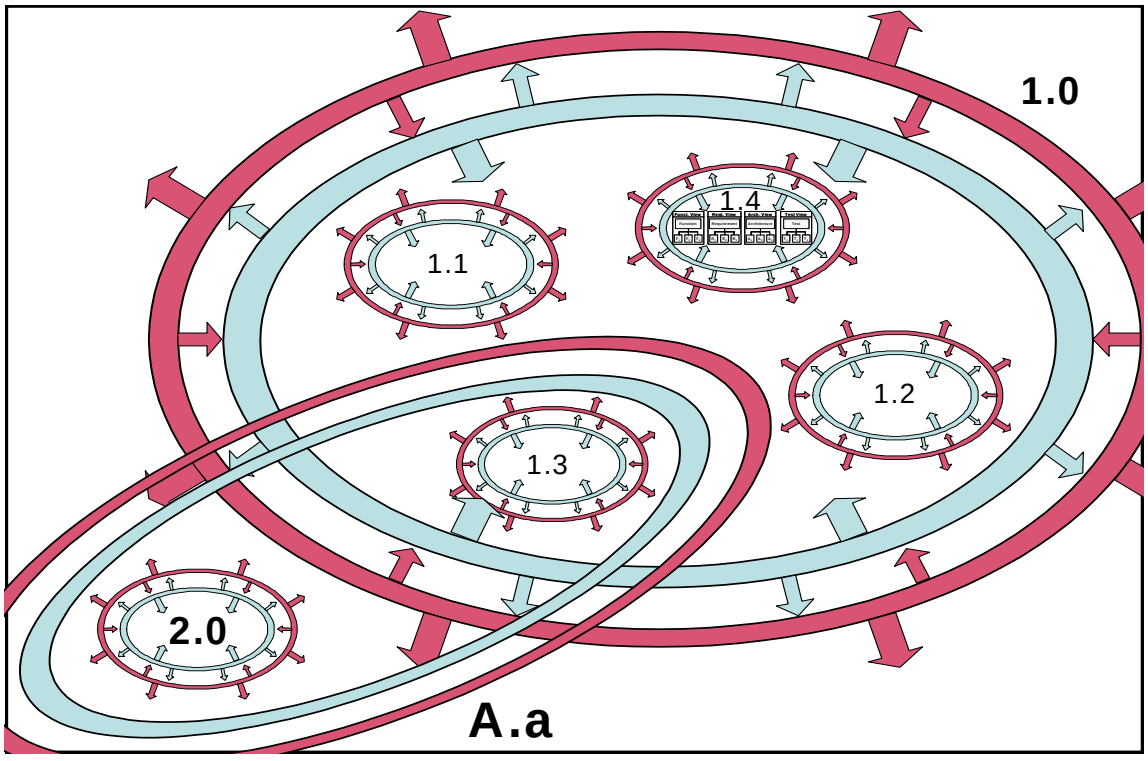

Figure 5.5-2 General Connection Types

This third connection type is a inward connection from “context A to concept A.” These general connection types are shown in Figure 5.5-2. Standard hierarchal system decompositions can be described as shown with system 1.0 and system 2.0 in Figure 5.5-2 or more networked types of system can be represented by system A.a. The context database table consists of the following attributes:

- Context_id – serial – int4
- Context_name – char – 64
- Context_description – text
- Context_type – type_id (FK)
- Concept_connection – concept_id (FK)
- Context_model – model_id (FK)
- Context_db_model – db_model_id (FK)
- Context_document – document_id (FK)

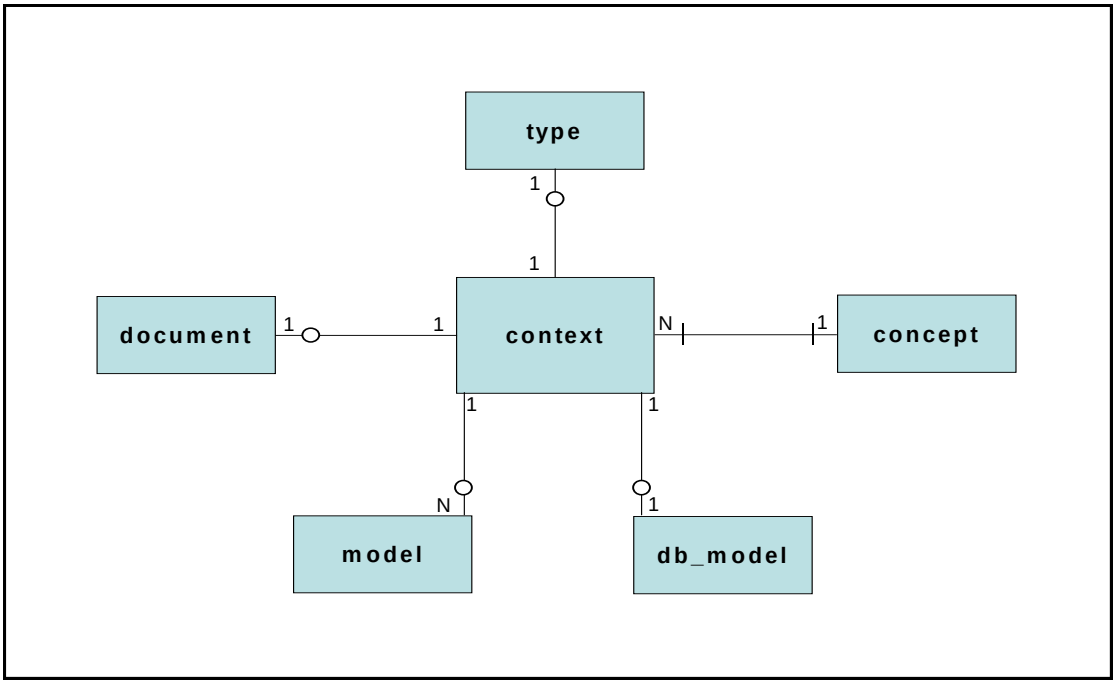

Figure 5.5-3 Context Logical ER Model

The concept table contains one row for each system concept view represented in the model. The concept view is focused on an inward view of the system and details the concepts associated with entities that make up the interior system structure. The concept view can be of two general types; one type that contains other system context views only and another type that contains function, requirement, architecture and test views only. These relation ships are shown in Figure 5.5-2. The concept logical ER model is shown in Figure 5.5-4.

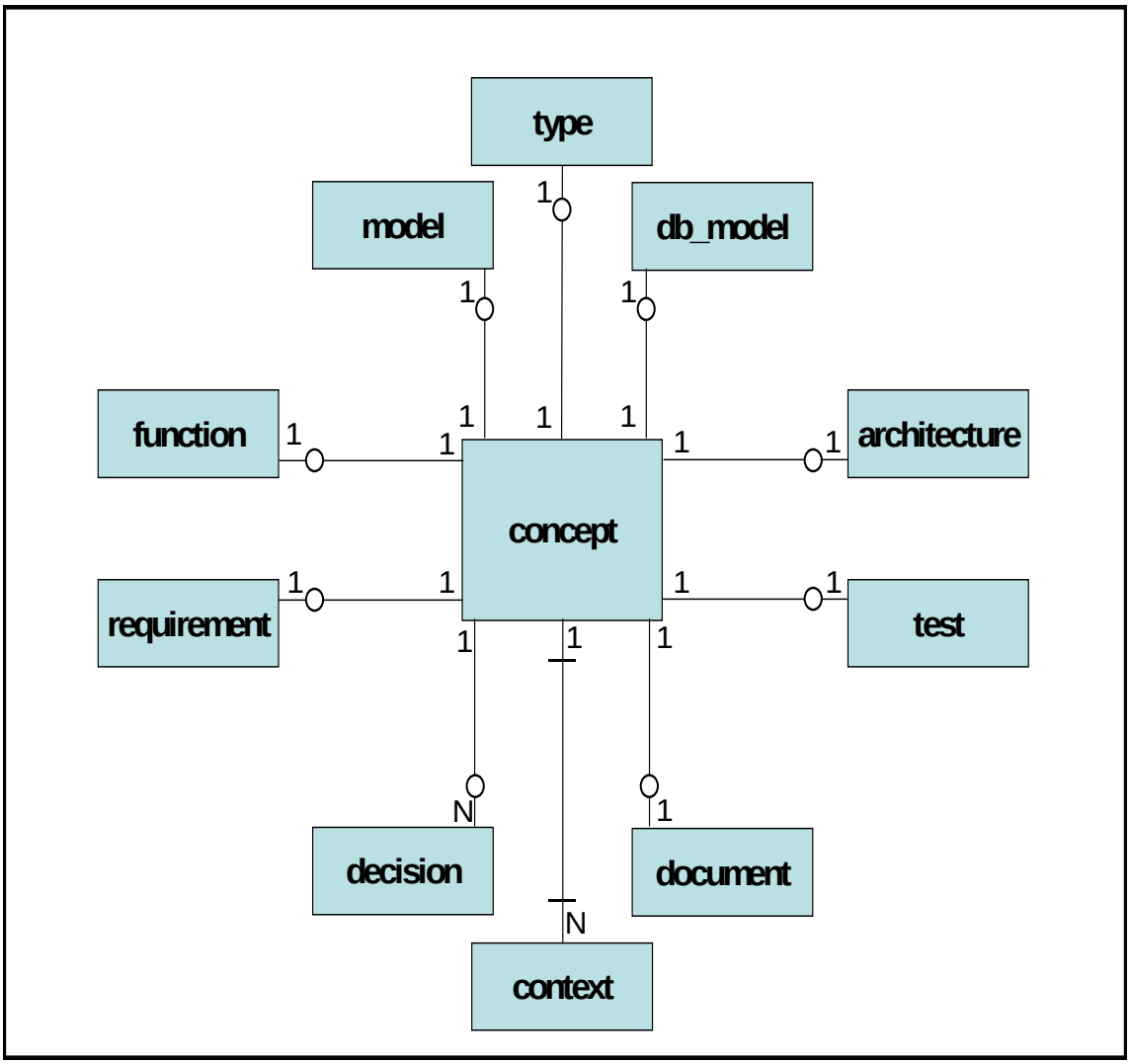

Figure 5.5-4 Concept Logical ER Model

The concept database table consists of the following attributes:

- Concept_id – serial – int4
- Concept_name – char – 64
- Concept_description – text
- Concept_model – model_id (FK)
- Concept_db_model – db_model_id (FK)
- Concept_type – type_id (FK)
- Context_connection – context_id (FK)
- Function_connection – function_id (FK)
- Requirement_connection – requirement_id (FK)
- Architecture_connection –architecture_id (FK)
- Test_connection – test_id (FK)
- Concept_decision – decision_id (FK)
- Concept_document – document_id (FK)

The function table contains one row for each system function view represented in the system model. The system functions will be determined and modeled in the manner described in the associated concept view. The function ER logical model is shown in Figure 5.5-6. The function database table consists of the following attributes:

- Function_id – serial – int4
- Function_name –char – 64
- Function_description – text
- Function_type – type_id (FK)
- Concept_connection – concept_id (FK)

- Fr_link – fr_id (FK)
- Fa_link – fa_id (FK)
- Function_model – model_id (FK)
- Function_db_model – db_model_id (FK)
- Function_decision – decision_id (FK)
- Function_document – document_id (FK)

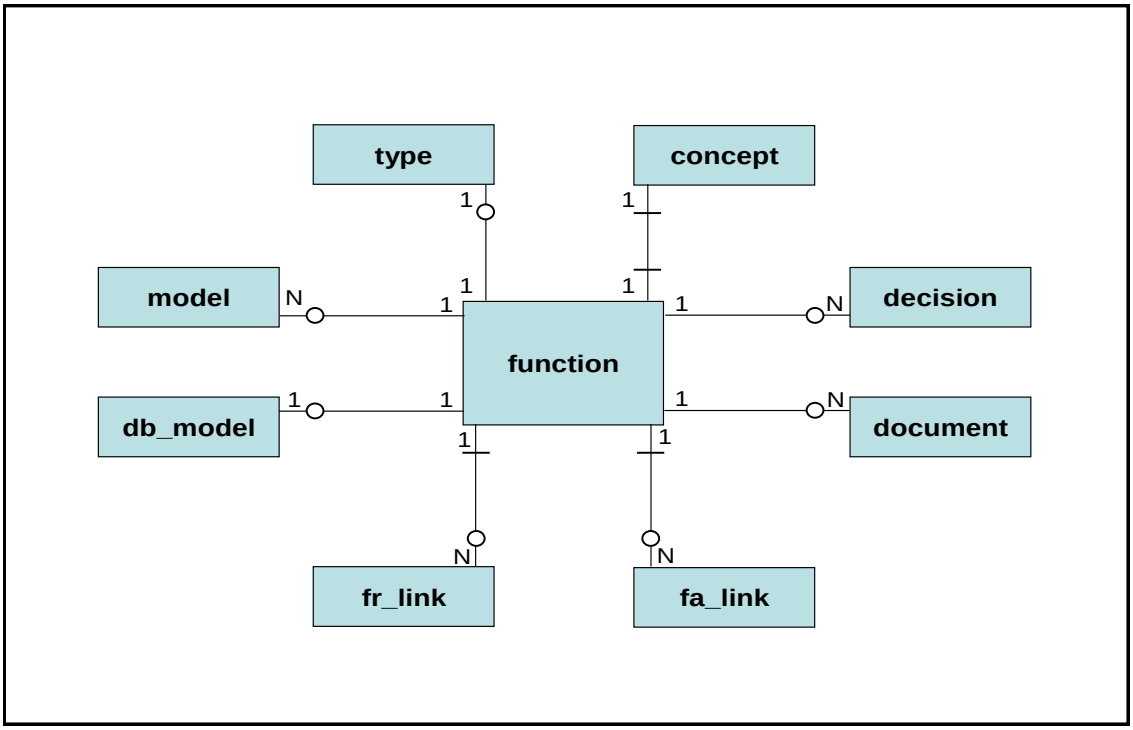

Figure 5.5-6 Function Logical ER Model

The requirement table contains one row for each system requirement view. Each requirement is directly connected to one or more function statements to create a basic problem statement. The requirement logical ER model is shown in Figure 5.5-7. The requirement table consists of the following attributes;

- Requirement_id – serial – int4
- Requirement_name – char -64

- Requirement_description – text
- Requirement_type – type_id (FK)
- Fr_link – fr_id (FK)
- Concept_connection – concept_id (FK)
- Requirement_model – model_id (FK)
- Requirement_db_model – db_model_id (FK)
- Requirement_decision – decision_id (FK)
- Requirement_document – document_id (FK)

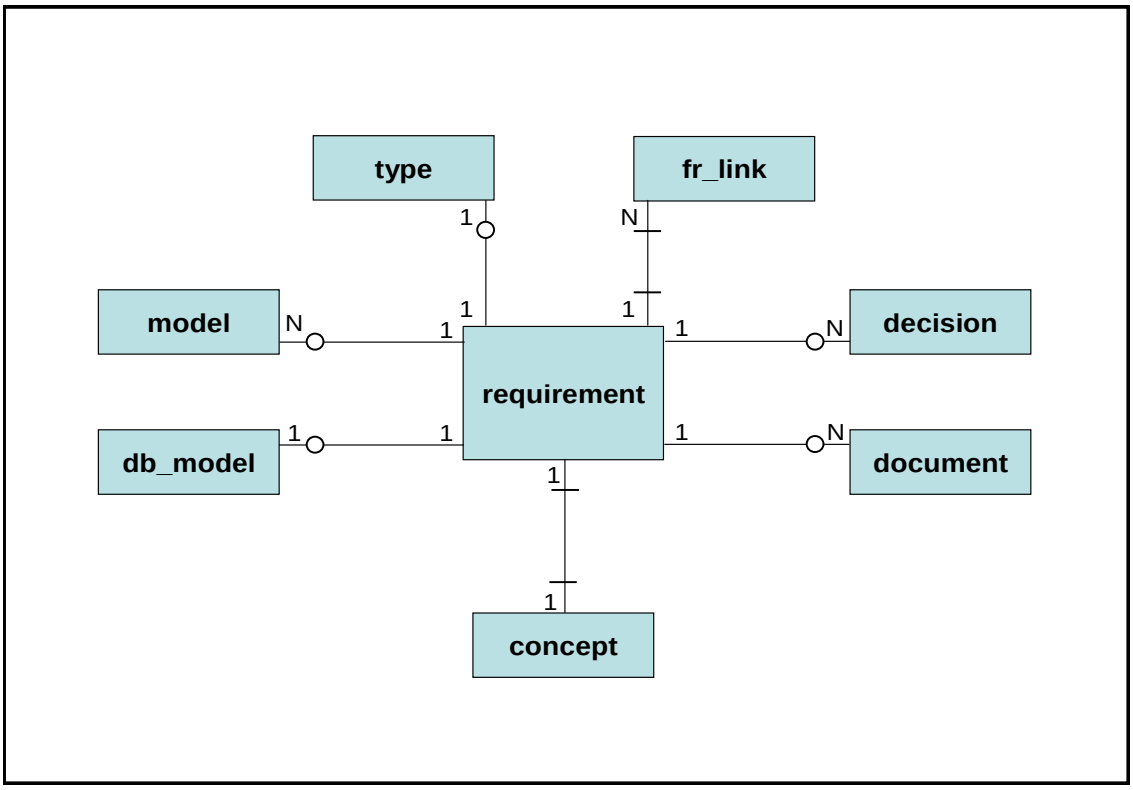

Figure 5.5-7 Requirement Logical ER Model

The architecture table contains one row for each candidate architecture solution that was considered for inclusion into the system solution. The architecture is the

system solution that answers the “function and requirement” problem statement. The architecture database table consists of the following attributes;

- Architecture_id – serial – int4
- Architecture_name – char – 64
- Architecture_description – text
- Architecture_type – type_id (FK)
- Concept_connection – concept_id (FK)
- At_link – at_id (FK)
- Fa_link – fa_id (FK)
- Architecture_model – model_id (FK)
- Architecture_db_model – db_model_id (FK)
- Architecture_decision – decision_id (FK)
- Architecture_document – document_id (FK)

The test table contains one row for each test view that is represented in the model. The test view focuses on recording the tests and procedures that are used to assure that the selected architecture will perform the required function as well as the requirement states. The test database table consists of the following attributes;

- Test_id – serial – int4
- Test_name – char – 64
- Test_description – text
- Test_type – type_id (FK)
- Test_model – model_id (FK)
- Test_db_model – db-model_id (FK)

- Concept_connection – concept_id (FK)
- Test_decision – decision_id (FK)
- Test_document – document_id (FK)

The decision database table contains one row for each decision made during the system development. Decisions and decision types will range from the recording of expert judgment about a subject to fully developed trade studies involving many people and experts. The decision database table consists of the following attributes;

- Decision_id – serial – int4
- Decision_name – char – 64
- Decision_description – text
- Decision_outcome – text
- Decision_document – document_id (FK)
- Decision_model – model _id (FK)
- Decision_type – type_id (FK)
- Decision_db_model – db_modle_id (FK)

The type table contains a row for every type of classifier used in the system model. These classifiers or types are used for grouping aspects of the system concepts in convenient workable units. The type table contains the following attributes:

- Type_id – serial – int4
- Type_name – char – 64
- Type_description - text

The document table contains a row for every document or specific document fragment developed during the system development activity. The document table contains the following attributes:

- Document_id – serial – int4
- Document_name – char – 64
- Document_content – text
- Document_author – char – 64
- Document_link – document_id (FK)
- Document_link_1 – document_id (FK)

The model table contains one row for each model used in the system development process. The model table contains the following attributes:

- Model_id – serial – int4
- Model_name – char – 64
- Model_description – text
- Model_executable - object

The database table contains one row for each database model used in the system development process. The database model is used to store database scripts, functions and other database specific models used in the system development process. The database table contains the following attributes:

- Db_modle_id – serial – int4
- Db_model_name – char – 64
- Db_model_description – text

The preceding twelve database tables are used as the core basis of the systems engineering information model presented here. A set of link record tables will be developed next to facilitate the "many to many" relationships found in the model. A link record will break a "many to many" relationship between two tables into two "one to many" relationships between three tables. The link record table is the third table that is designed for this purpose.

The core function, requirements, architecture and test are the tables that have the "many to many" relationships. These "many to many" relationships are: function table to requirement table, function table to architecture table and architecture table to test table. The following link record tables are used to create the required "one to many" relationships: fr_link_table, fa_link_table, at_link_table. The fr_link_table has the following attributes:

- Fr_id – serial – int4
- Function_id (FK)
- Requirement_id (FK)

The fa_link_table has the following attributes:

- Fa_id – serial – int4
- Function_id (FK)
- Architecture_id (FK)

The at_link_table has the following attributes:

- At_id – serial – int4
- Architecture_id (FK)
- Test_id (FK)

5.6. MODEL APPLICATION

Large distributed systems development projects are becoming the norm in many areas of industry. The increasing technical complexity of our products coupled with a large portion of system development being outsourced to other companies and other countries creates a situation where there is a great need to apply structured systems engineering practices to control system design and development. The systems engineering information model developed in this thesis lays the foundation for the development of standard relational database models to support distributed engineering management applications in a networked environment. A key feature of the database models is adaptability. Systems engineering and management tasks have their own specialized computer based models and systems that are already deployed in many organizations. The system model presented here allows for the identification of these other information resources and their management by either associating them with a specific system support model or handling them as a complete system.

Database connections to configuration management, risk management, earned value, process management, and project management computer-based tools provide the capability to manage all aspects of a project from one platform. The internal system logical structure provides the capability to create new systems, system interfaces and new system views without impacting the existing data structure or database in an adverse manner.

The systems engineering information model developed in this work places a strong emphasis on modeling and tracking the incremental development of customer requirements and needs. The complete system development lifecycle is modeled in a

manner that enhances technical communication by establishing a core set of system models that can be used in any type of system development. This generic approach to system modeling will establish a viable and credible foundation for the communication, storage and evaluation of technical system data. Generic systems development patterns and associated information model patterns will be used to communicate complex sets of technical data.

Historical tracking of system development patterns and system deployment techniques is a systems engineering practice that provides great value but a great cost for labor and configuration management of the material. The general approaches to system analysis and development supported by the methods presented in this thesis will greatly reduce the level of effort required to capture, analyze, and use this historical pattern data.

The CCFRAT approach provides a coordinated set of technical information constructs that are recorded and linked in a logical manner. The core information store consists of text data, executable model data and process meta-data that provide a mechanism to produce a detailed audit trail if one is required. One advantage of a systems engineering management data store developed using this information model is independence from any set of hard copy documents. The content of documents can be entered into the data store but the logic of the data relationships is not based on the existence of a structured "document tree." If a set of documents is required, by the customer, to populate a document tree, then these documents could be generated directly from the system engineering data store.

The CCFRAT systems engineering information model is designed around an incremental, modular recursive approach to system design. In this type of system

design, each phase of system development is carefully designed and bounded to consider all necessary factors at the current level of system abstraction. The systems engineering data store is used as a repository for all system related data and is organized in a manner that allows detailed tracking of design and decision threads through the data store.

The technical decision record is a standard by-product of a CCFRAT based systems engineering data store. The decision value set and decision matrix for each phase of system development and each trade study is stored in the database. The decision record can be produced for evaluation at any time.

6. DISCUSSION

6.1. APPLICATION RISKS

Systems engineering practices have been used to develop and deploy many complex technical and social-technical systems. There are always risks associated with these types of activities. It is argued here that structured, well-known, open-system approaches to systems engineering information models and databases will reduce overall system risk by increasing the quality, quantity, and precision of systems engineering data, in both the technical and programmatic domains. The relational database structure presented in this work has a flexible conceptual foundation that allows the structure to adapt to changing system models and organizational objectives. Other organizational data sources can be linked with the system to leverage organizational information stored in other formats and information systems.

6.2. ISSUES IN ORGANIZATIONAL DEPLOYMENT

The primary issues associated with the deployment and use of a systems-engineering database and other decision support tools are centered in the area of systems ease-of-use and interoperability. Large distributed systems development programs are decentralized activities that require coordination of large groups of people over a wide geographic area. The support tools must be network enabled and provide access control and configuration to control the access to system data.

The database tables and models presented in this document are considered to be the data tier in a standard three-tiered information management system. The systems engineering applications logic and access to the database would be located at the middle

tier, while the system user interface and be located at the client tier. The application server at the middle tier would add further process logic and data control to the overall systems engineering data management system. The application logic and would be separated from the data store and provide a more adaptive system architecture.

6.3. REQUIRED ORGANIZATIONAL ENVIRONMENT

The successful deployment of any automated support tool rests upon the organizations level of training and understanding of the tools benefits. Open source based tools provide a further benefit by allowing the organization to customize the tool to support the specific task at hand. Low-cost or no-cost relational database management systems provided in an open source manner can be effectively used to establish a customized, distributed systems engineering information management tool set. Existing data systems could be quickly modified to establish interfaces to any systems engineering management tool set based on the approach outlined in this document.

6.4. APPLICATION BENEFITS

An automated systems engineering data management tool will provide a mechanism to define the organization's systems engineering data products and processes. The tool can be used to reinforce the correct processes and practices necessary to produce useful products. Near real-time access to program information combined with a detailed technical decision history will benefit the organization in the

current program as well as develop a set of historical procedures and work patterns that can be used in future system development activities.

APPENDIX A.

POSTGRESQL DATABASE TABLES

The postgresql database tables developed for this thesis are listed in this appendix.

The detailed design structure for the system table is:

Column	Type	Modifiers
system_id	integer	not null default nextval (‘”system_system_id_seq”’:: text) (Primary key: system_pkey)
system_name	character(64)	not null
system_description	text	
system_acronym	character(64)	
system_type	integer	(Foreign key)
system_context	integer	(Foreign key)
system_concept	integer	(Foreign key)
system_model	integer	(Foreign key)
system_db_model	integer	(Foreign key)
system_decision	integer	(Foreign key)
system_document	integer	(Foreign key)

The detailed design structure for the context table is:

Column	Type	Modifiers
context_id	integer	not null default nextval(‘”context_context_id_seq”’:: text) (Primary key: context_pkey)

context_name	character(64)	not null
context_description	text	
context_type	integer	(Foreign key)
concept_connection	integer	(Foreign key)
context_model	integer	(Foreign key)
context_db_model	integer	(Foreign key)
context_document	integer	(Foreign key)

The detailed design structure for the concept table is:

Column	Type	Modifiers
concept_id	integer	not null default nextval (‘”concept_concept_id_seq”’:: text) (Primary key: concept_pkey)
concept_name	character(64)	not null
concept_description	text	not null
concept_model	integer	(Foreign key)
concept_db_model	integer	(Foreign key)
concept_type	integer	(Foreign key)
context_connection	integer	(Foreign key)
function_connection	integer	(Foreign key)
requirement_connection	integer	(Foreign key)
architecture_connection	integer	(Foreign key)
test_connection	integer	(Foreign key)

concept_decision	integer	(Foreign key)
concept_document	integer	(Foreign key)

The detailed design structure for the function table is:

Column	Type	Modifiers
function_id	integer	not null default nextval (‘”function_function_id_seq”’:: text) (Primary key: function_pkey)
function_name	character(64)	not null
function_description	text	not null
function_model	integer	(Foreign key)
function_db_model	integer	(Foreign key)
function_type	integer	(Foreign key)
concept_connection	integer	(Foreign key)
fr_link	integer	(Foreign key)
fa_link	integer	(Foreign key)
function_decision	integer	(Foreign key)
function_document	integer	(Foreign key)

The detailed design structure for the requirement table is:

Column	Type	Modifiers
requirement_id	integer	not null default nextval (‘”requirement_requirement_id_seq”’::

		text) (Primary key: requirement_pkey)
requirement_name	character(64)	not null
requirement_description	text	not null
requirement_model	integer	(Foreign key)
requirement_db_model	integer	(Foreign key)
requirement_type	integer	(Foreign key)
concept_connection	integer	(Foreign key)
fr_link	integer	(Foreign key)
requirement_decision	integer	(Foreign key)
requirement_document	integer	(Foreign key)

The detailed design structure for the architecture table is:

Column	Type	Modifiers
architecture_id	integer	not null default nextval (‘”architecture_architecture_id_seq”’:: text) (Primary key: architecture_pkey)
architecture_name	character(64)	not null
architecture_description	text	not null
architecture_model	integer	(Foreign key)
architecture_db_model	integer	(Foreign key)
architecture_type	integer	(Foreign key)
concept_connection	integer	(Foreign key)
fa_link	integer	(Foreign key)

at_link	integer	(Foreign key)
architecture_decision	integer	(Foreign key)
architecture_document	integer	(Foreign key)

The detailed design structure for the test table is:

Column	Type	Modifiers
test_id	integer	not null default nextval (‘”test_test_id_seq”’:: text) (Primary key: test_pkey)
test_name	character(64)	not null
test_description	text	not null
test_model	integer	(Foreign key)
test_db_model	integer	(Foreign key)
test_type	integer	(Foreign key)
concept_connection	integer	(Foreign key)
at_link	integer	(Foreign key)
test_decision	integer	(Foreign key)
test_document	integer	(Foreign key)

The detailed design structure for the model table is:

Column	Type	Modifiers
model_id	integer	not null default nextval (‘”model_model_id_seq”’:: text) (Primary

		key: model_pkey)
model_name	character(64)	not null
model_description	text	not null
model_executable	oid	

The detailed design structure for the db_model table is:

Column	Type	Modifiers
db_model_id	integer	not null default nextval (‘”db_model_db_model_id_seq”’:: text) (Primary key: db_model_pkey)
db_model_name	character(64)	not null
db_model_description	text	not null
db_model_executable	text	

The detailed design structure for the type table is:

Column	Type	Modifiers
type_id	integer	not null default nextval (‘”type_type_id_seq”’:: text) (Primary key: type_pkey)
type_name	character(64)	not null
type_description	text	not null

The detailed design structure for the decision table is:

Column	Type	Modifiers
decision_id	integer	not null default nextval (‘”decision_decision_id_seq”’:: text) (Primary key: decision_pkey)
decision_name	character(64)	not null
decision_description	text	not null
decision_model	integer	(Foreign key)
decision_db_model	integer	(Foreign key)
decision_type	integer	(Foreign key)
decision_outcome	text	not null
decision_document	integer	(Foreign key)

The detailed design structure for the document table is:

Column	Type	Modifiers
document_id	integer	not null default nextval (‘”document_document_id_seq”’:: text) (Primary key: document_pkey)
document_name	character(64)	not null
document_content	text	not null
document_author	text	not null
document_link	integer	(Foreign key)
document_link_1	integer	(Foreign key)

The detailed design structure for the fr_link table is:

Column	Type	Modifiers
fr_id	integer	not null default nextval (‘”fr_link_fr_id_seq”’:: text) (Primary key: fr_link_pkey)
function_id	integer	not null
requirement_id	integer	not null

The detailed design structure for the fa_link table is:

Column	Type	Modifiers
fa_id	integer	not null default nextval (‘”fa_link_fa_id_seq”’:: text) (Primary key: fa_link_pkey)
function_id	integer	not null
architecture_id	integer	not null

The detailed design structure for the at_link table is:

Column	Type	Modifiers
at_id	integer	not null default nextval (‘”at_link_at_id_seq”’:: text) (Primary key: at_link_pkey)
test_id	integer	not null
architecture_id	integer	not null

APPENDIX B.

INFORMATION MODEL TABLES

The detailed systems engineering relationship models are presented in this appendix.

System Behavior Model Classes are shown in the Table B-1 below.

Class 1	Relationship	Class2
Behavior	Contains zero or more	Input/Output
Behavior	Contains zero or more	Functions
Behavior	Contains zero or more	Control Operations
Input/Output	Generates and consumes	Functions
Function	Ordered by	Control Operation
Non-triggering I/O	Type of (effect)	Input/Output
Triggering I/O	Type of (effect)	Input/Output
Triggering I/O	Triggers	Function
Non-conditioning I/O	Type of (condition)	Input/Output
Conditional I/O	Type of (condition)	Input/Output
Conditional I/O	Provide criteria for	Selection
Selection	Type of	Control Operation
Sequence	Type of	Control Operation
Iteration	Type of	Control Operation
Concurrency	Type of	Control Operation
Parallel function	Type of	Concurrency
State	Type of	Concurrency

The system input and output model is shown below in Table B-2.

Class 1	Relationship	Class 2
Material I/O	Type of (physical nature)	Input/Output
Energy I/O	Type of (physical nature)	Input/Output
Information I/O	Type of (physical nature)	Input/Output
Non-Triggering I/O	Type of (effect)	Input/Output
Triggering I/O	Type of (effect)	Input/Output
Non-Condition I/O	Type of (condition)	Input/Output
Condition I/O	Type of (condition)	Input/Output
Stationary I/O	Type of (longevity)	Input/Output
Transitory I/O	Type of (longevity)	Input/Output
Local I/O	Type of (access)	Stationary I/O
Global I/O	Type of (access)	Stationary I/O
Replica	Type of (storage)	Stationary I/O
Stock	Type of (storage)	Stationary I/O
Triggering no content	Type of (content)	Triggering I/O
Triggering with content	Type of (content)	Triggering I/O

The system behavior and structure model is shown below in Table B-3.

Class 1	Relationship	Class 2
Classification	Type of	Structure View
Aggregation	Type of	Structure View
Context Diagram	Type of	Structure View

Assembly Diagram	Type of	Structure View
Structure View	Is associated with	Object model
Object Model	Defines interfaces to	Object behavior map
Object behavior map	Encapsulates functions of	Behavior model
Behavior model	Represented by	Behavior views
Parallel function	Type of	Behavior view
I/O Control	Type of	Behavior view
State	Type of	Behavior
Functions I/O	Type of	Behavior
Event control	Type of	Behavior
Function control	Type of	Behavior
State control	Type of	Behavior

Table B-4 shows the system requirements information model relationships.

Class 1	Relationship	Class 2
Initial information	Type of	Available information
Developed information	Type of	Available information
Initial text requirement	Part of	Initial information
Heritage information	Part of	Initial information
User information	Part of	Initial information
Initial model	Part of	Initial information
Operations concepts	Part of	Initial information
Derived requirement	Part of	Developed information

Developed model	Part of	Developed information
Implied requirement	Part of	Developed information
Adjudicated constraint	Part of	Developed information
Operational concept	Narrate	Initial model
Developed model	Extends	Initial model
Initial text requirement	Trace to	Initial model
Initial text requirement	Trace to	Developed model
Implied requirement	Trace to	Developed model
Implied requirement	Trace to	Derived requirement
Derived requirement	Trace to	Derived requirement
Reference requirement	Type of (by origin)	Initial text requirement
Original requirement	Type of (by origin)	Initial text requirement
Reference requirement	Point to	Original requirement
Resolution	Trace to	Derived Requirement
Resolution	Trace to	Adjudicated Constraint
Resolution	Trace to	Issue
Interface requirement	Type of	Initial text requirement
Functional requirement	Type of	Initial text requirement
Temporal performance	Type of	Initial text requirement
Nontemporal performance	Type of	Initial text requirement
Design requirement	Type of	Initial text requirement
Design requirement	Trace to	Issue

Table B-5 shows the system effectiveness measure creation model.

Class 1	Relationship	Class 2
Effectiveness Measure	Ranked by	Priorities
Priority survey	Generate	Priorities
Cost function	Establish	Priorities
Effectiveness Measure	Compute with	EM equations
EM from modeling	Type of (source)	Effectiveness Measure
EM from preferences	Type of (source)	Effectiveness Measure
EM from attributes	Type of (source)	Effectiveness Measure
EM survey	Generate	EM from preferences
Effectiveness Measure	Determine alternatives	Subject system design
Cost function	Selects	Subject system design
Execution engine	Compute	EM from modeling
Execution engine	Executes behavior	Subject system behavior
Subject system design	Is composed of	Component
Component	Built from	Component
Component	Describe structure	Structure operations
Component	Have	Object interface
Attributes	Are arguments of	EM equations
Attributes	Have	Values
Values	Provide	Value computation
Measurement	Type of	Value computation
Simulation	Type of	Value computation

Estimation	Type of	Value computation

Table B-6 shows the text requirement, behavioral and context model components.

Class 1	Relationship	Class 2
Object (c)	Part of	Object (c)
Object (c)	Has	Object role (c)
Input/Output (c)	Type of (kind of role)	Object role (c)
External systems (c)	Type of (kind of role)	Object role (c)
Subject system (c)	Type of (kind of role)	Object role (c)
Component (c)	Type of (kind of role)	Object role (c)
Component (c)	Part of (two or more)	Subject system (c)
Subject system (c)	Interface	External system (c)
External system (c)	Connects to	Interface (c)
Subject system (c)	Connects to	Interface (c)
Subject system (c)	Responds by	Response threads (b)
External system (c)	Executes	Scenarios (b)
Scenarios (b)	Respond to	Response threads (b)
Scenarios (b)	Part of	Behavior (b)
Response threads (b)	Part of	Behavior (b)
Function (b)	Part of	Behavior (b)
Input/Output (b)	Part of	Behavior (b)
Structure Operations (b)	Define behavior	Behavior (b)

	hierarchy	
Function (b)	Ordered by	Control Operations (b)
Function (b)	Trace to	Functional requirements (t)
Function (b)	Budget to	Temporal requirement (t)
Function (b)	Limit choice	Adjudicated constraint (t)
Function (b)	Generate and consume	Input/Output (b)
Interface requirement (t)	Type of (by use)	Text requirement (t)
Functional requirement (t)	Type of (by use)	Text requirement (t)
Temporal requirement (t)	Type of (by use)	Text requirement (t)
Nontemporal requirement (t)	Type of (by use)	Text requirement (t)
Design (t)	Type of (by use)	Text requirement (t)
Resolution (t)	Trace to	Text requirement (t)
Adjudicated constraint (t)	Trace to	Resolution (t)
Resolution (t)	Trace to	Issue (t)
Issue (t)	Trace to	Design (t)

The text requirements, structure and context model isshown below in Table B-7.

Class 1	Relationship	Class 2
Object	Part of	Object
Object	Has	Object role
Input/Output	Type of	Object role

External system	Type of	Object role
Subject system	Type of	Object role
Component	Type of	Object role
External system	Defines interfaces	Subject system
External system	Connects to	Interface
External system	Operate by	Scenarios
Subject system	Connects to	Interface
Scenarios	Composed of	Text operations concept
Scenarios	Respond to	Response threads
Subject system	Respond to	Response threads
Scenarios	Part of	Behavior
Response threads	Part of	Behavior
Structure operations	Define hierarchy	Behavior
Behavior	Allocate to	Component
Component	Defined by	Component Interface
Component	Assigned to	Functions
Component	Budget to	Temporal requirement
Component	Have	Attributes
Component	Trace to	Adjudicated constraints
Adjudicated constraints	Trace to	Resolution
Resolution	Trace to	Issue
Issue	Trace to	Design
Interface requirement	Type of	Text Requirement

Functional requirement	Type of	Text Requirement
Temporal requirement	Type of	Text Requirement
Non-temporal requirement	Type of	Text Requirement
Design	Type of	Text Requirement

Table B-8 shows the tradeoff analysis model.

Class 1	Relationship	Class 2
Effectiveness measure	Ranked by	Priority
EM from modeling	Type of	Effectiveness measure
EM from Preferences	Type of	Effectiveness measure
EM from Attributes	Type of	Effectiveness measure
EM survey	Generates	EM from preferences
EM equation	Computed with	Effectiveness measure
Subject system	Alternative selection	Effectiveness measure
Priority survey	Generate	Priority
Priority	Establish	Cost function
Cost function	Selects	Subject system design
Object	Part of	Subject system design
Object	Describe structure	Structure operations
Object	Have	Object interfaces
EM equations	Arguments for	Attributes
Attributes	Arguments for	Non-temporal equation
Non-temporal requirement	Computed with	Non-temporal equation

Attributes	Have	Values
Value	Provide	Value Computation
Measurement	Type of	Value Computation
Simulation	Type of	Value Computation
Estimation	Type of	Value Computation
Execution engine	Compute	EM from modeling
Execution engine	Generate	Time lines
Time lines	Validate	Temporal requirement

Table B-9 shows the Sequential Build and Test Model.

Class 1	Relationship	Class 2
Components	Part of (tested)	Subject system
Component	Assembled under direction	SBTP
SBTP	Monitored by	Tracking and reporting
SBTP	Accounts for	Business realities
Time to market	Type of	Business realities
Funding rate	Type of	Business realities
Risk	Type of	Business realities
Competition	Type of	Business realities
Validation of progress	Type of	Business realities
Validation of progress	Show response	Response threads
Technical risk	Type of	Risk
Schedule risk	Type of	Risk

Cost risk	Type of	Risk
Risk survey	Establishes	Technical risk

Table B-10 shows management information relationships.

Class 1	Relationship	Class 2
Customer Needs	Determine	System Plan
Organization Plan/Status	Constrain	System Plan
Long-term Plan	Type of	System Plan
Incremental Plan	Type of	System Plan
Long-term Plan	Constrains	Incremental Plan
Risk Management Plan	Type of	Long-term Plan
System Process Definition	Type of	Long-term Plan
Increment Definitions	Type of	Long-term Plan
Work Product Description	Type of	Long-term Plan
Size, Cost, Schedule Est.	Type of	Long-term Plan
Estimate of the Situation	Type of	Long-term Plan
Estimate of the Situation	Constrain	Risk Management Plan
Risk Management Plan	Determine	Incremental Definitions
Risk Management Plan	Determine	System Process Definition
Risk Management Plan	Determine	Size, Cost, Schedule Est.
Work Product Description	Determine	Size, Cost, Schedule Est.
Incremental Requirements	Type of	Increment Definitions
Resource Allocations	Type of	Incremental Definitions

Size, Cost, Schedule Est.	Determine	Resource Allocations
Resource Allocations	Constrain	Incremental Requirements
Development Goal	Part of	Incremental Plan
Incremental Requirements	Define	Development Goal
Inc. Size, Cost, Sch. Est.	Part of	Incremental Plan
Resource Allocations	Determine	Inc. Size, Cost, Sch. Est.
Success Criterion	Part of	Incremental Plan
Tailored Process	Part of	Incremental Plan
Development Goal	Determine	Success Criterion
Development Goal	Determine	Tailored Process
Tailored Process	Determine	Inc. Size, Cost, Sch. Est.
System Process Definition	Determine	Tailored Process
Organization Plan/Status	Part of	Org. Planning Info.
Org. Standard Proc.	Part of	Org. Planning Info.
Org. Standard Proc.	Constrain	System Process Definition
Organizational Plan	Part of	Organization Plan/Status
Organizational Status	Part of	Organization Plan/Status
Organizational Status	Determine	Organizational Plan
Incremental Results	Type of	System Develop. Results
System Status	Type of	Increment Results
System Status	Determined by	Increment Plan
Increment Results	Determine	System Status
Technical Risk Data	Part of	Increment Results

Technical Risk Data	Determine	Risk Management Plan
Incremental Results	Determine	Long-term Plan
System Work Products	Part of	Increment Results
Lessons Learned	Part of	Increment Results
Lessons Learned	Determine	Tailored Process
System Develop. Results	Determine	Incremental Definitions
System Develop. Results	Determine	Org. Planning Info.

Table B-11 shows the Design and Verification Relationships.

Class 1	Relationship	Class 2
System Requirement	Part of	System Design
Functional Architecture	Part of	System Design
Physical Architecture	Part of	System Design
System Design Plan	Constrain	System Design
System Design	Incorporate	Reusable Assets
Physical Architecture	Implement	Functional Architecture
Performance	Type of	System Requirement
System Requirement	Formalize	Customer Needs
Physical Architecture	Meet	Performance
Physical Architecture	Determine	CI Requirements
Behavioral	Type of	System Requirement
Functional Architecture	Implement	Behavioral
Constraints	Type of	System Requirement

External Interfaces	Type of	System Requirement
Functional Elements	Part of	Functional Architecture
Functional Interfaces	Part of	Functional Architecture
Human Elements	Part of	Physical Architecture
Physical Interfaces	Part of	Physical Architecture
CI	Part of	Physical Architecture
Hardware CI	Part of	CI
Software CI	Part of	CI
System Eval., V&V Info.	Compare With	Customer Needs
System Evaluation Results	Part of	System Eval., V&V Info.
V&V Results	Part of	System Eval., V&V Info.
I&T Procedures	Part of	System Eval., V&V Info.
I&T Procedures	Determined by	System Design
V&V Results	Verify	I&T Results
V&V Results	Validate	System Requirements
V&V Results	Verify	Functional Architecture
V&V Results	Verify	Physical Architecture
Detailed I&T Procedures	Elaborate	I&T Procedures
Recommendations	Part of	System Evaluation Results
Recommendations	Determine	System Design
Proposed Improvements	Part of	System Evaluation Results
Recommendations	Determine	Proposed Improvements
System Design	Incorporate	Proposed Improvements

Trade Studies	Part of	System Evaluation Results
Trade Studies	Determine	Recommendations
Completeness	Part of	V&V Results
Trace-ability	Part of	V&V Results
I&T Procedures	Determine	V&V Results
Integration	Part of	I&T Procedures
Test	Part of	I&T Procedures
Analysis	Part of	I&T Procedures

Table B-12 shows the conceptual information model for the AP-233 architecture.

Class 1	Relationship	Class 2
Presentation Information	Type of	Support Information
Configuration Data	Type of	Support Information
External Document	Type of	Support Information
Admin Information	Type of	Support Information
Data Types	Type of	Support Information
Classification	Type of	Support Information
Properties	Type of	Support Information
Support Information	Provides	Engineering Process
Support Information	Provides	Specification Elements
Support Information	Provides	System Architecture
Engineering Process	Records	System Architecture
Engineering Process	Records	Specification Elements

System Architecture	Defined by	Specification Elements
Requirements Data	Type of	Specification Elements
Functional Architecture	Type of	Specification Elements
Physical Architecture	Type of	Specification Elements
Requirements Data	Allocates to	Functional Architecture
Requirements Data	Allocates to	Physical Architecture
Functional Architecture	Allocates to	Physical Architecture

BIBLIOGRAPHY

1. Hughes, Thomas P. *Rescuing Prometheus*. New York: Vintage Books, 1998.

2. International Council on Systems Engineering (INCOSE) Technical Board. *Systems Engineering Handbook, 2nd ed*. Seattle INCOSE, 1998.

3. Mar, Brian. *Introduction to the Engineering of Complex Systems*. Seattle: University of Washington, 1996.

4. Chase, Wilton P. *Management of Systems Engineering*. Malabar: Robert E. Kreiger Publishing Company, Inc. 1974.

5. INCOSE Tools Database Working Group, 2003, *IEEE-1220 Process to SE Tools Mapping*. http://www.incose.org/tool/ieee1220top.html (10 Aug. 2003).

6. Grady, Jeffrey O. *Systems Requirements Analysis*. Boca Raton: CRC Press, 1994.

7. Defense Systems Management College. *Systems Engineering Management Guide.* Fort Belvoir: Defense Systems Management College, 1986.

8. Simpson, J. J. and Simpson, M. J., 2000, "U.S. Dept. of Defense Systems Engineering and Implications for SE Implementation in Other Domains." *Proceedings of the 2nd European System Engineering Conference*, 139-145.

9. Government Electronics & Information Technology Association. *Process for Engineering a System, EIA-632*. Denver: Global Engineering Documents, 2000.

10. Government Electronics & Information Technology Association. *Systems Engineering Capability Model, EIA-731*. Denver: Global Engineering Documents, 2000.

11. Bremen University, 2003, *GD250: Lifecycle Process Model "V-MODEL"*. http://www.informatik.uni-bremen.de/gdpa. (10 Aug 2003).

12. United States Department of Defense. *Defense System Software Development, DoD-STD-2167 A.* 1988.

13. Ramesh, B., Powers, T., Stubbs, C. and Edwards, M. *A Study of Current Practices of Requirements Traceability in Systems Development.* Monterey: Naval Postgraduate School, 1993.

14. Ramesh, B., Harrington, G., Rondeau, K., and Edwards, M. *A Model of Requirements Traceability to Support Systems Development.* Monterey: Naval Postgraduate School, 1993.

15. Chase, Wilton P. *Management of Systems Engineering.* New York: Vintage Books, 1998.

16. Forsberg, K. and Mooze, H., 1991, "The Relationship of Systems Engineering to the Project Life Cycle," *Proceedings of the First Annual International Symposium of the National Council on Systems Engineering,* 57-68.

17. Alford, M., 1991, "Strengthening the Systems Engineering Process," *Proceedings of the First Annual International Symposium of the National Council on Systems Engineering,* 1-9.

18. Alford, M., 1992, "Strengthening the Systems/Software Engineering Interface for Real Time Systems," *Proceedings of the Second Annual International Symposium of the National Council on Systems Engineering,* 411-418.

19. Mar, B. W., 1992, "Back to Basics," *Proceedings of the Second Annual International Symposium of the National Council on Systems Engineering,* 37-43.

20. Oliver, D. W., 1993, "Automated Optimization of Systems Architectures for Performance," *Proceedings of the Third Annual International Symposium of the National Council on Systems Engineering,* 259-266.

21. Oliver, D. W., 1993, "Descriptions of Systems Engineering Methodologies and Comparison of Information Representations," *Proceedings of the Third Annual International Symposium of the National Council on Systems Engineering,* 97-104.

22. Oliver, D. W., 1996, "An Analysis of the Model Based Systems Engineering Process," *Proceedings of the Sixth Annual International Symposium of the International Council on Systems Engineering,* 369-379.

23. Oliver, D. W., Kelliher T. P., and Keegan J. G. *Engineering Complex Systems with Models and Objects.* New York: McGraw-Hill, 1997.

24. Levis, A. H. and Wagenhals, L. W., 2000, "C4ISR Architectures: I Developing a Process for C4ISR Architecture Design," *Systems engineering,* 3(4), 225-248.

25. Wagenhals, L. W., Shin, I., Kim, D. and Levis, A. H., 2000, "C4ISR Architectures: II A Structured Analysis Approach for Architecture Design," *Systems Engineering,* 3(4), 249-288.

26. Shin, I., Bienvenu, M. P. and Levis, A. H., 2000, "C4ISR Architectures: III An Object Oriented Approach for Architecture Design," *Systems Engineering,* 3(4), 289-301.

27. Mar, B. W., 1997, "Back to Basics Again, A Scientific Definition of Systems Engineering," *Proceedings of the Seventh Annual International Symposium of the International Council on Systems Engineering,* 229-236.

28. Karangelen, N. E. and Hoang, N. T., 1994, "Partitioning Complex System Design Into Five Views," *Proceedings of the Fourth Annual International Symposium of the National Council on Systems Engineering,* CDROM.

29. Frank, R. F., 1998, "System Architecture – A View Perspective," *Proceedings of the Eighth Annual International Symposium of the International Council on Systems Engineering,* CDROM.

30. Maier, M. W., 1998, "Reconciling Systems and Software Architecture," *Proceedings of the Eighth Annual International Symposium of the International Council on Systems Engineering,* CDROM.

31. Percivall G. S., 1997, "Application of a System Architecture Standard to a Federated Information System," *Proceedings of the Seventh Annual International Symposium of the International Council on Systems Engineering,* CDROM.

32. Grady, J. O., 1995, "The Necessity of Logical Continuity," *Proceedings of the Fifth Annual International Symposium of the National Council on Systems Engineering,* CDROM.

33. Mar, B. W. and Morais, B. G., 2002, "FRAT – A Basic Framework for Systems Engineering," *Proceedings of the Eleventh Annual International Symposium of the International Council on Systems Engineering,* CDROM.

34. Simpson, J. J. and Simpson, M. J., 2001, "U.S. DoD Legacy SE & Implications for Future SE Implementation," *Proceedings of the Eleventh Annual International Symposium of the International Council on Systems Engineering,* CDROM.

35. Vigland, A. R., 1993, "Developing a Systems Engineering Automation Strategy," *Proceedings of the Third Annual International Symposium of the National Council on Systems Engineering,* CDROM.

36. Martin, J. N., 1995, "Requirements Mythology: Shattering Myths About Requirements and the Management Thereof," *Proceedings of the Fifth Annual International Symposium of the National Council on Systems Engineering,* CDROM.

37. Jones, D. A., 1995, “Some Lessons Learned in Requirements Management,” *Proceedings of the Fifth Annual International Symposium of the National Council on Systems Engineering,* CDROM.

38. Hellouin, L., Beaugrand, J. and Sahraoui, A., 2001, “Requirements Process and Traceability Issues in a SE Methodology,” *Proceedings of the Eleventh Annual International Symposium of the International Council on Systems Engineering,* CDROM.

39. LaBudde, E. V., 1995, “Requirements Management: Contrasting Traditional Systems and Software Engineering Methods,” *Proceedings of the Fifth Annual International Symposium of the National Council on Systems Engineering,* 489-494.

40. Stemm, D. J., 2001, “Requirements Management Versus Models,” *Proceedings of the Eleventh Annual International Symposium of the International Council on Systems Engineering,* CDROM.

41. DeGRegorio, G., 2000, “Enterprise-wide Requirements & Decision Management,” *Proceedings of the Tenth Annual Symposium of the International Council on Systems Engineering,* CDROM.

42. Brown, D. D., 1993, “Implementation of a Requirements Verification Database,” *Proceedings of the Third Annual International Symposium of the National Council on Systems Engineering,* 623-629.

43. Helm, J. C., 2002, “Decision Metrics Matrix Process,” *Proceedings of the Eleventh Annual International Symposium of the International Council on Systems Engineering,* CDROM.

44. Jones, D. A., 1997, “Executable Requirements Management Model,” *Proceedings of the Seventh Annual International Symposium of the International Council on Systems Engineering,* CDROM.

45. Griffith, P. B., 1994, “Different Philosophies / Different Models: RDD and IDEF,” *Proceedings of the Fourth Annual International Symposium of the National Council on Systems Engineering,* CDROM.

46. Douglas, K. S., 1993, “The Need for Automated Verification Tools for the Systems Engineer,” *Proceedings of the Third Annual International Symposium of the National Council on Systems Engineering,* 607-613.

47. Sutinen, K., Almefelt, L. and Malmqvist, J., 2002, “Supporting Concept Development Using Quantitative Requirements Traceability,” *Proceedings of the Eleventh Annual International Symposium of the International Council on Systems Engineering,* CDROM.

48. Crowe, D., Smith, H., Haberli, G., Cohen, R. and Lykins, H., 1998, "Adaptation of a Software Requirements Engineering Method to the System Level for Software-Intensive Systems," *Proceedings of the Eighth Annual International Symposium of the International Council on Systems Engineering*, CDROM.

49. Baker, L., 1994, "Systems Engineering: Communication From a System Design Database," *Proceedings of the Fourth Annual International Symposium of the National Council on Systems Engineering*, CDROM.

50. Carson, R. S., 1995, "A Set Theory Model for Anomaly Handling in Systems Requirements Analysis," *Proceedings of the Fifth Annual International Symposium of the National Council on Systems Engineering*, 515-522.

51. Mar, B. W., 1994, "Requirements for Development of Software Requirements," *Proceedings of the Fourth Annual International Symposium of the National Council on Systems Engineering*, CDROM.

52. Grady, J. O., 1998, "Systems Beget Systems," *Proceedings of the Eighth Annual International Symposium of the International Council on Systems Engineering*, CDROM.

53. Simpson, J.J., 2002, "Innovation and Technology Management," *Proceedings of the Eleventh Annual International Symposium of the International Council on Systems Engineering*, CDROM.

54. Hall, A. D. *Metasystems Methodology, A New Synthesis and Unification.* New York: Pergamon Press, 1987.

55. Checkland, P. and Scholes, J. *Soft System Methodology in Action.* New York: John Wiley & Sons, 1999.

56. Mar, B. *Introduction to the Engineering of Complex Systems.* Seattle: University of Washington, 1996.

57. Woods, T. W., 1993, "First Principles: Systems and Their Analysis," *Proceedings of the Third Annual International Symposium of the National Council on Systems Engineering*, 41-46.

58. Plastow, I., 1998, "How Do You Model a System," *Proceedings of the Eighth Annual International Symposium of the International Council on Systems Engineering*, CDROM.

59. Baker, L. and Long, J., 1998, "Specifying a System Using ERA Information Models," *Proceedings of the Eighth Annual International Symposium of the International Council on Systems Engineering*, CDROM.

60. Long, J. E., 1995, "Relationships Between Common Graphical Representation Used in Systems Engineering," *Proceedings of the Fifth Annual International Symposium of the National Council on Systems Engineering,* CDROM.

61. Yoo, I. S., Kim, J. C. and Park, Y. W., 2000, "A Developmental Guide of Robust System Architecture," *Proceedings of the Tenth Annual International Symposium of the International Council on Systems Engineering,* CDROM.

62. Steiner, R., 1998, "System Architectures and Evolvability: Definitions and Perspective," *Proceedings of the Eighth Annual International Symposium of the International Council on Systems Engineering,* CDROM.

63. Martin, J.N., 1997, "Process for Requirements and Architecture Definition," *Proceedings of the Seventh Annual International Symposium of the International Council on Systems Engineering,* CDROM.

64. Carson, R. S., 2000, "Global System Architecture Optimization: Quantifying System Complexity," *Proceedings of the Tenth Annual International Symposium of the International Council on Systems Engineering,* CDROM.

65. Gonzales, R. and Lovelace, N., 1997, "Using Stakeholder Mental Models to Create an Integrated Conceptual Model for Systems," *Proceedings of the Seventh Annual International Symposium of the International Council on Systems Engineering,* CDROM.

66. Armstrong, J. R., 1999, "Starting at the Finish Line," *Proceedings of the Ninth Annual International Symposium of the International Council on Systems Engineering,* CDROM.

67. Bilardo, V. J., Schlater, N.J., Forsberg, K. and Mooz, H., 1992, "Tailoring a Generic System Engineering Process for the Development of Research and Technology Projects," *Proceedings of the Second Annual International Symposium of the National Council on Systems Engineering,* 215-218.

68. DeLozier, R. and Snyder, N., 1993, "Engineering Performance Metrics," *Proceedings of the Third Annual International Symposium of the National Council on Systems Engineering,* 599-605.

69. Iliff, R. C., 1992, "Mapping Systems Engineering Principles and Practices to Commercial Sector Activity," *Proceedings of the Second Annual International Symposium of the National Council on Systems Engineering,* 201-205.

70. Sage, A. P. and Armstrong, J. E. *Introduction to Systems Engineering.* New York: John Wiley & Sons, 2000.

71. Blanchard, B. and Fabrycky, W. *Systems Engineering and Analysis.* New Jersey: 1997.

72. Tronstad, Y. D., 1997, "A Business Systems Engineering Model Architecture," *Proceedings of the Seventh Annual International Symposium of the International Council on Systems Engineering,* CDROM.

73. Vigland, A. R., 1993, "Developing a Systems Engineering Automation Strategy," *Proceedings of the Third Annual International Symposium of the National Council on Systems Engineering,* 769-774.

74. Muth, T., Herzberg D. and Larsen, J., 2001, "A Fresh View on Model-based Systems Engineering: The Processing System Paradigm," *Proceedings of the Eleventh Annual International Symposium of the International Council on Systems Engineering,* CDROM.

75. Jorgensen, R. W. and Philpott, I. W., 2002, "Architectural Abstractions," *Proceedings of the Eleventh Annual International Symposium of the International Council on Systems Engineering,* CDROM.

76. Denny, B. and Bennett, R., 2000, "One Engineering Process – Integrated," *Proceedings of the Tenth Annual International Symposium of the International Council on Systems Engineering,* CDROM.

77. Hyer, S. A. and Jones, M. W., 2000, "Realizing Complete Traceability With an Integrated Systems Engineering Environment (ISEE)," *Proceedings of the Tenth Annual International Symposium of the International Council on Systems Engineering,* CDROM.

78. Negele, H. and Wenzel, S., 2001, "Systems Engineering Meta-Tools for Complex Product Development," *Proceedings of the Eleventh Annual International Symposium of the International Council on Systems Engineering,* CDROM.

79. Adamsen, P. B., 1995, "A New Look at the Systems Engineering Process: A Detailed Algorithm," *Proceedings of the Fifth Annual International Symposium of the National Council on Systems Engineering,* CDROM.

80. Rochecouste, H., 1998, "Engineering Information Systems and the IEEE STD 1220-1994," *Proceedings of the Eighth Annual International Symposium of the International Council on Systems Engineering,* CDROM.

81. Lykins, H., Rose, S. and Scott, P. C., 1998, "An Information Model for Integrating Product Development Processes," *Proceedings of the Eighth Annual Symposium of the International Council on Systems Engineering,* CDROM.

82. Herzog, E. and Torne, A., 2000, "Ap-233 Architecture," *Proceedings of the Tenth Annual International Symposium of the International Council on Systems Engineering,* CDROM.

83. Herzog, E. and Torne, A., 1999, "Towards a Standardized Systems Engineering Information Model," *Proceedings of the Ninth Annual International Symposium of the International Council on Systems Engineering*, CDROM.

84. John, G., Hoffmann, M., Weber, M., Nagel, M. and Thomas, C., 2000, "Using a Common Information Model as a Methodological Basis for a Tool-supported Requirements Management Process," *Proceedings of the Tenth Annual International Symposium of the International Council on Systems Engineering*, CDROM.

85. DeGregorio, G., 1997, "An Information Architecture for Systems Engineering Progress, Examples, and Directions," *Proceedings of the Seventh Annual International Symposium of the International Council on Systems Engineering*, 325-332.

86. Schultz, A. P., Igenbergs, E. and Wehlitz, P., 2001, "Smart Information Architectures – Systems Engineering Information Within an Enterprise," *Proceedings of the Eleventh Annual International Symposium of the International Council on Systems Engineering,* CDROM.

87. Fossnes, T., 1999, "Systems Engineering and Information Management – a Local Need in a Global Perspective," *Proceedings of the Ninth Annual International Symposium of the International Council on Systems Engineering*, CDROM.

88. Battersby, D. J., 2000, "A Data Structure Approach to Systems Engineering," *Proceedings of the Tenth Annual International Symposium of the International Council on Systems Engineering*, CDROM.

89. McCartor, M. M. and Simpson, J. J., 1999, "Concept Mapping as a Communications Tool in Systems Engineering," *Proceedings of the Ninth Annual International Symposium of the International Council on Systems Engineering*, CDROM.

90. Geschwinde, E. and Schonig, H. *PostgreSQL Developer's Handbook.* Indianapolis: 2001.

91. Veitch, N. (ed), July 2003, "MySQL 4.0.12 Review," *Linux Format*, 28-29.

92. MySQL AB, 2003, *What Standards MySQL Follows.* http://www.mysql.com/doc/en/Standards.html (July 27th, 2003).

93. Hudson, P., Maher, J. and Toussi, F., 2002, *SQL Syntax.* http://hsqldb.sourceforge.net/doc/hsqlSyntax.html (August 3, 2003).

94. HudsonFog, 2003, *HudsonFog Readme.* http://hudsonfog.com/dev-doc/readme.html (August 3, 2003).

95. Ban, B., 2003, *JGroups – A Tool Kit for Reliable Multicast Communication.* http://www.javagroups.com/javagroupsnew/docs/hsqldbr.html (August 3, 2003).

96. Paragon Corporation, 2003, *PostgreS QL: An Open Source Object Relational Database Management System (ORDBMS).* http://www.paragoncorporation.com/ArticleDetail.aspx?ArticleID-11 (July 5, 2003).

97. Elmasri, R. and Navathe, S. B. *Fundamentals of Database Systems.* Red Wood City: The Benjamin/Cummings Publishing Company, Inc., 1994.

98. Jones, G. R. and George, J. M. *Contemporary Management.* New York: McGraw Hill, 2000.

99. Teorey, T. J. *Database Modeling & Design* San Francisco: Morgan Kaufmann Publishers, Inc., 1999.

VITA

Joseph James Simpson was born in Bremerton Washington on March 27th, 1949. Joseph J. Simpson's interests are centred 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.