A METHODOLOGY FOR THE MEASUREMENT AND EVALUATION OF COMPLEX SYSTEM DESIGNS

by

Michael Lane Talbert
Captain, USAF

Dissertation submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY
IN
COMPUTER SCIENCE

APPROVED:

Osman Balci, Chairman

Richard E. Nance

Wolter J. Fabrycky

Sallie M. Henry

James D. Arthur

December 1995
Blacksburg, Virginia

A METHODOLOGY FOR THE MEASUREMENT AND EVALUATION OF COMPLEX SYSTEM DESIGNS

by

Michael Lane Talbert

Osman Balci, Chairman

Computer Science

(ABSTRACT)

Most complex systems incorporate hardware, software and humanware elements operating synergistically under conflicting functional and nonfunctional objectives. These systems are usually embedded, mission-critical, performance-critical, real-time, distributed, highly integrated, heterogeneous, cost millions of dollars, and take many years to develop. Examples include space stations, combat vessels and aircraft, nuclear power stations, communication networks, and robotics-based manufacturing.

Early system design evaluation is essential to assess a design's potential for satisfying operational and budgetary requirements, since a significant percentage of the system life cycle cost is committed by design decisions made early in the system life cycle. However, at the design decision point, knowledge of technology, the operational environment, the political climate, etc., on which to make technically effective and cost efficient decisions is incomplete. Consequently, early design evaluation approaches are needed which yield credible results in the presence of incomplete knowledge.

This dissertation describes a multifaceted methodology for complex system design measurement and evaluation which exploits experience, techniques, and heuristics of technical and operational domain experts. The methodology is computer and knowledge based, and includes indicator-based assessment, visual simulation, the analytic hierarchy process, and fuzzy mathematics. The use of this methodology enables qualitative and quantitative measurement and evaluation of system designs at any level of detail desired. An independent assessment of the methodology by researchers and

systems engineering practitioners from the DOD, other federal government agencies, commercial industry, and academia affirmed the methodology to be a useful approach in the measurement and evaluation of complex system designs.

ACKNOWLEDGMENTS

This dissertation is dedicated to the memory of my wife's late father,
DeWitt S. True, M.D., Capt., USN (Retired)

The completion of this work marks the achievement of a goal I have cherished and strived for since high school. It is a worthy goal for a career, but not for a lifetime. I have set my sights on the more excellent goal of Eternity in the presence of the Creator. I hope to see you there.

Here on earth, I am deeply grateful to my major professor, Dr. Osman Balci, for many hours of consultation, brainstorming, and draft revision. His contributions have been in every way positively directed toward a successful, meaningful research experience. I also acknowledge and appreciate the willingness of my committee members to support this research. Their time is valuable, and I have borrowed from it mindful of the need to balance sufficiency with parsimony.

Deep thanks also to Sherry Barker, Dr. Bill Farr, and Dr. Harry Crisp. Their support in the evolution and evaluation of the methodology cannot be overstated.

My wife and children have sacrificed greatly for the cause of this degree. But their absolute, unwavering support has spiritually, physically, and mentally sustained me daily, sometimes hourly. My parents, siblings, in-laws, and other relatives have also been a tremendous source of encouragement in every way. Thank you all.

Bo, Chuck, Anders, Colin, John, Ghaleb, Dave, Ali, Utku, Levent, Randy, Marc, Trish, and Geneva: my association with each of you has been a joy and pleasure.

I am grateful to the congregations of Main Street B.C. in Christiansburg, VA, and Towerview B.C. in Belleville, IL. Thanks also to Jim and Emily, Jonathan and Kelly, and Mark B., from other local churches. The uplifting support of my Promise Keeper brothers has been crucial as well. The tie that binds us has proved its mettle many times over. Above all, to Him who supplies us all, I give my highest Thanks.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. PROBLEM DEFINITION AND OVERVIEW

1.1. INTRODUCTION

This dissertation describes a methodology for the measurement and evaluation of complex system designs. This chapter serves a two-fold purpose in the documentation of the entire research undertaking. First, it motivates the research by presenting a detailed description of the problem space for which the design evaluation methodology is an inroad to a solution. Second, provides and overview of the research endeavor.

Initially the research problem is introduced, identifying a need in the larger research community for which this work is a recommended solution. Complex systems are defined and characterized by their complex components. The process and problem of design evaluation is defined. The research objectives are enumerated, laying a foundation for assessment and substantiation of the research effort. The organization of the dissertation is also described to overview the direction of the research, and the remainder of the dissertation. Lastly, a summary of the research contributions is provided.

1.2. PROBLEM DEFINITION

As we plan for operational success in the future, within almost all industries including the military and advanced commercial industries, we increasingly rely on highly evolved technologies. Those who operate at the leading edge require dependable real-time performance from heterogeneous, integrated, computer-based systems in distributed, parallel, and/or embedded environments. Examples of this class of complex systems range from space stations, manned space vehicles, and combat vessels and aircraft, to nuclear power stations, pharmaceutical manufacturing plants and ultra-high speed communication networks. The potential fruits of mission success range from continuous space habitation and interplanetary exploration to real-time

global communication. The thorns of failure include diminution of national security and loss of human life.

What is the nature of the systems which bare the weighty burdens of operating on the leading edge? They are systems of systems; likely the integration of multiple large scale systems, old and new. They operate under complex and often conflicting objectives. They must demonstrate dynamically adaptive behavior, to adjust to changing operational environments, say from peacetime to hostile threat. The computational engines for these systems are typically highly parallel, use non-homogeneous resources, employ embedded software, are expected to be long-lived, and incorporate many conflicting functional and nonfunctional objectives. Hard real-time performance requirements demand continued operation even under adverse or otherwise degraded conditions.

For any realistic operational scenario, the successful employment of any of these systems requires synergy in operation of complex hardware, sophisticated software, and from one to hundreds of highly trained human operators (Figure 1.1). Conventional hardware components are the backbone and skeleton of complex systems. They range from cotton to concrete, from the materials composing the superstructure, hull, and fuselage, etc. to the smallest nut, bolt, and washer. To the distant observer, radar, or satellite, this configuration *is* the system. Other hardware is perhaps better referred to as "chipware." In the complex system, this is the microchip metropolis comprising the very large scale integrated (VLSI) computer circuitry. The high-speed VLSI hardware components provide the low-level logic needed to bear the burden for millisecond response time demands.

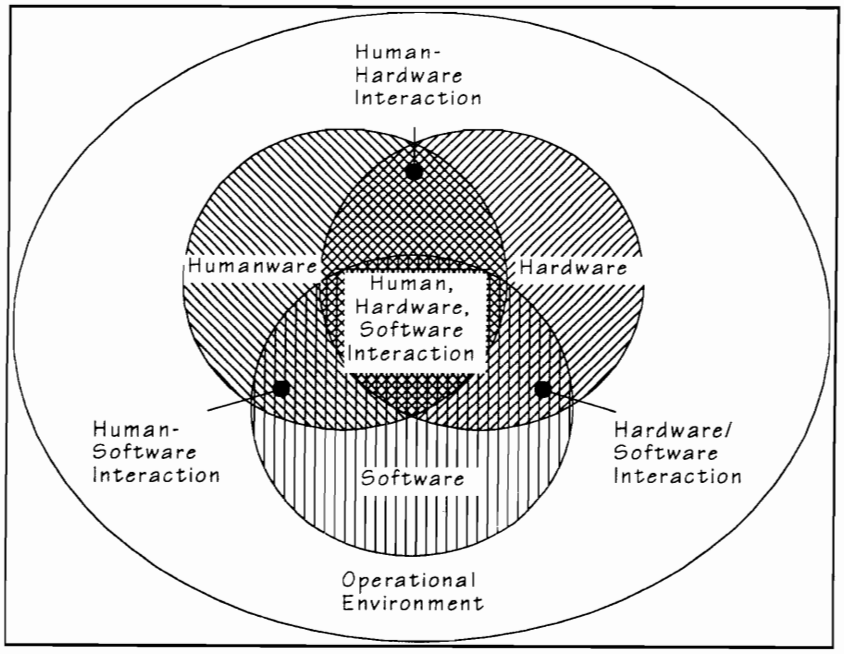


Figure 1.1. Major Components of a Complex System.

Software is the brain and firmware is the central nervous system. They provide the mid-level control logic which governs the dynamic behaviors of the majority of the system's major subsystems. Millions of lines of code written in high order languages (HOL) such as Ada, must integrate "seamlessly" with code written in other HOLs, assembly language, and device-specific machine language. Software systems provide guidance for threat detection, life support, propulsion, etc. The speed, reprogrammability, and multi-tasking nature of software-based components make up in part for the rigidity of hardware and the computational limits of humans. Even so, relative immaturity of the software system quality assurance process and the sheer complexity of massive software systems make this leg of the triad presently the most unreliable.

If the hardware is the skeleton and the software the brain of the complex system, the human operator is the heart — the most vital and most fragile. The dynamic, multi-dimensional, nondeterministic human keeps the system running. Timely resolution of crisis situations is dependent on the human's ability to retrieve and use relevant information, and implement them through interfaces to the system. But

the "man-in-the-loop," while almost infinitely and instantly reprogrammable, is also at once the most complex and the least predictable component in the system. The human alone is subject to the power of emotions such as fear, stress, and physical degradation such as fatigue and illness. The human is the sole element in the system who has a personal stake in the success of the mission. In the human, a family, life, liberty, and the pursuit of happiness share the same gray matter as do protocols, duty, honor, and country. In contrast, the flight deck, cannon, and radar are emotionally impervious to the threat of defeat or the joy of victory.

Physical, cognitive, psychological, and behavioral requirements for the human element of the system vary tremendously with respect to functional role, from "KP[1]" to "CINC[2]." System operators, like astronauts, seamen, pilots, and control room operators typically must undergo prolonged periods of extensive training. They also must demonstrate ongoing proficiency in periodic checkrides or simulators. Their job requires a person completely in charge of his faculties, to hedge against human error which has too frequently been diagnosed as the ultimate cause of disasters. These points punctuate the complex multi-faceted nature of the human operator in the complex system.

Humans, software, and hardware — elements within these domains must function both independently and at sophisticated junctures or interfaces. This requirement for cross-domain interaction coupled with the real-time hardness and gravity of the mission are characteristics which distinguish the complex system from other systems. Two additional distinguishing characteristics are system life-cycle cost and timeline. Defense and space program project budgets are commonly in the billions of dollars, attributable as much if not more to the development of complex control software as to physical materials used in system construction. Timelines for complex systems routinely span more than a decade from requirements capture to design to

---

[1] "kitchen patrol," a tedious, primitive, manual task, e.g. peeling potatoes and washing dishes.
[2] "Commander-in-Chief," the highest position or rank, especially the U.S. Armed Forces.

development to deployment. Extended operational lifetimes (the B-52 bomber has been in operation for nearly 40 years) and the engineering for system degradation, retirement, or recycling can make for a system life cycle approaching half a century.

A complicating consequence arises due to generally invariant cost and timing constraints. In general, fully satisfying these external constraints can prevent fully satisfying functional and performance requirements. As a result, designers are forced to strike a compromising balance between satisfying performance desiderata and staying within the external constraints. For competing designs, evaluation is an important means of determining which design best satisfies all requirements.

Two additional factors relevant to complex system cost and life cycle deserve mention, particularly regarding systems fiscally supplied by public funds. Midkiff and Fabrycky [91] point out that a significant percentage of the system life cycle cost results from design decisions made very early in the system life cycle, but the life cycle knowledge on which to base the decisions generally lags behind the timing requirement to do so (Figure 1.2). In effect, complex systems are programmatically not cost efficient. Additionally, when the nature of a system is subject to the caprices of political climate or national ideology, system requirements or funding or both may change every four years in the United States. Also, current U.S. military policy for the re-assignment of personnel allows for project continuity of only 30-42 months. As a result, a design for a system of national interest, e.g., the space station Freedom, or a space shuttle, may figuratively have to hit a moving target.

The foregoing can be summarized to make this point: the large scale, complex systems in demand for the present and future rely on elements from highly complex domains at various states of maturity which are subject to a broad spectrum of complicating influences. Stringent mission requirements, coupled with cost and timeline constraints for what are generally one-of-a-kind systems, beg for a methodology for system design which builds in fiscal stewardship and human life protection via *measurable* assurances of system success. The elements of such a

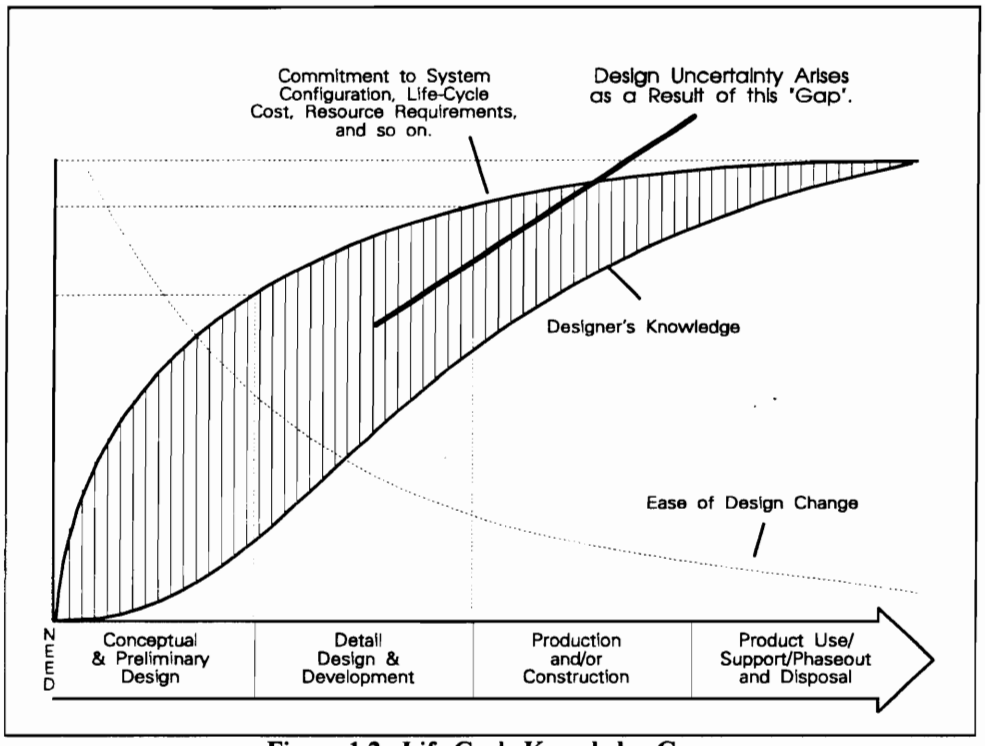


Figure 1.2. Life Cycle Knowledge Gap.

methodology are embodied in the life-cycle focused paradigm of the process of systems engineering.

The systems engineering process (SEP) generally applies to all large scale systems which require human, hardware, and software interaction, and generally have a high cost and lengthy time to deployment. The high level phases of the process are depicted in Figure 1.3 [Blanchard and Fabrycky 90].

A hallmark of the classic SEP is its adherence to a top-down stepwise refinement paradigm, both in the functional decomposition of a system under development and the decomposition of the development process itself. Included in the higher levels of the decomposition are several critical phases of design and subsequent design evaluation.

The three generally agreed upon design phases are (1) **conceptual design**, wherein alternative system solutions are evaluated based on operational, logistical, and financial factors; (2) **preliminary design**, wherein qualitative and quantitative system-

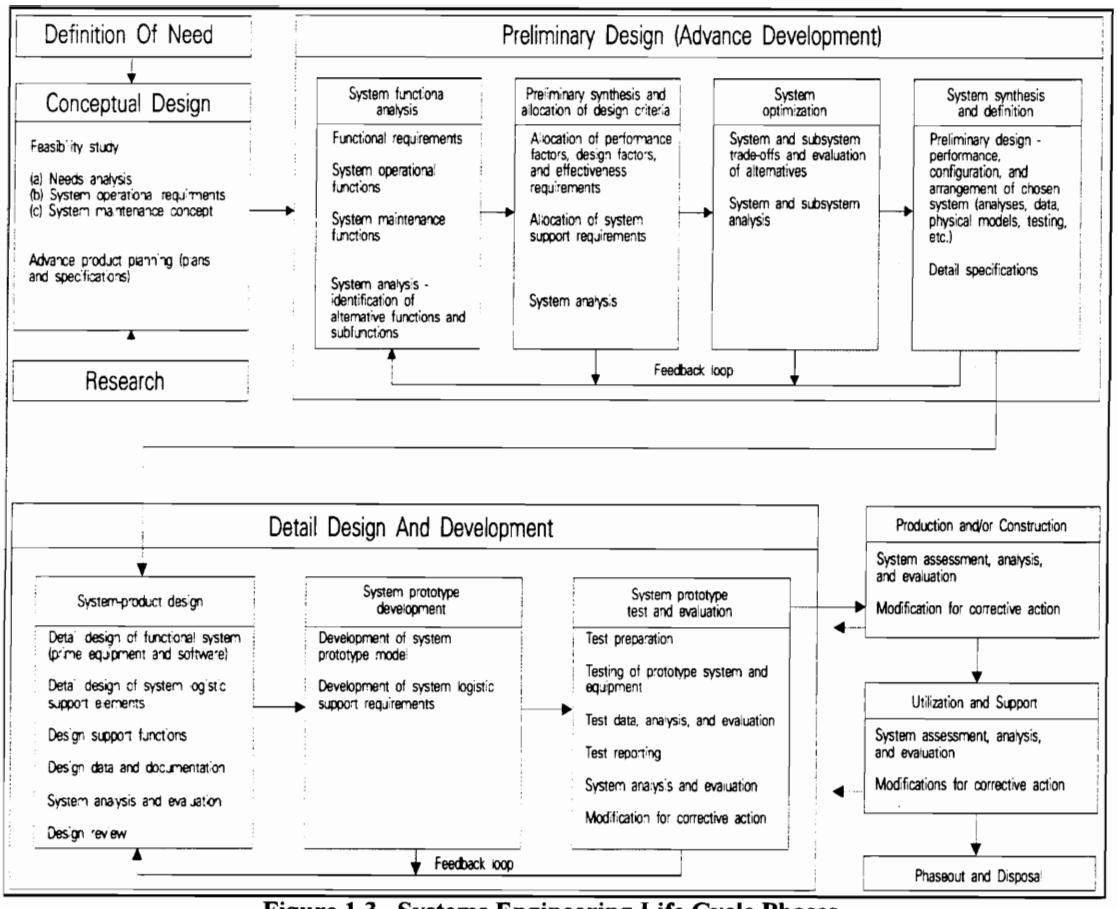


Figure 1.3. Systems Engineering Life Cycle Phases.

level requirements are established based on a system functional decomposition and functional requirements allocation; and (3) **detailed design**, in which subsystems are refined and detailed, functional hardware and software modules are developed, and basic system models are tested. Subphases are not excluded.

Given that consequences of programmatic cost inefficiencies as mentioned above can include reduced operational capability or even system project termination, the evaluation of the potential for system success early in the process is a primary focus of the SEP. Approaches to evaluation include cost, performance, logistics, reliability, etc., trade-off studies, and optimization at increasing levels of granularity as the system progresses through its life cycle. Diverse techniques and tools exist for piece-wise optimization, and as the technology to support tools and methodologies matures,

attention and efforts turn aggressively toward early integrated evaluation of the system design, a discipline still striving for maturity. More evaluation frameworks and tools are needed to support this ambitious endeavor, to narrow the gap between assurances of system efficacy and commitment of funds early in the system life cycle.

1.3. RESEARCH GOAL AND OBJECTIVES

The primary goal of the research is to develop a methodology which structures, guides, and assists in the measurement and evaluation of designs for complex systems. To enhance the credibility of the methodology and lay a foundation for its assessment, we present a set of objectives to serve as a requirements specification for the methodology.

1. The methodology should facilitate the measurement and evaluation of a complex system design's dynamic (i.e., time-critical, performance-critical, mission-critical) characteristics.
2. The methodology should facilitate the measurement and evaluation of a complex system design's real-time characteristics.
3. The methodology should facilitate the integrated measurement and evaluation of a complex system design with respect to its hardware, software, and humanware components, and their interfaces.
4. The methodology should facilitate the qualitative and quantitative evaluation of a complex system design.
5. The methodology should be generically applicable for a broad spectrum of complex system designs.
6. The methodology should lend itself to the incorporation of computer-aided assistance.
7. The methodology should possess inherent, piece-wise credibility.
8. The methodology should be easily usable by a design evaluation organization.

9. The methodology should be applicable for the preliminary and detailed design phases of the system engineering process.
10. The methodology should promote independent system design evaluation to prevent developer's bias.

1.4. DISSERTATION ORGANIZATION

Chapters 1-3 largely define the research problem and overview a solution. The present chapter defines the nature of complex systems, and the criticality of the design evaluation process. Chapter 2 is a review of techniques and methods currently employed as partial solutions to early system life cycle design evaluation. Chapter 3 is a largely pictorial overview of the methodology, enumerating the major phases and their subphases, and placing them in the larger design evaluation scenario.

Chapters 4-6 detail the mathematical techniques employed in the methodology. Chapter 4 presents the process of parsing a design into major indicator domains, and identifying design quality and utility indicators through hierarchical decomposition of the major domains. Chapter 5 details the application of Analytic Hierarchy Process to determining each indicator's relative criticality. Chapter 6 , and indicator scoring, respectively.

Chapters 7-9 provide synthesis of the methodology's components, and draw closure to the document. Chapter 7 describes how a computer-assisted expert knowledge base is used to support all phases of the evaluation process. Chapter 8 presents the methods and results of an independent assessment of the methodology by systems engineering researchers and practitioners from the DOD, other government agencies, industry, and academia. Chapter 9 provides conclusions from the research, and enumerates recommendations for future research.

1.5. SUMMARY OF CONTRIBUTIONS

Chapter 9 presents the contributions of this research. A well-formed foundation has been laid for the computer-assisted application of qualitative as well as quantitative

expert knowledge to the integrated measurement and evaluation of complex system designs. These contributions, summarized below, describe the components and their synthesis, which provides the integrated nature of the methodology.

- The domain dominance schema analysis method establishes the criticality of design elements respective of their domain. It also provides guidance for the composition of the evaluating panel, to the extent that domain expertise requirements must be commensurate with the domain dominance schema.

- The indicator-based evaluation approach provides for two dimensional division of the evaluation effort: in breadth, by dividing respective of domain, and in depth by hierarchical decomposition of indicators within each domain.

- Application of the Knowledge-Based Evaluation of System Designs (KBESD) software tool provides a graphically-oriented computer-based environment for implementing the indicator-based elements of the evaluation.

- Indicator hierarchy assessment increases assurance that the correct design evaluation problem is represented by the indicator hierarchy.

- The Analytic Hierarchy Process, adapted as we illustrate, is a credible means of determining individual indicator weights of criticality.

- Domain-specific design evaluation metrics incorporate objective assessment of known domain-specific quantities.

- System simulation provides a source for measurement and evaluation of the designed system's dynamic and time-critical elements.

- Interval and fuzzy mathematics provide accountability in the application of subjectivity, firmly based in the expertise of technical and operational domain expertise.

- The expert knowledge base approach provides computational and inference engines, comprising an integrating computer-based framework for applying the methodology.

2. LITERATURE SURVEY

2.1. INTRODUCTION

To introduce the reader to the general concepts and terminology used throughout this dissertation, we present a summary of the relevant literature. The literature survey addresses two separate aspects of design evaluation. We address each aspect in a separate major section. Section 2.2 surveys recent and classic literature addressing the design and design evaluation processes. Specific topics covered include the systems engineering process, design evaluation concepts, and the terminology associated with indicator-based evaluation. Section 2.3 presents methods and mathematical paradigms which have been employed in design evaluation. Section 2.4 states appropriate conclusions.

2.2. SYSTEMS DESIGN AND DESIGN EVALUATION

We begin with a discussion of the systems engineering process, which circumscribes the entire evaluation process. We follow with specific components of design assessment, leading up to indicator-based evaluation techniques.

2.2.1 The Systems Engineering Process

The "big picture" of the systems engineering process is presented in Chapter 1, Figure 1.3. The systems engineering process is continuous, iterative, and incorporates the feedback actions necessary to ensure convergence. We provide summaries of the requirements and design phases, excerpted from [Blanchard and Fabrycky 90] and elsewhere as indicated.

2.2.1.1 Phase I: Definition of Need/Requirements Capture

This phase of the process establishes first things first. Specific requirements for the system are established. These include: operational and functional performance, delivery time, mission profile, distribution and deployment, applicable effectiveness measures, environmental requirements, operational support and maintenance, and

expected lifetime. Lim *et al.* [92a; 92b], West *et al.* [90], Ramesh [92] and Rouse *et al.* [91] address what they consider the acute need to focus on human factors considerations at this early phase. There is an intensifying focus on the importance of accurate and traceable requirements capture. This is reflected in tools and models described in numerous works presented at recent annual meetings of the Complex Systems Engineering Synthesis and Technology Assessment Workshop (CSESAW). See e.g., [Palmer and Evans 94; Nallon and Edwards 94; Rundlet and Miller 94; Jeffords 94; Hugue *et al.* 94; Halligan 93; Karangelen 92]. Punctuating this increased emphasis, Ramesh [94] advocates a traceability model implementable at the *pre-requirements* stage, based mostly on experience and judgments from panels and working groups from particular classes of application domains.

2.2.1.2 Phase II: Conceptual Design

The conceptual design phase begins with the requirements specification. The activities of the previous discussion are often the front end of the conceptual design phase. Evaluators conduct feasibility analyses, establishing a valid need for a system and determining if existing technology can be integrated into a solution, or if development of new technology is warranted. Forging the foundations of the system maintenance concept and maintenance-related issues pertinent to advance product planning is also tackled in this early stage.

Given operational and maintenance requirements as goals, systems engineers conduct a preliminary system analysis to determine different technical means by which the system requirements can be met. Candidate solutions are evaluated to select the one most appropriate. The relative paucity at this stage, of firm values for system measures of merit, leads to creative analysis and evaluation techniques for choosing among alternatives. Dong [87], Verma [94] and Verma and Knezevic [94] employ fuzzy mathematics. Diteman and Stauffer [93] and Hyde and Stauffer [90] present a comparison of alternative comparison techniques. They find that absolute comparison of alternatives to a standard performs more reliably than pairwise comparison among

alternatives. Thurston [91] suggests a method for evaluating design alternatives which is based on utility theory, but with positive implications for the iterative nature of the re-design process.

At the end of the tradeoff studies, a ranking of candidate alternatives is established, and the "best" candidate is pursued for further development. A conceptual design review is conducted to ensure the design is traceable to stated requirements, and is appropriately and correctly documented. The design review documentation serves as a baseline for the preliminary design phase.

2.2.1.3 Phase III: Preliminary Design

This phase begins with the results of the conceptual design phase, normally a technical baseline for the system based on the detailed requirements specification. Requirements are mapped onto qualitative and quantitative requirements for functional subsystems. Functional analysis is performed to determine *what* the system is to do, not necessarily *how* the functions are to be accomplished. Both operational and maintenance functions are considered, in parallel when possible, to ensure sufficient maintenance functions are provided to support sustained operations. Functional and operational requirements, e.g. system effectiveness factors and physical maxima and minima, are allocated down to the functional subsystem, component, subcomponent, or even unit level. The same is done for support and maintenance requirements as well as life cycle cost factors.

Functional allocation involves determining what media will be used to implement a given function. Automated tools exist which aid design engineers in making functional allocation decisions. Midkiff and Fabrycky [91], Tarnoff [91], and Webster [87] describe a design decision support system for determining functional allocation between hardware, software, and firmware. The comprehensive *DeStinAtiOn* methodology [Howell, *et al.* 92, 91] developed for the Naval Surface Warfare Center, is used by design engineers for generating design allocation alternatives, and performing functional allocation trade-off analyses.

Trade-off and optimization are performed to attempt to achieve the most desirable level, or determine the most achievable level, of each indicator of measurement. Analytical methods and models are often required to implement the required computations and resultant representations. These aid the engineer in selecting achievable levels of each system parameter. Contemporary examples of optimization tools include *CEO*, a GUI-based cost effectiveness optimizer available for the Microsoft Windows™ platform [Pukite and Pukite 94] and SHARMA [Alqadi and Ramanathan 93], a tool for synthesizing heterogeneous computer systems for real-time complex system functions. These tools and those in their class can be quite sophisticated and were developed in response to the growing need for highly evolved automation in complex system design analysis.

As in conceptual design, there are many unknowns which must be grappled with in the trade-off process. Traditional utility theory techniques and Taguchi methods [Taguchi 86] are still widely used, but other recent techniques have been proposed. Total Quality Management is used to improve a product by improving the processes which lead to its development (see, e.g., Stoll and Zubas [94]). Hopgood [89] describes a knowledge-based trade-off and selection technique. Wood and Antonsson [91, 90, 89], Otto [93], and Otto and Antonsson [93] propose an alternative "method of imprecision" for conducting evaluation and tradeoffs under uncertainty.

System synthesis is concerned with combining system elements in a way which forms a functional system. Synthesis functions include allocation of requirements to functional and nonfunctional components, and determining techniques for testing, operating, and supporting system elements. Example tools and methodologies from the literature are discussed by Howell, *et al.* [92, 91], Min and Chang [93], and Zimmerman and Sebastian [93].

Results of trade-off and optimization studies lead to a set of design-to requirements which will be implemented in physical systems and algorithms in the

detailed design phase. The preliminary design review provides a comprehensive suite of documentation which serves as the baseline for the detailed design phase.

Phase IV: Detailed Design and Development

Following the results of the preliminary design phase, a system moves into the detailed design phase. True to its name, this phase is where all the details are hashed out and carefully documented. The activities of this phase include describing subsystems down to their lower level subcomponents, preparing detailed design documentation (e.g. detailed drawings, flowcharts, etc.), definition and development of computer software, development of system and/or subsystem prototypes, development of a comprehensive system model, and test and evaluation of the system model.

System effectiveness measures, technical performance measures, and cost effectiveness measures are to be designed into specific components at this stage. Computer Aided Design (CAD) and Computer Aided Engineering Design (CAED) tools are employed to assist in preparing detailed diagrams, etc. for circuit board layout, and system superstructure and infrastructure layout. Design engineers employ specialized decision support systems and analysis tools for evaluation of components. This is to gauge the degree to which prototype or actual components demonstrate compliance to the associated figures of merit, performance and functional indicators. A discussion of the general analysis model is given by Blanchard and Fabrycky [90] and Midkiff and Fabrycky [91]. Tools described in [Kim and Bacellar 93], [Howell, *et al.* 90], and [Andert and Peters 93] are representative of tools appropriate to this stage of design evaluation. Some of the tools referenced above are also used at this stage in system evolution.

Before the completion of this most rigorous of the design phases, the system design undergoes a formal design review. This includes equipment design reviews based on requirements specifications, diagrams, computer data, and component parts lists. Bugs, inconsistencies, and incompatibilities are reported and corrective action is prescribed. After resolving discrepancies uncovered in the formal design review, the

design is essentially frozen and undergoes a critical design review to ensure the discrepancy resolutions are satisfactory. It is important to note that a great deal of planning and participation is required of many engineers and specialists to ensure a thorough, valid, and effective critical design review.

2.2.2 Design Measurement and Evaluation

A summary of the design phases of complex system engineering is presented in Section 2.2.1. The key concepts which bear on this section are as follows:

(1) System design is a critical part of the system life cycle. Changes to the system are much less costly early in the life cycle. In later phases, when task allocation and commitment to particular architectures have been decided, design changes can result in delay of system delivery. Changes may also result in broken contracts, which can lead to expensive lawsuits and even premature project termination.

(2) Design evaluation, which occurs throughout the design phases, is driven by operational and functional measures of merit, established in the requirements solidification phase.

(3) Evaluation is often difficult, requiring computations under uncertainty, since complete information is rarely available to evaluators when it could be the most useful for ensuring life-cycle cost goals can be met.

In the following subsections we address issues pertinent to design measurement and evaluation. There are various terms used for the measurable quantities and qualities associated with a system and its associated design. These following terms are used almost synonymously in the literature: metric, measure, factor, attribute, index, figure of merit, merit index, measure of merit, and indicator. The term "indicator" is used herein to represent any of the above terms, except as otherwise indicated. An indicator can be a directly measurable quantitative value, or an indirectly measurable

qualitative value. Assessment and assignment of values to system design indicators are at the crux of the research approach described herein.

2.2.2.1 Integration emphasis

Complex systems require synergism of humanware, hardware, and software, whose performances intersect at well-established interfaces (refer to Figure 1.1). Integrated evaluation is required to ensure those synergistic requirements of the system which are satisfied if the design is implemented correctly [Blanchard and Fabrycky 90]. Take a naval surface vessel as an example. Based on individual domain evaluations, it may at some point be accepted that the computers and the radar and crew, etc. all separately meet specifications. Still, a very strong indication is needed that they all will work synergistically in all conceivable scenarios before the Navy would be willing to deploy the finished product and hundreds of lives in a hostile environment.

Not surprisingly then, any consideration of obtaining a system-level evaluation of the design should emphasize the integrated nature of the system. But the research heretofore has focused primarily on the evaluation of smaller pieces of the system, ranging from individual software modules up to complete subsystems. In recent years, the increasing emphasis on human factors issues (human-computer and man-machine interaction) has led to the standard practice of at least pairwise cross-domain evaluations. A likely reason for the slow maturity of system-level integrated evaluation is that it has been easier to develop evaluation tools for essentially autonomous systems, than for systems which interact with other systems. On the other end of the spectrum is the realization of the unfathomable size which a representation of every measurable quantity and quality of a complex system design would be.

Nonetheless, system-level gauging of design quality and system efficacy requires examination of humanware, hardware, and software systems not only separately but also as they interact with and depend on one another. For any reasonably comprehensive level of abstraction, this could lead to a large number of composite indicators to examine. Additionally, these indicators span a wide range of

not only engineering disciplines, but human factors and psychological disciplines as well. Experts from the many disciplines as well as interdisciplinary experts are required for a comprehensive evaluation.

2.2.2.2 Representing System-Level Indicators

It is at best a difficult task to collect a meaningful and manageable set of integrated system-level indicators. It is perhaps more difficult to accurately depict the sometimes complex relationships and dependencies among the indicators. But once a set of acceptable indicators has been determined, a taxonomy or hierarchy is an appropriate representation device. We now present a brief elaboration on taxonomies and hierarchies that provide sufficient background for their application in later sections.

2.2.2.3 Taxonomies

Hyde [89] has studied techniques for assigning and combining weighted scores for elements in a taxonomy for the purpose of system evaluation. He offers two criteria which a taxonomy should meet. The first is that it should express *perceptional orthogonality*. This means each element at a given level should be perceived as being mutually exclusive from every other. The implication is that no element should fall under any two "parent" elements in the taxonomy. We see in Chapter 4 how this principle must sometimes be violated. The second principle is that a taxonomy should be *exhaustive* over a domain, to be adaptable to any scenario within that domain. As Hyde [89] concedes, this, like the first, is certainly a restrictive requirement, which almost any publishable taxonomy violates. Nonetheless taxonomies appear in the literature in almost every discipline. The general paradigm for Hyde's approach is the positional role elements in a taxonomy play. Highest level elements are called *dimensions*. Dimensions are characterized by *attributes*, which represent a lower level of abstraction. Attributes in turn are assigned a *value* at the lowest level of the taxonomy.

Meister [89] discusses literally a taxonomy of taxonomies, generally without regard to structure. Two particular taxonomic forms applicable to this research are the *research taxonomy* and the *taxonomy of measures*. According to Meister [89 p. 120] "The great value of the research taxonomy is that it organizes and structures the variables the researcher must deal with, thus expanding the researcher's understanding of the interrelationships among those variables." This notion of understanding relationships and dependencies among parameters of interest is shown to be a key to design evaluation as we intend it. Meister [89] also asserts, without elaboration, that a taxonomy of measures should always accompany a research taxonomy. We suggest a reason for complementary nature of the two taxonomic types is that the set of indicators corresponding to a research parameter enable the researcher to gauge the degree to which variance of the parameter's value is accounted for.

2.2.2.4 Hierarchies

The hierarchy is a less restricted form of taxonomy. The characteristic of exhaustivity is still desirable, but the hierarchy recognizes multiple relationships between elements. Saaty [86] presents detailed axiomatic foundations of analytic hierarchies as a foundation for the Analytic Hierarchy Process (see Section 1.2.4). Saaty [90a,90b,93,94] extends the earlier work to include the priority-setting advantage of the hierarchical representation. Through pairwise comparisons among sibling elements relative to their parent, local priority among the siblings can be determined. These comparisons are based on a ratio scale technique, which lends special properties to the entire hierarchy. Multiplicative combinations of local priorities down the hierarchy allow establishment of global priorities for all elements. This priority generation, in various forms, augments the power of the taxonomy to include *degree* of relationship between elements. Both the existence and intensity of relationship prove important elements of the evaluation scheme described in Chapter 3.

An example of application of indicator hierarchies is found in the Objectives, Principles, and Attributes (OPA) framework for evaluation of software system development methodologies [Arthur and Nance 87,91]. The OPA framework is similar in structure to that of Hyde [89], except applied to a hierarchy instead of a taxonomy. At the top level of the hierarchy are the *objectives* for the entire software development effort. Governing *principles* guide the process of system development. Principles relate to objectives in a many-to-many relationship. Finally, adherence to the principles is indicated by a set of *attributes* which characterize the finished product. An example OPA hierarchy is presented in Figure 2.1.

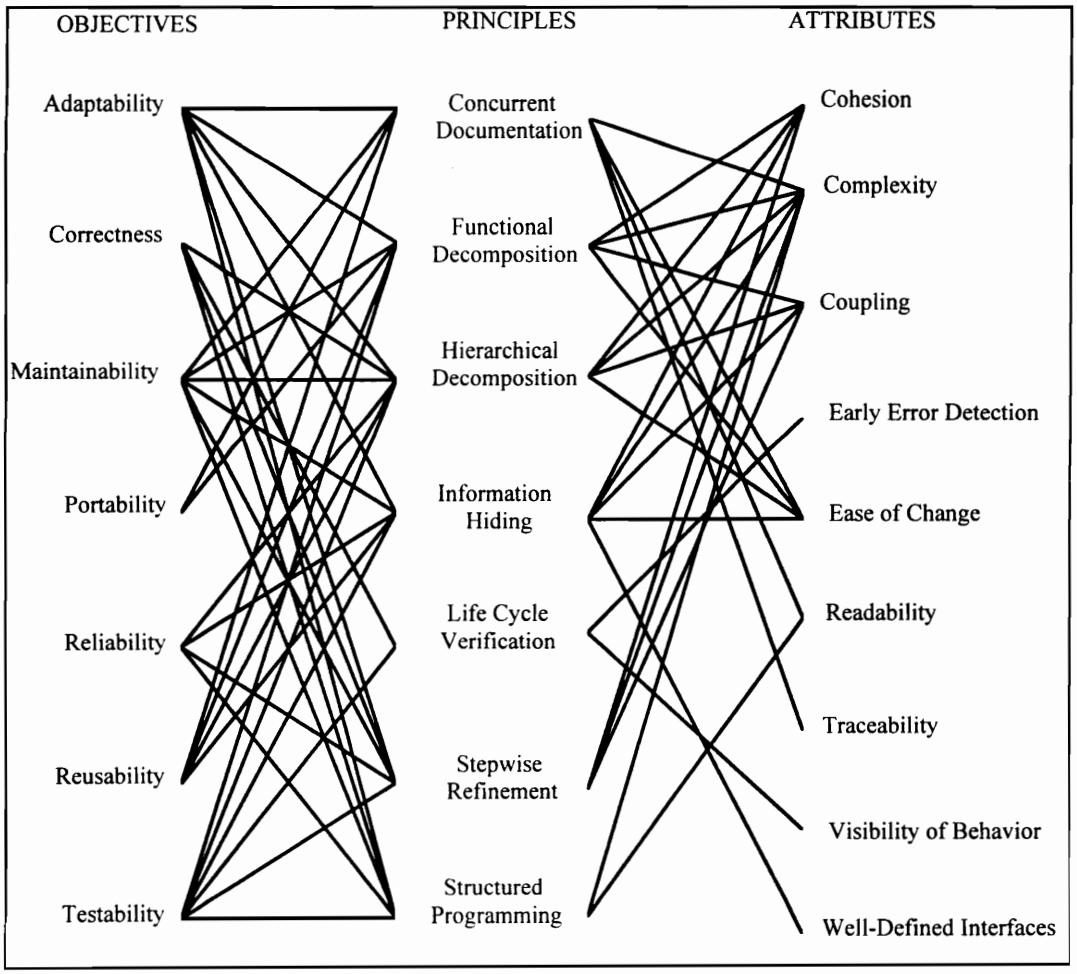


Figure 2.1. Objectives, Principles, and Attributes Diagram.

2.2.2.5 System-Level Indicator Hierarchies

Given that hierarchies of parameters are recognized as valuable tools for research and evaluation, then such structures are applicable to the representation of indicators for complex system designs. For years in the literature there have been examples of collections of indicators, little more than laundry lists and largely without regard to indicator relationships and dependencies. Blanchard and Fabrycky [90] have culled indicators from experience in systems engineering research. Similar collections of indicators are gathered in [Card and Glass 90; Möller and Paulish 93] for software metrics and in [Meister 89,91] measurement of the human in a system.

Also appearing in recent literature are some taxonomies developed for the expressed purpose of system evaluation, though none represent significant depth in structure or substantial numbers of indicators. We see these efforts to superimpose causal and comparative relationships on otherwise unordered lexicons of indicators as important first steps in system-level assessment and evaluation. Taxonomies or hierarchies from the literature which specifically deal with systems and system designs are presented in the remainder of this section. Diteman and Stauffer [93] follow Hyde [89] with a broad and general representation (Figure 2.2). Thurston [91] presents another high-level taxonomy of desirable performance characteristics for a system under design (Figure 2.3). Perhaps the most comprehensive effort to date is a taxonomy of system design factors proposed by Nguyen and Howell [92,94] shown in Figure 2.4. The authors have been involved in system-level design factors for several years, and are involved in a large effort to employ evaluation of complex systems designs by system-level indicators. Still, their taxonomy, while providing excellent coverage in breadth, does not represent many of the indicators at subsequent depths.

The shortcomings of these efforts point to a need for a more focused and yet more in-depth coverage of the indicators for complex system design evaluation. A new direction of attack, we believe, is appropriate for attaining both goals.

SYSTEM AREA	**Dimensions**	***Attributes***
LIFE CYCLE	Resource Management	*Lead-Times*
		Equipment Utilization
		Personnel Deployment
	Manufacturing	*Production*
		Quality Control
		Assembly
	Transportation	*Packaging*
		Shipping
		Storage
		Installation
	Performance	*Energy*
		Forces
		Geometry
		Dynamics
		Materials
		Signals
	Maintenance	*Preventative*
		Repair
		Product Life
	Disposal	*Degree of Hazard*
		Recyclability
HUMAN-MACHINE INTERFACE	Safety	*Direct Safety*
		Indirect Safety
		Warning
	Ergonomics	*Mental Ergonomics*
		Physical Ergonomics
	Aesthetics	*Visual Order*
		Perception of Function
		Cultural Style

Figure 2.2. Example System-Level Hierarchy.

Given the integrated approach to complex systems design evaluation, we choose a top-level parsing into seven major sections. These include "humanware," "hardware," "software," human-software, human-hardware, and hardware-software and the "hyper-interaction" domain in which all three major domain components interact. This launching point facilitates a more straightforward aggregation of the indicators described in the literature, which generally focus on a single domain or interface.

For examples from the literature which address a more narrow evaluation scope, see, e.g. [Aas, *et al.* 89; Amalberti 92; Arthur and Nance 91; Arthur, *et al.* 87; Ashlund and Hix 92; Ballas, *et al.* 90; Banks, *et al.* 88; Barnett 90; Becker, *et al.* 91; Blackman 90; Blackwell and Cuomo 91; Boy 87; Brown *et al.* 91; Butler and Finelli 93; Byers, *et al.* 91; Cacciabue 92; Callahan, *et al.* 90; Card and Glass 90; Carter and Wachtel 92; Chu, *et al.* 91; Crawford 92; Cuomo and Bowen 92; Cuomo and Rizzuto 90; Flach and Bennet 92; Goodman 93; Hahler, *et al.* 91; Halbert, *et al.* 92; Harbour and Hill 90; Heinecke 93; Hetzel 93; Hewett 90; Hollnagel 92; Jahanian 93; Johannesen and Woods 91; Jones and Biers 92; Jones and Mitchell 91; Jones and Mitchell 91a,91b; Jubis 90; Kapasouris, *et al.* 91; Kirkpatrick, *et al.* 90; Knapp and Vardaman 91; Knapp, *et al.* 91; Kopetz 93; Lala and Harper 93; Langen and Rau 90; Lee 93; LeMay and Comstock 90; Levendel 93; Lim, *et al.* 92; Locke 93; Malone, *et al.* 92; Meister 89; Merwin and Wickens 91; Meshkati 88; Min and Chang 93; Mitchell and Williams 93; Modrick 92; Möller and Paulish 93; Ntuen 91a,91b; O'Hara 91,92; Paula, *et al.* 93; Pawlowski and Mitchell 91; Peters 91; Plocher, *et al.* 91; Rasmussen and Vicente 89; Reichardt 93; Reid and Nygren 88; Rouse and Hammer 91; Rowe 92; Rueb, *et al.* 92; Salvendy 87; Sarno and Wickens 92; Selcon, *et al.* 91; Smith 93; Smith and Williams 93; Stassen, *et al.* 90; Staveland 91; Ujita 92; VanderWeil, *et al.* 93; Verfurth, *et al.* 91; Vidulich 88; Visciola, *et al.* 92; Weiland, *et al.* 92; Weisgerber and Kibbe 90; West and Adelman 90; Whitaker, *et al.* 90; Whitaker, *et al.* 90; Woods 90a,90b; Woods and Eastman 89; Woods, *et al.* 90; Woods, *et al.* 92; Xu and Parnas 93; Yoshimura, *et al.* 92; Zinser and Henneman 88].

Major Area	SubArea	SubSubArea
Management/ Economic		
	Development Cost	
	Profit Margin	
	Manufacturing Cost	
		Facilities
		Materials
		Tooling
		Energy
		Assembly
		Labor
	Manufacturing Flexibility	
		Ease of Assembly
		Number of Product Styles
		Parts Consolidation
		Product Turn-around Time
	Time to Market	
	Styling Flexibility	
	Expected Market Share	
Engineering/ Structural		
	Torsional Stiffness	
	Vibration	
	Yield Resistance	
	Weight	
	Buckling Resistance	
	Fatigue	
	Service Life	
	Temperature Resistance	
Customer Satisfaction		
	Useful Product Life	
	Fuel/Energy Efficiency	
	Reliability	
	Appearance	
	Service Call Rate	
	Corrosion Resistance	
Environmental		
	Recyclability of Process Scrap	
	Recyclability of Product	
	Waste and Emissions	Air
		Water
		Solid/Hazardous

Figure 2.3. Example System-Level Hierarchy.

- Performance
  - Response Time
  - Capability
  - Relative Activity
  - Speed
  - Throughput
  - Latency
  - Load Balancing
    - Information Load
    - Processing Load
  - Graceful Degradability/Load Shedding
  - Efficiency
  - Predictability
- Real-Time
  - Hardness
  - Hard Deadlines
  - Soft Deadlines
  - Temporal Distance
  - Tardiness
  - Number of Consecutively Missed Deadlines
  - Predictabilities
  - Graceful Degradation
- Computation/Processing Requirements
  - Importance
  - Usefulness
  - Priority
  - (Computing) Portability
  - Interrupt/Reset Capabilities
- Dependability
  - Reliability
  - Accuracy
  - Fault Tolerance
  - Graceful Degradability
  - Redundancy
  - Availability
    - Inherent
    - Achieved
    - Operational
    - Ease of Replacement
    - Crash Recoverability
    - Computation Heavy
- Process Effects
  - Quality
- Security
  - Classification
  - Type of Data
  - Percentage of Processing
- Time/Load
  - Encryption Type Requirements
  - Implementation Techniques
- Humanware
  - Ease of Use
  - Potential Operator Decisions
    - Operator Delay
    - Response Time
  - Operator Actions
    - Required No. Operators
    - No. of Concurrent Users
  - User Intensity
  - Avg. Time for Each Category
  - Potential Errors
- Physical Requirements
  - Size Requirements
  - Weight Requirements
  - Ruggability
  - Survivability
  - (Physical) Portability
  - Energy Requirements
    - Energy Consumption
      - Electrical
      - Fuel
      - Other
    - Energy Dissipated
  - Operating Environment
    - Geographical Location
    - Indoors/Outdoors
    - Temperature
    - Humidity
    - Acoustical Noise
    - Air Purity/Quality
    - Exposure (wind/EM)
    - Vibrations/Stability
  - Climate Control
  - Manufacturing Considerations
  - Computer
    - CPU
    - Memory
    - Storage
- Financial Requirements
- Time Projected
- Life Cycle
  - Testability
  - Maintenance
  - Obsolescence Lifetime
  - Reusability
- Future Needs Considerations
  - Adaptability
  - Expandability
  - Compatibility
  - Integrability
  - Interoperability
  - Integrity

Figure 2.4. System Design Factors.

2.2.3 Indicator-Based Evaluation

In this section, we describe the general technique of evaluation by indicators. Certainly, this technique has been around for many years most notably in the form of the utility theory from von Neumann and Morgenstern [47]. The general method is to assign a value to each indicator, weight the value by the importance of the indicator to the entity under evaluation, and then aggregate the weighted scores for an overall assessment. Recent publications in the literature show variations on the basic theme [Otto and Antonsson 91,93; Thurston 91; Keeney and Raiffa 93]. We present example techniques for indicator scoring and weighting, and for the process of arriving at an overall aggregate score.

2.2.3.1 Indicator Scoring

With the foregoing in mind, we turn now to variations for assigning scores to indicators which have appeared in the recent literature. In brief, we discuss simple single values, a probabilistic or fuzzy range of values, and a nonlinear function to assign values.

Crisp values are the most easily assigned. These correspond to metrics as defined above. They can also be values from a scale, say 1-10 (see discussions in [Webster 87; Hyde and Stauffer 90; Hart and Staveland 88; Reid and Nygren 88]). Hyde and Stauffer evaluated several popular types of psychometric scaled scoring methods and found global scales, both single- and multi-dimensional, to be more applicable to design evaluation than their guided (flow-chart-oriented) counterparts.

Values which are not crisp may be represented as ranges or as fuzzy values. A range includes confidence intervals generated through simulation, vague terms which stand for ranges, e.g. *low* or *high*, or vague terms which have ranges with fuzzy boundaries. The latter are often related to fuzzy values, which originated in the area of fuzzy mathematics [Zadeh 65]. An overview of the foundations of fuzzy mathematics is provided in Section 2.3.5. Examples of scoring via fuzzy values are found in [Ayyub and Eldukair 89; Cui and Zao 91; Dembicki and Chi 91; Dhingra and

Moskowitz 91; Dubois 92; Fukuda 89; Hadipriano 89; Hadipriano and Ross 91; Iqbal and Kinzel 92; Kraslawski, *et al.* 93; Kubic and Stein 88; Paek, *et al.* 92; Rao, *et al.*; Thurston and Carnahan 92; Vadde, *et al.* 92; Verma and Knezevic 94; Voland and Thuan 89; Wood and Antonsson 90; Zhang and Gaggioli 92].

A hybrid type of indicator score is discussed by Thurston [91]. It is adapted from general utility theory and the lottery method of Keeney and Raiffa [76,93]. For each indicator, Thurston adapts a range of values determined by a nonlinear utility function $U_i(x_i)$. The method generates a value which is computed as follows. For a single attribute (indicator), present the designer/evaluator with two alternatives: one with a value, A_1, which can hypothetically be achieved for certain, and one, A_0, for which there is a pair of extreme values, with probabilities p and 1-p of achievement, respectively. For the value A_1, it is determined at what value of p for A_0, the evaluator is indifferent between A_1 and the uncertain value at probability p.

The resultant value for p completes the ordered pair (A_1,p). A_1 is then adjusted to halfway between its value and the last value at which the uncertain alternative is preferred, and the process iterates. The series of ordered pairs form the utility function $U_i(x_i)$ for attribute x_i. Plotted on a graph, the abscissa, would span the range of values for A_1 and the ordinate would range from zero to one. Thurston references ASSESS, a computer program which assists in automating and displaying the value computations. Note that the range of values for A_0 could be generated by simulation, and the value for p could be a judgment of specialist or domain expert, depending on the nature of x_i.

2.2.3.2 Indicator Weighting

There are certainly as many variations on the theme of indicator weighting as for score assignment. The easiest and yet probably the least traceable or reliable method is simply *brute force*. An evaluator relies on experience or a standard of some sort and just assigns a criticality factor to weight an indicator. The weight can then be normalized by the sum of all weights for the collection of indicators. Depending on the purpose of the weighting scheme, normalization may not be desirable.

A more concrete approach is mentioned by Blanchard and Fabrycky [90]. They employ this technique in reliability assessment by assigning criticality to a component based on, e.g., the normalized number of subcomponents it contains. A similar approach can be applied, e.g. to the number of methods in an object, the number of external function calls in a software module, etc.

A probabilistic class of weighting techniques is employed in the various literature on modeling and prediction. The objective is to derive coefficients for a subset of indicators and a possibly non-zero constant, to form a regression. An example predictive regression equation for four indicators would be:

$$P = c + a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4,,$$

where P is the predicted value, c is the constant, $x_1..x_4$ are the variables, and $a_1..a_4$ are the regression coefficients. The coefficients are effectively the probabilistic weights applied to the variables. This technique is most appropriate to low level prediction scenarios, where the variables can actually be measured, as in the case of metrics.

Other methods apply techniques to manipulate human judgment. Thurston [91] employs a *scaling factor* in the general utility equation and a lottery technique similar to the score assignment procedure described above. The weight of criticality for a particular indicator is determined similarly, but for a single value as opposed to a function. In this procedure, all attributes are considered together. The hypothetically certain case is given where the indicator under evaluation is considered to be at its optimum value, and all others at their worst. The uncertain alternative is represented at the two extremes of all attributes at their best and worst, again with probabilities p and 1-p, respectively. The scaling factor, k_i, for indicator i, is the probability p at which the certain alternative is equally as preferable to the other alternative. Thurston attempts to draw a distinction between this scaling factor and the concept of a weight or relative criticality, but the argument lacks sufficient evidence to be persuasive. Additionally, Thurston employs the scaling factor in the same manner as other utility-

based methods apply a weight factor. Nonetheless, the approach is a valid one, and, as with the scoring technique, it can require simulation and expert judgment to implement.

A final weighting technique from the literature which mathematically manipulates human judgment comes from the analytic hierarchy process [Saaty 80,94] It is an enhancement of other pairwise comparison techniques such as discussed in [Otto 93]. The details of the technique are presented in Section 2.3. To be used as a weighting scheme, all the elements under consideration must be organized into a hierarchy. Then the evaluator makes pairwise comparisons of relative importance among all sibling children elements of a particular parent. From a matrix of the pairwise comparisons, a relative weight, local to each element in the immediate subset can be determined. The local weights within a subset sum to unity. For an entire hierarchy, global weights can be determined by multiplying children's' weights by the weights of their parents. The method proceeds in a bottom-up fashion. One benefit of the AHP is the attention to consistency in making pairwise judgments. The AHP is tailored to the normal inconsistencies in human judgment, and provides mechanisms for assessing and adjusting consistency as necessary. More discussion of consistency issues are provided in Section 2.3.4.2.

2.2.3.3 Score Aggregation

Even after the weights and the scores have been computed, there remain variations in the techniques to combine scores for an entire collection of indicators. The general form for the aggregate score U is the sum of products, $U=\Sigma w_i x_i$, where w_i and x_i are respectively the weight and score of indicator i. Hopgood [89] proposes an alternative approach, AIM (an improved method), for score aggregation:

$$U = \Pi\{[(w(j) - w\_mid\_point) * (p(i,j) - p\_mid\_point)] + shift\_term\}.$$

U is the aggregate score, $w(j)$ is the weight of indicator j, $p(i,j)$ is the performance of indicator j relative to its parent indicator, i. The mid_point terms are the respective midpoints for the weight and performance score ranges. The shift_term is the minimum number required to ensure the combined weight and performance rating is positive. The product is taken over all j, children of parent i.

The key advantage to AIM is its greater discriminability among elements from the four combinations obtained from low/high score and low/high weight. Hopgood points out that the usual method does not discriminate well between a low scored element with a high weight and a high scored element with a low weight. With AIM, there is the power to make such distinctions more vivid. For example, regarding design evaluation, a low score on a highly weighted element should act to diminish the alternative, but a high score on a lowly weighted alternative should not diminish it. Conversely, AIM tends to over bias a high score on a lowly weighted indicator. This, Hopgood asserts, is a desirably more conservative approach than the usual method.

2.3. DESIGN EVALUATION TOOLS AND TECHNIQUES

In this section, we provide an introduction by way of overview of some of the key mathematical and logical tools which are applied in the methodology. The Analytic Hierarchy Process is employed for indicator weighting, Fuzzy Mathematics is a key method used in indicator scoring, and application of a rule-driven knowledge base is the center of a computational engine to compute and analyze indicator aggregate scores.

2.3.1 The Analytic Hierarchy Process

The Analytic Hierarchy Process (AHP) has its roots largely in management and political decision theory, where Thomas L. Saaty employed an eigenvector model to determining priorities from a matrix of relative comparison judgments [Saaty 72]. Since then hundreds of adaptations, extensions, criticisms, and applications of the AHP have been published in many languages. Application of the process to industry problems continues to proliferate as industrial systems and artifacts become more and

more complex, and decision-makers rely less on metric precision and more on human judgment in the face of critical "best-evidence-based" decisions.

Several authors who track and document development and application of the AHP have provided introductory overviews of the AHP over the years [Zahedi 86; Saaty 87; and Forman 93]. More specific, technical, or mathematically rigorous discussions appear in [Apostolou and Hassell 93; Basak 93; Blankmeyer 87; Dadkhah and Zahedi 93; DeTurck 87; Harker 87a,87b; Harker and Vargas 87; Jensen and Hicks 93; Masuda 90; Moreno-Jimenez and Vargas 93; Murphy 93; Saaty 84,86,87,93,94; Saaty and Vargas 84; Salo 93; Triantaphyllou and Mann 90; Wang and Raz 91; Wedley 93; Wedley, *et al.* 93; Zahedi 87].

Among the published applications of the AHP one can find numerous instances relevant to complex systems engineering. Table 2.1 lists of some of the most relevant topics.

2.3.1.1 Saaty's Original Method

The basic steps in applying the AHP are as follows:

1. Define the decision issue and bound its scope.
2. Structure a hierarchy of the (potentially subjective) elements contributing to a decision.
3. In a bottom-up fashion, construct a pairwise comparison matrix of the relevant

Table 2.1. Applications of AHP to Complex Systems.

Application Area	*Source*
Static Evaluation of Combat Force Potential	Lee and Ahn 91
Evaluation of Distributed Control Systems	Peterson 84
Site-Suitability Analysis	Banai-Kashani 89
Cognitive Task Allocation	Papantonopoulos and Salvendy 93
Job Labor Intensity Evaluation	Shen, et al. 90
Computer Interface Evaluation	Mitta 93
Software Evaluation	Toshtzar 88
Software Reliability	Zahedi and Ashrafi 91
Software Development Productivity Factors	Finnie, et al. 93
Software Development Effort Prediction	Lee 93
Project Risk Assessment	Mustafa and Al-Bahar 91
Project Evaluation and Selection	Khorramshahgol, et al. 88

contribution or impact of each element on its governing criterion in the next higher level. Saaty [1980] has developed a scale for use in computing relative worth of decision elements (Figure 2.5).

4. Using the eigenvector method described by Saaty [1980], compute the relative contribution of each element and test the individual matrices for consistency (reflects soundness of judgment, understanding of the interdependencies of criteria, etc.).
5. Working downward through the hierarchy, use hierarchical composition to combine the weight vectors and arrive at global and local relative contributions (priorities) of each element.
6. Perform overall consistency check on the completed hierarchy of weights. Reassign relative weights contributing to a consistency ratio out of tolerance.

2.3.1.2 Criticisms and Extensions

The AHP has not been without its detractors. Triantaphyllou and Mann [90] examine the effect of the discrete scale {1/9, 1/8, 1/7,...,1,...7,8,9} generally employed in the AHP. They demonstrate how the AHP does not derive correct membership values for elements of a fuzzy set, with known membership values. They suggest the fault lies in the AHP's requirement to restrict pairwise comparisons to the

COMPARATIVE IMPORTANCE	DEFINITION	EXPLANATION
1	Equal importance	Two elements contribute equally to the property
3	Moderate importance	Experience and judgment slightly favor one element over another
5	Essential or strong importance	Experience and judgment strongly favor one element over another
7	Very strong importance	An element is strongly favored; its dominance is demonstrated in practice
9	Extreme Importance	The evidence favoring one element over another is of the highest possible order of affirmation
2,4,6,8	Intermediate judgment values	Compromise is needed between two adjacent judgments
Reciprocals		When element i compared to j is assigned one of the above numbers, then element j compared to i is assigned its reciprocal.

Figure 2.5. Scale of Judgments Used in Pairwise Comparisons.

discrete values enumerated above. Forman [93] counters this assertion and several others. Saaty [80,90,94] and Harker and Vargas [87] posit that although any real or integer scale can be used for establishing priority, the 1-9 scale was experimentally demonstrated to be the most widely applicable and practical scale.

Another common point of contention with the AHP is the need to deal with unacceptable inconsistency in pairwise comparisons. Murphy [93], and Apostolou and Hassell [93] attack the fallibility of the consistency requirement, but DeTurck [87], Harker [87a, 87b], Blankmeyer [87], and Wedley [93], offer constructive alternatives to consistency determination and adjustment, which have reinforced the credibility and extended the applicability of Saaty's work.

The mathematical soundness of the process has been demonstrated repeatedly. Rigorous mathematical proof can be found in [Dadkhah and Zahedi 93; Harker and Vargas 87; Saaty and Vargas 84], and [Saaty 80,90,94]. Proofs of soundness cover the mathematical nature of the hierarchy itself, the ratio scale of comparisons, the eigenvector method, and consistency issues. Of notable importance is Saaty [86], wherein complete axiomatic proofs of the AHP are provided. Ultimately, for our purposes of assigning weights of criticality to system indicators, the AHP is competent to provide the needed accuracy.

2.3.2 Fuzzy Arithmetic

Much has been written on the subject of fuzzy mathematics since its introduction as an extension of set theory [Zadeh 65]. There are numerous textbooks which thoroughly examine the subject and its extensions [Kandel 86; Zimmerman 91; Klir and Yuan 95; Kaufmann and Gupta 85; Tzafestas and Veretsanopolous 94]. We cursorily discuss this time-tested topic briefly from the most general of perspectives. In this section we provide an introduction to the concepts, terminology, and operations associated with fuzzy arithmetic. This is sufficient to lay a foundation for the way in which we employ fuzzy arithmetic to design evaluation. We gratefully acknowledge

the work of Verma [Verma and Knezevik 94] from whose well-developed primer on fuzzy arithmetic we draw the greater part of the discourse in the next section[1].

The World Book Dictionary defines *fuzzy* as "blurred" or "indistinct" [World Book 87]. In the realm of fuzzy mathematics, it is the boundaries of sets or numerical intervals which are blurred or indistinct. We introduce the subject with the concept of a fuzzy set, of which a fuzzy number is a special case.

Let X be the field of reference encompassing a definite range of objects (in other words, a relevant universe). Consider the subset $\tilde{A}$ where the transition between membership and non-membership is gradual rather than abrupt. In other words this fuzzy subset of the relevant universal set obviously has no well defined boundaries. For example, assume $\tilde{A}$ to be the set of all short men in a military squadron. There are men in the organization who are definitely members of this set (very short men), and those who are definitely not (very tall men). But there are also men who are on the borderlines.

In the field of crisp sets, every element in the relevant universe is assigned a discrete membership value of 1 if it is definitely a set member. An element is assigned a membership value of 0 if it is definitely not in the set. In the realm of fuzzy sets, elements are allowed to have membership values anywhere in the continuous range of real numbers [0,1] so that the demarcation of membership is gradual rather than abrupt. It follows that the more definitely an element belongs to a set, the closer is its membership value to 1, and conversely so, relative to 0, for approaching definite non-membership.

Formally, for a given fuzzy set, $\tilde{A}$, there exists a membership function, $\mu_{\tilde{A}}(x)$, for every element, x, of the relevant universe. The membership function, $\mu_{\tilde{A}}(x)$, maps x to a value in the range [0,1], representing its degree of belonging to the set $\tilde{A}$. A

---

[1] In some cases, without further acknowledgment, exact wording has been excerpted from the original work, for which we have the permission of the author.

particular benefit of the membership function concept is the immediate ordering of elements in the set, which in many cases is of more direct worth than the actual membership value. Consider, e.g., how the relative membership values of alternative design configurations under a particular set of constraints would be of use in selecting the most suitable alternative.

Related to fuzzy sets are fuzzy *linguistic variables* [Zadeh 75]. Such a variable is a vague natural language word or phrase, e.g. "low" or "much above average." When humans deal in imprecise valuations, it is more natural to think in terms of linguistic categories than numerical ranges. A collection of these terms for a particular concept is called the *Term-set* for the concept [Verma and Knezevic 94, p 125]. A term-set need not be finite, as in the following example, for the concept *Height*. Term-set(Height) = tall + not tall + very tall + not very tall + very very tall + ... + short + not short + very short + not very short + very very short + ... Here, the "+" represents a union operator. In practical applications, finite term-sets require less discriminability on the part of the user.

In design evaluation, however, when the process requires numerical values for score computation, linguistic terms must be mapped to real numbers. Verma and Knezevic [94, p. 25] describe the concept of a *compatibility function*, which is similar to a membership function for a fuzzy set. The compatibility function provides a mapping from elements in the relevant universe to membership in a linguistic term. For example, consider the linguistic concept of weight, and the term *heavy*, which is a restriction on the total interval for *weight*. We assume *Weight* has a bounded interval from 5 to 1000 kilograms. The compatibility of the weight values 50, 200, and 999 with the fuzzy restriction labeled *heavy* might be 0.01, .12, and 1.0, respectively.

Notice that the compatibility function need not be linear. Additionally, there may be non-linear functions which map between linguistic variables. For example, we may choose to represent the compatibility function, CF of *very heavy* as CF(*very heavy*) = CF(*heavy*)^2. The CF is tweaked to represent the degree of acceptable values

applicable to a particular variable, for a particular situation. Herein is the essence of the application of fuzzy values to real systems assessment.

We consider now the concept of a qualifier, α, for membership in a set. The α qualifier has the effect of narrowing the membership of elements in a set to only those whose membership is $\mu_{\tilde{A}}(x) \geq \alpha$. Thus $\tilde{A}$ under α becomes $\tilde{A}_\alpha = \{x \mid \mu_{\tilde{A}}(x) \geq \alpha\}$, $\alpha \in [0,1]$.

There are two additional restrictions which may be applied to membership functions. *Convexity* recursively requires that every ordinary subset of $\tilde{A}$ is also convex, i.e. that the membership function is continuous and has no dips or local minima. The requirement of *normality* guarantees that for at least one element x in the relevant universe, $\mu_{\tilde{A}}(x)=1$, i.e. that at least one element definitely belongs to the set [Gupta and Kaufmann 91; Klir and Yuan 95]. The figures below demonstrate these concepts. Figure 2.6 is both convex and normal. Figure 2.7 is neither convex nor normal.

A fuzzy number is a special case of a fuzzy set which is both normal and convex, as in the figure on the right. The fuzzy number's value is a placeholder for a single variable, but whose value is represented by an interval. This interval is distinct from a confidence, as used in probability, for which the interval represents probability of a value taken on by a random variable. According to Kaufmann and Gupta [85, p. 14], the random variable is an objective datum, a *measure*; however, a fuzzy variable is subjective, a *valuation*. This aspect of the fuzzy number is particularly valuable in the application of subjective measures, as would occur at some high-level stages in design evaluation.

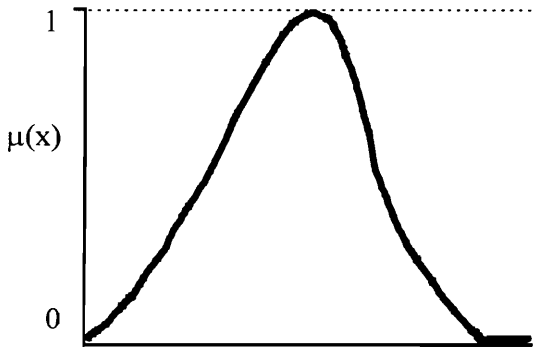


Figure 2.6. Convex, Normal Membership Function.

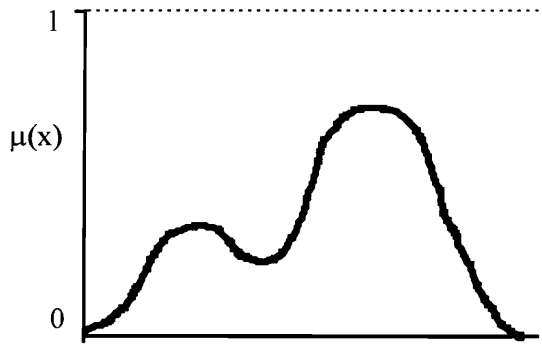


Figure 2.7. Non-convex, Non-normal Membership Function.

Combinations of fuzzy numbers follows the rules below [Kaufmann and Gupta 85]. In these examples, A_α and B_α are fuzzy intervals $[a_1^{(\alpha)}, a_2^{(\alpha)}]$ and $[b_1^{(\alpha)}, b_2^{(\alpha)}]$, which bound the fuzzy variables **A** and **B**, under a restriction, $\alpha \in [0,1]$.

Addition: $A_\alpha+B_\alpha= [a_1^{(\alpha)}, a_2^{(\alpha)}] + [b_1^{(\alpha)}, b_2^{(\alpha)}] = [a_1^{(\alpha)} + b_1^{(\alpha)}, a_2^{(\alpha)} + b_2^{(\alpha)}]$.

Subtraction: $A_\alpha-B_\alpha= [a_1^{(\alpha)}, a_2^{(\alpha)}] - [b_1^{(\alpha)}, b_2^{(\alpha)}] = [a_1^{(\alpha)} - b_2^{(\alpha)}, a_2^{(\alpha)} - b_1^{(\alpha)}]$.

Multiplication: $A_\alpha \bullet B_\alpha= [a_1^{(\alpha)}, a_2^{(\alpha)}] \bullet [b_1^{(\alpha)}, b_2^{(\alpha)}] = [a_1^{(\alpha)} \bullet b_1^{(\alpha)}, a_2^{(\alpha)} \bullet b_2^{(\alpha)}]$.

Division: $A_\alpha/B_\alpha= [a_1^{(\alpha)}, a_2^{(\alpha)}]/[b_1^{(\alpha)}, b_2^{(\alpha)}] = [a_1^{(\alpha)}/b_2^{(\alpha)}, a_2^{(\alpha)}/b_1^{(\alpha)}]$, $b_2^{(\alpha)} > 0$.

As examples, consider the intervals A_α and B_α as [-1,6] and [2,8], $\alpha \in [0,1]$.

$A_\alpha+B_\alpha= [1,14]$,

$A_\alpha-B_\alpha= [-9,4]$,

$A_\alpha \bullet B_\alpha= [-2,48]$, and

$A_\alpha/B_\alpha= [-1/8, 3]$.

This concludes our summary introduction to fuzzy arithmetic. We have laid the groundwork for the operations and concepts which are employed in our proposed methodology.

One example from the literature is particularly appropriate to the topic of design evaluation. Paek, *et al.* [92] apply a weighted fuzzy score scheme to the evaluation of designs for housing complexes. Of particular interest is the scheme for aggregating

scores in the indicator taxonomy they employ. We give an overview of the technique here.

Scores for indicator values may take the form of a fuzzy interval $Z_{i,h}(x)$, for the i^{th} criterion of design alternative x, at level h in the taxonomy. Using expert opinion regarding the best ($BESZ_i$) and worst ($WORZ_i$) values the indicator is expected to take, the score is converted to an index interval $S_{i,h}(x)$,:

$S_{i,h}(x) = [(Z_{i,h}(x) - WORZ_i)/(BESZ_i - WORZ_i)$, for $WORZ_i \leq Z_{i,h}(x) \leq BESZ_i]$

$S_{i,h}(x) = [(Z_{i,h}(x) - WORZ_i)/(BESZ_i - WORZ_i)$, for $BESZ_i \leq Z_{i,h}(x) \leq WORZ_i]$

and for which in the first case, $S_{i,h}(x) = 1$, for $Z_{i,h}(x) \geq BESZ_i$, 0, for $Z_{i,h}(x) \leq WORZ_i$, and for the second case, $S_{i,h}(x) = 1$, for $Z_{i,h}(x) \leq BESZ_i$, 0, for $Z_{i,h}(x) \geq WORZ_i$.

In the two cases listed, the first applies when a value for the best score is expected to exceed that of the worst score (as in crew living space). Conversely, the second applies to the case where the best score is expected to be lower than the worst (as in response time). Level by level the scores are aggregated in the following manner:

$$L_{j,h}(x) = \left\{ \sum_{i=1}^{n_j} w_{i,j} \left[S_{i,h,j}(x) \right] p_j \right\}^{1/p_j}, \text{ where:}$$

$L_{j,h}(x)$ = the aggregate score for all indicators in group j at level h for alternative x

n_j = number of elements in the sibling indicator group of parent indicator, j

$S_{i,h,j}$ = index value for the i^{th} indicator in group j at level h

$w_{i,j}$ = the weight reflecting the importance of indicator i within group j

p_j = the balancing factor for group j, inherited from the parent indicator, j.

As described in later sections, a weighted-score aggregation scheme such as this, with the additional balancing factor, offers the flexibility in scoring and the tools for strategic balancing which can be extended for application to the evaluation of complex system designs.

2.3.3 Knowledge-Based Systems

In this section, as in the previous, we provide only an overview of the subject matter, since the general concepts are widely known. We focus on the salient features common to most knowledge-based systems. This discussion lays a sufficient foundation for the reader to appreciate how we propose to employ this technology into our design evaluation methodology. We are particularly interested in the components of the systems which enable experts from an application or operational domain to input knowledge and rules of inference. These facts and rules can then be accessed in the process of the evaluation of complex system designs.

Hopgood's [93] text provides a broad-based overview of the subject of knowledge-based systems. In particular, he analyses applications to design assessment and decision-making. His knowledge-based system model is shown in Figure 2.8.

The knowledge acquisition module may include a computer-based series of questions presented to an expert or other source of knowledge. The expert responses would be formatted into the syntax of the knowledge base, indexed, and stored. This function could be quite complex and involve natural language parsers, specialty dictionaries and thesauri, etc. A greater discussion of this function is found in [Frakes and Baeza-Yates 92].

The explanation module offers the capability to re-trace the reasoning followed in arriving at a conclusion. This could be as simple as a sequential recounting of the

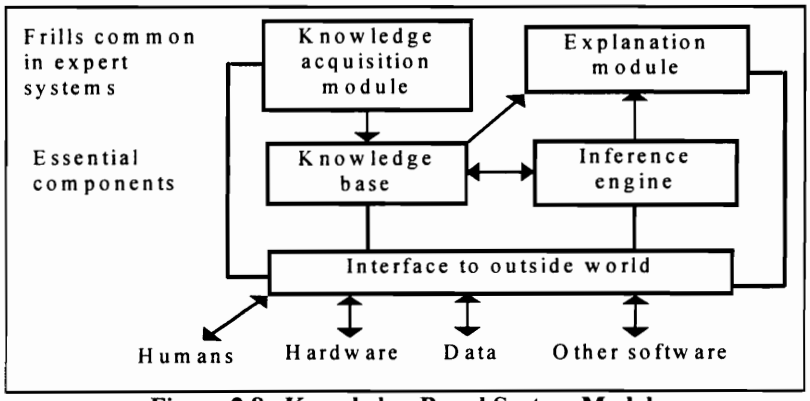


Figure 2.8. Knowledge-Based System Model.

steps, or an elaborately presented, natural-language description. Even in the brief explanations we have presented of these functions, it is understandable how they are considered frills, and are more likely to be found in very large knowledge-based systems, such as for rapid medical diagnosis.

The more basic components of any knowledge-based system are the knowledge base itself, and an inference engine for deriving other facts and arriving at conclusions. The knowledge base ultimately contains a structured set of facts and rules. *Facts* are presented as statements, which represent relationships between elements in the knowledge system. An example statement could be "**MiG-29** is WEAPON_TYPE *fighter*" or "**Battleship** has HULL_COLOR *gray*." For illustrative purposes, different font types and cases are used to represent different components of a fact. In this example, the words in bold represent **instances** of a knowledge element — nouns. Similarly, upper case terms indicate ATTRIBUTE TYPES for an element. Finally, in this simple example, words in italics represent *attribute values* — adjectives. These parts of speech are parsed by the inference engine to match facts against other facts and rules in the process of reaching conclusions.

We do not wish to suggest that facts in a knowledge base are so simple. There are likely to be thousands of both simple and compound facts in the knowledge base of a large-scale practical system. Additionally, the attribute values may likely be vague linguistic terms as discussed in Section 2.3.5. An example of a compound fact with a fuzzy variable might be "**Exocet** is WEAPON_TYPE *anti-ship* SOURCE *French* DELIVERY_METHOD *fighter-launched* SPEED_CATEGORY *high-speed*," where the vague term *high-speed* could demonstrate the fuzzy characteristics described earlier. As a result, inferences on fuzzy facts require fuzzy mathematical techniques. Numerous textbooks and references in the literature describe fuzzy expert systems, see e.g. [Gupta 85; Negoita 85; Sanchez and Zadeh 87; Kruse 91; Kandel 92].

Rules are the other type of statement in a knowledge base. They are the key elements used in the inference engine. The inference engine uses facts from the

knowledge base to match variables in the rule base to derive solutions to queries. Rules relate two or more facts with an action. A rule has a condition part and an action part, referred to respectively as the *antecedent* and the *consequent* in expert system parlance. A simple rule for a knowledge-based advisor for a weapons control system might be "if TIME_TO_IMPACT < *target_acquisition_time* then SIGNAL_IMPACT_IMMINENT and SOUND_GENERAL_ALARM." In this example, the phrase in small capital letters represents the consequent, or the action to be performed if the antecedent is satisfied. We again point out the near trivial nature of this rule and stress that in large-scale systems, the rules would likely be compound and contain fuzzy variables requiring, as in this case, extremely fast resolution.

We now make note of a particularly nagging concern with knowledge-bases, which has direct bearing on the responses the system can generate. We are referring to the well-documented *closed-world problem* [Hopgood 93]. This problem describes the dilemma which arises due to the finite amount of facts in the knowledge base. An immediate consequence is that queries to the system may address topics for which no facts have been entered, nor can be derived by either induction, deduction, or abduction from the extant facts.

Consequences for an ill-designed knowledge-based system could be disastrous, including system failure and ultimately the loss of human life. Still, knowledge-based systems are widely employed and increasingly so, and continue to improve in reliability and reality of advice. The closed world problem can be alleviated somewhat with larger rule and fact bases and knowledge-generating engines, or machine-learning, which have yet to achieve operational maturity.

In summary, the basic elements of knowledge-based systems are given as follows:

(1) Human expert knowledge is converted to the facts of the knowledge base.

(2) Expert operational heuristics become condition-action rules in the inference engine.

(3) Facts and rules can be fuzzy, since they are rooted in imprecise human knowledge.

(4) Computational resources must handle fuzzy, imprecise information with limited information.

(5) Large-scale systems are increasingly reliant on knowledge-based controllers and advisors.

2.4. SUMMARY

This chapter provides a broad based overview of the literature relevant to complex systems, and complex system design evaluation. Specifically, the chapter addresses the systems engineering process as a the circumscribing framework for all design evaluation. Secondly, it provides a large list of references to the evaluation of specific complex system components, including hardware, software, hardware, and the multi-way interfaces among these elements. Finally, it addresses particular tools and techniques which have been applied to design evaluation work, including evaluation by indicators, the analytic hierarchy process, fuzzy mathematics, and knowledge-based systems.

3. METHODOLOGY OVERVIEW

3.1. INTRODUCTION

The problem description in Chapter 1 and the survey of relevant literature in Chapter 2 punctuate the complex, judgment-based nature of complex systems design evaluation. It is by no means simplified, standardized, serialized, or solved by extant techniques. In this chapter, we introduce the major components of a methodology for complex system design measurement and evaluation. Greater detail, variations, and explanations are provided in subsequent chapters. This overview chapter is largely pictorial, laid out in flow-based and schematic figures with detailed callout text. In the sections to follow, we weave the pictures into a coherent whole. Section 3.2 describes how the evaluation methodology fits in the system life cycle. Section 3.3 outlines the steps in the individual phases of the methodology. In Section 3.4, we summarize and close the chapter.

3.2. DESIGN EVALUATION SCENARIO

Our design evaluation methodology enters the system life cycle after the conceptual design has been evaluated, and a solution system to meet operational mission requirements has been selected. Figure 3.1 depicts the scenario in overview form. It shows the organizations involved in the process and information flow between them; dashed lines represent feedback. In the next subsections we identify and briefly describe the role of each organization.

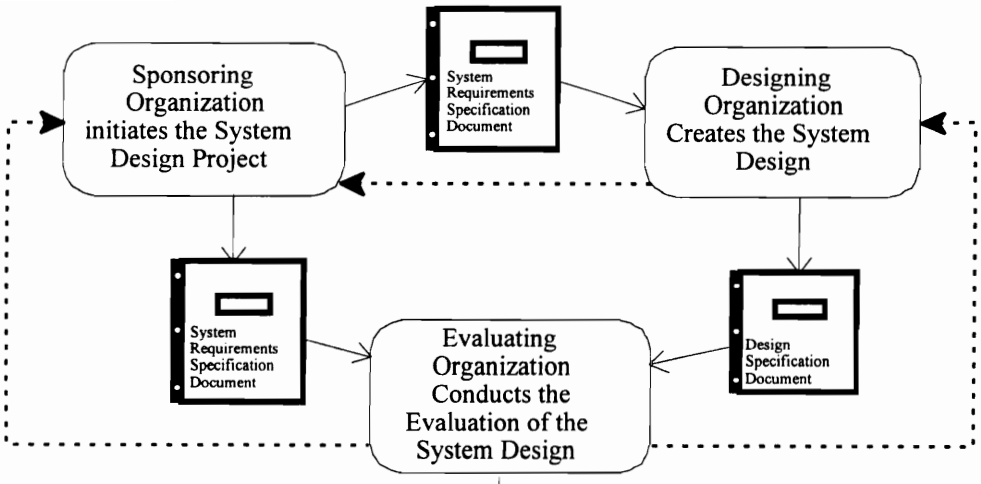
3.2.1 Organizations

There are three organizations involved in the requirements engineering, design, and design evaluation for a complex system. They are respectively the sponsoring organization, the designing organization, and the evaluating organization. In the next three sections, we briefly describe the roles of the three organizations involved in the system design evaluation scenario.

3.2.1.1 Sponsoring Organization

The sponsoring organization is the one with the need for a new or upgraded complex system. Typically this is a government agency, the military, an electric power conglomerate, or an advanced technology-based industry. They have an operational need for which a complex system is a proposed solution. This organization will operationally employ the system in the context of a larger mission, e.g., national defense, space exploration, public service, or capital profit. They are the source for operational mission scenarios, budget constraints, and operating resources, including personnel. Typically they do not possess the technical expertise or resources to design and develop the system, so they contract with a development organization. Additionally, as is the case with the Federal Government, the sponsoring organization may be mandated by law or other public statute to contract out large-scale development projects. Finally, the sponsoring organization may or may not possess project management personnel to provide sufficient full-scale project oversight.

The sponsoring organization typically works with a systems engineering organization to conduct requirements engineering. Efforts at this stage concentrate on defining the operational need, performing feasibility studies, establishing system operational requirements, and defining the system maintenance concept. The preliminary systems analysis follows, and the results are documented in the system specification, or system requirements specification. This document serves as the driver for the system concept definition and conceptual design. After conceptual design evaluation, the system requirements are further refined and a designing organization is contracted to create a (or one of several competing) preliminary or detailed design(s). The system requirements specification is the key input to the designing organization.



The **Sponsoring Organization** defines the operational need. With Systems Engineers, they conduct Requirements Engineering functions, feasibility analyses, and generate a **system concept** for satisfying the operational need.

Alternative systems are evaluated against the requirements [Verma 94] and the system requirements specification is prepared.

The **system requirements specification** includes the operational requirements, maintenance concept, system functional diagram and interfaces, physical and performance effectiveness measures of merit, logistical support plan, conceptual design documentation, and quality assurance provisions [Blanchard and Fabrycky 90].

The **Designing Organization** develops the preliminary and/or detailed design for the system, based on the system requirements specification. System and subsystem components are defined or selected.

The design specification as well as the system requirements specification are to be evaluated *independently* by a third-party **evaluating organization** . This organization is contracted by the sponsoring organization to ensure the design both addresses the correct requirements, and correctly satisfies the requirements.

The evaluating organization uses the methodology described herein to assess the quality and requirements satisfiability of the design, providing a basis for comparing the relative quality of competing designs.

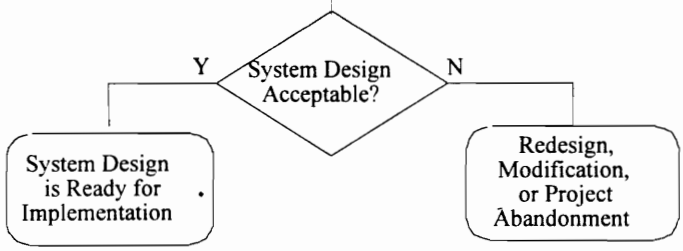


Figure 3.1. Complex System Design Evaluation Scenario.

3.2.1.2 Designing Organization

This organization is typically a network of subcontracted design organizations. In the aggregate, the designing organization possesses the technical expertise, design development resources and facilities, project management, and oversight personnel required to produce the complex system design.

Based on the system requirements specification, the designing organization prepares a preliminary and/or detailed design of the system which fulfills the sponsoring organization's requirements. The process involves allocating functional and non-functional system requirements to system components from the software, hardware, and humanware domains [Webster 87, Tarnoff 91]. The design and documentation of the design process itself are provided to the evaluating organization for their expert assessment.

3.2.1.3 Independent Evaluating Organization

Independence in evaluation is commonly applied in an attempt to mitigate effects of politics, pressure, or bias. Similarly, this methodology advocates an independent assessment of complex system designs. Beginning with the preliminary design, a third-party organization, contracted by the sponsor, should evaluate both the design and the processes which created the design. This evaluation is intended not to obviate but to supplement the in-house design review process, Their findings are of benefit to both the sponsoring and the designing organizations, and multiple feedback loops are present between the three organizations. Subsequent to evaluation, the design is deemed either acceptable for future consideration, or deficient in one or more aspects of the governing requirements.

The evaluating organization possesses the expertise and resources to evaluate the integrated aspects of the design. Expertise includes expert knowledge from technical disciplines which are intersected by the functional and nonfunctional requirements of the proposed system (e.g., software engineering, human factors, etc.). Evaluation resources include computer automation, knowledge-based environment, and

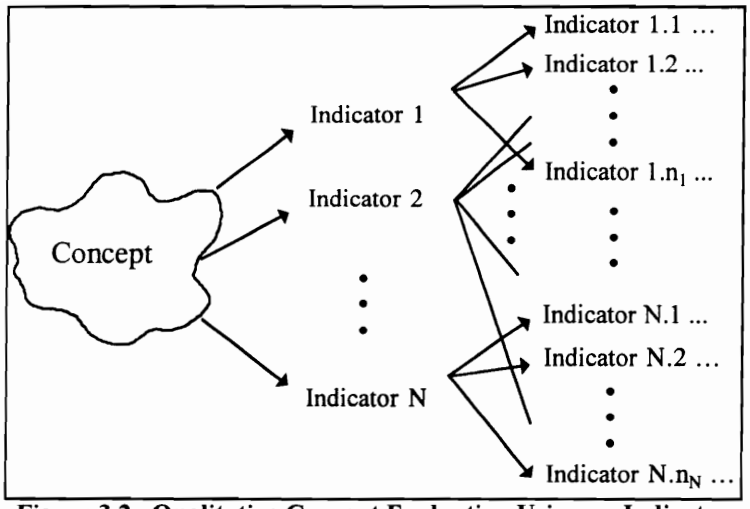


Figure 3.2. Qualitative Concept Evaluation Using an Indicator Hierarchy.

visual simulation environment. These resources aid in collecting, structuring, and processing data to assess the quality of the design and the process which generated the design.

3.2.2 Measurement Framework

The notion of "design quality" is a soft concept; it is qualitative, not quantitative. A qualitative concept is not directly measurable, but is assessed by a collection of *indicators*. We use this term in the same sense as other disciplines use various related terms, e.g., *metric*, *figure of merit*, *index*, *scale*, *factor*, *measure*, etc. Indicators themselves may be qualitative, requiring subindicators for their assessment.

Recursive application of indicator refinement leads naturally to a hierarchical structure, as depicted in Figure 3.2. In an indicator hierarchy, the recursion terminates at "leaf" nodes. These base-level nodes must be evaluated and their scores combined with those of their siblings to yield an aggregate score for their parent. Continuing in this manner up the hierarchy, a value is obtained for the root concept.

Assessing the value of an indicator is itself not necessarily a precise procedure. Especially where human judgment weighs in the determination of indicator's value, some error is naturally involved. We assert that the more expert the evaluator of an

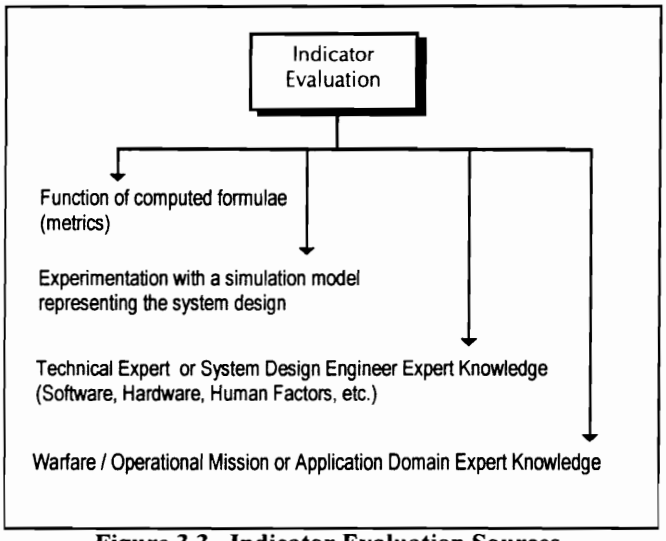


Figure 3.3. Indicator Evaluation Sources.

indicator, the more negligible the error. Thus, we advocate *expert evaluators* for scoring indicators.

There are four sources for determining the value of an indicator (Figure 3.3). Metric-based indicator scores use a function of directly computable values. Simulation-based indicator scores rely on results of experimentation with a simulation model representing the system design. Simulation is required to assess indicators related to the behavior of the system being designed. Indicator scores based on technical expert knowledge include software engineering, human factors engineering, and systems integration engineering expertise, among others. Operational domain knowledge-based scores rely on the judgment and expertise of veterans from the operational environments in which the proposed system is to be employed.

3.2.3 Software Support for Design Evaluation

Two software products, developed at Virginia Tech, provide the required computer-aided assistance in the application of our design evaluation methodology: VSE (Visual Simulation Environment) and KBESD (Knowledge-Based Evaluation of System Designs).

The VSE is a new technology in visual simulation created as a result of experimental research for over a decade under funding from the U.S. Navy [Balci et al.

1995]. It can also be characterized as a computer-aided simulation model development environment. The VSE facilitates the creation of visual simulation models of complex system designs. Experimentation with the simulation model representing a system design enables us to estimate the dynamic characteristics of the designed system. Such estimation is extremely important for evaluating the designs of dynamic systems which are performance-, mission-, and time-critical. The second category of indicators in Figure 3.3 is assessed by way of using simulation.

The VSE currently runs under NEXTSTEP Unix Operating System on the following hardware platforms: Motorola 68040, Intel 80486 and Pentium, Sun SPARC, and HP PA-RISC. It will be ported to OpenStep, Windows 95, Windows NT, Solaris, and other operating systems in 1996. The major features of the VSE are given below [Balci, et al., 95].

- Intended for discrete-event type of visual simulation modeling for problem solving.
- General purpose and domain independent.
- Can be used to visually simulate any complex system at any level of detail desired.
- Provides a picture-based approach to visual simulation modeling. A model component pictorially represents a system component by using a scanned photograph or any drawing of that component.
- Employs a multifaceted conceptual framework for guiding the modeler in model construction.
- Enables the creation of truly object-oriented models with encapsulation, inheritance, and message passing.
- Facilitates the development of a model in a graphical and top-down hierarchical manner.
- Enables top-down hierarchical decomposition of dynamic objects into component objects.
- Enables the creation of "intelligent" objects by specifying logic for dynamic objects and their components.
- Enables the creation of models using the machine-oriented view, material-oriented view or a combination of the two views.
- Enables the modeler to follow the paradigm "What You See Is What You Represent."
- Provides a state-of-the-art human-computer interface.

- Provides an English-like object-oriented (with encapsulation, inheritance, and message passing) scripting language for model logic specification.
- Reduces the amount of model logic specification and localizes it to a model segment.
- Supports all phases of visual simulation model development and experimentation.
- Facilitates the validation, verification, and testing of simulation models.
- Facilitates the credibility assessment of simulation study results.
- Provides a multimedia learning support system and on-line assistance.
- Provides the automation-based software paradigm where the central focus is on creating and maintaining the model specification and automatically generating the executable code.
- Traces execution errors all the way back to the model specification.

The KBESD prototype software tool (Figure 3.4) provides computer-aided assistance for most of the phases of our methodology. Upon the development of a production version, the KBESD is expected to provide a wide range of automated support including:

- a **graphical user interface** to the indicator database,
- an **indicator hierarchy browser** to provide a view of several levels of the hierarchy,
- an **indicator inspector** which allows viewing and editing of all information (attributes) specific to an individual indicator,
- a **computational engine** for indicator score aggregation,
- an **expert system shell** for user input of facts and rules relative to the design evaluation,
- an **expert system inference engine** for applying rule-based knowledge to the evaluation, and
- a **pop-up notes panel** for indicator-specific remarks by individual evaluators.

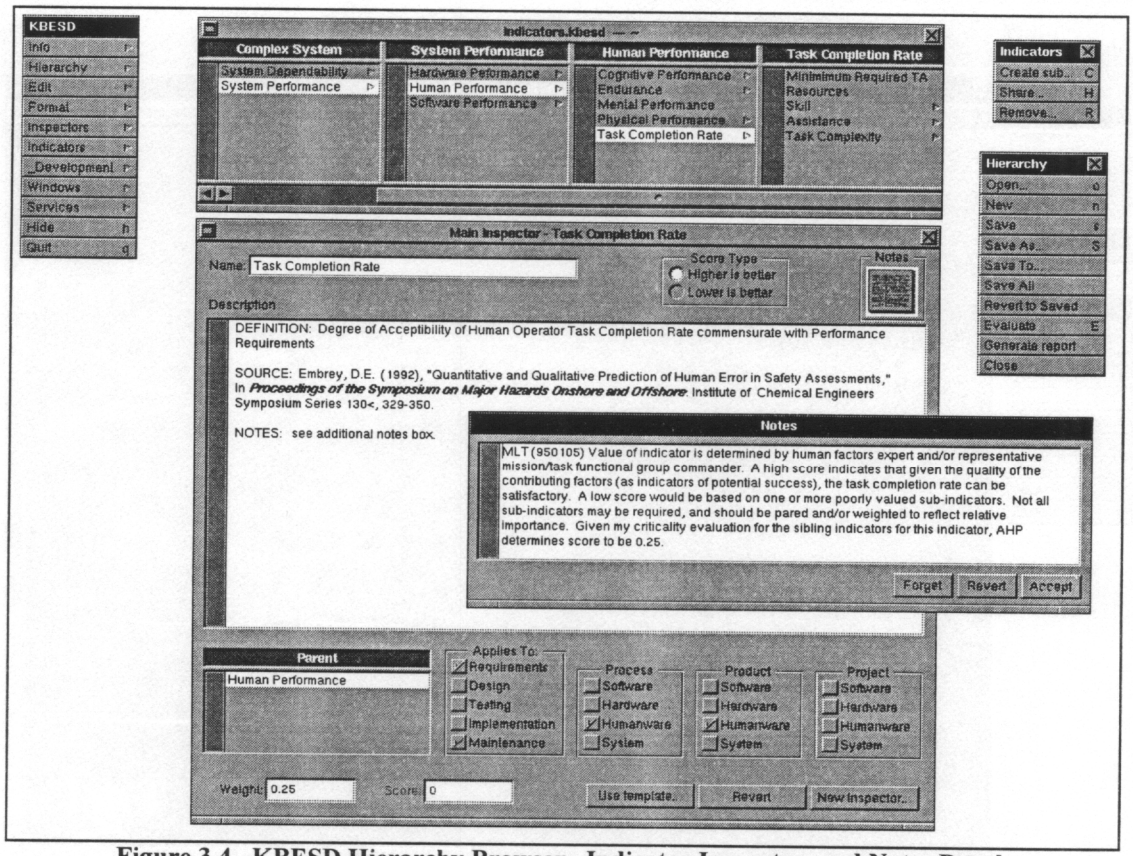


Figure 3.4. KBESD Hierarchy Browser, Indicator Inspector, and Notes Panel.

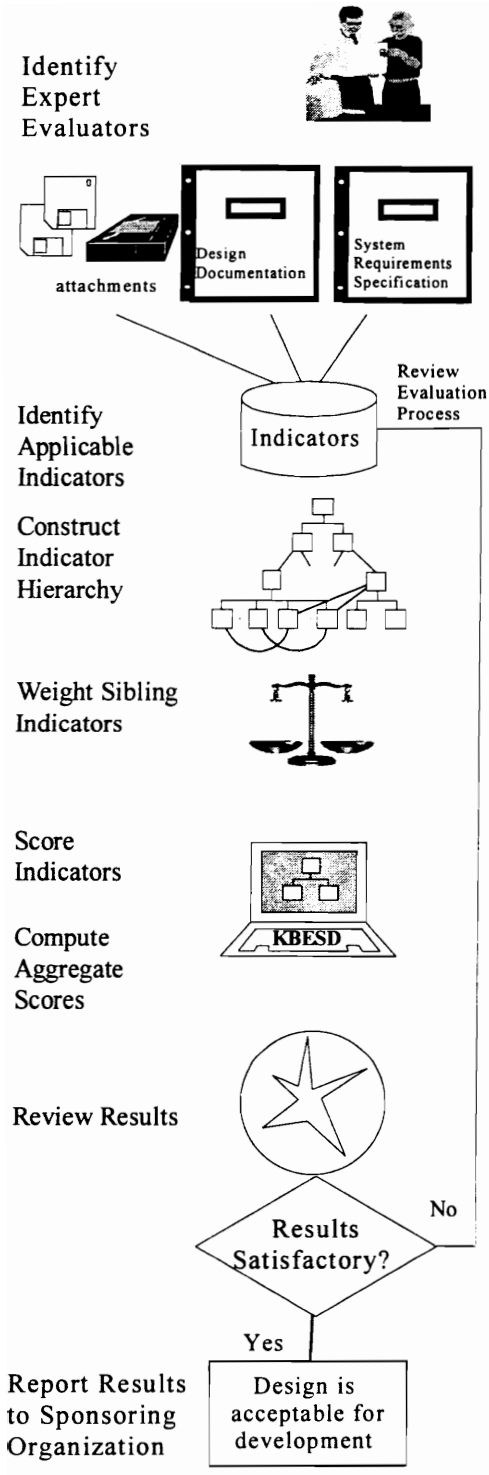
3.3. COMPONENTS OF THE METHODOLOGY

The methodology presented herein is based on evaluation of a system design via a structured collection of design quality indicators. Other facets of the methodology relate to strategies for identifying, weighting, scoring, and combining indicators, with the use of proven mathematical techniques, and application of a knowledge-based system. The general approach is depicted in Figure 3.5. In the subsections to follow, we overview the specific components of the design evaluation methodology. For each component, we provide a graphical depiction of the major steps, with explanatory text.

3.3.1 Identification of Expert Evaluators

A complex system is typically a system of systems from the major domains of *people*, *hardware* (including "*chipware*") and *software*, employed in some operational environment. A complex system requires the synergistic operation of perhaps hundreds of humans, tons of structural hardware exposed to the (possibly hostile) elements, complex integrated circuitry, and in the aggregate, millions of lines of software. But elements from these domains interact at pairwise interfaces, e.g., human-hardware (man-machine), human-software (human-computer), and hardware-software (firmware). Multi-way interfaces are experienced in the operation of a complex system.

Now consider evaluating a design for such a system. A preliminary design addresses decomposition and allocation of functional and nonfunctional requirements to elements of the domains (individually and pairwise), e.g., software or firmware, etc. Evaluating the feasibility and utility of the design with respect to user requirements requires personnel with expertise spanning all the functional and nonfunctional areas addressed by the design. Additionally, experts are needed from the operational environment in which the system will be employed. In Figure 3.6 we depict these four general categories of expert evaluators, drawn along lines of interaction between elements of the system, and between the system and its operational environment. We briefly address these four expert classes below.



The evaluating organization is contracted by the sponsoring organization to perform the design evaluation. Initially, the evaluating organization identifies the **technical and operational expertise** requirements and hires expert evaluators to conduct the evaluation.

The evaluators study the portions of the documentation related to their area of expertise and **identify indicators** which are used to evaluate the design

Through **recursive refinement**, experts in each evaluation domain **define lower level indicators**, establishing and depicting causal and dependent **relationships** between them. Indicators are entered into the KBESD software tool, with other indicator-specific information. We currently have compiled a hierarchy of over 400 indicators.

Evaluators **determine weights** for each indicator in a sibling group. Where probabilistic or regression coefficients are not appropriate, we recommend the **Analytic Hierarchy Process**, which is widely applied to establish relative criticalities among items.

Evaluators compute or assign scores to base-level indicators. Indicator scores come from generally four sources: By application of **metrics**, **simulation** of system behavior, technical/systems **engineering knowledge**, and **operational expert knowledge**.

Weights and scores are entered into the KBESD. The KBESD employs **rules and facts** related to indicator interrelationships, acceptable score ranges, and evaluator expertise. These rules are created by the evaluators and evaluation organization personnel. With the use of **fuzzy and crisp** mathematics, a knowledge base is employed to aggregate scores and assess the credibility of the evaluation.

Kiviat Graphs are a convenient and effective device for displaying relative performances of a collection of indicators. A kiviat diagram can be displayed for each indicator sibling group, or any desired collection of indicators.

An evaluation which meets or **surpasses established thresholds** can be considered acceptable. Individual indicator groups which do not meet thresholds can be revisited at an appropriate point in the process. Design weaknesses are reported to the designing and sponsoring organizations.

Figure 3.5. Major Phases of the Design Evaluation Methodology.

Metricians and single-domain specialists include software engineers, hardware engineers (both structural and silicon), and perhaps behavioral psychologists, sociologists, etc. These technical domain engineers would provide evaluative inputs regarding requirements satisfiability of individual domain components, as well as inputs to evaluators of inter-domain indicators.

Inter-domain evaluators may be experts in human factors, programmable integrated circuitry, communications switches, software and hardware maintenance, industrial psychology, etc. They prepare valuations of the utility of the design to the extent that it addresses pairwise interactions among elements of the individual domains. These evaluators may also provide inputs which are useful to the *systems integration engineers*, who consider synergistic interactions across three or more domains.

Operational domain experts are typically seasoned veteran operators of the type of system under design. These evaluators are armed with heuristics, gut feelings, and insights which are ingrained by experience in various operational domains. They provide judgments on system-behavioral indicators with respect to various operational scenarios.

Thus far, we have described a need for a comprehensive spectrum of evaluator expertise and suggested a practical suite of evaluator types. In practice, an evaluating organization would ensure an evaluator staff with sufficient expertise appropriate to the system design being evaluated. A guiding influence on evaluator expertise composition within the evaluating organization is the relative prominence of the major domains which the system design intersects. More on this subject is discussed in Chapter 4.

3.3.2 Identification of Design Feasibility/Utility Indicators

Figure 3.7 provides a depiction of the indicator identification process. In this section we discuss the basic steps of the process.

The design documentation and the system requirements specification are the launching points for the indicator identification process. While Figure 3.7 depicts these inputs as single entities, they may be voluminous, multi-part, and even multi-

media items, taking the form of figures, charts, software, spreadsheets, even photographs and motion video.

Indicator identification proceeds in a top-down fashion, yielding several upper-level main categories of indicators and subsequently, finer resolutions of subcategories. Systems engineers and operational domain experts determine the highest level indicators, which in turn are refined by interdomain specialists, and so on until lower- and base-level indicators are identified by single domain specialists. At the time of indicator identification, the scoring source should be determined as one (or a combination) of the four types described in Section 3.2.2. Either after indicator identification or concurrently with it, evaluators can begin to construct hierarchies from the indicators. We describe this process in the next section.

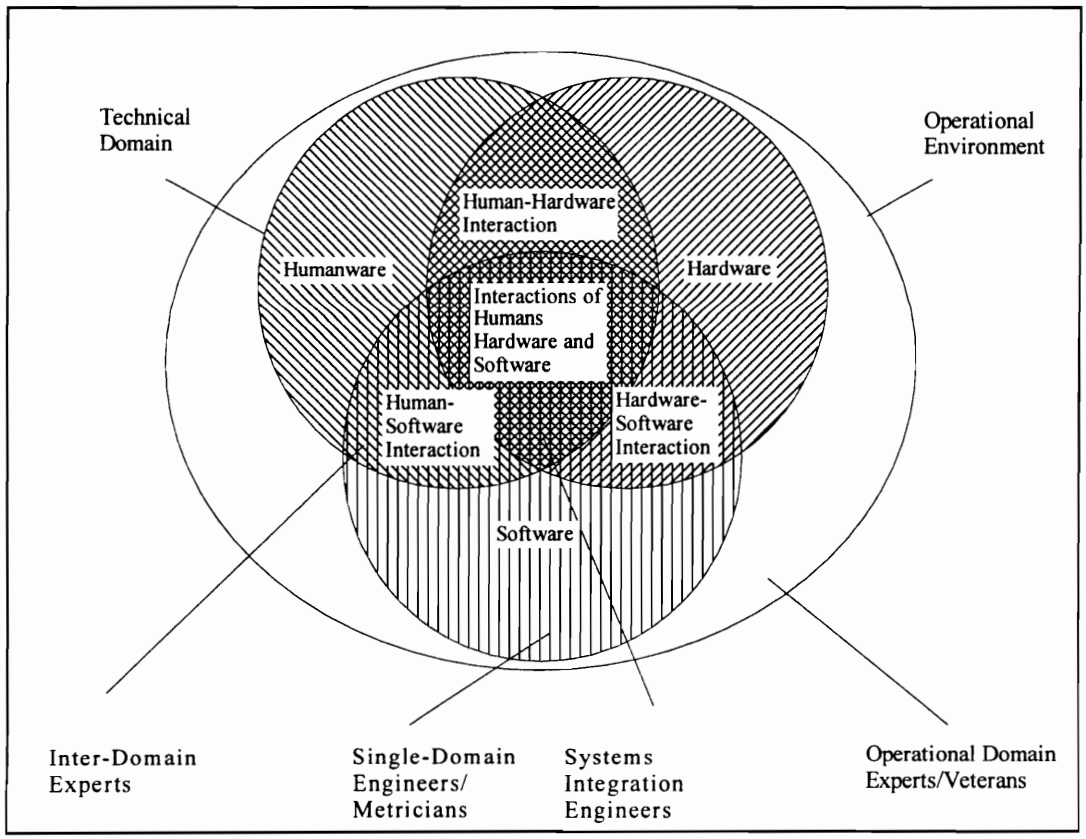
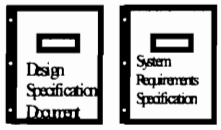


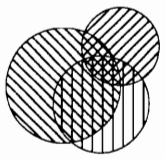
Figure 3.6. Evaluator Expertise Sources.

Expert Evaluators Review Requirements and Design Specification Documents



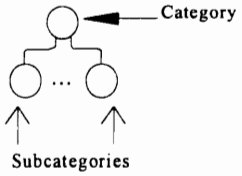
The driving influence for selection of indicators is the **requirements specifications document** which contains a statement of the system-level measures of merit desired by the sponsoring organization. The requirements documentation may be a paper document, software, graphics, spreadsheets, etc. The design is to be evaluated based on the requirements, augmented by the insight and experience of the technical specialists, metricians, and operational domain experts.

Experts Discern the Domain Dominance Schema, Inherent in the System



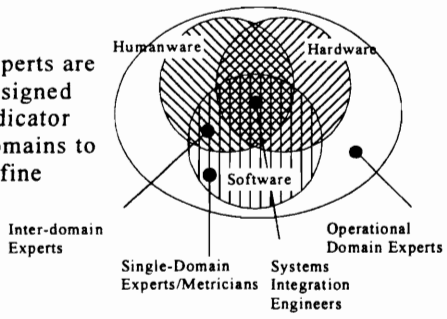
Based on the nature of the system and the application/mission for which it is intended, **a domain dominance schema** will be evident. From the schema, experts should identify major indicator domains which are refined into subcategories as needed. A starting point is the system-level measures of merit included in the requirements. These may be, e.g., *Performance*, *Dependability*, *Producibility*, etc.

Evaluators Create Indicator Categories and Subcategories



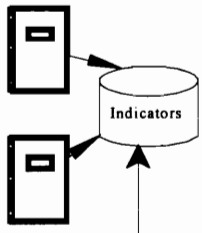
System-level categories are likely to involve design elements which include Human, Hardware, and Software domains. Indicator selection at this level should be evaluated by **systems integration engineers**. Pairwise inter-domain elements could require experienced personnel from each separate domain as well as experts in the **interdisciplinary** arena. Design elements which encompass only a single domain may not require interdisciplinary experts, but, e.g., software engineers, hardware engineers, structural engineers, or behavioral psychologists. Defining indicators related to operational scenarios are the task of **operational experts**.

Experts are Assigned Indicator Domains to Refine



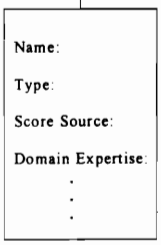
Indicator subcategories are then **assigned to specific expert evaluators**. Several expert evaluators from the same indicator domain may be desired.

Evaluators Identify and Define Indicators from their Domain of Expertise



Evaluators review the applicable portions of the requirements and the design to **identify and define indicators** related to their assigned subcategory. Upper-level indicators are refined into lower-level indicators, based on the experience and expert knowledge of the evaluators.

Evaluators Determine and Specify Attributes for the Indicators



For each indicator, there is a suite of attributes whose values may be unique to the indicator. Evaluators determine and specify values for these attributes which become part of the indicator entity. These attributes are employed in the knowledge-based phase of the design evaluation.

Experts Identify Indicators for Determining the Quality of the Design Process

Evaluators also identify **indicators** for assessing the quality of the **requirements engineering and the processes** which created the design. This assessment will provide the sponsoring organization with a complete evaluation coverage, increasing the credibility and value of the evaluation.

Figure 3.7. Indicator Identification Process.

3.3.3 Construction of Indicator Hierarchies

The process of indicator hierarchy construction is pictorially represented in Figure 3.8. Beginning with the indicators identified in previous phase, or concurrently with it, evaluators begin to organize indicators into a directed acyclic graph or hierarchy based on indicator influence relationships. Relationships may be many-to-one, yielding a more network-like structure than simply a tree. However, we recommend observation of the rule-of-thumb of seven plus or minus two (7±2) as an appropriate number of children for a parent node. Larger numbers tend to lead to additional work in future phases of the methodology, particularly in scoring and weighting. We discuss specific scenarios under which the sibling group size is a factor in Chapters 5 and 7.

Relationships between indicators include both direct *structural* and indirect *non-structural* relationships. Non-structural dependencies reflect proportional relationships between indicators and are represented in the fact and rule base of the KBESD. They are used as a check-up during the scoring and aggregation processes. Structural relationships are directly represented as links within the KBESD hierarchy browser tool. They reflect the parent-child relationships between children indicators and a composite parent indicator.

At the time of entering indicators into the hierarchy browser, evaluators may make liberal use of the KBESD *inspector* and its *note panel* for providing details for each indicator and its role in the evaluation of its parent indicator. A template for relevant information in the inspector has been suggested by Nguyen and Howell [94]. Additional indicator information indicates whether a high value or a low value of an indicator is desirable, which of the score sources is applied to assess the indicator's score, and to which phase of the system life cycle the indicator most applies.

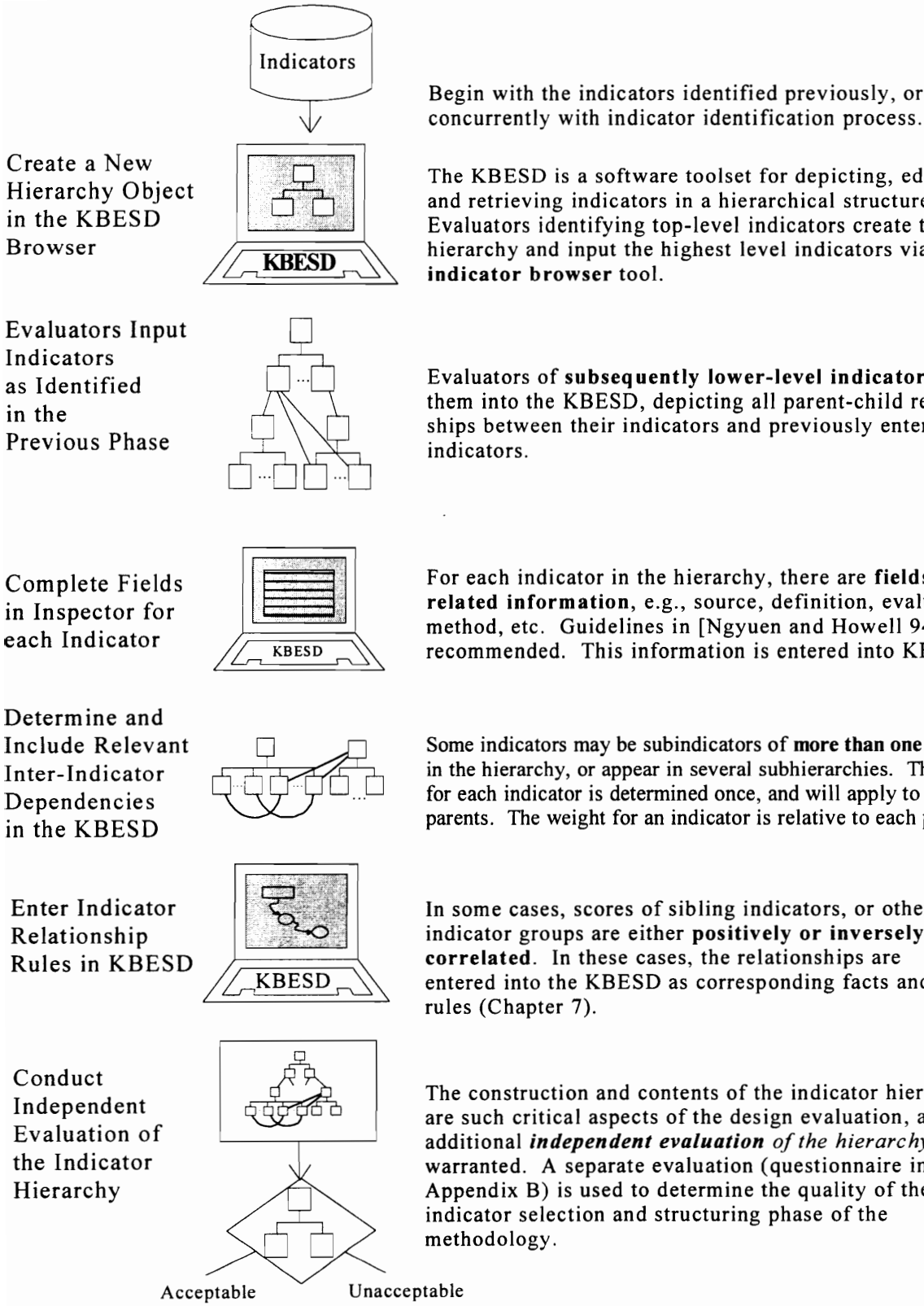


Figure 3.8. Indicator Hierarchy Construction Process.

3.3.4 Relative Criticality Weighting of Indicators Using AHP

Figure 3.9 provides a pictorial representation of the major steps involved in determining indicator weights. We recognize that the combination of sibling scores to derive an aggregate score for their parent indicator is not necessarily a simple linear combination of the children indicator scores. In some cases, particularly with metric- and simulation-based indicator scores, regression coefficients may be derived and used to combine components of the overall indicator score. In more complex cases, the training of a neural network of the children indicators may be desired to establish the sibling weights. In this section, we briefly describe how the Analytic Hierarchy Process (AHP) is used in the methodology to determine weighting coefficients for sibling indicators. Greater detail is presented in Chapter 5.

The procedure for incorporating the AHP into the methodology is relatively straightforward, since from previous steps, the evaluators have generated a hierarchy of indicators in the KBESD. Next, evaluators make pairwise comparisons of relative importance between sibling indicators. Consistency of comparisons is next computed using methods described in Chapter 5. When satisfactory consistency has been achieved, the relative weights of the indicators in a sibling group are computed.

The computation of and convergence toward consistency is included in the iterative process of making pairwise comparisons. *Expert Choice* computes consistency automatically and offers several options for adjusting comparisons, including re-presenting the indicators in a random order to remove bias due to the order of presentation. More sophisticated methods for consistency adjustment are discussed in sources presented in Chapter 5. These methods may be implemented in software tools such as *Mathematica* or some scientific spreadsheets.

Evaluators can easily perform the related functions in a PC-based AHP tool, such as *Expert Choice* [Saaty 93]. Evaluators can reproduce subhierarchies, perform pairwise comparisons, and have consistency automatically computed in *Expert Choice*. Results from these off-line computations are entered into the KBESD.

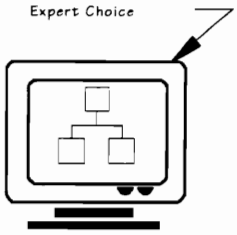
Determine Weight (Coefficient) Source for Indicator Siblings

$$S_j = \sum_i w_{i,j} S_i$$

weight coefficient

Quick Overview: The **aggregate score** (Sj) for an intermediate indicator, j, is a function of the scores (si) of its children indicators. Some scores may be **combined** non-linearly, polynomially, etc., and **normalized** to a uniform range (e.g., [0,1] or [0,100]). Expert evaluators for an indicator determine the **childrens' score combination function** as weights of criticality

Apply Analytic Hierarchy Process



Many Sibling Groups are combined **linearly**, so weights **sum to unity**. For these cases, the Analytic Hierarchy Process (AHP) is used. The AHP is applied to determine child indicator weights (wij) **within a sibling group**, relative to the common parent indicator.

Enter Indicator Hierarchies into AHP Tool

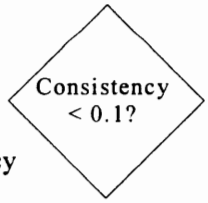
The AHP can be applied with a **PC-based software tool** such as *Expert Choice*. Evaluators for an indicator sibling group enter the **indicators** into **Expert Choice**, depicting their hierarchical relationships. An indicator with **multiple parents** is **weighted separately for each** sibling group in which it belongs.

Perform Pairwise Comparisons between Sibling Indicators

	a	b	c	d
a	1	2	6	7
b		1	3	6
c			1	5
d				1

The AHP **reduces complexity** by relying on pairwise comparisons between indicators in a sibling group. *Expert Choice* facilitates making pairwise **comparisons of elements**. Comparisons for two elements (**a**,**b**) are recorded as values, *v*, from the **range [1,9]**. *v*=1 means **a** *is equally important as* **b** and *v*=9 means **a** *is exceedingly more important than* **b**. The value for (**b**,**a**) is 1/*v*. By definition, (**a**,**a**) =1.

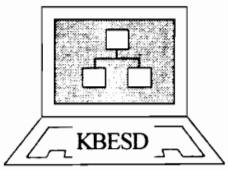
Compute/Examine Subhierarchy Consistency and Perform Consistency Adjustments as Necessary



Using techniques of [Harker 87a,b; Wedley, et al. 93], evaluators initially perform a **minimum set** of comparisons. In the worst case, a single group of *n* siblings could require $n(n-1)/2$ comparisons.

Expert Choice **automatically computes consistency.** If the **consistency ratio** is unacceptable (> 0.1), techniques by [DeTurck 87; Blankmeyer 87; Hughes 93; Wedley 93; Golden and Wang 89] may be applied.

Record Indicator Weights in KBESD



When consistency is achieved for an indicator sibling group, the weights are applied to the individual indicators in the appropriate fields in the KBESD hierarchy browser.

Figure 3.9. Indicator Weight Determination Process.

When consistency among individual sibling groups is satisfactory, a large-scale computation of the consistency of the hierarchy should be performed as a final check. Saaty [94, p. 126] advises that a measure "in the neighborhood of 0.1" is sufficient.

We do not recommend against the lottery method of Keeney and Raiffa [93] or other similar techniques as a means of assigning indicator criticality factors. In some indicator groups where pairwise comparisons pose a particular problem due to company politics or otherwise, we recommend these techniques as alternatives. Details of applying this procedure appear in e.g., [Thurston 91].

Figure 3.10 depicts an illustrative hierarchy of indicators for the quality concept of "System Survivability." The main concept has been refined into five indicators, each of which has been further refined. In practice, each lowest-level indicator shown is next either further refined, or determined by expert evaluators to be at its base level. Illustrative pairwise comparisons for the top-level indicators are displayed in Table 3.1. Based on the pairwise comparisons in the table, the relative importance for the five indicators, respectively is computed to be (.153, .378, .304, .103, .061), with consistency of .099. Formulae for computing these values are presented in [Saaty 94].

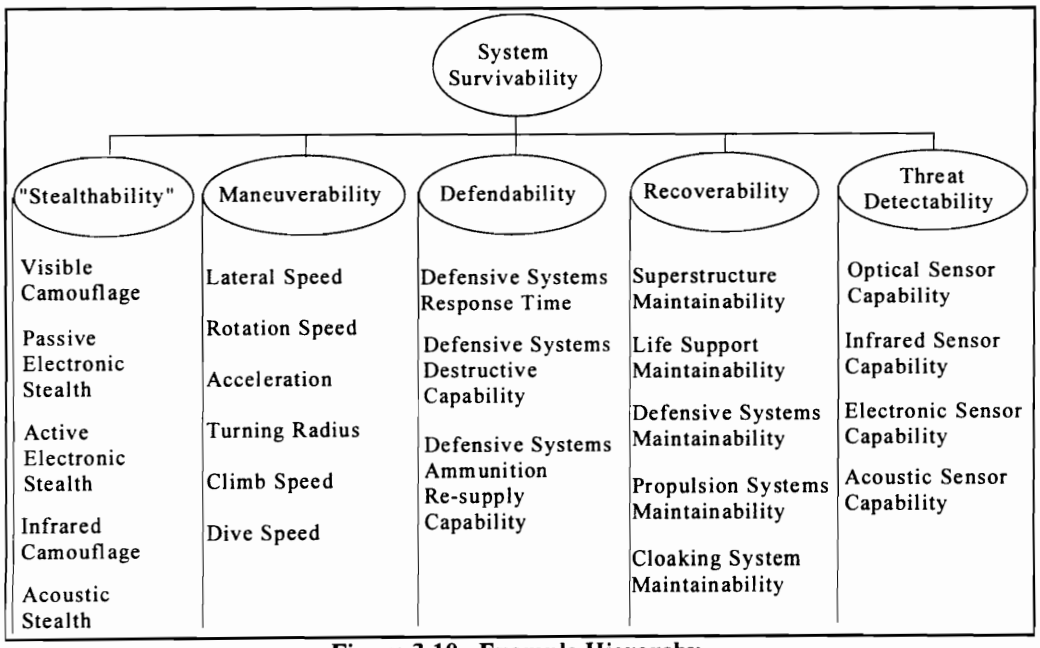


Figure 3.10. Example Hierarchy.

Table 3.1. Example Pairwise Comparisons.

Survivability	Stealth	Maneuver	Defend	Recover	Threat Det
Stealth	1	1/2	1/3	2	3
Maneuver		1	2	5	4
Defend			1	5	3
Recover				1	4
Threat Det					1

3.3.5 Determining and Assigning Scores for Indicators

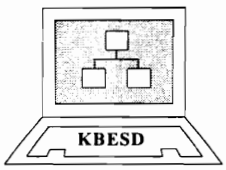
In Section 3.2.2 and Figure 3.3 we address four basic indicator score sources. In Figure 3.11 we outline the score assignment process. Figures 3.12, 3.13, and 3.16 provide detailed steps for applying each of the score types.

From Figure 3.11, we see the following steps in the scoring process, regardless of the score type: (1) Determine the score range which is desirable for each indicator, (2) Compute or assign the indicator score, (3) enter the score into the KBESD, and (4) respond to any flags raised by the KBESD knowledge base regarding the score. Determination of score ranges is based on published standards, expert knowledge, or system requirements. Score Assignment is based on the defined score source types. Scores are entered into the KBESD, along with knowledge-based rules establishing warning conditions which raise flags. Addressing flags is part of the evaluation.

3.3.5.1 Metrics-Based Scores

Some base-level indicator scores can be a function of directly observable (table lookup), measurable, or computable metrics (Figure 3.12). As an example, consider a metric for software design architectural complexity [Card and Agresti 88]. To measure architectural complexity, we require the intramodule complexity and the intermodule (structural) complexity. The structural complexity is itself another computed metric. It is a combination of the "fanout" of each module (directly measured), and the number of modules (directly observed). Our intent is to point out the potential complexities in dealing with metrics. In Chapter 6 we address this issue in considerably more detail.

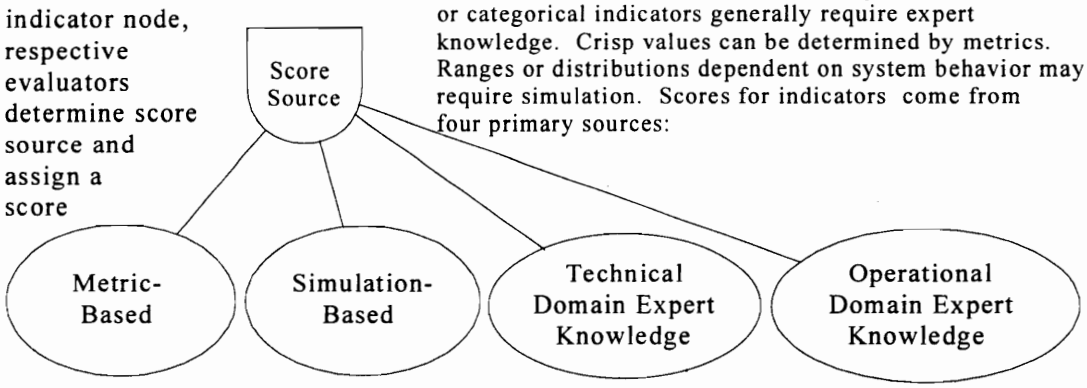
For indicator nodes, Expert Evaluators determine acceptable score ranges



Evaluators assign scores to the indicators within their subcategory or subdomain of expertise. Only leaf nodes require score assignment. Other nodes require the determination of acceptable score ranges. Experts in the respective subdomains enter the ranges for indicator scores in the KBESD knowledge base.

For each leaf indicator node, respective evaluators determine score source and assign a score

A key step is determining the most appropriate **source for scoring** leaf indicators. Highly qualitative or categorical indicators generally require expert knowledge. Crisp values can be determined by metrics. Ranges or distributions dependent on system behavior may require simulation. Scores for indicators come from four primary sources:



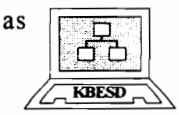
1) A function of formula-computable metrics. This is a **crisp score**. It can be normalized based on the desirable ranges of values which the function of the metric scores can take.

2) An interpretation of the results of, e.g., system behavior simulation or simulation of system-level components. This score may be based on a **confidence interval** or a probability distribution.

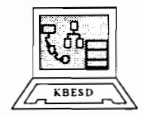
3) A synthesis of Technical Domain Expert Knowledge. This score may be **qualitative**, and can be represented and manipulated as an interval or **fuzzy value**. Normalization is an integral part of the membership of the value in its fuzzy set.

4) A synthesis of Operational Domain Expert Knowledge, similar to technical expert scoring. This score comes from a seasoned veteran in the operational domain for which the indicator is applicable. It is likely a **fuzzy** value as in 3).

Assign Score as Determined According to Type



Enter Score and Rationale in KBESD



Respond to alerters from KBESD expert system

Each evaluator's raw score and rationale for the score are entered into the KBESD. The KBESD's computational engine computes a final, normalized score (converts fuzzy interval to crisp value) and **performs checks** on each entry to check for spurious values, based on information entered in previous stages. Evaluator's rationale becomes important when seemingly spurious scores are detected.

Responses to alerters include re-evaluating a score, altering a triggering rule, or providing strong justification for the score to stand as is.

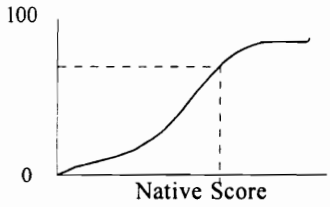
Figure 3.11. Indicator Score Assignment Procedure.

Determine Applicable Metric(s)

$\{m_1, m_2, \ldots, m_n\}$

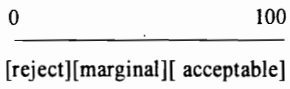
When an indicator score is to be determined as a function of computable metrics, evaluators for the indicator must determine the applicable metric(s) and their formulae. If several evaluators are assigned to the same metric, each evaluator may employ a separate set of metrics for determining the eventual indicator score.

Determine Mapping from Metric(s) to Indicator Score



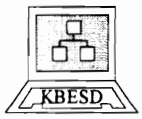
The assigned indicator score may be a **combination** of several metrics. When this is the case, the evaluator must determine the mapping from the raw metric value(s) to the indicator score.

Determine Applicable Score Ranges



The combination function leads to the definition of a **worst case range** of values which the combined score could take. From within this range, evaluators need to determine the **acceptable range** of values a score can take. This range is used to normalize the scores into [0,100] and must be **entered into the KBESD**.

Record Ranges in KBES



Ranges entered into the KBESD are simply intervals. The KBESD uses the range boundaries and rules in the knowledge base to perform checks on input scores against the ranges. Many applicable rules can be applied to enhance the credibility of the evaluation.

Compute Metric(s)

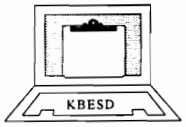
When computing a score for a metric-based indicator, the evaluator of the indicator computes the applicable metrics.

Map Metrics t Score

$S_i = f(m_1, m_2, \ldots, m_n)$

After the raw metrics are computed, the evaluator **combines the metrics** to obtain a raw score for the indicator and input the score into KBESD.

Make Notations in KBESD Regarding Results of Score



The KBESD computational engine uses the score ranges from above to ensure the raw indicator score falls within the established ranges. An unacceptable score **triggers** a warning from the expert system, which requires evaluators to re-examine the score. If the score is to be accepted anyway, the evaluator should **place notations in the KBESD** to justify keeping the score as acceptable

Figure 3.12. Indicator Score Determination based on Computable Metrics.

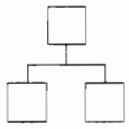
Metrics yield scores which are unit-less, or in a native unit particular to the metric. Evaluators must understand how the "goodness" of the raw metric score varies as the score itself spans its possible or practical range. For example, scores on the upper end of the native range may be the most desirable. Other *score desirability* mappings are possible. Such a mapping ultimately converts a native metric score to the standard range [0,100] which we uniformly apply throughout the methodology. If the results of more than one metric, or different results from multiple evaluators are used to obtain a score, then a combination function is likely required to map the metric values to the raw indicator score. Evaluators indicate ranges for indicator scores (e.g., acceptable, reject, marginal). Based on the ranges, evaluators create score-vs.-range-related facts and rules in the KBESD. KBESD uses the rules to perform integrity checks and other functions, after the scores are entered. Chapter 7 describes the knowledge-based aspect of metrics-based scoring in greater detail.

3.3.5.2 Simulation-Based Scores

In the course of evaluating the design of a complex system, numerous indicators are identified which evaluate the design's accommodation of dynamic or operational aspects of the system (e.g., *high-load computer system response time*, *minimum vessel turnabout time*, and *facility evacuation completion time*). These dynamic characteristics are to be assessed with respect to various operational scenarios, e.g., steady state, idle, and escalated activity level, in environmental extremes, and in peace time, threat, or emergency.

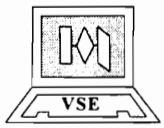
For system components or small subsystems, mockups or prototypes are often used. However, for system-level evaluation, this approach is often limited by the proportionately increasing cost of including a necessary degree of realism in the evaluation. An alternative to physical models are computer simulation models of various system behaviors. Simulation models are less expensive, reconfigurable, tolerant of error in application, and highly reusable. An outline of the process of applying simulation to indicator evaluation is presented in Figure 3.13.

Determine Indicators which require Simulation to Score



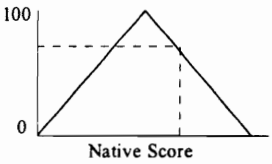
Indicators related to **dynamic** aspects of the system under design are the most likely candidates for evaluation by simulation. Evaluators decide which indicators should be scored through simulation, taking into consideration all operational scenarios in which the indicator's value may be applicable to design evaluation.

Develop / Test Simulation Model



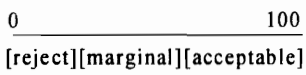
Evaluators with expertise in system simulation use the **Visual Simulation Environment (VSE)**, a state-of-the-art simulation environment, to **rapidly** develop **simulation models** of complex systems. An **object-oriented scripting language** facilitates construction of complex models and reduces development time.

Determine the mapping from Simulation output to Indicator Score

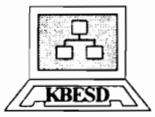


The output from simulation may be a **confidence interval**, a **mean** and an interval half-width. etc. These results may be **combined** or manipulated by some function to give the raw indicator score. Otherwise, an expert evaluator may apply knowledge from experience with the indicator to directly assign a value to the indicator based on interpretation of the simulation results.

Determine Applicable Ranges for Score

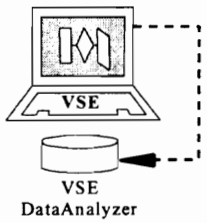


Enter Score Ranges in KBESD



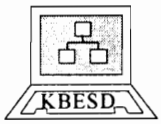
As with metric-based scores, evaluators need to consider all possible values for the indicator score, and determine **what range is acceptable** for a simulation-based indicator. The range is used to **normalize** the raw score and **provide a check** on the score's value against the stated range. This information must be entered into the KBESD's knowledge base.

Experiment with the Simulation and Obtain Results



The simulation experiments are run using the Visual Simulation Environment (VSE) and **output data** are collected. **Analysis** and interpretation of the data is generally required before an actual indicator score can be assigned. The VSE DataAnalyzer can be used.

Map Simulation Results to Indicator Score



By the means determined by the expert evaluator, simulation results are used to **assign a score** for the indicator. This score may later be **normalized** into the range [0,100] for the score aggregation process.

Input Scores into KBESD

Evaluator(s) **enter the score** into the KBESD.

Figure 3.13. Determining Indicator Scores based on Simulation.

In dealing with the results of simulation, we do not anticipate discrete values. Instead, we anticipate results such as probability of occurrence, frequency of occurrence, and expected values, expressed as confidence intervals about an estimated mean value. As a consequence of this, we are required to address issues such as: (1) the appropriate confidence level and the range of possible values it defines, (2) relationships between simulation-derived estimates and the operational scenario under which they are obtained, and (3) the *desirability* mapping from any value in the confidence interval to the standard score range [0,100]. We address the mathematics of these issues and others in detail in Chapter 6. Knowledge types appropriate for interpreting simulation-derived values are described in Chapter 7.

An aid to simulation model development is the Visual Simulation Environment (VSE), developed at Virginia Tech [Balci, *et al.* 95] (Figure 3.14). The VSE is an object-oriented environment for easing the development of simulation models. The VSE allows simulation of the dynamic aspects of a system under design at essentially any stage in the design process, from the requirements specification forward. As more details and system design parameters become available, more detailed models can be constructed and nested. Ultimately, a hierarchy of models can be used to simulate variously detailed behaviors of the system. This is particularly useful for observing simulated behavior of the integrated system, since lower-level models feed results into the upper levels, allowing a view at any desired level of abstraction.

Deriving reliable and useful results from this aspect of the methodology requires repeated experimentation with the simulation model at various levels of detail, for perhaps a suite of operational scenarios. The output for further analysis and inclusion into the evaluation process would be confidence intervals for the modeled aspects of the system. These results would be normalized and mapped to value ranges consistent with other indicators in the hierarchy.

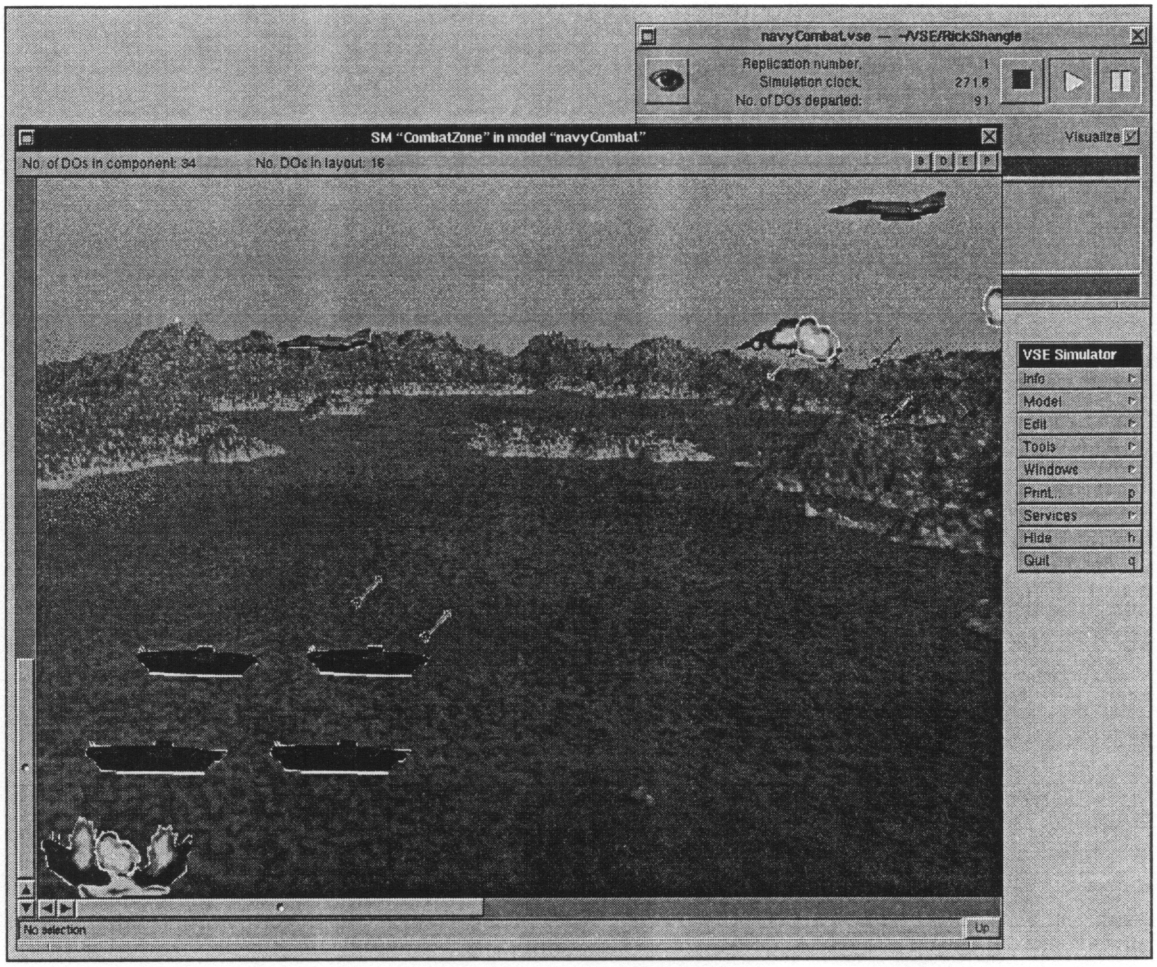


Figure 3.14. Example VSE Simulation of a Naval Combat Scenario.

3.3.5.3 Technical Domain Expert Knowledge-Based Scores

For design quality indicators for which neither computable metrics nor simulation results alone provide a basis for a valuation, evaluators in technical disciplines rely primarily on their expert knowledge in the field(s) encompassing the indicator. Categories of the expert knowledge which apply include: relationships between indicators within and between major indicator domains, maximum and minimum acceptable values for indicators, and relative quality of a design indicator compared to others they are familiar with from experience.

In dealing with the scores from a subjective perspective, interval and fuzzy mathematics provide a broad class of arithmetic concepts and operations. A brief review of basic fuzzy value concepts is in order. The field of fuzzy mathematics [Klir and Yuan 95; Zimmerman 91] provides the *membership function* (Figure 3.15) as an instrument to map elements of the universe of values to a *degree of belonging* to a *fuzzy set*. *Alpha cuts* are used to narrow the range of acceptable elements from the universe by requiring a minimum membership value of $\alpha \in [0,1]$.

Membership functions can be triangular, trapezoidal, polynomial, etc. In Figure 3.14, the triangular membership function may be represented as the triplet (a_1,a_2,a_3), where a_1 and a_3 are the extreme min and max values from the universe which bound possible members, and a_2 is the most desirable value. Functions for which the min or max value are the most desirable may have a triplet of the form (a_1,a_1,a_3) or (a_1,a_3,a_3), respectively.

Fuzzy values are often assigned instead of crisp ones, especially when qualitative elements are assessed. One method of incorporating fuzziness is the mapping of a crisp value into a *fuzzy membership value* via the membership function. In other cases, an *interval of confidence* is more appropriate. In this case, an evaluator assigns an interval as a score, reflecting the best and worst values the indicator is believed to be worth. In our methodology, a combination of these two methods is

allowed. The basic method of applying fuzzy measures in the methodology is depicted in Figure 3.16. Details are provided in Chapter 6, but we provide process highlights here.

The process initiates when evaluators determine that an indicator requires direct application of expert knowledge instead of metrics or simulation results. First, evaluators determine if the indicator can be considered so critical to the entire evaluation that if the indicator cannot be scored above its "rejection" threshold, the entire design fails. There are relatively few such indicators, but the methodology allows for their identification and treatment.

Scores above a rejection level are considered *acceptable*, but to draw knowledge-based conclusions (Chapter 7) regarding *degrees* of indicator quality, the acceptable range is further subdivided and qualified. In the figure, the demarcation between acceptable and *marginal* can be placed on the score continuum by superimposing a fuzzy membership function and applying an alpha cut, α.

$$\mu(x) = \begin{cases} 0, & x < a_1 \\ \dfrac{x - a_1}{a_2 - a_1}, & a_1 \le x \le a_2 \\ \dfrac{a_3 - x}{a_3 - a_2}, & a_2 \le x \le a_3 \\ 0, & x > a_3 \end{cases}$$

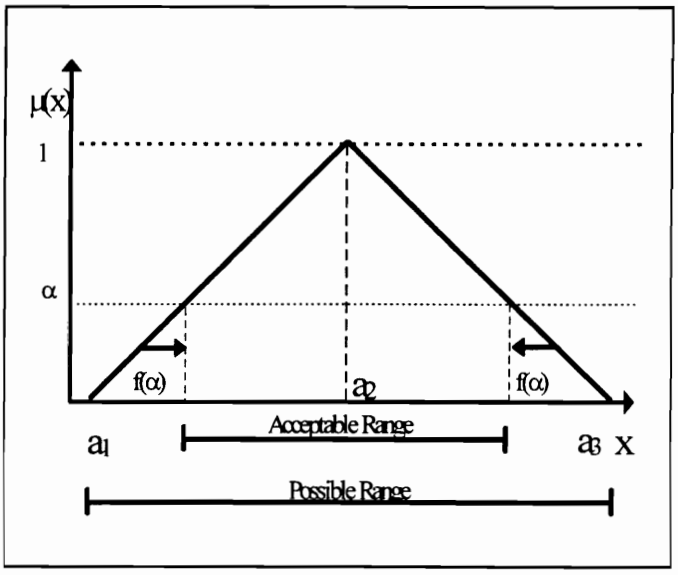


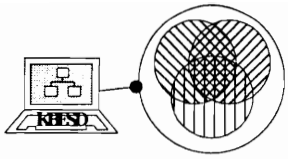
Figure 3.15. Example Triangular Fuzzy Membership Function and Graph.

Next each evaluator assesses the quality of the indicator by assigning an interval from the range [0,100], say [65,75]. Using techniques described in Chapter 6, interval scores from each evaluator are combined into one. The result of the aggregation is a single interval representing the (perhaps weighted) mean of the intervals assessed by evaluators.

Computationally, crisp values are easier to deal with than intervals. Thus, a means of extracting the *essence* of the interval as a crisp value is needed. We refer to this as the process of *defuzzifying* the interval. The key to interval defuzzification is the truncation and collapse of the interval by a criticality factor, $k \in [0,1]$, whose value is associated with pessimistic, moderate, or optimistic evaluation posture. Thus, the value of an indicator may be evaluated differently at varying values of k, using the technique $(1-k)$*lower bound + k*upperbound. For example, a low k factor (pessimistic) would collapse the fuzzy interval near the upper bound, and a high k value (optimistic) would collapse the interval near the upper bound. Intermediate values of k collapse the interval at various points in between. Employing what-if analysis by varying the value of k is discussed in Section 3.3.9.

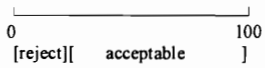
Ultimately, interval aggregation and defuzzification are performed by the KBESD. The evaluators must determine the score ranges, criticality factors and score intervals. The KBESD applies knowledge supplied earlier by the evaluators to perform various score checks. Under the rules in the knowledge base, the evaluators are notified of scores which fall outside of desirable intervals. The KBESD performs other knowledge-based functions described in Chapter 7.

Determine which indicators must be scored from expert knowledge



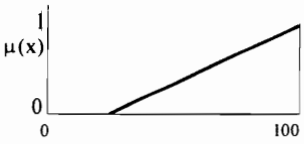
Expert evaluators determine that some scores cannot be assigned based on metric combinations or simulation results. These scores require the **application** of **technical or operational domain expert knowledge**.

Determine Accept/Reject ranges for scores



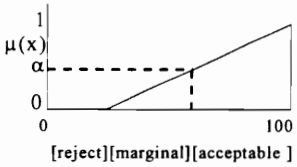
Evaluators determine "reject" thresholds for scores and enter the ranges into the KBESD.

Define Fuzzy Membership Functions for Score



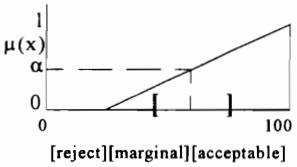
Scores based on expert knowledge are treated as fuzzy" and manipulated via Fuzzy Mathematics. The fuzzy set is the set of possible scores which can be assigned to a particular indicator. Membership functions represent desirability of possible scores, in the range [0,1].

Determine Alpha Cut for acceptable scores



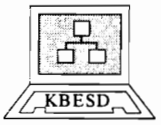
An alpha-cut is a threshold for minimum membership in a fuzzy set. An evaluator selects an alpha-cut to delineate the acceptable vs. marginal range of scores. This information is used in the knowledge base for evaluation enhancement.

Assess Score as an Interval in [0,100]



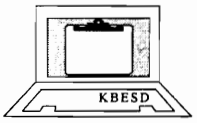
Each evaluator for the indicator assesses the value of the indicator as an **interval**, since a crisp score over the range of [0,100] is too precise to accept.

Input Score into KBESD



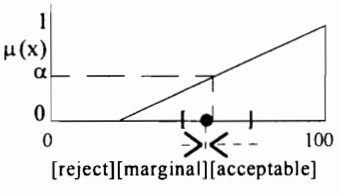
Each evaluator enters the score interval into the KBESD. The intervals for all evaluators are combined according to the axioms of fuzzy mathematics, to yield an aggregate. However, each interval is retained in the KBESD for knowledge-based analysis.

Input Score Rationale into KBESD



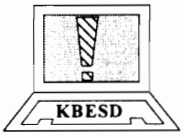
In this case of scores based on expert knowledge, the evaluator enters **rationale** explaining how the score was determined. This is a **useful information** resource for other evaluators to **consult**, especially if the score is not consistent with others.

KBESD Defuzzifies Score with respect to desired level of "what-if" analysis



The KBESD can collapse the fuzzy score with respect to any degree of pessimism or optimism within the score interval. Doing so allows "what-if" analysis to be performed on the entire hierarchy depending on the level of conservative bias desired. The resultant score can be assigned a **membership value**, based on the indicator's membership function. The knowledge base assesses the design based on these values.

Evaluator Responds to KBESD Triggers



Information such as other judges' **scores**, **rules** for score **relationships**, score **ranges**, etc. are input into the KBESD as part of the indicator identification process. These rules are used to determine **seemingly spurious** scores and to **raise flags** requiring attention.

Figure 3.16. Application of Technical and Operational Domain Expert Knowledge to Scoring.

3.3.5.4 Operational Domain Expert Knowledge-Based Scores

Some indicators can only be evaluated by expert knowledge of veteran operators from the operational environment in which the designed system is to be employed. Expert knowledge of this type may consist of a collection of rules of thumb or heuristics, based on many years of experience in the operational domain of similar systems. Indicator classes of this nature include e.g., *threat assessment*, *survivability*, and *environmental influence* for a military system.

These indicators are scored using interval values in a similar fashion as domain knowledge-based scores (Figure 3.16). The intervals are defined as fuzzy linguistic ranges, and treated as fuzzy sets. This topic is discussed in detail in Chapter 6. Additionally, an alternative method adapted from [Klir and Yuan 95] may be employed.

3.3.6 Application of an Expert Knowledge Base

In the previous two sections, we refer to the application of technical and operational domain expert knowledge in complex system design evaluation. We discuss the application of the heuristics as an aid and guide to scoring indicators. Heuristics of this type, such as ranges and tolerances, are not so useful printed in a document or laid out in voluminous checklists. They would be more useful if they were incorporated in the knowledge base of a computer-based expert system. There, combined with many similar rules, they could serve as flags and guides to evaluators who must input scores for related indicators.

There are three important aspects to consider regarding the incorporation of an expert knowledge base: creating it, accessing it, and applying it. In this section we describe how those tasks are accomplished using the knowledge-based system capability of the KBESD.

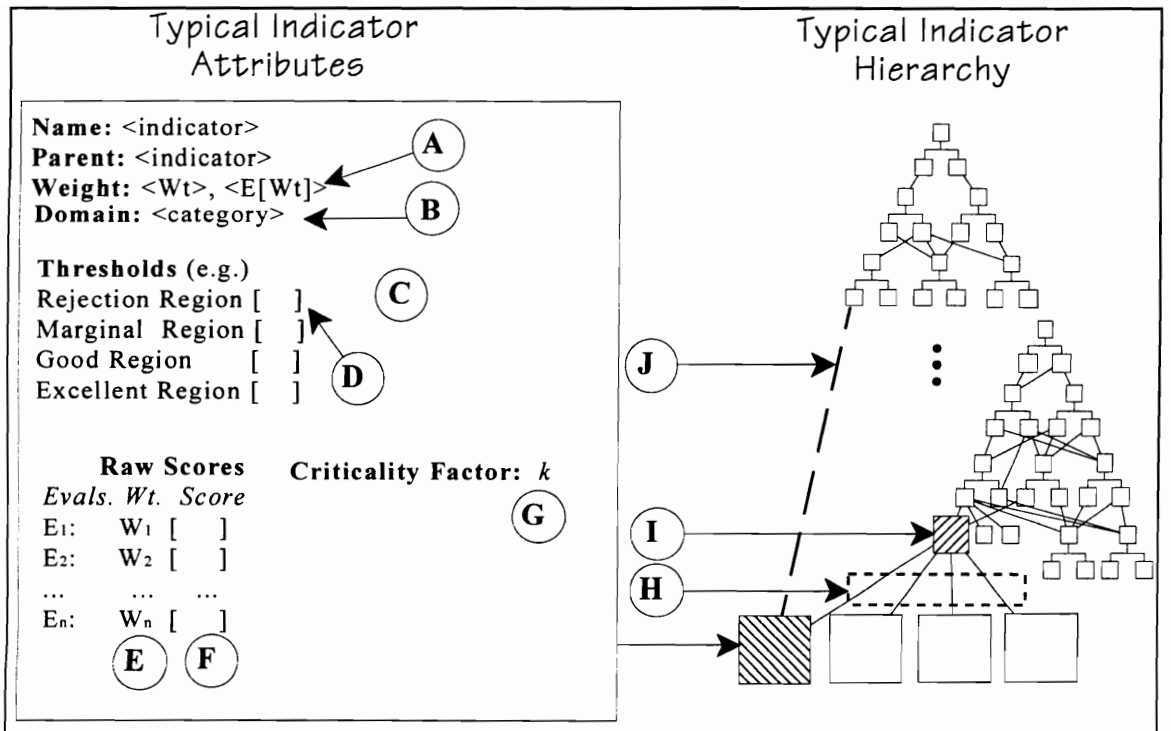
We facilitate the creation of the knowledge base by providing an expert system shell in the KBESD tool. Using the shell, expert evaluators and evaluation organization personnel use the shell language to input indicator-, domain-, and/or

evaluator-related facts and rules into the knowledge base. Example facts include upper/lower bounds (ranges) on indicator scores and expert precision and bias factors. Experts also input rationale for the facts and rules, if desired or required by the evaluation organization's policies. These rules would be available in pop-up windows in the application sessions described below. Chapter 7 presents the knowledge base details, but an overview of the types of facts and rules employed in KBESD is depicted in Figure 3.17.

Accessing the knowledge base can take place in two ways. First, it can be accessed directly, when bare facts and rules are available for editing as during its creation or other editing sessions. Second, the knowledge base is accessible during an interactive session with the KBESD. A user in the process of entering a score for a particular indicator may seek the "expert advice" from the knowledge base for an appropriate range, or ballpark figure.

A pictorial representation of uses of expert knowledge and rules is provided in Figure 3.17. In a typical scenario, as an expert evaluator inputs knowledge-based scores, the inference engine of the KBESD would be performing checks on the data. If an input figure is out of the ranges the experts have previously offered, a series of rules would be triggered resulting in the user being notified. At this point the user would have the opportunity to consult with the knowledge base. Consultation may include viewing a trace of the logic which fired the warning, and browsing any textual comments placed in the system during the knowledge base creation sessions.

Subsequently, if the expert evaluator is confident of the input, or knows he has broached an area as yet not considered by the other experts, he may override the "experts" and make notations of his own rationale. Used in this way, the KBESD provides both an impetus and a forum for surfacing and addressing these differences. The goal is that it works to the overall good of the evaluation.



Evaluation-Enhancing Knowledge Types Pertain To:

A- The ratio of the weight to its expected value (1/n, see G)
B- Relationships between this indicator and other indicators by category
C- Ranges for the absolute and relative threshold widths
-- based on operational scenario, indicator relative weight, domain, etc.
Relationship between indicator weight and upper bound of Rejection Interval
D- Requirement for designating Rejection threshold with "Design Critical"
E- Determination of relative weights for evaluators
Dispersion, Domination, Dwarfing of weights among evaluators
F- Scoring advice from standards or historical data
Dispersion of scores
Width of score interval
Value of score relative to standard/historical data
Fuzzy Interval combination methods
G- What-if analysis based on min/max of score ranges, driven by *k* factor
Determination of Optimum *k* factor for minimum acceptable aggregate scores
Desirable values for *k* to be used in what-if analysis
H- Inter-sibling trade-off analysis comparisons by weight, score
Dispersion, Domination, Dwarfing of weights among siblings
I- Aggregate score extremes analysis based on weights, *k* factor
J - Consistency of scores of indicators in non-structural relationships
Adherence to constraints in non-structurally related indicators

Figure 3.17. Applications of Knowledge in Design Evaluation.

3.3.7 Aggregating Indicator Scores

Figure 3.18 depicts the process of combining weighted sibling scores to get a score for the parent indicator. After completion of the previous phases of the methodology, all base-level indicators are already scored and weighted. The final step in the scoring aspect of the methodology is the aggregation of children scores to determine the parent's score. Figure 3.18 shows two parent indicators, *j* and *k*, and their sibling indicators. Since base-level indicator x_v is a child of both parents *j* and *k*, it must have a relative weight for both sibling groups of which it is a part. Notice that the performance score for x_v is the same for both parents. The hierarchy aggregation process then involves the recursive application of the aggregation function up the hierarchy.

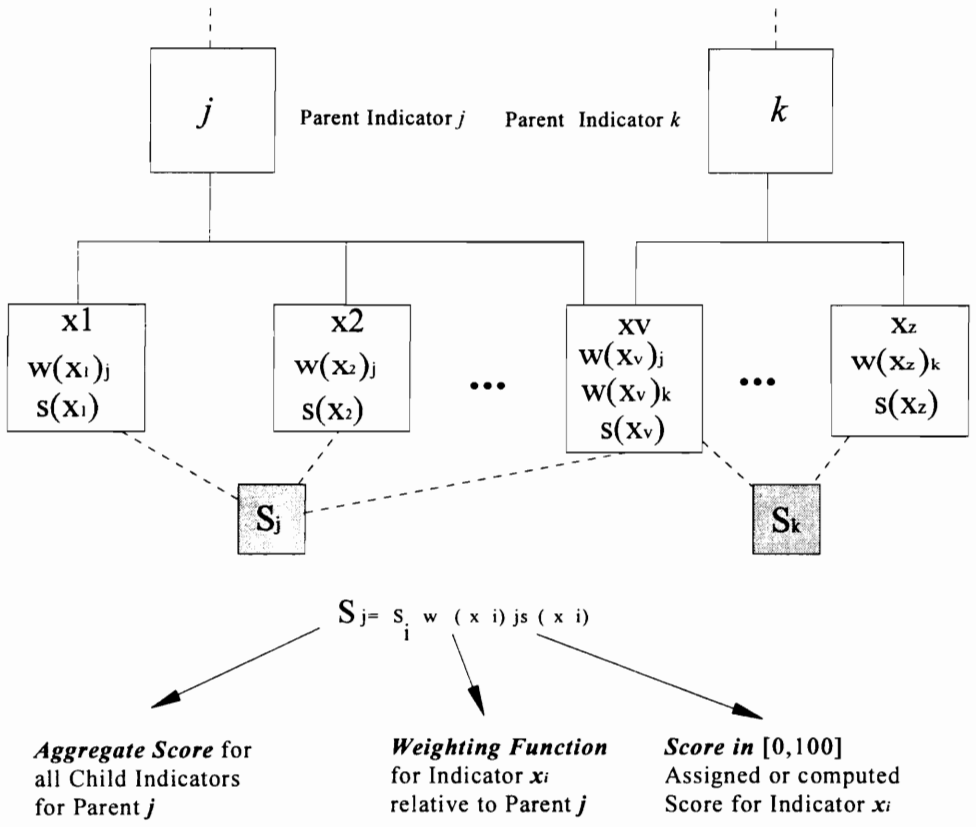


Figure 3.18. Score Aggregation Method.

During this phase, scores from multiple evaluators are combined for each indicator. Also, all native scores from metrics and simulation-derived confidence intervals are mapped to the range [0,100]. Finally, interval scores and fuzzy intervals are collapsed to extract a discrete value.

3.3.8 Representation of Indicator Scores using Kiviat Graphs

The Kiviat graph is a visual representational device for depicting several indicator scores simultaneously. It has been a particularly effective device for representing the scores of computer performance-related measures [Ferrari, *et al.* 83, p. 195]. Some indicators indicate desirable performance when the score is low, e.g., *response time*. For others, e.g., *throughput*, the higher the score, the better it is. The Kiviat diagram is capable of displaying both types of indicators simultaneously. Since the evaluation methodology we are describing is indicator based, and the indicators are expected to be analyzed in groups of less than a dozen, the Kiviat graph is a suitable display device.

After aggregate scores are computed, Kiviat diagrams can be produced for indicator sibling groups, as desired. Desirable indicator groupings could include sibling indicators from a parent, or any collection of related indicators. Evaluators for the particular domain to which the indicators are relevant perform analysis of the diagrams as one means of comparing results to established thresholds. We propose that the diagrams and the analyses be included in the design review documentation.

A Kiviat graph, as in Figure 3.19, is constructed by assigning positions of a collection of indicators to points along the circumference of a circle. An indicator's score is marked on the graph as a point along the radius from the origin to the indicator's position on the circumference. A high number is marked close to the circumference. A low number is marked close to the origin. The scores of all indicators in the particular collection are indicated in this fashion. The polygonal region between the score points and the origin is then shaded, yielding, for example, the star-like shape depicted in Figure 3.19.

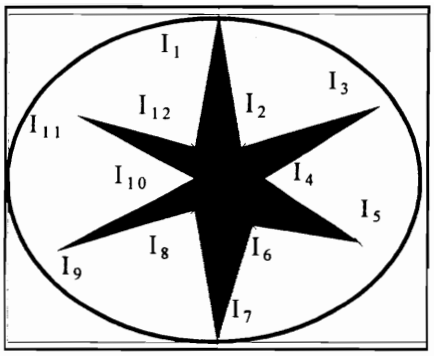


Figure 3.19. Example Kiviat Diagram.

To interpret a Kiviat graph, one must know which indicators are desired to have high scores, and which are desired to have low scores. Then, quickly by inspection, one can discern the goodness or badness of scores for the entire collection of indicators by the shape of the graph. For example, in Figure 3.19, assume the odd-numbered indicators are desired to have high scores, and even-numbered indicators are desired to have low scores. By inspection of the figure, every indicator appears to have a desirable score. Many other high-low indicator score combinations are possible.

3.3.9 Interpretation of the Results

In this section, we address the interpretation of the results obtained by using the methodology we have described. The purpose of the methodology is to guide experts through the complex and tedious task of system design evaluation. Particularly the evaluation of the design is with respect to design utility and satisfiability of requirements. The evaluation relies on applicable metrics and generally on the judgments of experts in the technical domains intersected by the system under design. To that end, the evaluation cannot be precise and its ultimate value requires interpretation.

For consistency and ease of comparison of alternatives, this method systematically bounds all base-level scores to a uniform range, e.g., [0,100]. Intuitively, as we aggregate scores up the hierarchy, the probability of maximizing the range quickly diminish to zero. The issue then becomes one of *degree* of acceptability.

Thus, a threshold is needed to drive the interpretation of the final score. Ultimately, the degree to which the score satisfies the threshold determines the acceptability of the entire design. However, since the designs addressed here involve complex hardware and software and highly skilled human operators, both intra- and inter-domain thresholds become important. Indeed, the unacceptability of the design within a single domain could invalidate the entire design.

This consideration becomes particularly important in the unlikely, yet conceivable scenario in which the final score is deemed acceptable, yet an intermediate result is not.

We do not leave the evaluating organization with simply the final aggregate score as the terminal point of the hierarchy-based evaluation. The KBESD allows for *what-if analysis* on the basis of the criticality factor, k. In what-if analysis, evaluators may apply a k value uniformly for all indicators or variously for different categories of indicators. The domain dominance scheme (above and in Chapter 4) for the hierarchy is a recommended starting point for determining differing values of k to employ. The KBESD knowledge base can also be used to contain information based on expert knowledge, which assesses the value of k for which the most realistic or plausible aggregate score *can* be attained.

Naturally then, the key to realistic and meaningful evaluation becomes the determination of the thresholds. As with the indicator valuations themselves, inputs from the respective domain experts and upper management decision makers are required. In the lower tiers of the indicator hierarchies, metricians and technical domain experts are the source for threshold establishment. At the points of inter-domain integration, operational experts make use of the results from lower-levels. At the system-level, operational domain experts join systems integration design engineers to come up with thresholds based on the results that are passed up. Ultimately, responsibility for passing judgment on the final results rests with project upper

management, under the counsel of systems integration engineers and system operational experts.

How does the threshold establishment and interpretation process unfold? Consistent with the systems engineering process, the evaluation process is iterative. The cycle involves establishment of initial thresholds, indicator weighting, evaluation, comparison with thresholds, possible re-weightings or re-evaluation, what-if analysis, then reconsideration of the thresholds themselves. We acknowledge this cyclic method could lend itself to varying degrees of score, weight, or threshold "rigging" to arrive at a desired score-to-threshold ratio. However, we trust that in the hands of responsible and reputable metricians, engineers, and project management personnel, the potential occurrences and effects of these actions would be controllable and negligible.

3.4. SUMMARY

An overview of a multifaceted methodology for evaluation of complex system designs is presented. The methodology is intended for implementation by independent evaluating organizations, responsible for evaluating the requirements satisfiability of the design. As a presentation format, we pictorially depict the conduct of the major phases of the methodology. We textually thread the phases together, providing motivation and commentary as necessary. Detailed information for the techniques to be applied in methodology phases are provided in subsequent chapters of the main document, of which this is a part.

The software centerpiece of the methodology is the KBESD, a software tool for the Knowledge-Based Evaluation of System Designs. The KBESD provides information storage, retrieval, and display for the components of the evaluation. The KBESD also contains the computational engine and expert system capability needed to conduct the knowledge-based evaluation of the system design. The VSE is also a vital software tool for use in the methodology. It provides an object-oriented visual

simulation capability which is useful for estimating indicators values relative to the operationally-dependent, dynamic characteristics of the designed system.

The methodology is indicator-based. Experts, from the technical/operational domains intersected by the system under design, examine the design and identify scenario- and domain-specific indicators to evaluate the design. Identification of indicators requires a step-wise refinement approach, which results in a (possibly large) hierarchical network of design quality indicators. High-level indicators decompose into more refined indicators until base-level indicators are identified.

The analytic hierarchy process is the recommended means of determining the relative criticalities between the children indicators of a composite indicator. Evaluators employ the techniques of AHP external to the KBESD (e.g., through the use of *Expert Choice*) to determine weights to assign to sibling indicators. Computed weights are entered into the KBESD.

Quality scores for the indicators come from four primary sources: computable metrics, experimentation with *visual* simulation models of the designed system, technical domain expert knowledge, and operational domain expert knowledge. Crisp or fuzzy scores are assigned to the indicators by evaluators in the respective domains. The weighted scores are combined up the hierarchy to yield an aggregate score for the design.

Expert knowledge employed through the KBESD is a key to enhancing the credibility and facilitating the usability of the methodology. Domain-specific expert knowledge is represented as facts and rules in the KBESD. The knowledge is used to serve as checks on the values of indicator scores as well as inter-indicator score parameters. Additionally, knowledge is applied to evaluate aspects of the indicator hierarchy and the evaluators themselves.

Results of the application of knowledge to the indicator hierarchy are graphically represented as Kiviat diagrams. These diagrams are particularly useful for simultaneously displaying the values of a collection of indicators.

4. IDENTIFICATION OF DESIGN QUALITY INDICATORS

4.1. INTRODUCTION

The initial phases of the methodology pertain to the identification of design quality indicators by technical and operational experts. Issues pertinent to these phases include the selection of expert evaluators, identification of major indicator domains, decomposition of major domains into indicators, and structuring the indicators into a hierarchy to represent influence relationships between indicators. A final concern is the assessment of the hierarchy itself. In this chapter, we elaborate these initial procedures.

The chapter flows as follows. Section 4.2 introduces the concept of design indicator domain schema analysis. Section 4.3 discusses evaluation panel considerations regarding the expert evaluator panel composition. Section 4.4 details the process of the identification of design quality indicators, and Section 4.5 details issues concerning the construction and assessment of the indicator hierarchy. Section 4.6 is a description of a generic hierarchy of over 400 indicators culled from the literature. Section 4.7 provides a summary and conclusions for the chapter.

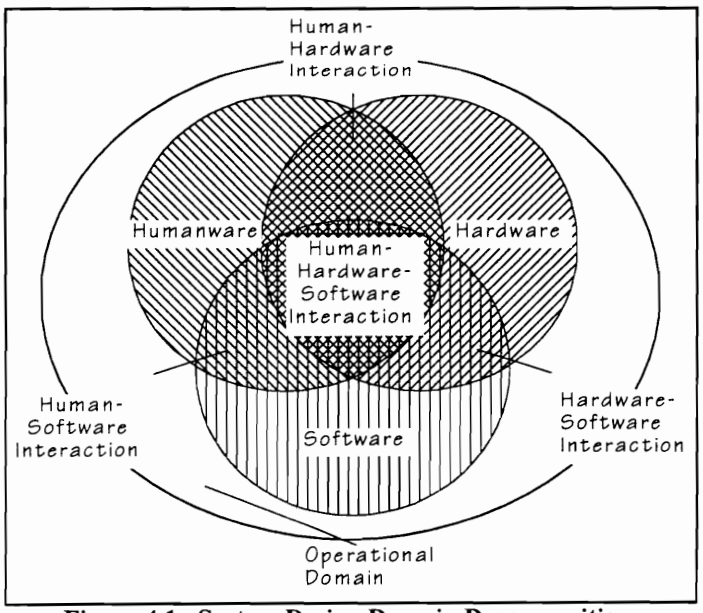


Figure 4.1. System Design Domain Decomposition.

4.2. DESIGN INDICATOR DOMAIN SCHEMA ANALYSIS

Our indicator-based methodology for complex system design evaluation, consistent with systems engineering methods, begins with a system decomposition into major functional and nonfunctional domains. A top-level decomposition provides a general domain schema like the one depicted in Figure 4.1. Specific system dimensions such as Performance, Dependability, etc. are superimposed on the major domains to yield high-level domains such as "Humanware Performance" and "Software Dependability." Domains recursively decompose into subdomains, from which indicators of design quality are defined.

Indicator domain schema analysis focuses on two key issues which must be addressed at the outset of the evaluation process: (1) what are the major indicator domains which characterize the system under design, and (2) what is the relative dominance of each indicator domain. The direction charted by the resolution of these issues guides decisions to be made in subphases described later in this chapter. We briefly discuss each issue below.

4.2.1 Domain Definition

Chapter 1 illustrates and expounds on the multifaceted nature of complex systems. The chapter's discussion conveys the distinct and interactive role of the major system components. Evaluating the design of a complex system requires accommodating the multi-domain nature of the system. Beginning with the system-level perspective, the roles of individual domains and their interactions must be defined, understood, and accounted for — this is *domain schema analysis.*

Domain schema analysis is a process of defining and understanding the organization and interplay of indicator source domains inherent in the system as designed. This analysis is accomplished by reviews of the system requirements documentation, system specification documentation, and similar documents/media, with attention to defining major domains and their interactions. Experienced systems

engineers and related-system operational experts review the documents and are able to discern and define an indicator schema for the current system as designed.

Figure 4.1 depicts three major domains, whose overlaps yield seven total top-level indicator domains to be addressed in design evaluation. An eighth circumscribing domain pertains to the operational environment in which the system is to be employed. Table 4.1 shows characterizations of these domains. The characterizations are intentionally general, and are provided for reference and completeness of the section. In practice, systems engineers may apply different mappings between the indicator domains we have listed and their characterizations (e.g., attributing some otherwise pure "software" characteristics to "human-software interaction"). With definitions of the key

Table 4.1. Characterization of Major Indicator Domains.

DOMAIN	CHARACTERIZATION/EXAMPLES
Hardware	(1) Structural, facilities, superstructure, expendable materials, consumables, load bearing parts, moving parts, containers. (2) Computer circuitry ("chipware"), fixed positions, embedded, logic-bearing.
Software	Computer programming language source code; object code; code storage, archival, and access methods; documentation.
Humanware	System operational and maintenance personnel; physical, mental, cognitive, and interpersonal activity; instructions; procedures; group dynamics; teamwork.
Human-Software Interaction	Human physical and logical input-output interface with computer-based systems; includes operation and maintenance of interface software.
Human-Hardware Interaction	Human physical input-output interface with computer-based systems, and other system hardware; includes operation and maintenance of interface hardware.
Hardware-Software Interaction	Embedded software in computer chipware ("firmware"); includes software-driven hardware; operated without human operator involvement.
Human-Hardware-Software (system-level) Interaction	System-level, operational domain interactions; requires synergistic human manipulations of system software and hardware via input-output interfaces.
Operational	Interaction of the system with operational environment; hostile, friendly, rapidly changing, marked by extremes.

constituents established, indications of system-level domain domination take the reins in steering the direction of the evaluation. This topic is addressed in the following section.

4.2.2 Domain Dominance

Chapter 5 is dedicated to determination of relative priority among a unilateral collection of entities: such techniques are not invoked in this preliminary phase. Rather, application of rigorous mathematical approaches is obviated by generally self-evident domain domination at the system level. Consider a few permutations of possible scenarios in the Figures 4.2(a-c).

Scenario (a) represents a system dominated by hardware and software components and their interplay. A class of example systems includes software guided munitions, rockets, and unmanned space vehicles, where human involvement is required for initial configuration and maintenance, but not in on-line system operation. Scenario (b) depicts a software-dominated system, where human and hardware involvement is involved with maintenance and execution of the mission-controlling software. Example system classes include robot-controlled manufacturing and production, where software-driven precision characterizes the value-added contribution of automation. Scenario (c) depicts a humanware dominated system. As an instance of this class of system, consider a maintenance facility. Here, although software and hardware diagnostic tools are involved in mission accomplishment, human labor and decision-making dominate the operational scenario.

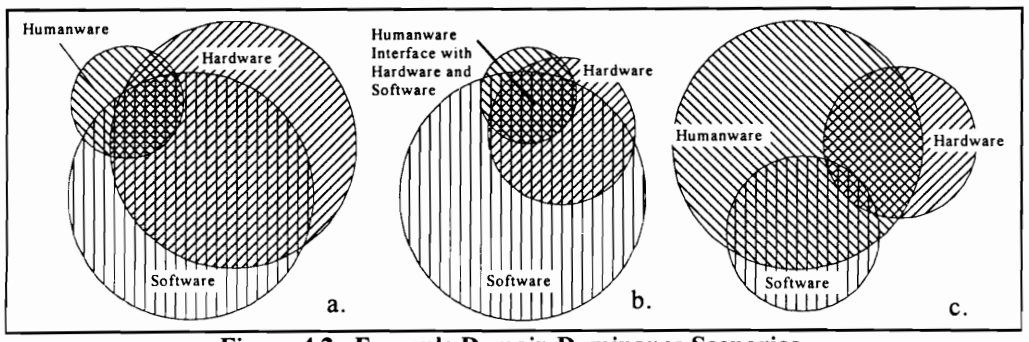


Figure 4.2. Example Domain Dominance Scenarios.

Having analyzed the proposed system under expected operational scenarios, indications of domain dominance become clear. This high-level information is to be applied in two subsequent phases of the evaluation. First, domain priority information is one class of knowledge employed in the knowledge-based portion of the design evaluation. Chapter 7 discusses the acquisition and employment of expert knowledge in the KBESD as an integral part of the evaluation.

Second, domain dominance information should weigh significantly in recruitment and selection of the evaluating organization expert panel. This is to ensure the availability of expertise in qualities and quantities commensurate with the domain dominance. Considerations for selection of evaluation panel are discussed in the next section.

4.3. EVALUATION PANEL IDENTIFICATION

In the following sections, we address considerations pertaining to the identification and selection of expert evaluators for system design quality indicators. The quality of the evaluation panel is critical to the quality and credibility of the evaluation results. Thus, at the outset of the discussion of the methodology, we offer guidelines for building in evaluator quality.

4.3.1 Expert Evaluator Selection

Existing consulting companies who function as independent evaluating organizations (e.g., Booz, Allen, and Hamilton, consultants to the DOD) may maintain a relatively static expert panel, with an expertise base heavily slanted in one major domain or another, such as software engineering. This is particularly likely to be the case if the company specializes in evaluating a narrow class of systems, e.g., avionics software. However, for an evaluating organization offering more broadbased evaluation services, the particular panel of design evaluation experts may vary from contract to contract. Thus, each contract to conduct an evaluation of a given complex system design may require recruitment and hiring of experts from major indicator domains not already

represented in the organization. The domain dominance scheme for a particular system should logically drive the composition of the evaluation panel.

Consider the example system classes represented in Figure 4.2. For system class (a), the unmanned system, an evaluation panel would be expected to be characterized by a minimal complement of humanware domain experts, e.g., behavioral and organizational psychologists, training specialists, and human factors engineers. A more practical complement would include more hardware and software performance engineers, embedded software system specialists, and operational domain experts familiar with operating characteristics of similar systems. It follows that the quality (i.e., amount, cumulative experience, technical reputation, etc.) of domain-specific, inter-domain, and system-level evaluators should be proportional to the degree to which their respective domain is prominent in the proposed system. Operational domain experts are required respective of the range of operational scenarios and operating environments in which the designed system will be employed.

4.3.2 Identifying Expertise Variance among Evaluators

For a very large complex system design, an evaluating organization may assign several evaluators to a single indicator or indicator class. Among the expert evaluators there likely exists a variance of evaluative expertise. This is a consequence of variances in lifetime experiences and human judgment. Consider a field of evaluators with varying years of experience as an expert practitioner in a particular domain. The judgments of those with the most expertise are likely to be more precise initially, and be applied with greater emphasis than the judgment of those with less experience. For less experienced evaluators, the emphasis of individual indicator assessments may need to be less than those in the first category. Our methodology provides procedures for assessing and responsibly accounting for such variances and applying them for the benefit of the evaluation.

The general procedure is to make a numerical assessment of the relative expertise of each evaluator respective of a single indicator (highly localized), an indicator sibling group, or an entire indicator domain (globalized). We require a combination of objective and subjective factors to make the overall multifaceted assessment. Based on the assessment, evaluators are ranked and *emphasis coefficients* (weights)[1] are assigned for each evaluator with respect to the indicator basis on which they are being compared (e.g., local, global, etc.). The resulting coefficient reflects the degree to which his or her judgments are incorporated into the evaluation. Such an emphasis is relative to those of peer evaluators who possess greater or lesser degrees of expertise in a particular domain. In practice the coefficients are used to yield a single score for each indicator by obtaining a weighted mean of the scores from all evaluators.

Techniques for assessing degrees of emphasis among a group of indicator category evaluators fall under the domain of multi-attribute decision making, priority theory, and others. The Analytic Hierarchy Process is useful for obtaining global weights which may serve as defaults for each evaluator. At a lower level, an interesting technique is provided by Klir and Yuan [95], based on evaluator assessments on a sibling group of indicators. We do not prescribe a particular technique, but recommend and discuss the Analytic Hierarchy Process in Chapter 5. Additionally, Bhargava [95] enumerates numerous other decision techniques.

Regardless of the particular method of arriving at evaluator emphasis coefficients, evaluating organization management personnel should participate heavily in this process, both by determining the evaluation criteria, and assigning values. Example expertise assessment factors include:

- *experience as a practitioner* in the design/development of systems in the domain
- *experience as an evaluator* of subsystems in the particular domain
- *evaluator's personal risk* or stake in the evaluation
- *education and training* in the technical domain

[1] While the concept of emphasis coefficient and weight are mathematically equivalent, we use the former term since is logically conveys a more correct sentiment of the manner in which the values are applied.

- *previous performance* in design evaluation (thoroughness, reliability, timeliness, consistency)

Finally, we do not consider that a lower emphasis applied to the assessments made by the less experienced evaluators is intended punitively or in a personally denigrating manner. Realistically, varying degrees of expertise are to be expected on any large project. Attaching emphasis coefficients to individual evaluators based on technical merit is a useful mathematical device for making the best use of expertise variances when dealing with professionals. Any other use of this device could likely prove detrimental to the entire evaluation effort.

4.3.3 Applying Evaluator Emphasis Coefficients

The ranking coefficients obtained in the previous step are not required through the scoring phase of the evaluation. In fact, knowledge of the coefficients may unintentionally affect the actual raw scores applied to an indicator by an evaluator. This is because an evaluator who is aware of his or her coefficient may apply an implicit fudge factor, to be taken into account at the point of scoring. Instead, the factors may be more appropriately computed *after* the indicator scoring phase is complete, to allow unbiased observation of each evaluator's expertise in action. Observation may then allow real-time assessment of factors such as *thoroughness*, *timeliness*, *confidence*, etc.

Whenever they are actually determined, the coefficients are used when several expert evaluators assign a value to the same indicator (or larger indicator basis). An aggregate score for the indicator is obtained by a weighted arithmetic mean of the scores from each evaluator for each indicator. The general formula is:

$$\sum_{i=1}^{n} W_i * S_i \qquad (4.1)$$

where:
n = the number of evaluators scoring on the indicator
i = the *i*th evaluator

w_i = the emphasis coefficient of evaluator i
s_i = the score assigned by evaluator i.

One final topic regarding the application of emphasis coefficients is applicable. Evaluator-specific information such as the score assigned to individual expertise factors, may be tracked in the KBESD knowledge base. Additionally, rules relating (1) the distribution of the factor scores among all evaluators, (2) factor scores for a single evaluator, and (3) weights to factor scores, may also be included in the KBESD knowledge base. These entries can be used to provide automated assistance in the assessment of the final emphasis coefficients. Details of the knowledge base and its application are provided in Chapter 7.

4.4. INDICATOR IDENTIFICATION

Design feasibility and requirements satisfiability indicators are identified and later assigned quality scores by expert evaluators, appropriate to their area and level of technical or operational expertise. Indicator identification should progress in a divide and conquer, stepwise refinement fashion, beginning with system-level indicators and refining these to greater levels of detail.

4.4.1 Indicator Subdomain Refinement and Definition

Systems engineers and problem domain experts (i.e., similar-system operational experts) begin the indicator decomposition process by establishing system-level indicators such as *system effectiveness (*e.g., *performance*, *dependability*, etc.). An example upper highest level operational domain indicator is *survivability*. These higher level indicators are then partitioned along humanware, hardware, etc., lines, yielding indicators like *software performance requirements satisfiability*, *humanware dependability requirements satisfiability*, etc. Senior evaluators in the respective technical specialties partition these indicators further into subindicators, such as *software timing requirements satisfiability*, *humanware reliability requirements satisfiability*, etc. Evaluators with expertise in more highly refined subdomains further decompose these

intermediate indicators until base-level indicators are established. As a general practice, the 7±2 rule of thumb should be observed when decomposing an indictor into lower levels. Computationally, three to seven indicators are easier to work with than higher numbers. Groups larger than this should be split, and smaller groups of indicators may need to be combined to make other tasks, e.g., applying the AHP, as efficient as possible.

Base-level indicators require no further refinement, and can be "directly" assigned a score using applicable methods from Chapter 6. Some indicators may reach base-level relatively soon in the refinement process. These indicators may be scored based on system-level operations heuristics, for which only expert judgment from operational veterans can provide a credible subjective valuation. A discussion of sources for assigning indicator scores is provided in next section.

Notice the explicit references to "requirements satisfiability" in the names of upper level indicators. We include this nomenclature to encourage a traceability to requirements. In many cases, indicators very deeply nested in the hierarchy may contain the "requirements traceability" label. Generally at the lowest hierarchy levels, where metrics, simulation, or expert knowledge are applied for indicator scoring, there may be no directly traceable link to requirements. However, the lowest level indicators are scored ultimately to provide assessments for the degree of satisfaction of the requirements traceable indicators in the upper levels.

4.4.2 Indicator Score Source Determination

Chapter 3 introduces four sources of indicator scores, i.e., metric-based, simulation-based, technical domain expert knowledge-based, and operational domain expert knowledge-based. At the point of identifying a base-level indicator, a determination can be made regarding the most appropriate source for providing an evaluation of the indicator. During the indicator refinement process, the score source may be revealed as self-evident. That is, some indicators may be directly computable from metrics, as we discuss in Chapter 6. For situations where the appropriate source is

not obvious, experts must make the call based on experience, available resources, manpower, technology, etc.

4.4.3 Indicator Score Range Assignment

As the indicator refinement subphase progresses, intermediate indicators and base-level indicators are defined. We require the definition and assignment of ranges bounding (possibly overlapping) regions of scores within a uniform interval [0,100]. The regions are used both numerically and logically in the evaluation. In this section we discuss the process of assigning scoring regions and mapping raw or subjective indicator scores to a uniform interval.

4.4.3.1 Defining Score Range Regions

Two fundamental regions are required for each indicator: a *rejection* region and an *acceptable* region. We make a case for additional desirability regions later, but we proceed to motivate the two fundamental regions first.

A *reject region* generally bounds the range of scores for which the indicator is not acceptable at all. This does not imply that the entire design (or subdomain or major domain) also fails. Instead, we can say that the particular indicator, if only for comparison or general interest purposes, is not deemed satisfactory with respect to the system design objectives.

A special case is defined to account for the more serious reject scenario. In the case that a failing score for a single indicator invalidates a sibling group, a subdomain, a major domain, or the entire design, the reject region is also the *failure* region. To make the distinction, systems-level evaluators determine the worst case damage a rejection score could cause to the evaluation (or more localized indicator grouping). In most cases, especially below the upper few levels in the hierarchy, no single indicator's score should invalidate the entire design. However, we allow for earmarking an indicator with its highest potential detrimental impact to the design. Practically, experts from the

evaluating organization populate the knowledge base of the KBESD with facts and rules related to both the determination and interpretation of failure regions (see Chapter 7).

The second major scoring region is the *acceptable region*. This is clearly the region of scores which are at least minimally acceptable with respect to the design objectives. To allow for more detailed analysis of the degree of acceptability of individual and aggregate scores, we advocate partitioning the acceptable region into three regions: a *marginal* and *excellent* region on the lower and upper tails, respectively.

Figure 4.3 depicts a few possible scoring region scenarios. We expect many range width variations exist in a large hierarchy. Here we describe several more common combinations. Figure 4.3, example (A) depicts a likely typical score range distribution. Example (B) shows what is likely a very critical indicator, with a large failure region, a modest accept region, and no excellent region. Example (C) depicts an indicator which has a broad tolerance for low scores, and which is likely not critically weighted in the evaluation. Example (D) depicts an indicator with an implied essentially linear acceptability function. More on the topic of acceptability functions is provided in the next section and in Chapter 6. As a final note regarding the uniform score range of [0,100], it need not be the case that the most desirable scores are on the high end of the range. We allow for the reverse situation where scores closer to 0 are more desirable. In those cases, either the ranges are reversed or the scores are re-

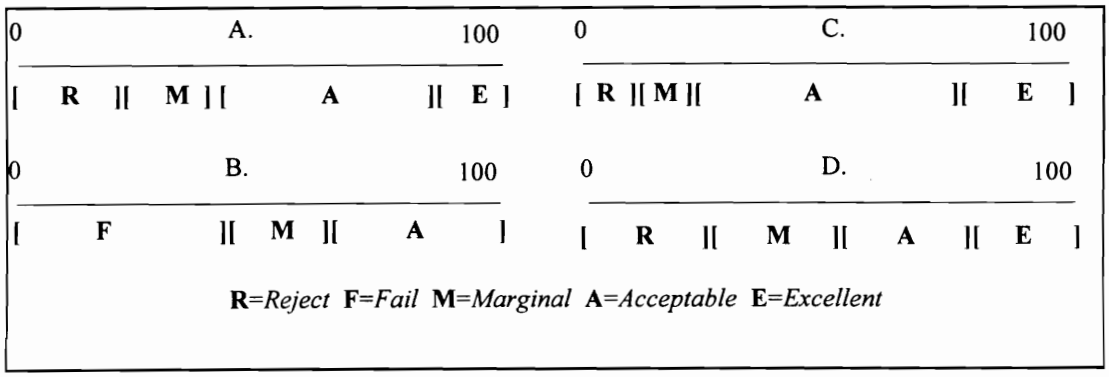


Figure 4.3. Example Score Desirability Regions.

interpreted as [100 - <score>]. For example, a high score for "throughput" is desirable, but a desirable score for "response time" would be low. Computationally, the "response time" score would be converted to its additive inverse with respect to 100.

4.4.3.2 Determining Score Range Regions

Notice in Figure 4.3 that all score ranges are in the uniform range [0,100]. This is an artificial constraint, which greatly eases application of the methodology. In this section we briefly address the mapping of scores to this standard range, then direct the reader to Chapter 6 for more detail and examples.

Base-level scores derived from metrics or simulation are initially assigned ranges based on the actual units with which the indicator is measured. For example, a particular indicator, *throughput* may be determined to have an actual range of [1200, 57600] for some unit, say *bits per second.* These scores, x, in such an interval, [a, b] could be mapped *linearly* to the range [0,100] with the boundary conditions: $f(a) = 0$, $f(b) = 100$, using Equation 4.2. Non-linear mapping functions may also be employed. Details regarding this are discussed in Chapter 6.

$$f(x) = \frac{x - a}{b - a} * 100 \tag{4.2}$$

Judgment-based scores can also be interpreted as being based on fuzzy intervals. In these cases, the interval [0,100] correlates to the membership interval [0,1], associated with fuzzy mathematics. Since Chapter 2 presents an overview of basic fuzzy mathematics concepts, it is sufficient to state at this point that the α-cut device, $\alpha \in (0,1]$ is employed to demarcate the boundary between the reject and acceptable regions. Additional α levels may be applied to demarcate the marginal, excellent, or additional regions. Figure 4.4 illustrates the concept. Application of fuzzy mathematics to the scoring process is described in more detail in Chapter 6. Score

range information is used in the knowledge-based portion of the design evaluation methodology as discussed in Chapter 7.

4.5. INDICATOR HIERARCHY CONSTRUCTION

The indicators identified via the decomposition approach described above should fall naturally into a hierarchical structure. Since the hierarchy is the critical data structure in the indicator-based evaluation methodology, we provide a discussion regarding both the construction and independent assessment of the indicator hierarchy.

4.5.1 Indicator Hierarchy Construction

In this section, we discuss the process of constructing a hierarchy of indicators which are employed in the design evaluation. We address two key parts: (1) accounting for indicator relationships and (2) inclusion of additional indicator-specific information in the KBESD.

4.5.1.1 Indicator Relationships Representation

Intuitively, construction of the basic hierarchy should be straightforward. As each intermediate indicator is refined into a collection of subindicators, a tree data structure may naturally be generated. However, due to the integrated nature of the complex system

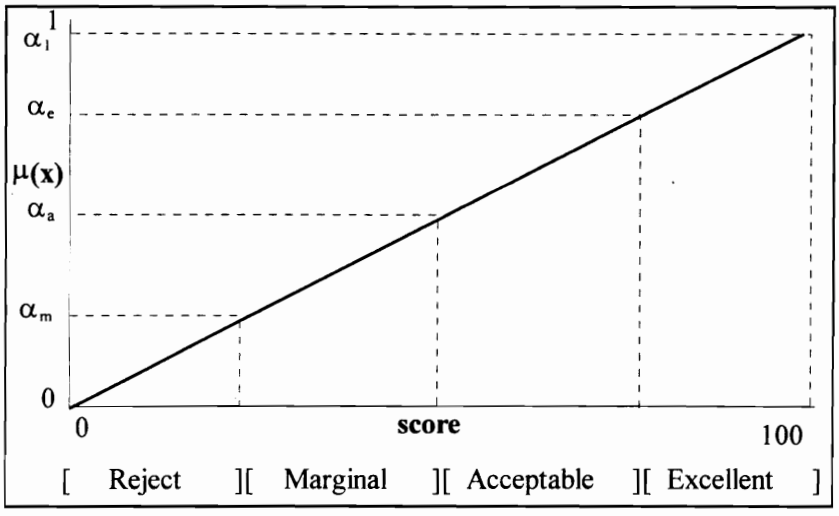


Figure 4.4. Example Score Ranges Imposed by Alpha Levels.

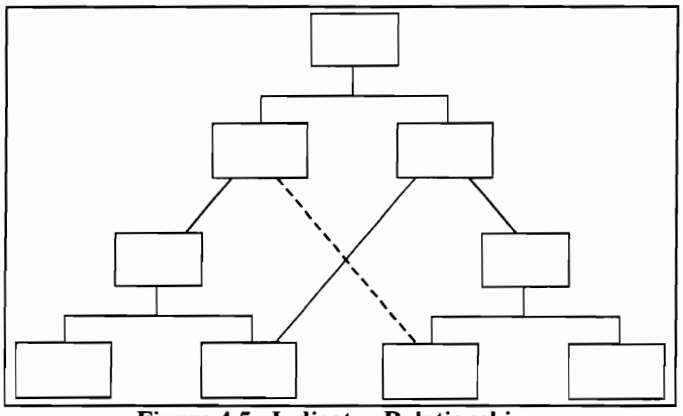


Figure 4.5. Indicator Relationships.

whose design is evaluated by indicators, there should be relationships defined between indicators which result in a directed acyclic graph. The most likely reason for departure from a tree structure is an indicator which is a "child" of two "parent" indicators.

There are two general types of indicator relationships. We term as *structural* a relationship between two indicators which is a consequence of the decomposition of one indicator into a set containing the other (i.e., parent and child indicators). Structural relationships are represented in Figure 4.5 as solid lines.

Non-structural relationships may be *proportional* (dashed line in Figure 4.5) reflecting a *quid pro quo* interplay between two intermediate indicators or entire indicator classes, in which changes in one would expect to generate corresponding changes in the other(s). For example, projected increases in software **reliability** would be expected to result in projected increases in software **cost** (Figure 4.6a), and

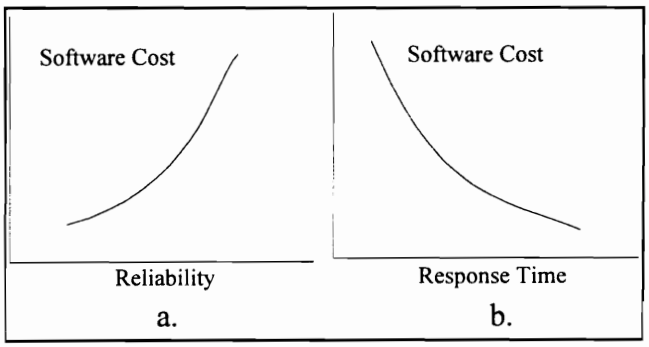


Figure 4.6. Proportional Indicator Relationships.

decreases in human-software interface **response time** would be expected to be reflected in increases in the **cost** of the interface (Figure 4.6b). Non-structural relationships may be linear or non-linear, involving integer, real, or even complex coefficients.

Structural relationships are directly represented in the KBESD as parent-child relationships, and these relationships are used in aggregating indicator scores up the hierarchy (Chapter 6). Non-structural relationships are reflected as facts and rules in the knowledge base, and are used for secondary evaluation of the design quality (Chapter 7).

4.5.1.2 Additional Indicator-Specific Information

The KBESD indicator inspector and hierarchy browser allow a great deal of information to be included for each indicator. We recommend the use or adaptation of the System Design Factor template of Ngyuen and Howell [94] (Figure 4.7). While not all fields are applicable or necessary for every indicator in the hierarchy, this template represents a comprehensive collection of indicator-related information fields. Condensing from [Ngyuen and Howell 94, pp. 4-1 - 4-2] , we provide brief descriptions of the fields from the figure.

> The **Name** item is a slot holder for the name of a specific design factor (e.g., # Reliability of Beam Former). The **Type** item is a slot holder for the classification of the factor (e.g., probability). The **Range** item is a slot holder for the minimum and maximum value or the cardinality of the factor (e.g., 0.0 to 1.0). The **Units** item is a slot holder for the unit of measurement of the factor (e.g., Units of Probability). The **Methods/Principle** item is a slot holder for the approaches or techniques that the designer/customer considered to be associated with this factor (e.g., Fault Tolerance, Highly Reliable Component). The **Rationale** item is a slot holder for the reason that this factor applies to a specific component/object (e.g., Life Critical Function). The **Relationship** item is the slot holder for the list of closely associated factors (e.g., Availability, Fault Tolerance). The **Relational Expression** field in this item provides the slot for the list of correlations between this factor and closely associated factors (e.g., Positive Correlation, Negative Correlation). The **Quantification** item contains the Type and Formula fields. The **Type** field in this item is the slot holder for either integer, float, double, short, or long. The **Formula** field in this item currently provides the slot for three mathematical expressions: (1) actually calculated (e.g., R(t) = 1 - F(t)), (2) required to be a specific value (e.g., 0.989), and (3) budgeted by designer or customer (e.g., 1.01 * 0.989). The **Consistency Rule** item consists of By-Aggregation, By-Type, By-Design Factors, By-View, and By-Component rules. For example, By-Aggregation field provides a slot that holds the rule for governing this factor consistently throughout the hierarchy (e.g., Use Rule X and Rule Y). The **Reference** item is a slot holder for the source of reference or the name of the author that formulated this factor. The **Definition** item is a slot holder for the clarity for this factor. The **Annotation** Item is the slot holder for commenting on relevant information or providing warnings related to this factor. Lastly, the **Next Template** item is not completely

defined at this time, but it is the slot holder for any detailed specification that may not require the customer's direction.

In addition, we have incorporated fields directly in the KBESD inspector which identify the life cycle phase in which the indicator is applicable, as well as the general indicator domain, e.g., "software" These fields are represented as check-boxes on the inspector panel for each indicator (Figure 3.4).

4.5.2 Indicator Hierarchy Quality Assessment

In Chapter 3 and in the immediate foregoing, we present indicator identification as the basis for the design evaluation and we establish the foundational role of the indicator hierarchy in the design evaluation methodology. To briefly recap the evaluation scenario, a design for a complex system is being evaluated by an independent evaluating

1	**Name:**	Reliability of Beam Former	
2	**Type:**	Probability	
3	**Range:**	0.0 to 1.0	
4	**Units:**	Units of Probability	
5	**Methods/Principle:**	Fault Tolerance, Highly Reliable Component	
6	**Rationale:**	Life Critical Function	
7	**Relationship:**	Availability, Fault Tolerance, ...	
	a. Relational Expression	Positive Correlation, Negative Correlation	
8	**Quantification**		
	a. Type		
	b. Formula	R(t) = 1 - F(t)	Actual
		0.989 entered	Required
		1.01 * Required	Budgeted
9	**Consistency Rule**		
	a. By aggregation	Use Rule X and Y; Rule X: The probability of the component in series is the product of its probabilities. Rule Y: The probability of the component in parallel use one of rating voting scheme.	
	b. By type		
	c. By design factor		
	d. By view		
	e. By component		
10	**Reference:**	Author(s) name	
11	**Definition:**	Text Book	
12	**Annotation:**	Comments	
13	**Next Template:**		

Figure 4.7. System Design Factor Fields Template.

organization, on behalf of a sponsoring organization. To provide an independent evaluation of the quality of the design, a comprehensive collection of indicators is identified and assembled by a group of experts. These indicators are the elements whose individual and collective quality scores are used to indicate the quality of the entire design, and critical aspects of the design process.

However, the design evaluation is not based on a *simple* collection of indicators, but on a *structured* collection. Indicators are structured into a hierarchy. We submit that the quality of the hierarchy (quality, quantity, degree of, and logic behind the interrelationships between indicators, among other indicators) is reflective of the quality of the process of indicator derivation and definition, i.e., the *application of expertise* to the problem. It follows that the quality of the evaluative process lends to the credibility of the entire design evaluation. Thus, the degree to which the indicators are expertly selected and structured is an indicator to the sponsoring organization of the credibility of the results the evaluation *can* yield (Figure 4.8). With the goal of ensuring high inherent credibility which produces great confidence in the ultimate results of the design evaluation, we impose a *meta-evaluation* of the indicator hierarchy at this point in the methodology.

After a moment's reflection on the potential consequences of requiring an independent evaluation of the indicator hierarchy, three issues may strike the reader as significant. (1) *Why* another independent assessment? (2) *Who* performs the hierarchy assessment? (3) *What* criteria are used to assess the indicator hierarchy? We address each of these issues below.

4.5.2.1 Motivation for Indicator Hierarchy Assessment.

One may argue that an independent evaluation (by a 3rd party) of the *design itself* is already a superfluous endeavor, practical only, if at all, for the most costly and mission critical complex systems (e.g., a space station, a nuclear-powered aircraft carrier, a multi-acre chemical processing complex, etc.). Practically, the designing organization has already performed several design reviews, and the evaluating organization has also

undertaken to evaluate the design. How, then, can we justify the requirement for essentially *fourth* party in the process? Our reasoning is derived from the critical nature of the mission and costly development and life cycle of the system.

The original three parties play what we view as a *strategic* role in the system design process. They satisfy large-scale, "global" objectives, i.e., bringing an efficacious system into being in a traceable, accountable, feasible manner. That is, their contributions to the system design cycle (requirements, design, and design evaluation) are fundamental to the entire process. In our view of complex system design evaluation, the absence of any of these three contributions should bring into question any further progress on the entire process. In contrast, the mission we define as evaluation of the evaluation hierarchy can be viewed as *tactical*. It satisfies a specific "local" objective within the design evaluation process, in much the same way as intermediate design reviews function within the larger design process.

What is that local objective? In short, it is to answer this question: "*if the design is evaluated according to this structured collection of indicators, will it give a sufficient measure of the goodness or badness of the design*?" An affirmative answer gives assurance, both to the evaluating organization and the sponsoring organization, that the evaluation problem has been well defined [Balci and Nance 85]. In that sense, it is also the assurance that the correct evaluation problem has been addressed by the evaluating

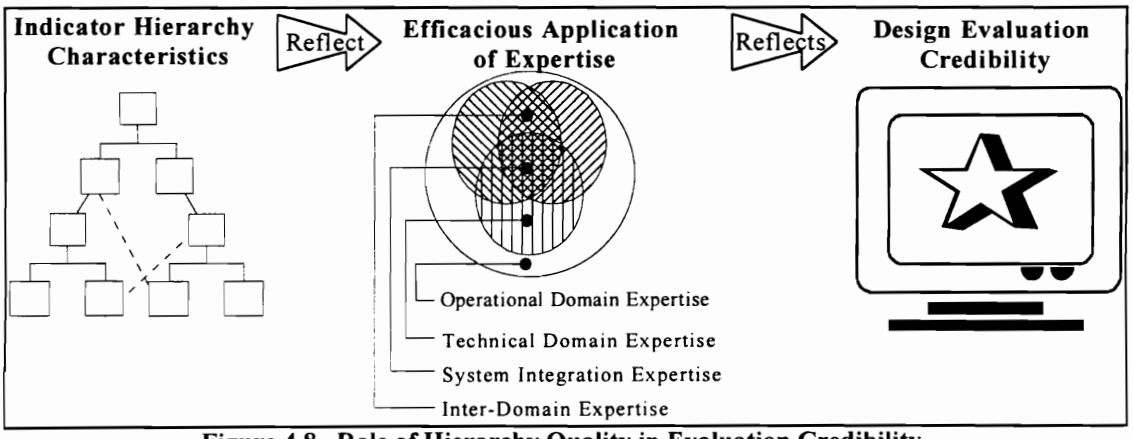


Figure 4.8. Role of Hierarchy Quality in Evaluation Credibility.

organization, and that their address of the problem is thorough and demonstrates technical competence. This is analogous to the service provided by the evaluating organization to the sponsor regarding the design itself, i.e., that the correct problem is addressed by the design, and that the design correctly addresses the problem.

We recommend a parsimonious approach to allocating cost and effort resources to hierarchy assessment. While the task is an essential (sub)component of the overall evaluation, it can be done with considerably less effort than the engulfing design and design evaluation phases. Figure 4.9 depicts a recommendation for the relative allocations of resources among the design subphases.

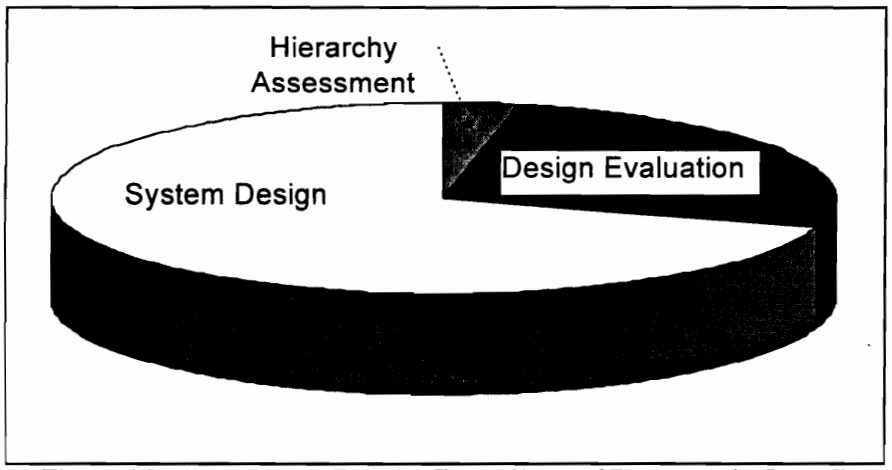


Figure 4.9. Approximate Relative Expenditure of Resources in Overall Design Phase.

4.5.2.2 Identification of a Panel for Indicator Hierarchy Assessment

We approach the identification of an independent expert panel to assess the evaluation hierarchy from the perspective of the sponsoring organization. We do so because it is this organization which assumes fiduciary responsibility for the corporation, customers, or taxpayers who ultimately supply the capital to conduct the project. And, it is the sponsoring organization which stands to derive the most benefit from a positive assessment of the indicator hierarchy, which we have suggested, leads to a quality operational product. Consequently, we submit that a "fourth party"

organization is selected for the purpose of hierarchy assessment at the direction of the sponsor.

The hierarchy assessment panel must consist of experts in the area of design or design evaluation for complex systems. As with the design evaluation organization, expertise from individual and multi-way intersections of technical and operational domains is required. We submit however that the degree of individual technical expertise should be nearly eclipsed by systems- and operational domain expertise. This is primarily due to the *aggregate* nature of the hierarchy assessment task, which depends more on the expertise of those with an overall system's perspective, and not largely from detailed technical knowledge applicable within specific domains.

Practically speaking, there are several reasonable candidate organizations which could provide a credible assessment of the fitness of the indicator hierarchy. Some suggestions include an organization who bid exclusively for the job, an organization within the sponsoring organization, another designing organization, or another design evaluation organization. On an individual organization basis, serious consideration must be given for the potential for "corporate grudge." Since it is not the purpose of this document to *prescribe* a source for assessment of the indicator hierarchy we stop short of making any more specific requirements beyond those of independence and expertise.

4.5.2.3 Method for Indicator Hierarchy Assessment

Overall hierarchy quality is revealed in both the content and the structure of the hierarchy. That is, the quality of the design evaluation hierarchy is assessed by examination of not only the actual indicators, which are the staple contents of the hierarchy, but also by the way they are organized in the hierarchy. Recall from Figure 4.8 that the quality of the hierarchy is a reflection of both the understanding of the design problem and application of expertise that went into the evaluation. Thus, to make assessments of this nature regarding an indicator hierarchy, naturally a collection of appropriate indicators is required.

We have found in the literature only sparse references to methods for hierarchy assessment as we propose here. In particular, regarding the assessment of taxonomies (pure trees), Fleishman and Quaintance [84], well known for taxonomy-based evaluations of the human aspect of system performance, offer qualities of external validity such as logic, parsimony, reliability, and exhaustivity. We propose the following, which overlap and extend those of Fleishman and Quaintance.

- Breadth of coverage
- Depth of coverage
- Representativeness of indicator interrelationships
- Validity of individual indicators
- Sufficiency of indicator decomposition
- Life cycle representation
- Traceability to requirements

These indicators may appear broad, and do so intentionally. We assert that these indicators should be applicable for any design evaluation hierarchy, since the contents are about a *hierarchy*, not about a *design*. Consequently, although the *contents* of any given hierarchy are evaluated relative to a particular operational domain, the *qualities* of the hierarchy can be subject to an operationally independent set of criteria.

We have developed a structured questionnaire to be used to assess the potential of an indicator hierarchy for yielding a credible evaluation of a complex system design. The questionnaire is designed to elicit feedback from the experts in the hierarchy assessment group. The questionnaire is presented in Appendix B.

4.6. A GENERIC HIERARCHY OF DESIGN EVALUATION INDICATORS

We have conducted an extensive literature search for indicators from individual domains as well as from multi-way intersecting domains. The major fields surveyed included hardware (both structural and computer), software, humanware, system producibility, and system cost. Based on the indicators extracted from the literature, we have constructed a *generic* hierarchy of indicators.

4.6.1 Purpose of the Generic Hierarchy

This generic hierarchy is provided to serve as a template for the evaluation of essentially any complex system as we have defined in Chapter 1. It is not our intention that this hierarchy is suited for direct application to any given design evaluation. Rather, any design evaluation organization should be able to use this hierarchy as a starting point, or template, tailoring it to the particular system design under evaluation. Greatest room for individual tailoring remains in the lowest levels of the hierarchy. There, particular techniques, metrics and subjective expert knowledge are the most organization dependent.

4.6.2 Description of the Generic Hierarchy

We now provide a description of the generic hierarchy sufficient to familiarize the reader with its structure and breadth of coverage. The complete hierarchy is provided in the KBESD software tool. In Appendix A we present the complete listing of the hierarchy, giving indicator names only.

Figure 4.10 depicts a typical KBESD hierarchy representation. The hierarchy *browser* is shown in the upper portion. The lower portion is the associated indicator *inspector*, for the rightmost highlighted indicator. The top level of the hierarchy (leftmost column in the browser) contains the soft concept "Complex System Design Quality." Since the quality of a design of a complex system is dependent on numerous factors, we have decomposed this indicator into a second level of indicators. These include System Cost Requirements Satisfiability, System Effectiveness Requirements Satisfiability, System Life Cycle Applicability, and System Physical Requirements Satisfiability. We have adopted the "Requirements Satisfiability" rubric for most upper level indicators since the quality of the design is to so large an extent dependent on the degree to which the design satisfies the sponsoring organization's requirements, stated and implied. We include "implied" requirements, because one of the adjunct functions of the design evaluation is to assure the sponsoring organization that the design satisfies the stated

requirements, but that *all* the right requirements have been satisfied, whether explicitly stated in the requirements specification or not. This prevents what Balci and Nance [85] refer to as a *Type III error*, the error of solving the wrong problem. In this case, the wrong problem is a design based on incompletely specified requirements. The expert evaluation panel possesses the knowledge to discern the degree to which such a Type III error has been avoided by the designing and sponsoring organizations.

By including so direct a tie to requirements, we build in and enforce a requirements traceability function for the methodology. In our hierarchy assessment questionnaire given in Appendix B, we explicitly request feedback regarding the degree to which an indicator hierarchy maps the original requirements to indicators.

We now present a brief description of the upper level indicators from the generic hierarchy. The hierarchy is extensive, containing over 400 indicators. Even adequate coverage of more than the uppermost levels is not plausible within the bounds of this document.

Cost Requirements Satisfiability and the lower level indicators of its subhierarchy come largely from the work of Blanchard and Fabrycky [90] who define many system life cycle cost indicators. We acknowledge the determination of cost acceptability is as much an accounting function as it is an engineering function. However, it is obvious that a design for a system, functionally acceptable but fiscally infeasible over its life cycle, is a design which will never be executed. To determine if the system cost requirements can be satisfied, a collection of subindicators for the cost requirements satisfiability within several subcategories is needed. These subindicators are themselves further decomposed to four levels and in some cases more.

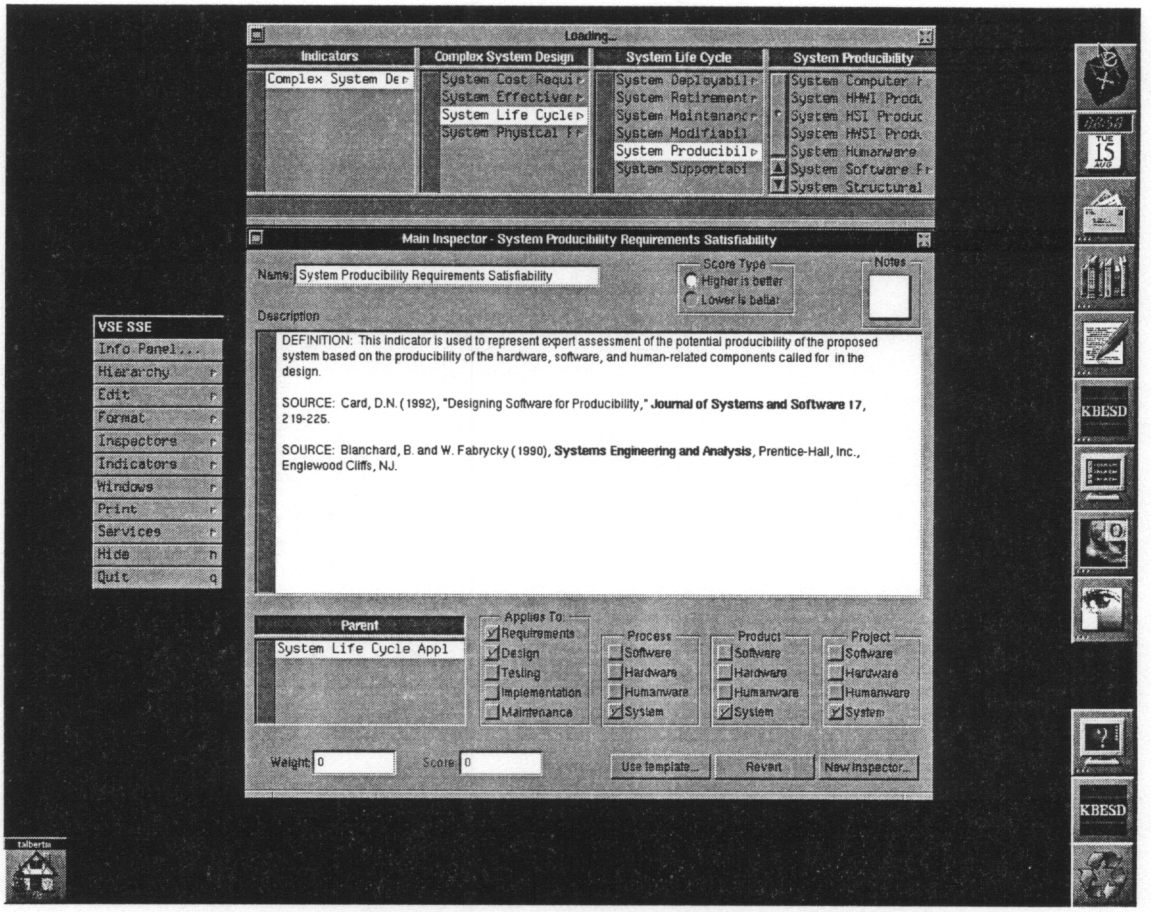


Figure 4.10. Upper Levels of the Generic Indicator Hierarchy.

The application of this subhierarchy of indicators is not to arrive at an aggregate dollar amount which is simply compared to a budget "bottom line" figure. The purpose of including cost requirements acceptability in this hierarchy is to make provision for prioritizing cost areas via the weighting mechanism discussed in Chapter 5. In this way, the KBESD can be used to facilitate cost trade-off considerations, in the event that some indicators are assessed to fall in their respective "rejection region." From the perspective of design evaluation, as opposed to budget creation, the contribution made by this indicator subhierarchy is a depiction of degree to which the subsystems and components specified in the design do, or can be made to, satisfy the budgeted requirements.

Modern systems engineering methods focus on a complete life cycle approach to systems design and analysis. *Life Cycle Applicability* is the aggregate indicator for the expert assessment of the degree to which the design accommodates all life cycle phases of the proposed system. Inclusion of this indicator and its subhierarchy is in accordance with the design evaluation criteria of Verma [91].

Determination of a value for this indicator relies on the values of a subhierarchy of lower level subindicators. For *Life Cycle Applicability* (highlighted in the second column in Figure 4.10), a first line decomposition produces indicators for the major phases of the system life cycle. The subindicators address Producibility, Deployability, Maintainability, Modifiability, Supportability, and Retirement. *Producibility Requirements Satisfiability* is selected in the third column of Figure 4.10. The maintainability and supportability indicators are included to re-enforce the modern trend toward concurrent engineering, which includes design for maintenance, production, and operational support concurrent with the design of the system itself. These subindicators represent the aggregate expert assessment of the degree to which the design respectively addresses their particular phase. These subindicators are individually further decomposed. The next level of refinement yields, e.g., subindicators for *System Producibility Requirements Satisfiability* as shown in the rightmost column of the browser of Figure 4.10.

The KBESD also makes provision to identify the applicable life cycle phases for every indicator in the hierarchy. The inspector for each indicator contains check boxes which are used to indicate at which phase in the system life cycle an indicator is applicable, as shown in the lower half of Figure 4.10. The KBESD makes use of these fields in a knowledge-base of facts, rules, and assertions relating an indicator's applicable life cycle phases to other attributes such as its weight, score, score source, and others.

We now delve a little deeper into the hierarchy, and describe one of the indicator subhierarchies further down the Life Cycle Applicability branch. *Software Producibility Requirements Satisfiability* and its lower level indicators address the evaluation of a design for a complex software system with respect to the feasibility with which it can be (correctly) produced. The upper levels of this subhierarchy come largely from the examination and extension of the work of David Card [92]. This indicator domain serves a role similar to the cost domain. That is, no matter how *functionally* complete a design for a system is, if implementing the design requires human, hardware, and software resources which cannot be made available to meet the schedule of the operational need, *at any cost*, then the system cannot meet the sponsoring organization's needs.

This indicator also addresses the potential for correctly producing the desired system. Lower level indicators like *design structure complexity* and its subindicators give a measure of the degree to which a *correct* software system can be effected through implementation of the design. At this depth, commonly known and applied metrics (e.g., Henry and Kafura's Information Flow metric [Henry and Kafura 84]) may be computed to yield leaf-level scores for this branch of the hierarchy.

We note also at this depth in the hierarchy we have encountered indicators which are applicable to more than one parent. In particular, *correctness* and related indicators are also appropriate for evaluation of some of the *performance* and *dependability* requirements satisfiability indicators which are themselves separate subhierarchies. Our

point is to introduce at this point the networked nature of the hierarchy due to multi-way influences among indicators. Many instances of this are found in the hierarchy.

Returning to the topmost levels of the hierarchy, we note *System Effectiveness Requirements Satisfiability* is decomposed into two main branches, relating to indicators for system performance and system dependability. *System Performance Requirements Satisfiability* and lower level indicators in its subhierarchy come from a large number of sources in the literature. These indicators are used by the evaluating organization to: (i) measure known entities, (ii) predict unknown quantities by simulating system behavior based on known quantities, and (iii) apply subjective expert knowledge based on the state of the art of available technology and experience in the operation of similar or related systems.

The reader can observe from the hierarchy in Appendix A large number of subindicators from a wide range of disciplines which populate the system performance subhierarchy. Indicators cover disciplines from human performance (both physical and mental), software performance, computer hardware performance. These design quality subindicators cover the degree to which design accounts for such diverse topics as task complexity, hardware I/O response times, human operator cognitive ability variations, communications signal to noise ratio tolerances, and many others. We include such detailed indicators, and recommend the inclusion of many others, tailored to the particular system design. This is because a design for a complex system comprised of interacting elements from the humanware, hardware, and software domains needs to consider system performance with respect to variances in performance of these lower level indicators.

System Dependability Requirements Satisfiability covers a wide range of dependability issues. It primarily relates to the aggregate expert assessment of the degree to which the system design accounts for the dependability requirements for the proposed operational system. For the first level of decomposition, and a few additional levels in some cases, we have based our dependability decomposition on the work of Laprie [92]. This work has become a *de facto* standard for terminology in the dependability literature.

We have incorporated Laprie's Reliability, Availability, Security and Safety attributes of dependability with our set of major domains and their multi-way interfaces, i.e., Software Security Requirements Satisfiability, Humanware Availability Requirements Satisfiability, etc. Unfortunately, references exploring these attributes as distinct indicators are rare. The exceptional case is that of Reliability, which has better representation both in the literature and in our hierarchy.

Issues of reliability intersect all three major domains of hardware, humanware, and software, as well as their intersections. A great amount of literature is available particularly for the humanware, human-hardware, and human-software interfaces. These, too, are proportionately represented in the hierarchy. In recent years, more has been published regarding software reliability. Almost without exception, these measurements are based on Markovian models and Petri nets, both deterministic and nondeterministic. Clearly score sources for this category of indicator would rely heavily on system, software, hardware, and human behavioral models.

Widespread interest is found in the literature on the issue of *human reliability.* This is particularly important for complex systems such as nuclear power plant control rooms, military vessels or vehicles, space vehicles, or aircraft, where a human operator is the ultimate controller of the function of the system. Consequently, we provide coverage of considerable depth in this area in the hierarchy.

Returning finally to the upper regions of the generic hierarchy, we now address *Physical Requirements Satisfiability.* This indicator has its basis in the work of Ngyuen and Howell [94] with extensions from varied sources in the literature. This subhierarchy of indicators may be based largely on known quantities (weights, sizes, durability) which are fixed according to the allocation of system functionality to subsystems and components in the design. Like the cost indicator subhierarchy, the indicators in this subhierarchy are not intended for use as an aggregate value for system weight, size, etc. Instead, the intent is to provide an aggregate measure of the degree to which the design specifies components which would allow the implemented system to remain within

specified requirements. Again, as with cost indicators, the KBESD facilitates trade-off considerations based on indicator weights.

There are two overlapping subsets of physical requirements indicators. Weight and volume and survivability-related indicators are particularly applicable for designs of vehicles, mobile, or portable systems. A second major subset of these indicators are applicable for technologically advanced industry applications such as robotics controlled manufacturing, intelligent-software controlled industrial processing or manufacturing, or nuclear power plant control rooms. Indicators applicable to this class of system relate more to floorspace, capacity, and energy requirements, with a distinct de-emphasis on mass- and mobility-related indicators.

4.7. SUMMARY

In this chapter we describe the initial phases of the design evaluation methodology. Determination of dominance of the system design by one or more indicator domains is an initial step. Its results are represented throughout the evaluation, even bearing on evaluator selection and expertise allocation. Indicator domain refinement, indicator definition, and the determination and assignment of indicator-related information are the next major focus. Finally, the indicator hierarchy construction process is detailed, accounting for the various types of indicator relationships. As a checkup, the indicator hierarchy itself is to be independently assessed for representativeness and potential to yield credible evaluative results.

We also provide and describe a generic hierarchy of complex system design evaluation indicators. The complete listing of the current hierarchy is provided in Appendix A. The software version of the entire hierarchy can be made available for use on the KBESD.

5. DETERMINING INDICATOR WEIGHTS

5.1. INTRODUCTION

Chapter 4 details several procedures culminating in a hierarchy of design quality indicators. Not all indicators contribute equally to the assessment of the design, however. This hierarchy's nested structure partially accounts for this by virtue of the vertical decomposition of indicators. Unilaterally, among siblings, indicator score criticalities with respect to the aggregate parent score may also vary. In this chapter, we provide a procedure to determine relative weights for indicators within a sibling group. Our chosen method for determining these weights is the Analytic Hierarchy Process (AHP), proposed by T.L. Saaty [80, 94].

The chapter contents are as follows: Section 5.2 provides a detailed explanation of each step in the process. Section 5.3 provides an example of the application of the AHP to the determination of weights among sibling indicators. Section 5.4 overviews other techniques for relative weight determination. Section 5.5 concludes the chapter with a summarizing case for our choice of the AHP as our primary method.

5.2. THE ANALYTIC HIERARCHY PROCESS

The steps of the AHP, with explanations from the literature are given as follows:

Step 1. *Define and validate the goal or decision to be analyzed*: If a decision is to be made, then it must be a valid one and its scope must be bound to a workable size.

Step 2. *Choose indicator set*: This is perhaps the most difficult of all the steps. In this step, the decision makers must identify a set of indicators which represent all the aspects of the decision problem which must be weighted and measured. This step is critical because it reflects the decision makers' understanding of the elements required

to make a valid decision. Adjustments to the collection of indicators may need to be made during the next phase.

Step 3. *Construct the decision hierarchy*: This phase is nearly as critical as the previous one. The resulting hierarchy of decision elements is a direct reflection of the decision makers' understanding of causal and cooperational dependencies among decision elements.

Step 4. *Perform pairwise comparisons*: For each set of decision elements which are "children" under the same "parent" node in the hierarchy, relative comparisons are made between pairs of elements. The ratio scale typically used for comparison is shown in Figure 5.1. Other scales, including real numbers and a wider range are discussed by J. Saaty [93]. Discussions of the theory of the ratio scale are provided by Harker and Vargas [87], T. Saaty [93] and Forman [87,93].

The pairwise comparisons are recorded in an n-by-n judgment matrix, where n is the number of elements in the subset. Comparisons can also take other forms. Saaty [94] and Salo [93] describe comparisons in the form of interval judgments instead of a single ratio. Jensen and Hicks [93] discuss a technique which involves pairwise comparisons given ordinally only, without the use of a relative scale. Regardless, the maximum number of comparisons required for n elements is $(n)(n\text{-}1)/2$, since only the

DEGREE OF IMPORTANCE	DEFINITION	EXPLANATION
1	Equal importance	Two elements contribute equally to the property
3	Moderate importance	Experience and judgment slightly favor one element over another
5	Essential or strong importance	Experience and judgment strongly favor one element over another
7	Very strong importance	An element is strongly favored; its dominance is demonstrated in practice
9	Extreme Importance	The evidence favoring one element over another is of the highest possible order of affirmation
2,4,6,8	Intermediate values between two judgments	Compromise is needed between two adjacent judgments
Reciprocals		When element i compared to j is assigned one of the above numbers, then element j compared to i is assigned its reciprocal.

Figure 5.1. Ratio Scale for the Analytic Hierarchy Process.

upper triangle of the matrix needs to be completed. The lower triangle contains the inverses of their symmetric entries, and the diagonal, of necessity, contains all ones. This special matrix is called a square reciprocal matrix. It demonstrates some desirable mathematical properties which ease computations for consistency.

Relatively simple mathematics reveal the fast growing nature of the number of pairwise comparisons required. The number grows large especially for a hierarchy which contains a large number of indicators. If we let n be the average number of children per parent node and d be a depth of the hierarchy (root at $d=1$), the number of pairwise comparisons required is $(n^d-n)/2$. As depicted in Table 5.1, the number of children per parent increases downward and depth in the tree increases to the right. To obtain the values in the table, we assume a complete tree of n nodes per parent at each depth. For example, a complete hierarchy with a depth of five (5) and an average of five (5) child indicators per parent indicator would require at worst case 1560 pairwise comparisons.

Fortunately, the worst case need not always prevail. In many example hierarchies provided in [Saaty 80, 90a, 90b, 94] and elsewhere in the literature, it is likely that the hierarchy is not necessarily full, reducing the number of required comparisons. Regardless, when the number of comparisons to be made becomes larger than would appear manageable, alternative means for completing massive comparison matrices or matrices with missing judgments have been derived, as discussed below. Another instance in which judgments might be left undetermined includes the scenario in which a decision maker simply does not have enough information on which to make a comparison. Yet another occurs when e.g., disagreements or company politics make it inexpedient to come out with a firm comparison for two key decision elements. Saaty [90a, 90b, 94] addresses the specific issue of conflict resolution. Solutions also addressing the problem of excessive numbers of comparisons are discussed below.

Table 5.1. Number of Pairwise Comparisons for a Full Hierarchy.

Nodes/Depth	2	3	4	5	6	7
3	3	12	39	120	363	1092
4	6	30	126	510	2046	8190
5	10	60	310	**1560**	7810	39060
6	15	105	645	3885	23325	139965
7	21	168	1197	8400	58821	411768
8	28	252	2044	16380	131068	1048572
9	36	360	3276	29520	265716	2391480
10	45	495	4995	49995	499995	4999995

Harker [87a, 87b] proposes that as few as n comparisons suffice to generate all entries for the judgment matrix. However, the entries must comprise a *minimum spanning tree* for a graph representing the collection of elements plus one redundant entry. This is necessary to ensure at least one path exists between every pair of nodes (elements), whose entry in the matrix is represented at position a_{ij}. Based on this graph theoretic approach, Harker iteratively estimates values not yet assigned as the *geometric mean* of the intensities of all paths from element i to element j. Such a path consists of all nonzero matrix entries, a_{lm}, in the path $a_{im},\ldots,a_{lj}$. Wedley, *et al.* [93] suggest that significantly greater initial accuracy in choosing the first n-1 comparisons is achieved if the elements are first ordinally ranked, and the lowest ranked element used as a baseline for the initial n-1 comparisons.

Harker [87b] also proposes a method of determining when enough comparisons have been performed to achieve a stable set of element priorities. If w^k is the vector of relative weights computed after k comparisons, then a decision maker could stop after $k+1$ comparison if $|w^{k+1} - w^k|/w^k \leq \alpha$, where α is a prespecified threshold. Harker found that for $\alpha=0.05$, a significant number of comparisons could be eliminated. See Table 5.2.

Table 5.2. Reduction in Pairwise Comparisons.

n	n(n-1)/2	Reduced Comparisons Required	Percentage of Comparisons Required
6	15	10.90	72.67
7	21	12.24	58.29
8	28	13.56	48.43
9	36	14.36	39.89

Step 5. *Compute consistency of judgments*. The concept of consistency is simple, yet critical to application of the AHP. Consistency is related to transitivity. Consider the judgment matrix A, with entries a_{ij}. Transitivity implies that entry a_{ik} should be equal to entry a_{ij} times a_{jk}, $i<j<k$. If every such triad of judgments in A is transitive, then the matrix and the judgments are perfectly consistent. In the realm of human judgments, departures from perfect consistency are likely the rule rather than the exception. For example, it is quite common to like bananas more than apples, apples more than grapes, but prefer grapes to bananas. This is human nature, and the AHP was developed with the inconsistencies of human judgment in mind.

Consistency is computed from the maximum eigenvalue of the judgment matrix which solves Equation 5.1:

$$Aw=\lambda_{max}w \tag{5.1}$$

where A is the judgment matrix, w is the right eigenvector for A, and λ_{max} is the maximum eigenvalue for A.

A *consistency index* (CI) for an n-by-n matrix is computed as follows:

$$CI = (\lambda_{max}-n)/(n-1) \tag{5.2}$$

The CI is normalized by a *random index* of consistency, RI. The RI is computed as the average CI for a large number of n-by-n reciprocal matrices, filled with random entries from the ratio scale referenced earlier. Random indices for several values of n appear in Table 5.3. The desired *consistency ratio*, CR, is simply:

$$\text{Consistency Ratio} = \text{CI/RI} \tag{5.3}$$

Saaty [80, 94] has suggested that a CR of greater than 0.1 indicates an unsatisfactory consistency in judgments, which could lead to invalid decisions. See [Apostolou and Hassell 93] for evidence that suggests a confidence interval about 0.1 is

Table 5.3. Random Consistency Indices [Saaty 94].

n	1	2	3	4	5	6	7	8	9
CI	0.0	0.0	0.52	0.89	1.11	1.25	1.35	1.40	1.45

more appropriate.

The issue of consistency has been examined variously in the recent literature. Murphy [93] demonstrated an inherent bias in the nine-point ratio scale toward inconsistency, especially when several judgments are assigned values on the high end of the scale. She notes, however, that this has the effect of canceling a portion of judgment error due to inflated ratios.

Harker [87a, 87b] offers a technique for determining whether and which additional comparisons are required to achieve desired consistency. The consistency of an *n*-by-*n* reciprocal matrix improves as the largest right-side eigenvalue for the matrix decreases towards *n*. Consequently, a pending comparison would improve consistency if it reduced the (estimated) value of the eigenvalue. It follows that the next comparison to be performed should be the one which would reduce the eigenvalue the most. Harker makes use of the derivative of the eigenvalue for the matrix with respect to any given missing entry as a guideline. Dadkhah and Zahedi [93] offer a simpler approach. The comparison to perform next is the one in position a_{ij}, which yields the highest differential of the λ_{max}. Harker cautions that the resultant a_{ij} is at best a *guideline*, and that a ranking of the top few candidates would be most helpful to the decision maker.

Other consistency determination approaches have recently appeared in the literature. Wedley [93] extends the works of Harker [87a, 87b] and Forman [88] with multiple regression equations to both detect significant inconsistency and predict ultimate consistency of an incomplete square reciprocal matrix. He also provides guidelines for incorporating the prediction equations into an AHP software program. Salo [93] proposes a framework employing extended regions, similar to predictive right eigenvectors, for attaining early consistency gauges when pairwise judgments are missing or in the form of interval judgments. It derives bounds on element priorities as new comparisons are made, providing decision makers with early indications of element *ranking*. Jensen and Hicks [93] propose an additional coefficient of

consistency applicable to the special case when judgment matrices are simply ordinal, and not based on a ratio scale, as we mention in Step 4. The reader is referred to the source for details.

Step 6. *Make necessary adjustments for consistency.* If the CR is greater than 0.1, there are several techniques for making adjustments to achieve a desirable consistency. The simplest technique, suggested by T. Saaty [80,94] and J. Saaty [93] is to spot-check the pairwise comparisons, looking for obvious examples of gross intransitivity, such as with the fruit preferences mentioned above. More sophisticated methods exist, as described in Step 5.

In addition, DeTurck [87] suggests an iterative procedure for improving consistency in increments of approximately 10%. His approach is computationally intensive, but is warranted for key decision elements for which minimal inconsistency is critical. Blankmeyer [87] compared the eigenvector method of AHP to four techniques based on minimizing the Euclidean distance between observed consistency and perfect consistency. He found the eigenvector method was least likely to result in reversal of ranks as judgments were adjusted, but that it was the only technique examined for which priority determination was not symmetric with respect to the transpose of the judgment matrix. The point offered by the author was that the judgments which are more natural to make are those for the degree of being dominated (column entries) versus the degree of dominance (row entries). These judgments would be represented as the transpose of the AHP matrices as we show them.

Step 7. *Compute relative weights of elements.* This step will have already been completed in the process of making consistency determinations in prior steps. The right-hand eigenvector for the judgment matrix A contains the relative weights for the elements which have been compared. Saaty [94] offers as many as four methods for computing or approximating the eigenvector. The first involves a straightforward

normalization of matrix entries by the sum of the entries in the row. The second technique involves normalization by the maximum element in the column. The third technique involves the geometric mean of the matrix row elements, and the fourth technique is the exact solution from matrix theory. All these techniques are well known and are not repeated here. Software tools such as *Mathematica* or *Expert Choice* are used in practice to provide exact solutions.

Step 8. *Compute consistency of the hierarchy.* Assuming satisfactory consistency has been attained for the individual element subsets, a final consistency measure for the entire hierarchy could be computed. Saaty tersely describes the process as follows:

> The consistency of a hierarchy is obtained by first taking the sums of products of each consistency index CI with the composite priority of its criterion (parent). Then the ratio is formed of this number with the sums of the products of the random consistency index for that order matrix with the composite priority of its criterion [Saaty 94, pp. 126-127].

As with the individual matrix, the consistency ratio for the hierarchy should be less than 0.10 in order not to cause concern for needed improvements in the judgments. Saaty does not elaborate on the advantages of hierarchical consistency and provides only cursory mention of it in the several works where it is discussed [Saaty 80, 86, 90a, 90b, 94 and J. Saaty 93].

Step 9. *Adjust for consistency as needed.* Adjustments to hierarchical consistency have to be made at the level of the individual matrix [Saaty 93]. Generally, attempts must be made to reduce the inconsistency of the matrices which are either highest ranked, the most inconsistent, or both.

Step 10. *Compute global weights for all indicators.* The global weight for a leaf element is simply the product of its local weight with its parent's recursively computed global weight. Depending on the use of the weighted elements in a decision, only the local weights may be required, since the global weights only apply to the leaf nodes. We see in the applications portion of this proposal how the local weights are used in a design evaluation scheme.

5.3. APPLICATION OF AHP FOR INDICATOR WEIGHT DETERMINATION

To illustrate the application of the AHP, we use an example hierarchy, adapted from the illustrative hierarchy given in Chapter 3. This must be considered only a portion of a larger hierarchy, but it is large enough for our purposes. We begin with the subhierarchy shown in Figure 5.2. We proceed downward through the hierarchy, applying the AHP. Where necessary (e.g., for consistency adjustment, etc.) we apply corollary techniques as proposed in the sources from the literature.

5.3.1 Making Pairwise Comparisons among Sibling Indicators

The first three steps of the AHP procedure have already been accomplished as of this step, since the goal of "Acceptability of System Survivability" has been

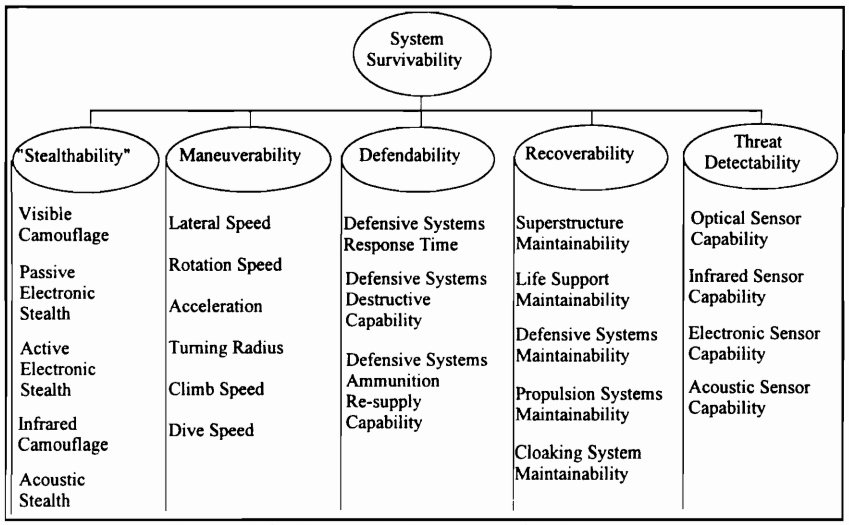


Figure 5.2. Example Indicator Subhierarchy.

established, and the decision indicators have been identified and assembled in a hierarchy. In the hierarchy of Figure 5.2, there are no multiple-parented indicators. If there were, the indicator would be judged for a weight with respect to each parent. Step 4 requires pairwise comparisons be performed between indicators in a sibling group. The highest level sibling group contains the indicators {"Stealthability", Maneuverability, Defendability, Recoverability, and Threat Detectability}. We now describe the process of making pairwise comparisons to determine relative weights of indicators. Additional methods are referenced in Section 5.4.

5.3.1.1 Preliminary Exercises to Promote Consistency

Before beginning the process, there are some preliminary exercises which have a positive impact on the numerical credibility and practical acceptability of the comparisons. To be successful in the effort, evaluators should first conduct a preemptive discussion of the indicators in a sibling group with the purpose of determining a rough *rank ordering*. The discussion should allow for ties. Second if possible, the discussion should result in an identification of best and/or worst candidates among the sibling indicators (ties allowed). In both cases, no discussions of *degree* of order or of goodness are required.

The rough rank ordering provides a basis for at least *preferential* transitivity in making the pairwise comparisons. It is an indicator of the degree to which comparisons based on evaluator preference (judgment) demonstrate general transitivity. For example, if the group thinks indicator A is more important than B, and B more important than C, then transitivity in preference would lead us to expect a preference of A over C. While this need not always be the case, it is an indicator that the resulting relative weights are based on logically sound reasoning.

The establishment of a best/worst indicator can provide a basis for comparisons which should lead to improved numerical consistency. This follows the recommendations of [Basak 93; Jensen and Hicks 93]. Both exercises should promote a consensus among evaluator preferences. This is important so disparities in judgment

are not played out on the numerical battlefield when judgments are averaged to obtain a single value. In this case, the final numerical assignment made more closely agrees with the values from and intentions of the evaluators.

5.3.1.2 Determining a Value for a Pairwise Comparison

From Section 5.2, Step 4, the actual process of making numerical pairwise comparisons is straightforward. The general process involves making entries into a judgment matrix, of size n by n, for n indicators in the sibling group. Entry (i,j) contains the value for the comparison of indictor i versus indicator j. The reciprocal of this value appears in position (j,i).

For two indicators in the same sibling group, an evaluator indicates the degree of preference (or importance) of one indicator over the other. In our example, judgments for top-level indicators are provided in Figure 5.3. Judgments and resulting weights and consistency ratios for their respective sibling groups appear in Figures 5.4-5.8. The figures are images from the AHP software tool *Expert Choice*. We now address the contents of the output.

For the pairwise judgments, entries in each row relate the relative importance of the row head to the importance of the column head. If the value is in parenthesis, the opposite is meant. Judgments are made with respect to the scale {1/9, 1/8, ..., 1/2, 1, 2, ..., 8, 9}. Discussions of the scale are provided in Figure 5.1. For example, in Figure 5.3, *Maneuverability* is rated moderately more important than *Stealthability* (3.0), instead of the other way around.

Computationally, the parenthetical entries are treated as reciprocal values in the judgment matrix. The parenthetical notation is simply a typographical convenience. Note in this example that *Threat Detectability* and *Recoverability* are judged to be equally important, as are *Defendability* and *Maneuverability*. Explanatory notes in the figure provide definitions for the indicator abbreviations and the comparison scale employed.

```
                          JUDGMENTS WITH RESPECT TO
                                    GOAL

            STEALTH    MANEUVER  DEFEND     RECOVER    THREATDT
 STEALTH                 ( 3.0)    ( 3.0)     ( 5.0)     ( 6.0)
 MANEUVER                            1.0      ( 2.0)     ( 2.0)
 DEFEND                                       ( 1.5)     ( 2.0)
 RECOVER                                                   1.0
 THREATDT

Matrix entry indicates that ROW element is ___
  1 EQUALLY  3 MODERATELY  5 STRONGLY  7 VERY STRONGLY  9 EXTREMELY
more IMPORTANT than COLUMN element unless enclosed in parenthesis.

              GOAL: Acceptability of System Survivability

DEFEND   --- Quality of overall system defendability capability
MANEUVER --- Quality of overall system maneuverability capability
RECOVER  --- Quality of capability to recover system if damaged
STEALTH  --- Quality of stealth capability of the system
THREATDT --- Quality of ability to detect external threats
```

Figure 5.3. Top-Level Pairwise Comparisons.

5.3.1.3 Combining Comparisons from Several Evaluators

We anticipate that in many cases, there will be several expert evaluators involved in making the pairwise comparisons. In these cases, how are the final values for each pairwise comparison determined? Saaty [94] suggests taking the geometric mean of the respective evaluator's values. The geometric mean has been demonstrated to be more capable of dealing with ratio or reciprocal values, which are common when applying the AHP [Aczél and Saaty 83]. Using the arithmetic mean for the values 3 and 1/4, the aggregate value would be (3+.25)/2 = 1.625. Using the geometric mean, the value would be SQRT(3*1/4) = .866. The geometric mean (Equation 5.4) yields the more conservative value.

We provide in Chapter 4 a means to logically and mathematically discriminate between expertise levels of evaluators, deriving *evaluator emphasis coefficients (EEC)*. An EEC is applied to each pairwise judgment of each evaluator, to ensure that the

$$\textit{weighted geometric mean} \equiv \prod_i c_i^{w_i} \tag{5.4}$$

comparisons take fullest accounting for variances in expertise. Thus, we incorporate this coefficient in computing a weighted geometric mean by the formula in Equation 5.4.

```
                        JUDGMENTS WITH RESPECT TO
                                STEALTH < GOAL

          VISUCAMO  PASSELEC  ACTVELEC  INFRCAMO  ACOUSTIC
 VISUCAMO           ( 6.0)     ( 4.0)     ( 2.0)     ( 1.5)
 PASSELEC                        1.5        3.0        4.0
 ACTVELEC                                   3.0        4.0
 INFRCAMO                                              1.0
 ACOUSTIC

Matrix entry indicates that ROW element is ___
  1 EQUALLY  3 MODERATELY  5 STRONGLY  7 VERY STRONGLY  9 EXTREMELY
more IMPORTANT than COLUMN element unless enclosed in parenthesis.


                          GOAL: System Survivability

ACOUSTIC --- Acoustic Stealth
ACTVELEC --- attainment of active electronic stealth (jamming)
INFRCAMO --- Infrared Camouflage Capability
PASSELEC --- Attainment of Passive Electronic Stealth (Undetectable)
STEALTH  --- Successful Application of Stealth Technologies
VISUCAMO --- Visual Camouflage Capability

                                  PRIORITIES
0.067
VISUCAMO
0.401
PASSELEC
0.317
ACTVELEC
0.117
INFRCAMO
0.098
ACOUSTIC

                       INCONSISTENCY RATIO = 0.008.
```

Figure 5.4. Comparisons, Weights and Consistency Ratio for Stealthability.

JUDGMENTS WITH RESPECT TO
MANEUVER < GOAL

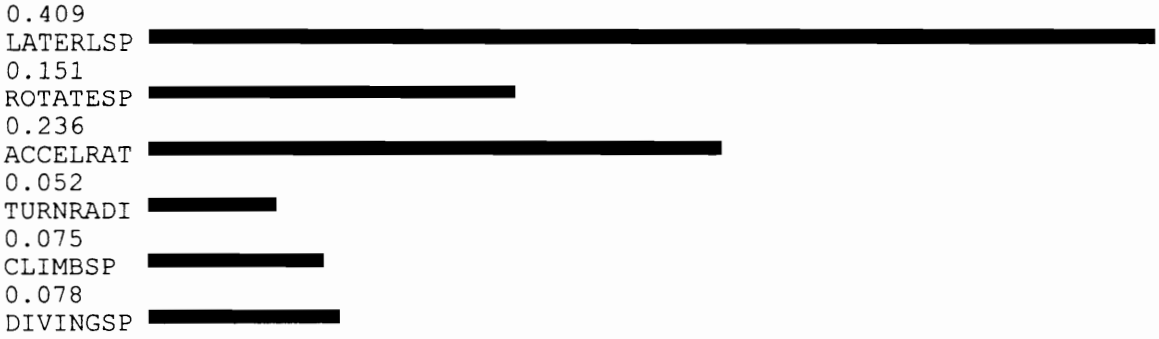
	LATERLSP	ROTATESP	ACCELRAT	TURNRADI	CLIMBSP	DIVINGSP
LATERLSP		2.0	4.0	6.0	6.0	6.0
ROTATESP			(5.0)	1.0	5.0	3.0
ACCELRAT				5.0	2.0	2.0
TURNRADI					(3.0)	(3.0)
CLIMBSP						1.0
DIVINGSP						

Matrix entry indicates that ROW element is ___
1 EQUALLY 3 MODERATELY 5 STRONGLY 7 VERY STRONGLY 9 EXTREMELY
more IMPORTANT than COLUMN element unless enclosed in parenthesis.

GOAL: System Survivability

ACCELRAT --- Acceleration Capability
CLIMBSP --- Climb Speed Capability
DIVINGSP --- Dive Speed Capability
LATERLSP --- Lateral Speed Capability
MANEUVER --- Successful Incorporation of Maneuverability Capability
ROTATESP --- Acceptability of Rotation Speed Capability
TURNRADI --- Vehicle Turning Radius

PRIORITIES



INCONSISTENCY RATIO = 0.166.

Figure 5.5. Comparisons, Weights and Consistency Ratio for Maneuverability.

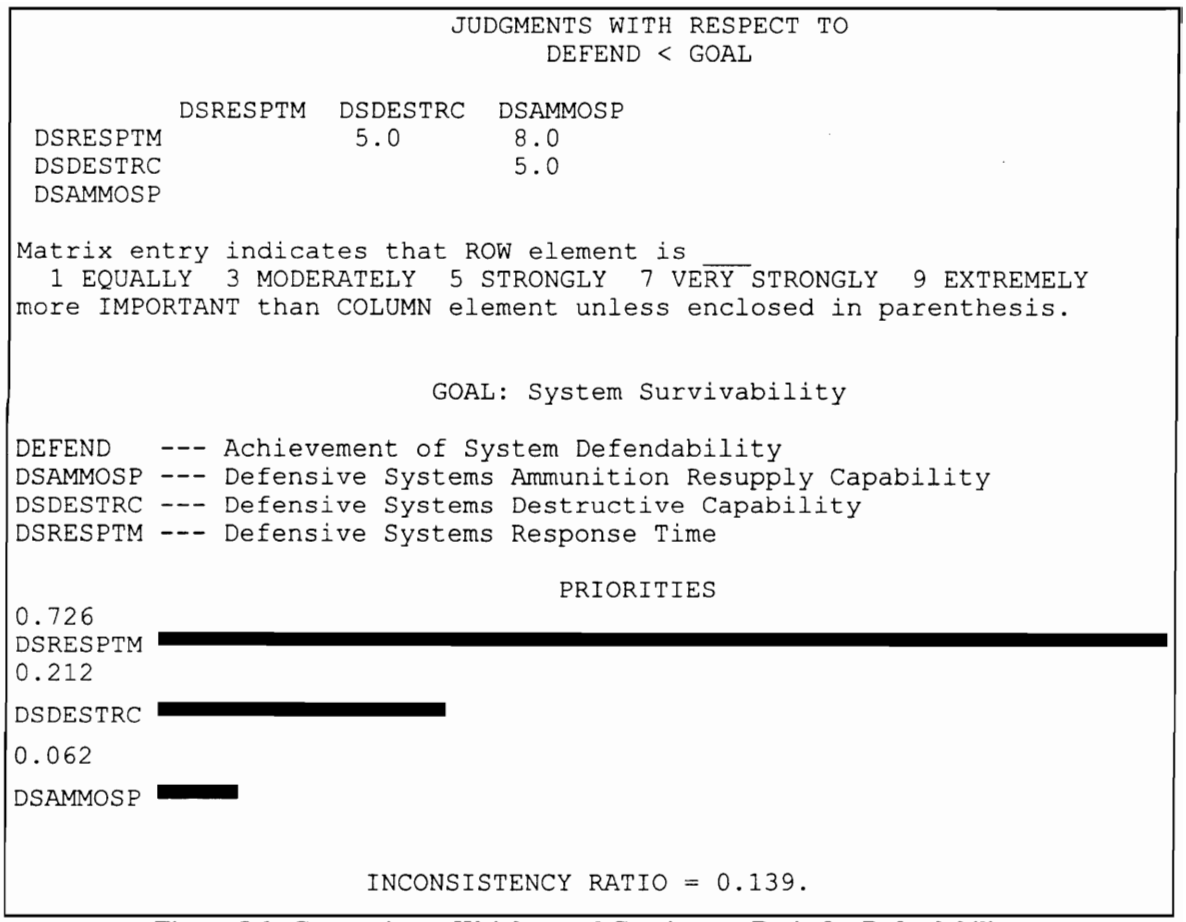


Figure 5.6. Comparisons, Weights, and Consistency Ratio for Defendability.

JUDGMENTS WITH RESPECT TO
RECOVER

	STRCTMNT	LIFSPMNT	DEFSYMNT	PRPSYMNT	CLOAKMNT
STRCTMNT		3.0	1.0	1.5	8.0
LIFSPMNT			1.0	(2.0)	5.0
DEFSYMNT				1.0	5.0
PRPSYMNT					5.0
CLOAKMNT					

Matrix entry indicates that ROW element is ___
1 EQUALLY 3 MODERATELY 5 STRONGLY 7 VERY STRONGLY 9 EXTREMELY
more IMPORTANT than COLUMN element unless enclosed in parenthesis.

GOAL: System Survivability

CLOAKMNT --- Cloaking System Maintainability
DEFSYMNT --- Defensive Systems Maintainability
LIFSPMNT --- Life Support Systems Maintainability
PRPSYMNT --- Propulsion System Maintainability
RECOVER --- attainment of capability to recover system if damaged
STRCTMNT --- Superstructure Maintainability

PRIORITIES

0.336
STRCTMNT
0.160
LIFSPMNT
0.227
DEFSYMNT
0.236
PRPSYMNT
0.040
CLOAKMNT

INCONSISTENCY RATIO = 0.027.

Figure 5.7. Comparisons, Weights, and Consistency Ratio for Recoverability.

JUDGMENTS WITH RESPECT TO
THREATDT < GOAL

	OPTICSNS	INFRASNS	ELECTSNS	ACOUTSNS
OPTICSNS		1.0	(6.0)	1.0
INFRASNS			1.0	3.0
ELECTSNS				5.0
ACOUTSNS				

Matrix entry indicates that ROW element is ___
1 EQUALLY 3 MODERATELY 5 STRONGLY 7 VERY STRONGLY 9 EXTREMELY
more IMPORTANT than COLUMN element unless enclosed in parenthesis.

GOAL: System Survivability

ACOUTSNS --- Acoustic Sensor Capability
ELECTSNS --- Electrical Sensor Capability
INFRASNS --- Infrared Sensor Capability
OPTICSNS --- Optical Sensor Capability
THREATDT --- ability to successfully detect external threats

PRIORITIES

0.138
OPTICSNS
0.278
INFRASNS
0.485
ELECTSNS
0.099
ACOUTSNS

INCONSISTENCY RATIO = 0.116.

Figure 5.8. Comparisons, Weights, and Consistency Ratio for Threat Detectability.

5.3.2 Computing Relative Weights

Step 5 calls for the computation of relative indicator weights and assessment of judgment consistency. Details of the procedure are described in Section 5.2. For illustration and comparison, we compute the consistency using four relatively simple methods, then give the exact results using the *Expert Choice* software. Each method is described in [Saaty 90].

To begin computation of relative weights, we present for reference the complete reciprocal matrix shown in the left side of Table 5.4. To arrive at the results of Method 1, we employ what Saaty [90] refers to as the *crudest* technique. In this case, each row sum is simply divided by the sum of all the sums. For Method 2, the *better* technique, the reciprocal of each column sum is divided by the sum of the reciprocals. Method 3, called *good*, requires dividing each entry by its column sum. The row sums are then computed and each is divided by the number of elements in the row. Method 4, also *good*, begins with the geometric mean (*n*th root of the product) of each row's entries, where there are *n* entries in each row. Each row mean is normalized by the sum of the row means. The results and some intermediate values are provided in the right side of Table 5.4.

The exact value requires raising the judgment matrix to arbitrarily large powers and normalizing each row sum by the sum of all elements in the matrix. The solution vector in this case is $(0.055, 0.160, 0.170, 0.293, 0.321)^T$, which to three decimal places agrees with the results of Method 3. In fact, accuracy in approximating this

Table 5.4. Matrix of Computations for Weight Determination.

SURVIV-ABILITY	Stealth	Maneuver	Defend	Recover	Threat Detect		ROW SUM		Meth 1	Meth 2	Meth 3	Meth 4
Stealth	1.00	0.33	0.33	0.20	0.17		2.03		0.055	0.056	0.055	0.056
Maneuver	3.00	1.00	1.00	0.50	0.50		6.00		0.163	0.158	0.160	0.161
Defend	3.00	1.00	1.00	0.67	0.50		6.17		0.168	0.172	0.170	0.170
Recover	5.00	2.00	1.50	1.00	1.00		10.50		0.286	0.298	0.293	0.292
ThreatDt	6.00	2.00	2.00	1.00	1.00		12.00		0.327	0.317	0.321	0.321
COLSUM	**18.00**	**6.33**	**5.83**	**3.37**	**3.17**		**36.70**		**1.00**	**1.00**	**1.00**	**1.00**

vector's true value increases from Method 1 to Method 3. Method 4 provides a suitable estimate in the presence of larger inconsistency.

5.3.3 Determining Consistency of Judgments

The solution vector which provides indicator weights is also the right *eigenvector* for the judgment matrix, introduced in Equation 5.1. From Equation 5.2, we see the role of the maximum eigenvalue, λ_{max}, of the judgment matrix, in the determination of a consistency index of the judgments (CI). From the CI we can derive the (in)consistency ratio, CR. Henceforth, the terms "consistency ratio" and "inconsistency ratio" are used interchangeably with the former term being preferred.

We can estimate the eigenvalue by solving Equation 5.1 for λ_{max}. First, take the product of the judgment matrix and the right eigenvector (computed from any of the methods above). Divide the resultant vector by the eigenvector estimate. Next take the arithmetic mean of the elements of the quotient vector. For our example, the resultant values for the product and quotient vectors and the eigenvalue estimate are respectively:
$(.278, .804, .853, 1.468, 1.608)^T$, $(5.008, 5.011, 5.006, 5.014, 5.011)^T$, and 5.01.

Applying Equation 5.2, using $n=5$, the CI for the judgment matrix is: $(5.01-5)/(5-1) = .0025$. Applying Equation 5.3 and Table 5.3 (for $n=5$, RI = 1.11), we obtain an (in)consistency ratio, CR of $.0025/1.11 = .002$. This value is well below the 0.1 desired upper bound, so we proceed with confidence in the consistency of our judgments for this set of indicators. For other sets of judgments, such a desirable consistency is not achieved the first time, in the following sections we discuss handling less than desirable consistency.

5.3.4 Handling Inconsistency in Pairwise Comparisons

Upon review of Figures 5.4-5.8, the reader may notice that the consistency ratios for the sibling groups of *Maneuverability* (.166), *Defendability* (.139), and *Threat Detectability* (.116) exceed the desirable threshold of 0.1. We note in Section

5.2 references to several mathematical techniques for converging to an acceptable consistency, e.g., [Wedley 93; Harker 87a]. In this section we provide methods for determining an approach to handling inconsistencies in judgments. Several factors should be considered individually and in combination:

- *degree* of inconsistency
- *weight* of the parent indicator (e.g., Maneuverability)
- *expected weight* of the parent indicator
- *number* of siblings in the sibling group
- *agreement* among evaluators regarding the original pairwise comparisons

In light of consideration of the factors, there are three courses of action:

1. Accept the level of inconsistency
2. Determine and adjust the most offensive pairwise comparison(s)
3. Conduct the entire comparison exercise again for the sibling group.

In the subsections to follow, we discuss techniques for each course of action. Techniques for the second and third course of action are well studied and in many cases, mathematically complex. In the hundreds of references to the AHP we have encountered, we have yet to find a direct reference to the first course of action. A notable contribution to the subject matter then, is the presentation of a logical and mathematical basis for determining when apparently excessive inconsistency can be tolerated. However, we cite a single, possible exception to our claim of original contribution. Apostolou and Hassell [93] conducted a study which revealed no significant statistical differences ($p = 0.05$) in mean weights determined by over 100 evaluators. This was observed even though the mean CR was 0.125, and half of the evaluators' comparisons had a CR > 0.10. The authors did not extend their findings to *methods* for interpreting inconsistency, and so we assert that our claim of a contribution is valid.

For the latter two options, we reference published techniques and point the reader to a software tool which assists in executing the options.

5.3.4.1 Accepting Excessive Inconsistency

A team of expert evaluators may actually be justified, mathematically and logically, in simply accepting greater than desirable inconsistency. Justification can be based on intuition or dogma alone, taking shelter in the maxim that a threshold of 0.1 is Saaty's *recommendation*, not a mandate. Practically speaking, since the pairwise comparisons are the product of human preference and judgment, perfect consistency need not necessarily be expected. Transitivity in human judgment simply is not required. It is our intention in using the AHP to account for that aspect of human judgment.

A more plausible justification for ignoring excess inconsistency is that the weight of the parent indicator is relatively low compared to its own sibling indicators. In such cases, the parent indicator weight may be considered "low enough" such that any negative consequences of inconsistency have insignificant impacts on the upper regions of the hierarchy.

A simple indicator of "lowness of weight" is the ratio of an indicator's actual weight to its *expected* weight, $1/n$, where the parent indicator is one of n sibling indicators. In our example, *Maneuverability* has a weight of 0.160. Given that it is one of five siblings, its expected weight is 1/5 = 0.2. The ratio is then 0.160/0.20 = 0.8. The indicator has a weight about half of that of either *Recoverability* (0.293) or *Threat Detectability* (0.321). Also, it has a weight *less* than would be expected if all indicators were considered equal.

In view of the foregoing discussion, we introduce a factor to guide the evaluators in determining when a degree of inconsistency exceeding 0.1 may be considered acceptable. In general, when the parent indicator of a sibling group has a weight *approaching or below* its expected weight, the *Lightweight Factor* (LF), can be applied. This factor is computed as in Equation 5.5:

$$\textit{Lightweight Factor} \equiv \frac{n_s}{CR^* w_p^* n_p} \tag{5.5}$$

where:

n_s is the number of indicators in the sibling group for which (in)consistency has been computed;

CR is the consistency ratio of the sibling group;

w_p is the weight of the parent indicator (e.g., Maneuverability); and

n_p is the number of siblings in the parent's sibling group (factors into the expected weight of the parent).

We now examine the components of the Lightweight Factor. The LF takes into consideration all but one of the items we list above with regard to determining how to approach inconsistency. It does not take into account the subjective factor "agreement among evaluators on comparisons." This subjective factor is well-suited to implementation through knowledge-based inferencing, as we address in Chapter 7.

Regarding the interpretation of the values of the Lightweight Factor, we have arbitrarily chosen to interpret *higher* values of this factor as greater reasons for *accepting* the excessive inconsistency. The components of the formula are discussed in the following paragraphs, and we give practical bounds in Table 5.5.

We have placed n_s, the number of sibling indicators in the comparison group, in the numerator of Equation 5.5. The reason may be intuitive: the more sibling indicators considered in the pairwise comparison process, the more likely intransitivity in judgments is to occur. Thus, with more siblings in the decision, we increase the tolerance for inconsistency.

We have placed the Consistency Ratio (CR) in the denominator. This allows lesser degrees of inconsistency to drive the value higher, increasing favor toward accepting the degree of inconsistency.

Table 5.5. Practical Bounds on Components of the Lightweight Factor.

Component	Bound	Commentary
n_s	[3, 9]	In keeping with practical sibling group sizes.
CR	(.1, .3]	We and [Saaty 90, 94] regard a CR exceeding .3 to be unacceptable. Less than .1 is *de facto* acceptable.
$n_p w_p$	[.2, 1.5]	Values exceeding 1.5 indicate a relatively critical indicator. Less than .2 is considered *de facto* negligible.

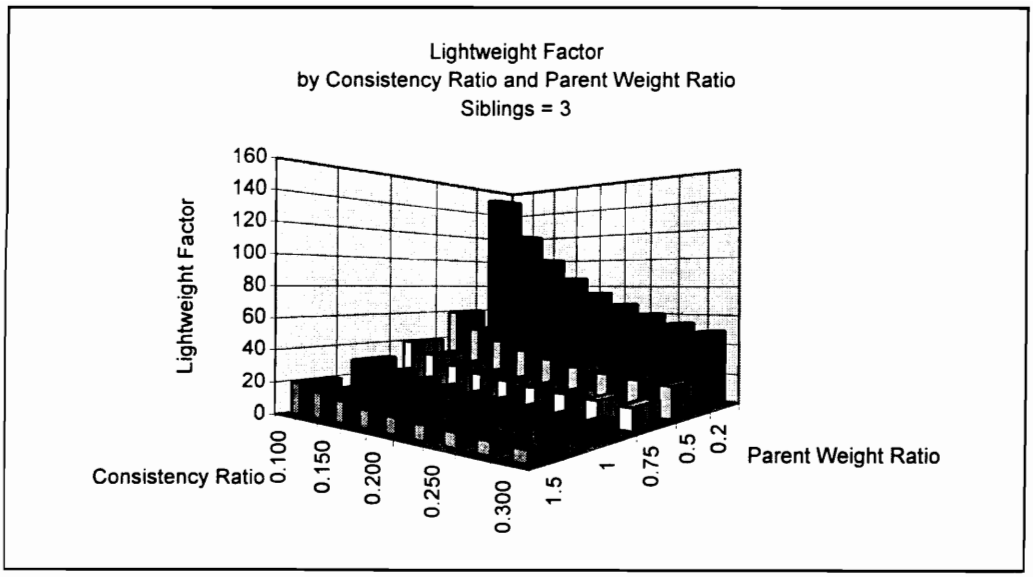


Figure 5.9. Lightweight Factor Values ($n_s = 3$).

The quantity $n_p w_p$ is the algebraic equivalent of the ratio of the weight of the parent indicator to its expected weight, $(1/n_p)$. The smaller this quantity is, the less critically weighted the parent indicator is. Thus the less critical becomes the consistency of the judgments pertaining to its children indicators in the hierarchy.

In light of the immediate foregoing and Table 5.5, we now examine the associated aggregate bounds on the Lightweight Factor. Using the most desirable, median, and least desirable values from the bounds, we respectively obtain (450, 41, 8). Note that from the above discussion, the most desirable bound for n_s is 9 instead of 3, and the median values are respectively (6, 0.2, 0.725).

The function representing the Lightweight Factor grows rapidly as the individual components approach the most favorable values. Figures 5.9-5.11 depict several graphs representing the Lightweight Factor over ranges of component values.

The reader may note the prominent spike in the graphs as the parent weight ratio approaches a minimum value. Also evident from the graphs is the nearly flat region as both the x and y axes approach their least desirable values. Appreciating these two regions of the graphs, we can now provide an interpretation of the Lightweight Factor to establish a minimum *threshold* for accepting inconsistency. We select as a threshold the value 45, approximately the median value. Above this threshold, a decision to accept inconsistency can be made without significant consequence, since each contributing factor would be on the more favorable end of its respective bound.

5.3.4.2 Determining and Adjusting the Source(s) of Inconsistency

In this section we treat cases where the consistency ratio was below 0.3 (the highest value considered by the Lightweight Factor approach) but the Lightweight Factor is below the threshold we have established. In these cases, there may be a prominent occurrence of inconsistency or intransitivity among one or more pairs of indicators, which may have been overlooked when a large sibling group was being compared pairwise.

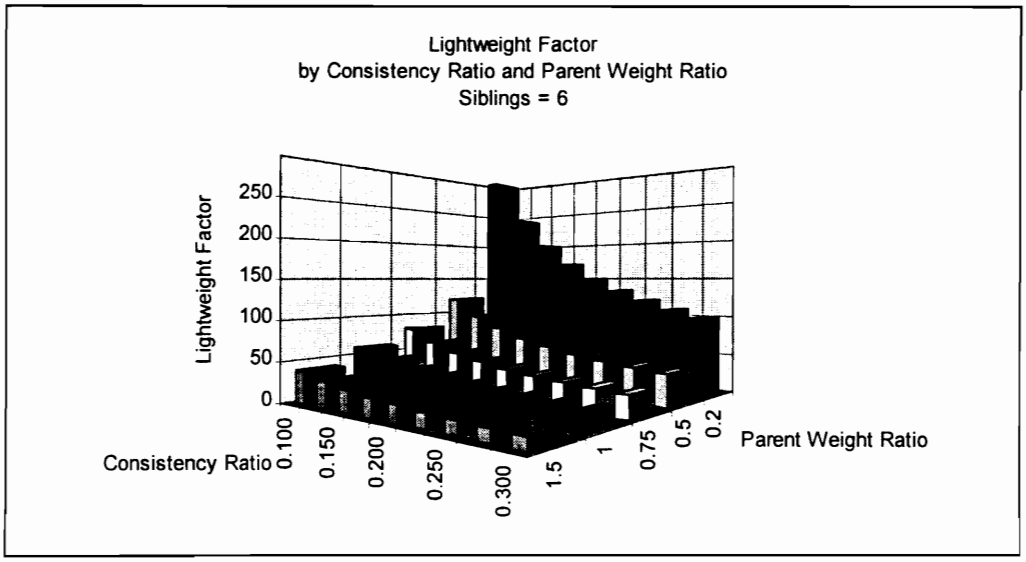


Figure 5.10. Lightweight Factor Values ($n_s = 6$).

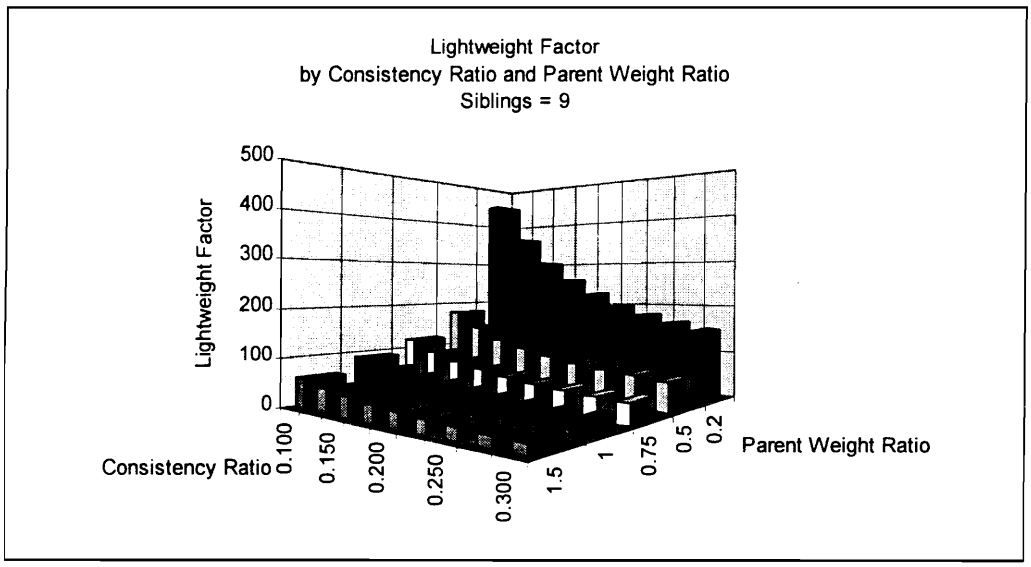


Figure 5.11. Lightweight Factor Values (n_s = 9).

One relatively simple method for detecting a possible source of inconsistency is by simple visual inspection. This technique is only recommended when no AHP software tool is available (e.g., when only a spreadsheet or calculator is available). In this technique, an analyst "eyeballs" the relative consistency of a matrix entry a_{ij} by multiplying two entries a_{ik}* a_{kj}. An entry a_{ij} which is considerably different from one or more of its constituent products is a good candidate for re-evaluation.

Obviously, there can be as many as n-1 such products for each entry. However, since judgment matrices with a moderate to low n_s are more likely to have unacceptable LF values than highly populated sibling groups, this visual checkup can be performed relatively painlessly, or assisted by a spreadsheet or pocket calculator.

An additional method, usable on most spreadsheets is a variation of a matrix multiplication technique, multiplying the judgment matrix by itself. The task is conducted by taking the row*column element-wise product as usual, but instead of summing them, take a mean (or maximum) value. A comparison of this value to position a_{ij} in the judgment matrix reveals which entries depart the most from "expected" (average or best case) consistency.

More complex mathematical procedures exist. Several sources are referenced in Section 5.2 which describe techniques for pinpointing the chief instances of *preferential intransitivity* [Dadkah and Zahedi 93; Salo 93; Wedley 93; Harker 87b; Blankmeyer 97]. The details of these techniques are beyond the scope of this document, and the reader is referred to the respective sources. However, we outline the technique of [Dadkah and Zahedi 93], since it is illustrative of the type of matrix algebra required.

The basic technique requires the left and right eigenvectors associated with λ_{max}, the eigenvalue of the complete judgment matrix. The iterative procedure takes the derivative of the judgment matrix with respect to each entry according to Equation 5.6.

$$\frac{\delta \lambda^{*}_{A}}{\delta a_{ij}} = \frac{v_i u_j - u_i v_j a_{ij}^{-2}}{v^T u} \tag{5.6}$$

where

λ^{*}_{A}	is the maximum eigenvalue for the matrix A
a_{ij}	is a position in the matrix A
a_{ij}^{-2}	is the square of the reciprocal of the value of position a_{ij}
u	is the right eigenvector for matrix A [i (j) is the ith (jth) position]
v	is the left eigenvector for matrix A [i (j) is the ith (jth) position]
v^T	is the transpose of the left eigenvector.

In lieu of applying Equation 5.6 to all a_{ij}, an alternative is to take the derivative with respect to only a select subset of "suspect" entries. Suspect entries are those whose values appear grossly inconsistent, even by visual inspection (or by using a technique as described above).

After applying Equation 5.6 to some or all of the entries in the judgment matrix, the ones with the largest departures from zero are the best candidates for change. If the derivative with respect to element a_{ij} is positive, a reduction in value is recommended. If the derivative is negative, an increase to the element is called for. In either case, the reciprocal value in position a_{ji} must be changed accordingly. Note that several or all changes may be made during a single iteration of this process.

This process is intended to be iterative, leading to a re-examination of matrix consistency after each iteration until consistency becomes acceptable. Note that for most accurate results, the eigenvalue and eigenvectors of the revised judgment matrix should be recomputed before each subsequent iteration.

Finally, we recommend the simplest method, making use of software tools developed by others, e.g., the tool *Expert Choice*. This tool automatically pinpoints the top several candidates for comparison adjustment, and recommends an appropriate value [Saaty 93]. However, as the authors of the tools point out, no pairwise comparison should be changed simply for the sake of improved consistency. The identification of gross intransitivities and recommendations for consistency improvement must be weighed against the experience of the evaluators who made the original pairwise comparisons. We reiterate that preferences need not be perfectly consistent.

5.3.4.3 Reassessing Consistency for an Indicator Sibling Group

In some cases, the process of making pairwise comparisons among an indicator sibling group may need to be reaccomplished entirely. This is most likely to be the case when:

- gross inconsistency has been realized, or
- poor agreement can be reached regarding either:
  - (i) which original comparisons to revise or
  - (ii) to what value they should be revised.

We propose in Section 5.3.1.1 two preliminary exercises which should minimize the likelihood of these conditions being prominent in a set of comparisons. In the event that consistency problems remain, we recommend two approaches, both facilitated by *Expert Choice*. The first approach involves simply reordering the indicators in the judgment matrix to stimulate a new line of thinking. The second technique involves "what-if" analysis. Especially when using an AHP-based tool, tuning one or more comparisons and allowing the software to quickly recompute

weights and consistency may indicate a flaw in the original application of the pairwise comparison procedure.

5.3.5 Adjusting for Inconsistency in the Example Hierarchy

For our example hierarchy, and from Figures 5.4-5.8, we obtain the weights, consistencies, parent information, and Lightweight Factor (LF) values in Table 5.6. Notice that two of the most highly weighted indicators (*Defendability* and *Threat Detectabilty*) have LF values falling below the threshold of 45. *Maneuverability* is right on the threshold and is considered acceptable. For illustrative purposes, we address only *Threat Detectability*, since it has the lowest LF value.

The judgment matrix for *Threat Detectability* is provided in Figure 5.12. We employ first the "eyeball" method then the matrix multiplication method to determine a likely candidate for reconsideration of the comparison value. By visual inspection, we can detect apparent logical inconsistencies with regard to the comparisons among ELECTSNS, OPTICSNS, and INFRASNS. We indicate that OPTICSNS and INFRASNS are of *equal* importance. The inconsistency comes in with the comparison of ELECTSNS to these two. ELECTSNS is judged to be *strongly to very strongly* more important than OPTICSNS, but *equally* as important as INFRASNS. A reasonable correction would be to reconsider the relationship between OPTICSNS and INFRASNS, since in no other comparisons are other indicators preferred the same to both. In fact, INFRASNS is judged to be *moderately* more important than ACOUTSNS, and ACOUTSNS is judged to be equally important as OPTICSNS.

Table 5.6. Lightweight Factors for Example Hierarchy.

Indicator	n_s	CR	n_p	w_p	LF
Stealthability	5	0.008	5	0.055	2273
Maneuverability	6	0.166	5	0.16	45
Defendability	3	0.139	5	0.17	25
Recoverability	5	0.027	5	0.293	126
Threat Detectability	4	0.116	5	0.321	21

	OPTICSNS	INFRASNS	ELECTSNS	ACOUTSNS
OPTICSNS	1	1	.166	1
INFRASNS	1	1	1	3
ELECTSNS	6	1	1	5
ACOUTSNS	1	.333	.200	1

Figure 5.12. Complete Judgment Matrix for Threat Detectability.

We take a conservative approach to adjustment and alter the judgments to reflect that INFRASNS is closer to *moderately* more important than OPTICSNS. This gives a value of 0.5 in position (2,1) and reciprocally a value of 2 in position (1,2). The resulting inconsistency ratio recomputes to 0.04, which is satisfactory.

Employing the matrix multiplication technique, we observe slightly different but similar results. Application of the multiplication and mean process yields the differences matrix in Figure 5.13. The entries in the figure represent the *difference* between the mean of the products from the multiplication operation and the original judgments. Using this technique, we find three top candidate matrix entries. Position

THREATDT	OPTICSNS	INFRASNS	ELECTSNS	ACOUTSNS
OPTICSNS	2.04	0.62	1.22	2.29
INFRASNS	2.75	1.33	1.09	1.00
ELECTSNS	-0.50	3.08	2.84	0.75
ACOUTSNS	1.88	1.13	1.02	2.13

Figure 5.13. Differences between Mean of Consistent Entries and Original Entries in the Threat Detectability Judgment Matrix.

(3,2) is the highest. The next one, in position (3,3), must be discounted since the original entry must be 1. Our "eyeballed" selection in position (2,1) is next.

Notice that the ELECTSNS vs. INFRASNS comparison is a key offender, just as we show earlier. Increasing the original value in (3,2) to 2 (position (2,3) becomes 0.5) reduces the inconsistency to .06, which is good, but not as good as the results of our original reduction effort. In either case, the original judgments have merely been refined, not fundamentally changed. Consequently, we can proceed with assurance that our original assessments of relative importance have not been compromised for the

sake of achieving numerical consistency. Incidentally, independently using the *Expert Choice* tool, we found that the judgment in position (2,1) was the most likely candidate for improved consistency. *Expert Choice* recommends a value of 4.3 for the position which would generate CR of 0.011. We caution that this adjusted value improves consistency, but may no longer reflect the true opinion of the evaluators.

5.3.6 Completing the Example

Finally, we tune the original judgments as needed to reduce inconsistency. For the *Threat Detectability* judgment matrix, we make changes according to the inspection method as discussed above. We set entry in position (2,1) to 2 and the reciprocal entry in position (1,2) to 0.5. The resultant CR is 0.04 and the weights of the indicators are changed as depicted in Table 5.7. For *Defendability*, we reduce the entry for DSRESPTM (response time) versus DSDESTRC (destructive capability) in position (1,2) to 4, ((2,1) becomes 0.25). The result is a new CR of 0.09. The resultant changes in indicator weights are presented in Table 5.8. Notice that the differences are not significant, lending credibility to the result of Apostolou and Hassell [93].

Table 5.7. Changes in Weights for *Threat Detectability* Sibling Group.

	OPTICSNS	INFRASNS	ELECTSNS	ACOUTSNS
Original	0.138	0.278	0.485	0.099
Adjusted	0.109	0.319	0.473	0.100

Table 5.8. Changes in Weights for *Defendability* Sibling Group.

	DSRESPTM	DSDESTRC	DSAMMOSP
Original	0.726	0.212	0.062
Adjusted	0.699	0.237	0.064

We turn now to steps 8-10 of Saaty's AHP technique. We find these steps to be essentially inapplicable for our purposes for the following reasons. Step 8 calls for the computation of the consistency of the entire hierarchy. In applications of the AHP where the element priorities are the entire basis for decision making, this step may be essential. In our methodology, the priorities obtained from the AHP are

complementary to the indicator scores themselves. Thus, we do not require computation of hierarchical consistency. Step 9, which calls for adjustments for hierarchical inconsistency, subsequently does not apply.

Step 10 calls for the computation of *global* weights of criticality for individual indicators. Since our methodology requires weighting and scoring of indicators only with respect to the "local" sibling group and immediate parent(s), we have no requirement for these values.

5.4. ALTERNATIVE WEIGHT DETERMINATION METHODS

In Sections 5.2 and 5.3, we detail and demonstrate the steps of the AHP. We recognize that numerous means of arriving at relative criticality weights exist that have been used in practice. We briefly introduce some techniques from the literature which do not involve pairwise comparisons. We include these techniques only for completeness of the discussion of indicator weighting. In so doing, we forego any details, but point the reader to instructive references.

Keeney and Raiffa [89, 93] offer a lottery method relying similarly on human judgments, but more greatly reflecting the conscious tradeoff process involved. It involves a series of "what-if" questions, with the objective of establishing a weight for one entity at which the entity is equally preferable to an existing one. The technique is employed with an illustrative example in [Thurston 91]. In some indicator groups where pairwise comparisons pose a particular problem due to company politics or otherwise, we recommend this technique as an alternative. We note, however, it is not as well suited for group judgments as the AHP.

A second technique stems from the application of neural networks. Neural networks are a popular development for use in high-speed computing environments for highly specialized applications, (e.g., pattern recognition and process control software). Key to employing neural networks is determining relative criticalities among the attributes of an input which allow it to be classified as one of many types of known

entities. The *training* of a network results in a finely tuned set of weights for a collection of sibling nodes. Excellent tutorials on the subject are provided in [Klir and Yuan 95; von Altrock 95].

5.5. SUMMARY

In view of our taking note of alternative weighting techniques, we conclude this chapter with a summary of the aspects of the AHP which make it our primary weighting procedure. The widespread applicability of the AHP supports a solid case for AHP as an appropriate process for assigning weights to both quantitative and qualitative indicators. We punctuate the salient features of the AHP which make it our choice.

As with the structuring of the hierarchy(ies) of indicators, the assignment of weights to indicators also reflects the evaluators' understanding of the intensity of relationships among them. Additionally, in an indicator-based evaluation such as ours, patterned after the utility theory, the criticality of the weighting aspect is obvious — it is one of the two key multiplicands in the formula. In view of the importance of the evaluation phases in the life of a complex system and its design, our specifications for our weighting scheme provide:

1. Simplicity of a data input
2. Traceability of data inputs and user motivation for data inputs
3. Capability of the scheme to allow human judgments as inputs
4. Adaptability of the scheme to tolerate absence of judgments
5. Robustness of the scheme to tolerate inconsistency in human judgment
6. Capability in the scheme to feedback a measure of consistency to the user
7. Flexibility of the scheme to allow for consistency adjustment
8. Capability of the scheme to provide guidance for improving consistency
9. Extendibility of the scheme to the domain of complex system design evaluation
10. Maturity of the scheme as tested, proven, and dependable in practical applications
11. Widespread acceptability of the scheme evidenced by the references in the literature

In what follows, we recount the AHP features which provide assurance that the AHP satisfies the characteristics listed above. Input of judgments while employing the AHP requires only simple, pairwise comparisons, relative to a (typically 1-9) ratio scale. Contrast this to the brute force technique of heuristically assigning weights to an entire collection of indicators simultaneously.

Final weightings are traceable from the judgment input matrices. The matrices are prepared manually as judgments are made by evaluators, or prepared dynamically by AHP software, if such a tool is used. The tool *Expert Choice*, prompts users for input using verbal, numerical, or interrogative techniques. With the inputs, it constructs the matrices and stores them for analysis, comparison, and printing. Additionally, *Expert Choice* provides the capability for users to input notes for the rationale behind a particular comparison.

We address the reality of the $O(n^2)$ number of pairwise comparisons which could be required by the AHP. However, we note the contributions of, e.g., Dadkhah and Zahedi [93], DeTurck [87], Harker [87a, 87b], and Wedley [93], which indicate that satisfactory consistency can be achieved with high assurance in $\Omega(n)$ comparisons. These extensions to the AHP allow for missing judgments under certain restrictions as discussed earlier. Additionally, extensions such as these provide evidence of the tried and proven nature of the process. Each extension increases the reliability of the process, and increases the scope of scenarios in which it can be applied.

Ensuring robustness of a process to tolerate the inherent inconsistency in human judgment yet produce reliable results is a non-trivial task. The original AHP with its extensions over the years has essentially accomplished the task, though at a cost of computational complexity. Given the critical nature of the routine operation of complex systems, the cost is certainly justified. In ensuring reliable performance in the presence of human inconsistency, the AHP has been extended to incorporate methods for pinpointing sources of inconsistency. Extensions also provide guidance for

converging to an acceptable consistency [Dadkhah and Zahedi 93; Wedley 93; Wedley, *et al.* 93].

The applicability of the AHP to complex system design evaluation is evidenced by numerous sources as we reference in the literature review of Chapter 2. The maturity of the AHP is evidenced by the plethora of references in the literature which either demonstrate its application or suggest improvements or extensions. We see these numerous augmentations not as evidence of flaws in the original process, but as fires of refinement, tuning the process for a wide range of reliable application. Finally, the multiplicity of references in the literature describing application of the AHP satisfy our requirement that the process is widely accepted. For examples of its application outside the realm of complex systems, see e.g., [Banai-Kashani 89; Dimenna 89; Masuda 90; Peterson 84; Shields 86; Srinivasan and Bolster 90; Wang and Raz 91; Weiss 87; Yan and Svedberg 91, and Zahedi 90].

6. ASSIGNING INDICATOR SCORES

6.1. INTRODUCTION

In Chapter 4, we describe methods pertaining to the construction of a hierarchy of complex system design quality indicators. Within indicator sibling groups in the hierarchy, indicators are assigned a weight of criticality. Weights are determined using the Analytic Hierarchy Process, as we describe in Chapter 5. In the present phase of the methodology, indicators are ultimately assigned a score in the range [0,100]. The score represents an objective or subjective measure of the goodness of the design quality represented by the indicator, and respective of the applicable system requirements.

In this chapter, we address assigning scores to the base-level indicators in the hierarchy, and ultimately determining an aggregate score for the entire hierarchy. More specifically, we describe methods for determining scores based on metrics, technical and operational domain expert knowledge, and experimentation with simulation models. In the process of indicator scoring, we may require knowledge of the evaluator emphasis coefficients and the sibling group(s) in which an indicator falls. For the score aggregation process, we require the weights of criticality, determined by techniques in Chapter 5.

The chapter is organized as follows: Section 6.2 describes the process of selecting and applying metrics as a basis for indicator scores. Sections 6.3 and 6.4 describe the process of applying subjective expert knowledge from technical and operational domain experts, respectively. Section 6.5 describes the application of the results of dynamic modeling of system behavior to scoring applicable design quality indicators. Section 6.6 describes the process of aggregating indicator scores to arrive at a score for the entire hierarchy. Section 6.7 provides a chapter summary and draws appropriate conclusions.

6.2. SCORES BASED ON METRICS

A *metric* as defined herein has a discrete value, and is either directly observed or measured, looked up in a table, or computed by one or more formulae. Example indicators whose scores can be based on metrics include those listed below. There are many others.

- software design structural complexity
- hardware component heat tolerance (using known components)
- task complexity
- cost
- hardware connectivity
- documentation readability
- human-hardware control panel layout complexity
- operator visual acuity or hearing ability
- information security classification
- operator intelligence quotient
- I/O data transfer rate (using known components)
- communications bandwidth (using known components)

6.2.1 Determining When to Apply Metrics

In short, a metric should be applied as an objective basis for indicator scoring whenever a reliable, traceable, replicable, appropriate one is known. Since metrics abound in the literature, in proprietary in-house collections, and in central metric sites, the critical task becomes one of selecting the metric(s) to use, once one or more candidate metrics have been found. In this section, we describe sources for metrics, and address the issue of metric appropriateness.

6.2.1.1 Sources of Metrics

In general, the task of selecting one or more metrics to be used in assigning an indicator score is straightforward. In many cases, the task is as simple as finding an appropriate metric from the literature. The well-studied Henry and Kafura information flow metric for software design structural complexity falls in this category [Henry and Kafura 81,84]. However, as we describe below, there are caveats which pertain to metric shopping in the literature, especially, but to metrics from other sources as well.

An alternative method of metric selection is a search of an in-house or central database of metrics appropriate for a particular system design indicator. One such database is the MIST database in use by the Naval Surface Warfare Center, Dahlgren Division. The MIST database is based on the System Design Factors collection of Ngyuen and Howell [94] and contains metrics applicable to various phases of system design evaluation. The present work is one part of a larger effort to provide a circumscribing framework for the use of design evaluation indicators such as in MIST.[1]

6.2.1.2 Metric Appropriateness

Regardless of the source of a metric, the issue of *appropriateness* must be a consideration. We have not found in the literature an explicit set of criteria for metric appropriateness, but Shepperd [89], Courtney and Gustafson [93], and Kitchenham, *et al.* [94], and Zage, *et al.* [95] present related works. We believe guidelines for assessing metric appropriateness are a necessary component of a methodology which employs metrics. The attributes in Table 6.1 must be considered when selecting a metric as a basis for indicator scoring.

Table 6.1. Metric Acceptability Attributes.

Attribute	Explanation
Widespread support in the literature	• Demonstrated by application with credible results. • Extensions or improvements offered. • Not mathematically disproven or contradicted. • No underlying assumptions invalidated.
Credibility of linear correlations	• No "shotgun correlations."
Design life cycle phase appropriateness	• Applicable to design, process, or product stage?
Ease/Accuracy of input parameter collection	• Use of automated data collection. • Use of structured, traceable manual data collection.
Computational efficiency	• Can be computed/derived with in-house, even desktop resources.
Stability	• Metric results do not show wide variance with small changes to input parameters.
Domain applicability	• Avoids inefficable crossover of metric from one well-studied domain to another, to which the metric is not as readily or credibly applicable.

[1] The KBESD tool described in Chapter 4 is a complement to such an indicator database. It facilitates the computation of aggregate indicator scores while employing a rule base for knowledge-based evaluation guidance.

One metric quality attribute of particular concern is *credibility of linear correlations*, which can be gauged by the degree of use of so called "shotgun correlations" described in [Courtney and Gustafson 93]. The authors express concern regarding the proliferation of new metrics based on linear correlations between software design elements and future life cycle phases of the software project, e.g., source code maintainability. Their objective is to identify elements of any study proffering a metric, in which more statistics than statistical understanding has been applied. In general, the following characterize a study which has made use of shotgun correlations:

- Small sample size relative to the number of parameters considered
- Hypothesis not explicitly stated
- Forces linear correlation coefficients
- Experiment involves many aspects of the software project
- Preliminary ideas about what may be found exist, but are not stated in a manner that matches the statistical tests conducted
- Insufficient demonstration of independence among independent variables
- Correlates as many measures as possible against one or more dependent variables
- Outlier removal without satisfactory explanation of their existence

6.2.1.3 Choosing from Among Similar Metrics

In some cases, multiple appropriate metrics are available for measuring the same design quality attribute. An example indicator is *software design structure complexity*, for which numerous metrics have been published. Examples we have found include those of Chen and Lu [93], Shepperd [89], Card and Agresti [88], Henry and Kafura [81, 84], Yin and Winchester [78], and McCabe [76]. Given that these and certainly more exist, how can a single software design complexity indicator be assigned a value? Clearly the individual measures of design complexity can contribute to the final value for the software design complexity indicator. But how is it done?

We first recommend a screening of the individual metrics with respect to the quality attribute list in Table 6.1. Particular attention should be given to the life cycle phase appropriateness and domain applicability. Weeding out metrics through these filters would prevent the generally inconclusive results from "combining apples and oranges" or "driving a nail with a socket wrench."

Second, with a suite of metrics which are all considered appropriate, another issue to consider is when one metric is subsumed by another, by nature of extension or improvement. For example, the information flow metric of Shepperd [89] is offered as an improvement on the information flow metric of Henry and Kafura. Before evaluators can proceed with the application of one or more metrics to the assignment of an indicator's score, they must decide which (all, some, or one) best represents the design quality attribute measured by the indicator.

On what bases can evaluators make such a decision? We propose some guidelines. The best line of attack is a search for metrics employed in *similar operational scenarios* cited in case studies from the literature. These case studies may indicate the suitability of a particular metric for the given system design. If no relevant case studies exist, *data availability* for metric computation may drive the decision. For example, the availability of design module fan-in and fan-out counts may make an information flow-based metric appealing. In still other cases, the *internal policy of the organization* may dictate the metric(s) to be used. In any case, the best information available should be reviewed and discussed by expert evaluators from the applicable domain. Subsequently, the results of the discussion are factored into the selection of one or a combination of metrics.

6.2.2 Mapping Metric Scores to Indicator Scores

Pursuant to determining that an indicator can be assessed by one or more appropriate metrics, the task requires generating a function to map the native metric score(s) to the uniform score range [0,100]. We address subjects of mapping a single indicator or a collection of indicators in this section.

6.2.2.1 Determining a Mapping Function

Once evaluators have determined a metric is applicable and appropriate as a basis for the value of an indicator, the metric's raw or *native* score must be mapped into the range [0,100]. The mapping requires first that the best and worst possible values for the

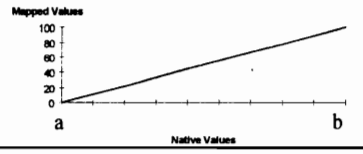
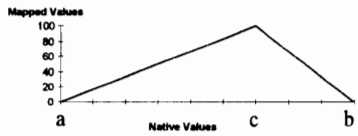
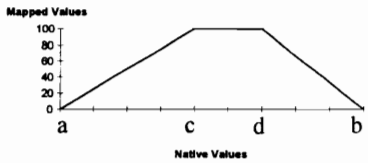
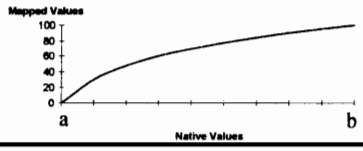
metric's native range of values be determined. In some cases the "best" value may be on the low end of the range. In other cases, the "best" value may be on the high end. We can handle either case. Additionally, if the best (or worst) value appears in the middle of the native score range, as with a trapezoidal score distribution, the least and greatest bounds for the best (or worst) value must also be defined as shown in Table 6.2.

The second piece of information required for the mapping relates to the variability of native scores for the metric. In particular, mapping native scores to the range [0,100] requires that we know how the *desirability* of native scores varies from the worst case to the best case, or vice-versa. By desirability we mean the following: are higher or lower native scores more desirable, and how does the desirability vary as the score varies over the range? In the general case, the most desirable scores map to 100 and the least desirable scores map to zero. The case for desirably low scores is also easily handled.

Some common score desirability functions vary linearly, logarithmically, trapezoidally, or triangularly. Other possibilities exist. Mapping functions for each of these score types are presented in Table 6.2. Note that in each case, the most desirable scores are near the upper end of the native range. The functions presented in the table can easily be manipulated to depict scores on the low end of the native range as more desirable. For example, if most desirable scores fall on the low end, the linear line in Table 6.2 (linear case) would be drawn from 100 at a, to 0 at b.

We now briefly describe the graphs as depicted in Table 6.2. In the linear case, a unit increment in score is equally desirable over the entire native range. In the triangular case, the most desirable scores are found away from the endpoints of the range. The trapezoidal case is a generalization of the triangular case, in which a subrange of native scores away from the endpoints are equally desirable. The logarithmic case accounts for a strong bias against scores on one end of the native range.

Table 6.2. Common Metric Native Score Mapping Functions.

Mapping Type	Function	Graph
Linear	$100 * \frac{(x-a)}{(b-a)}$	
Triangular	$100 * \frac{(x-a)}{(c-a)}; \quad a \le x \le c$ $100 * \frac{(b-x)}{(b-c)}; \quad c < x \le b$	
Trapezoidal	$100 * \frac{(x-a)}{(c-a)}; \quad a \le x \le c$ $100; \quad c \le x \le d$ $100 * \frac{(b-x)}{(b-d)}; \quad d < x \le b$	
Logarithmic	$100 * \log_{(b-a+1)}(x-a+1)$	

How is an appropriate score mapping function determined? In general, it is determined by the natural variance of possible scores from the metric itself. For example, if the metric varies linearly with linear variances in input, then the mapping function can be linear. If there is an optimization point or point of diminishing returns inherent in the original metric's score range, then the triangular or trapezoidal mapping can be used.

Note also that these two functions can be used to approximate the classic bathtub or cave score trends. As mentioned above, the logarithmic mapping function can be selected when there is a desire to markedly discount, or accent scores from one end of the native range. In extreme cases, an exponential mapping function may be applied. While we have not depicted one in Table 6.2, clearly the exponential curve (or its inverse) would show a steep rise (or decline) on one end of the native range. As a final note, we

point out that this mapping differs from the desirability range mapping discussed in Chapter 4. This mapping makes scores in the [0,100] range *possible*, while the desirability ranges determine which scores *within* [0,100] are acceptable, marginal, etc.

6.2.2.2 Combining Multiple Metrics

In some cases, an indicator can be assigned a value based on a combination of several metrics. In those cases, how are multiple metrics combined to give a single normalized score? There are two general conditions in which this situation is likely to occur. The first case is realized when a linear[2] equation relates several independent variables to a measure for a single dependent variable. The undesirable hazards of a shotgun correlation notwithstanding, this relation combines several metrics using correlation coefficients. Those coefficients are based on linear regression techniques, that determine which of a collection of metrics account for the greatest amount of variance in the dependent variable. The correlation coefficients reflect the degree of impact of each independent variable on the dependent one.

For an illustrative example, we present the design structure *object class complexity* metric for an object-oriented software design, found in Chen and Lu [93]. Constituent metrics include the following, all of which are either directly computable or scorable by a table lookup, from tables generated by the authors:

a) operation complexity
b) operation argument complexity
c) operation coupling
d) class coupling

Using linear regression, the authors derive the equation: Class complexity = 153.95+(-0.49)**a**+(-0.35)**b**+(-7.35)**c**+(-9.61)**d**, where the letters represent the corresponding metrics from the above list. To convert this metric to an indicator, the

---

[2] We acknowledge non-linear combinations are possible, but they are encountered far less frequently in the literature than linear metric combinations.

range of possible (and reasonable) values for the total native score would be mapped to the range [0,100] as in the previous section.

A second case for combined metrics occurs when *several* metrics are deemed appropriate as a basis for scoring an indicator. We list above several software design complexity metrics. How are these combined to give a meaningful indicator score?

To be consistent with one of the major thrusts of this work, we recommend a *weighted average* technique as a primary method. Under these circumstances, the native score from each individual metric used would be mapped to [0,100] and then weights of preference attached to each converted score. For example, using either the Analytic Hierarchy Process or another prioritizing method, evaluators determine the preference for each metric on which the score is to be based. We suggest the acceptability criteria described in Table 6.1 as the basis for prioritizing each metric. After a priority has been established, the weights must be normalized so that they sum to unity. The final score assigned to the indicator is then the weighted sum of the individual metrics' converted scores.

6.2.3 Combining Scores from Multiple Evaluators

Based on the techniques described in the previous sections, combining scores from multiple evaluators is straightforward. We describe two cases. In the first, all evaluators are computing the same metric. In the second case, different metrics may be used.

6.2.3.1 Using Identical Metrics

One would normally expect that owing to the computable, traceable, replicable nature of metrics, all evaluators computing the same metric should arrive at the same score. However, using again the Chen and Lu [93, p. 233] metric as a case in point, when there are *subjective* elements involved, differences in inputs to the metric's linear combination equation are possible. For example, the "operation complexity" metric listed

Table 6.3. Example Subjective Metric Component: *Operation Complexity Value.*

Rating	Complexity Value
Null	0
Very Low	1-10
Low	11-20
Nominal	21-40
High	41-60
Very High	61-80
Extra High	81-100

as item (a) above, is not computed, but is assigned from a table based on the subjective judgment of each evaluator. We have reproduced the table in Table 6.3 for reference.

From the table and the governing regression equation, it is evident that differences in evaluator's assessment of Operation Complexity could result in different values for the metric, class complexity. In these cases, the combination technique is similar to that for different metrics used by the same evaluator. That is, using *evaluator emphasis coefficients* (Chapter 4), the weighted arithmetic mean of the native metric value is taken. This value is then mapped to the range [0,100] using whatever mapping function has been selected as appropriate.

6.2.3.2 Using Different Metrics

We now consider the case where experts in a particular domain employ different metrics in assigning a value to an indicator. Whether an individual evaluator uses one or multiple metrics is irrelevant. If an evaluator employs *multiple* metrics, then the techniques described in Section 6.2.2, "Combining Multiple Metrics," are to be used. The result is a mapping of the scores to the range [0,100]. If an evaluator uses only a single metric, it is simply mapped to the standard range as before. For either case, as in the previous section, evaluator emphasis coefficients are applied as weights associated with each evaluator's resultant mapped score. The weighted mean of the scores from each evaluator is taken, and the final score results.

6.3. SCORES BASED ON TECHNICAL DOMAIN EXPERT JUDGMENT

In this section, we provide procedures for the application of technical domain expert knowledge to the assignment of a score for an indicator. We discuss determining applicability of this method for a given indicator, conducting judgment-based scoring through interval mathematics techniques, and incorporating the judgments of multiple experts.

6.3.1 Determining When to Apply Technical Domain Expert Knowledge

We describe in Chapters 3 and 4 circumstances under which technical domain expert judgment is a valid source for indicator score assignment. Example indicator categories include those within a technical domain (e.g., human-software interaction) whose names contain "degree of...," "acceptability of...," or similar expressions. In some cases, even indicators with these names are not at the base level of a hierarchy. In these instances, they have been decomposed into subindicators which can be scored based on objective means, as in the previous sections. The indicators relevant to this chapter are those base-level indicators requiring an expert's *subjective assessment* of degree or acceptability of a design aspect measured by a qualitative indicator.

The chief determinant, then, for basing an indicator score on subjective expert judgment is the *absence of any applicable objective basis*. This may seem obvious. However, a primary goal for this methodology is credibility of the proposed indicator assessment techniques. To that end, we offer the following as a primary guideline for assigning any indicator score based solely on expert judgment:

> *All reasonable and credible* objective *means of indicator score assignment should be considered and found inadequate before appreciable subjectivity is introduced into the evaluation process.*

Subjectivity reduces traceability, replicability, and verifiability. It is only to the extent that these properties can be reasonably retained throughout the methodology that subjectivity is tolerated. We facilitate retention of these properties through automation,

on-line documentation of indicator assessment rationale, and knowledge-based checks for consistency, sensibility, and appropriateness of indicator scores (Chapter 7).

It is not our intention to overstate the obvious with the foregoing declarations. Instead, given the composition and intended operational natures of the systems whose designs will be evaluated by this methodology, we desire only to punctuate the evaluative quality assurances inherent in the methodology.

6.3.2 Applying Technical Expert Knowledge to Score Assignment

The general procedure for applying technical domain expert knowledge to design quality indicators requires the direct assignment of a reasonably narrow *interval* of values within the standard range. A single value is simply too precise to be believable. To understand why interval scores are necessary, consider the subjective assignment of an integer score, x, from the range [0,100]. Such an assignment begs the questions, "on what basis is an assignment of x instead of x-1 to be preferred? What about x+1? Indeed, why not $x \pm 1$, even $x \pm k$?" Naturally, then, the quest for a more believable, acceptable score evolves into intervals. For some value, k, an interval is perceived as possessing inherently greater mathematical or believability. The next logical question is then, how big is k?

Out of a possible range of [0,100], interval score assignments of $x \pm 5$ reflect a relatively high-level of certainty of the "degree of..." or "acceptability of..." the particular quality measured by the indicator. This is essentially the same as assigning an integer value from a range of [0,10]. In the literature, discussions abound regarding how precisely a person can discern a distinction between items. Saaty's research in applying the AHP arrived at nine (9) items as a reasonable upper bound [Saaty 80].

We believe technical domain experts conducting design evaluation possess knowledge enough to discern differences of at least one in ten. We acknowledge such scores are acceptable only when assigned by evaluators who possess a great deal of knowledge and experience in the technical subdomain. We propose a brief, reasonable

list of factors to consider when gauging an acceptable interval width for subjective scoring.

- the "measurability" inherent in the design quality represented by the indicator
- the global weight of the indicator (product of local weights of ancestry)
- the relative weight of the indicator (local weight times number in sibling group)
- a large number of encounters with the indicator or design quality to compare with
- partial objective bases, such as number, presence, or absence of other quality indicators
- operational or mission-specific conditions under which the design quality is evaluated

Intuitively, the precision of a score interval should be consistent with the degree of satisfaction of the above criteria, and others. Rules in the knowledge base are to be created by technical experts to define guidance for assigning meaningful, believable, interval scores. The rules may relate interval widths, upper and lower bounds, etc., with respect to these and similar criteria. Such rules may be specific to an entire subdomain or an individual indicator. These rules are applied to give feedback regarding the credibility of indicator scores. This aspect of the evaluation is described in Chapter 7.

We have not explicitly incorporated the characteristic score desirability regions in a discussion of assigning indicator scores. We have not done so for the following reasons: (1) it is our intention in this scoring scenario that evaluators make a subjective assessment of the indicator's value on its own merit, and explicitly reckon with its desirability later in the evaluation; (2) since the evaluators are also contributors to the definition of the regions, they are already aware of at least the approximate range bounds. In sum, we desire that each indicator be assigned an interval score derived solely from expertise-based judgment, and without reference to a previously established bound denoting score desirability.

6.3.3 Combining Scores from Multiple Evaluators

In this section, we provide procedures for combining interval scores from several evaluators. We require the evaluator emphasis coefficients described in Chapter 4. We address combining scores where both an upper bound and lower bound are provided by

all evaluators, as well as the case when only a single bound can be provided by one or more evaluators.

6.3.3.1 Scores with Completely Specified Intervals

The procedure for combining interval scores is an adaptation of the techniques used for metrics-based scores. That is, we take a weighted arithmetic mean of the individual interval low end or high end scores using the evaluator emphasis coefficients as weights. The arithmetic is an extension of standard interval mathematics found in [Klir and Yuan 95]. The weighted arithmetic mean for interval values for an indicator, *i*, is as in Equation 6.1,

$$\left[\sum_{j=1}^{n} w_{ij} l_{ij}, \sum_{j=1}^{n} w_{ij} u_{ij}\right] \tag{6.1}$$

where:

n: number of evaluators who assign an interval score to the indicator;
l_{ij}: the lower bound of the interval from evaluator j for indicator i;
u_{ij}: the upper bound of the interval from evaluator j for indicator i;
w_{ij}: the evaluator emphasis coefficient for evaluator j for indicator i.

We now provide an example. Consider the intervals represented by line segments in Figure 6.1. The lower and upper bounds on the interval are positioned above the respective line segment ends. Each interval is assigned by a different evaluator. The evaluator emphasis coefficient for the evaluator appears in parentheses to the left of each interval. The weighted mean interval score is computed using Equation 6.1 as follows and appears in a darker tone at the bottom of the figure.

[(0.25*60+0.37*65+0.18*61+0.20*65), (0.25*80+0.37*75+0.18*72+0.20*84)]
=[63.03,77.51].

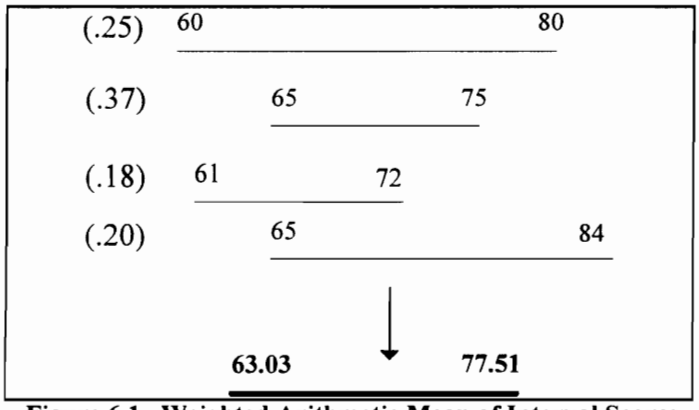


Figure 6.1. Weighted Arithmetic Mean of Interval Scores.

6.3.3.2 Scores with Incompletely Specified Intervals

Complete intervals need to be assigned by the evaluators. Our procedure is robust enough to handle the case where one or more evaluators are able to give only an upper or lower bound, but not the complete interval. For example, suppose the fourth evaluator offers only a lower bound of 65, and not the upper bound of 84 as depicted in Figure 6.1. The mean interval would be computed as in Equation 6.1, but with n=4 for the lower bound, and n=3 for the upper bound. However, now the emphasis coefficients no longer sum to unity.

There are several means to adjust the emphasis coefficients. Using the AHP, we would require a re-assessment of the pairwise comparisons for each remaining evaluator. The comparisons would, of course, be with respect to the original criteria used to determine the current emphasis coefficients. However, there are drawbacks to this method.

The literature on the AHP describes the phenomenon of "rank reversal" which can occur when a member is added or subtracted to a set of elements. Rank reversal implies, e.g., that if evaluator A had the highest emphasis coefficient originally, another evaluator, say, B, could be computed to have the highest coefficient after evaluator D has been removed. Debate continues regarding the likelihood and consequences of this phenomenon. The discussions in [Forman 93; Saaty and Vargas 84] provide mathematical explanations and counter scenarios. As a final point, in the event an

evaluator is *added* to the group, re-accomplishing the pairwise comparisons is required. If there is the removal of an evaluator, we offer a shortcut approximation to the exact solution.

For quick and practical application, we propose an intuitive and much more simple adjustment process. In the simple method, we normalize the coefficients of the remaining evaluators with respect to the sum of their coefficients. In this example the sum is 0.80. Normalizing each remaining evaluator's coefficient with respect to the sum yields, 0.25/0.80 = 0.31, 0.37/0.80 = 0.46, and 0.18/0.80 = 0.23. These new coefficients sum to unity as expected and required. Using them to compute a new upper bound, we get a value of approximately 76.

The new upper bound is down from the original mean of 78. This is not surprising, since evaluator 4's original upper bound was higher than 78, and has now essentially been given a weight of zero. Also, evaluator 2's upper bound was less than 78, and now receives an increased emphasis coefficient in determining the new mean. The net result is a reduced upper bound on the interval. The mean lower bound remains the same.

At this point in the scoring process, the assigned indicator score would remain an interval. The procedure for collapsing the interval to obtain a discrete value is discussed in the score aggregation section, Section 6.6.

6.4. SCORES BASED ON OPERATIONAL DOMAIN EXPERT JUDGMENT

Many similarities exist in the application of operational domain expert knowledge and technical domain expert knowledge. In this section we take advantage of existing similarities and replace redundant explanations with references to the previous discussion. However, there are some aspects of applying operational domain expert judgment which are distinct from the previous technique. As applicable, we make the differences clear.

6.4.1 Determining When to Apply Operational Domain Expert Knowledge

Similarly to the guidelines regarding the application of technical domain expert knowledge, operational domain expert judgment should be directly applied only after more objective means have been exhausted. Also, the kinds of indicators which are scored based solely on operational expert judgment include those containing "degree of..." and "acceptability of...," as before. The need which remains, then, is a means of discriminating between a requirement for knowledge from technical domain or operational domain experts.

We preface a discussion of the distinctions between what is "technical domain expert knowledge" and what is "operational domain expert knowledge" with the caveat that the distinction is not in all cases a solid line. It does not require much of a stretch into either the memory or imagination to conceive of an aerospace engineer becoming an astronaut. This one person, by nature of background and occupation, becomes both a technical domain expert in aerospace engineering, and an operational domain expert in the navigation and/or piloting of space vehicles. Consequently, an evaluation for an indicator from this person may be of either type of expert knowledge. In sum, the search for distinctions between the two types of knowledge need not necessarily be exhaustive nor exclusionary.

There are some attributes of an indicator which do make it a high-probability candidate for operational domain expertise-based evaluation. Perhaps a key distinction which can be drawn involves the depth to which a technical domain-based indicator is *embedded* in a particular operational mission. For example, a clinical psychologist can be expected to know a great deal about the effects of an individual's personality traits on his or her ability to perform team-oriented tasks. This is essentially single-domain expertise. In contrast, one would more likely require the judgment of a career NASA behavioral psychologist, a "shipboard shrink," or a career industrial psychologist to evaluate some of the mission impacts of human-human, human-machine, and human-software characteristics of a complex system design. Again, the distinguishing characteristic is the

degree to which technical domain expertise is embedded into a particular operational mission.

Another distinction is the *precision* with which qualitative judgments can be made and accepted as *meaningful*. To draw this point, we briefly review a similar discussion from Section 6.3 regarding this topic. As an example, one may expect an expert software engineer to know with appreciable precision the "acceptability of module cohesion." Such an assessment would be based on experiences with numerous software systems. The expert may take into consideration, e.g., the maintainability of code with a given degree of module cohesion. A sizable experience base would allow for (mental or literal) relative comparisons to previously encountered designs. These comparisons could be expected to yield a credible assessment in the form of an interval of 10 points out of a hundred.

In contrast, it is not so reasonable to expect that even a veteran industrial psychologist could assign a score of , e.g., [65,75], as a meaningful measure of the "acceptability of operator endurance demands," especially where unique operational scenarios are expected. There are fewer case studies on the basis of which to make a comparison, or fewer related indicators whose presence or absence could serve as guides in subjectively assigning a score.

Still, operational domain expertise is essential to complex system design evaluation. In complex system design evaluation, we encounter a subset of indicators for which only a career of operational mission experiences can provide *any* meaningful measure of quality, acceptability, degree, etc. And that measurement must be mapped to a quantitative form for use in the methodology. We show in the next section how this characteristic, i.e., the relative imprecision of operational domain expert knowledge impacts the scoring procedure for this category of indicators

6.4.2 Applying Operational Expert Knowledge to Score Assignment

The general procedure for applying operational domain expertise-based knowledge to assigning indicator scoring differs from that used for technical domain expert knowledge. Application of operational domain knowledge more directly applies fuzzy mathematical techniques. In saying that, we note that interval mathematics as applied in Section 6.3 is a form of fuzzy mathematics. However, in this section we introduce the application of fuzzy linguistic terms. Evaluators assign scores as *terms* instead of explicit numerical intervals.

6.4.2.1 A Quick Review of Fuzzy Concepts

We begin with a brief review of the basic concepts. Greater detail is provided in the literature survey of Chapter 2, and the overview of Chapter 3. A *linguistic term* is a word which stands for an interval of values, such as the terms "rejection" or "excellent" which we use to represent the desirability of a range of indicator scores. In the fuzzy mathematical literature, a *fuzzy* linguistic term represents a fuzzy set. Associated with a fuzzy set is a collection of members of the set and a measure of their degree of *membership* in the set. Membership ranges from [0,1] where 0 indicates an element is absolutely not in the set and 1 indicates that an element is absolutely, completely in the set.

Consider the fuzzy set of "days of the weekend." The elements "Saturday" and "Sunday" have a membership of 1. "Wednesday" has a membership of 0. "Monday," "Tuesday," "Thursday," and "Friday" have a membership in the range [0,1]. Table 6.4 depicts a membership distribution for those who like to start the weekend early.

6.4.2.2 Deriving Membership Functions for Fuzzy Linguistic Terms

Now, how do we apply fuzzy linguistic terms to scoring operational domain-related indicators? We can make almost direct use at this point of the subintervals established during the indicator definition phase, described in Chapter 4. We refer to these intervals as characteristic regions, representing the *desirability* of a given score. Evaluators establish these subintervals in the range [0,100] to represent the bounds for a "rejection" score, "acceptable score," etc. These terms can now be used to characterize an indicator's qualitative value based on the desirability of the assessed score. In this way, evaluators avoid directly assigning an actual subrange of values.

We discuss in Section 6.4.1 the inamenability of operational domain expert knowledge-based scores to relatively narrow numerical ranges. To make use of the labeled, pre-defined numerical subranges defined in Chapter 4, we must convert them to fuzzy linguistic sets. Mathematically, a fuzzy membership function for each characteristic subrange is required to define the membership of each value from [0,100] included in a fuzzy subrange. We offer an implementation method which is easy on the evaluators, and takes advantage of work already done.

To streamline the process, we make use of the intervals established during the indicator identification phase. We "fuzzify" the existing intervals by a given factor, δ. δ is derived from the appropriate degree of fuzzification based on, e.g., the operational scenario. To arrive at δ, we choose a fuzzification percentage, ν, of the total range width, e.g., 15 percent. Then, δ is computed as in Equation 6.2. Equations 6.3-6.5 can then be

Table 6.4. Membership of Days of the Week in the Fuzzy Set "Days of the Weekend."

Day	Membership
Monday	0.4
Tuesday	0.2
Wednesday	0
Thursday	0.6
Friday	0.8
Saturday	1.0
Sunday	1.0

used to generate the membership function for a fuzzification of any interval [a,b] using δ. where ν is a percentage determined by the evaluating organization. For an intermediate interval, as in Figure 6.2, whose upper and lower tail are not coincident with the extremes

$$\delta = \nu(b-a) \tag{6.2}$$

of the overall score range, membership is computed as in Equation 6.3.

$$membership(x) = \mu(x) = \begin{cases} \dfrac{x-(a-\delta)}{2\delta}; & a-\delta \le x \le a+\delta \\ 1; & a+\delta < x \le b-\delta \\ \dfrac{(b+\delta)-x}{2\delta}; & b-\delta < x \le b+\delta \end{cases} \tag{6.3}$$

For the lowest interval, e.g., "rejection," whose low end, *a*, is coincident with the lowest end of the overall score range, as in Figure 6.3, the expression becomes:

$$membership(x) = \mu(x) = \begin{cases} 1; & a < x \le b-\delta \\ \dfrac{(b+\delta)-x}{2\delta}; & b-\delta < x \le b+\delta \end{cases} \tag{6.4}$$

For the highest interval, e.g., "exceptional," whose upper end, *b*, is coincident with the highest end of the overall score range, as in Figure 6.4, the expression becomes:

$$membership(x) = \mu(x) = \begin{cases} \dfrac{x-(a-\delta)}{2\delta}; & a-\delta \le x \le a+\delta \\ 1; & a+\delta < x \le b \end{cases} \tag{6.5}$$

The effects of fuzzification of an interior, lower, and upper interval by application of Equations 6.3-6.5, respectively, are depicted in Figures 6.2-6.4. As an example,

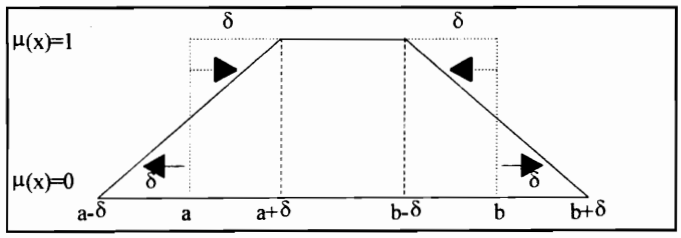


Figure 6.2. Effect of Internal Interval Fuzzification by δ.

consider the score desirability subranges for a given indicator. For illustrative purposes, let the original region boundaries be as depicted in Table 6.5, column 2. Using ν=0.15,

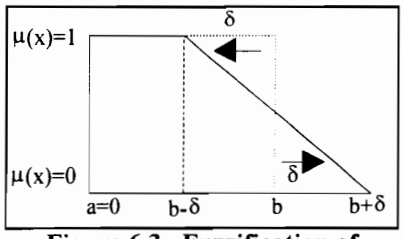


Figure 6.3. Fuzzification of Lowest Subrange.

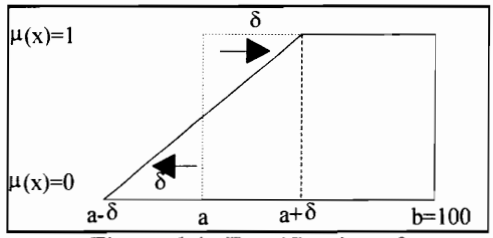


Figure 6.4. Fuzzification of Highest Subrange.

and Equations 6.2-6.5, the δ's to be applied are listed in column 3. Column 4 shows the new range boundaries. The resultant fuzzy linguistic membership graphs are shown in Figure 6.5.

In Figure 6.5, the top graph is the membership graph for the original ranges. Originally, every entry within the bounds of an interval has a membership of 1. That is, a score of 2 is equally as much a "reject" score as is a score of 19. After the fuzzification process, memberships vary within the regions, as depicted in the bottom graph in the figure. For example, the membership of 2 in "reject" is still 1, but the membership of 19 is now 0.67. Also, as a consequence of interval overlap, a score of 19 now has a membership of 0.33 in the "marginal" region. The knowledge base described in Chapter 7 makes use of memberships in multiple regions as a means of advising evaluators on

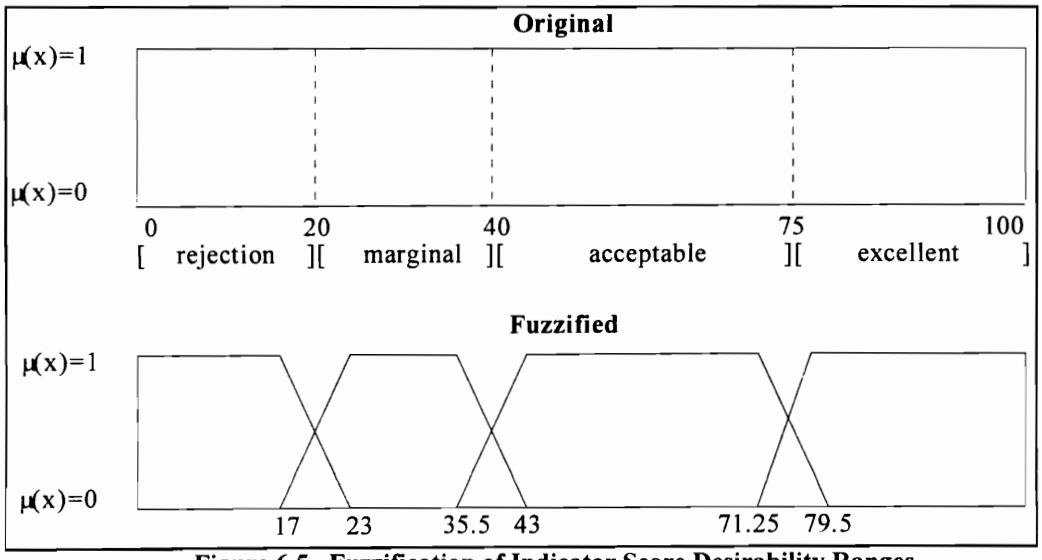


Figure 6.5. Fuzzification of Indicator Score Desirability Ranges.

Table 6.5. Fuzzification of Characteristic Subranges.

Linguistic Term	Original Range	δ	New Range
rejection	[0,20]	3	[0,23]
marginal	(20,40]	3	[17,43]
acceptable	(40,75]	4.5	[36, 80]
excellent	(75,100]	3.75	[71,100]

interpreting the evaluation results.

6.4.2.3 Assigning Fuzzy Linguistic Scores

The background provided in the previous two subsections makes the mechanics of actually assigning scores relatively simple. The critical parts are the determination of the characteristic subranges and the assessment of the goodness of the quality measured by the indicator. The first task is accomplished in the conduct of the identification of indicators, early in the design evaluation process. The second task requires the judgment, heuristics, opinion, and expertise gained during a career of operational domain experiences.

Mechanically, at this point, the evaluators need only assess and record the "score" for an indicator. The score for an indicator of the type addressed in this section is the degree of *attainment*, by the design aspect being evaluated, of the characteristic represented by the indicator. In the case of an indicator for which lower scores are more desirable, the score represents the degree of *avoidance*, by the design aspect being evaluated, of the characteristic represented by the indicator.

After the degree of attainment (or avoidance) has been assessed, it is associated with an appropriate linguistic term, e.g., "acceptable," and entered into the KBESD. The KBESD handles the mathematics of membership computation and several knowledge-based interpretations of the score.

6.4.3 Combining Scores from Multiple Experts

We mention at the start of Section 6.4 that similarities exist between some aspects of scoring by technical domain knowledge and scoring by operational domain knowledge. The score combination technique is the most similar aspect. In short, we follow Equation

6.1. What difference there is, arises due to the relatively larger interval widths associated with the fuzzy linguistic terms. Recall in the technique of Section 6.3, evaluators assign relatively narrow intervals, with great opportunity for overlap among them. Each interval could be individually defined by each expert. However, in dealing with pre-established linguistic intervals, the assigned score ranges are fixed, thus demonstrate little overlap, and are relatively wide. As a consequence, the aggregate interval can be quite wide.

This is most obvious when among a group of experts, the same indicator's score is represented by more than two linguistic terms. As an example, consider an indicator whose degree of attainment of the property it measures is determined by four experts as depicted in Table 6.6. The figures in the table are based on the contents of Figure 6.5 and Table 6.5. Applying Equation 6.1 to the last column in the table, the averaged interval is [37, 75], with the results rounded to the nearest integer.

The resultant range is quite wide, spanning nearly 40 units. This is a consequence of the imprecision with which the scores of this type of indicator can be meaningfully assigned. Ultimately, this is a consequence of the subjective nature of the design aspects measured by the indicators scored in this manner.

6.5. SCORES BASED ON SIMULATION OF SYSTEM BEHAVIOR

Simulation of any system relies on a representative and accurate model of the system or problem being studied [Balci and Nance 85, 87]. A model is a representation of the system of interest. In complex system design evaluation, we deal with systems of a very large scale, diverse composition, and critical operational nature. Simulation of the dynamic, operationally-dependent behaviors of the system allows evaluators to assess indicators relating to system behavioral and performance requirements.

Table 6.6. Values for Combination of Fuzzy Linguistic Ranges.

Evaluator	Emphasis	Score	[RangeMin, RangeMax]
A	0.25	Marginal	[17, 43]
B	0.37	Acceptable	[36, 80]
C	0.18	Acceptable	[36, 80]
D	0.20	Excellent	[71, 100]

6.5.1 Determining When to Apply Simulation

Indicators whose scores must be based on simulation are those which measure dynamic, operationally-dependent, scenario-based aspects of the designed system. These are system aspects which cannot be measured statically by examining the applicable portions of the design. Simulations of system behavior under varying operational scenarios must be used to gauge a design's probability of meeting system operational requirements.

To capture complex interplay between humans and system components in various operational scenarios, we move into an arena not far removed from wargaming. In these scenarios, the simulation model represents the behavior of the system hardware and software. Realistically modeling human logic and behavior, especially under time-critical, high-threat scenarios, requires sophistication not yet achievable by models. The entire field of human workload modeling is dedicated to this very task [Armando and Bagnara 92; Ballas *et al.* 92; Hollnagel 92; Lee and Moray 92; O'Hara and Hall 92; Rueb *et al.* 92; Sarno and Wickens 92; Ujita 92; Whitaker *et al.* 92; Peters 91; Becker *et al.* 91; Blackman 91; Lemay and Comstock 90; Stassen *et al.* 90; Weisgerber and Kibbe 90; Woods 90; Zinser and Hinneman 88]. To build in dynamic realism, the human element is incorporated live, in real-time, as a physical man-in-the-loop.

To gain usable results, we require networks of models, even geographically distributed models [Nance and Page 94]. An example system design aspect which requires this level of dynamic modeling would include the design's accounting for the *survivability* of the designed system, e.g., a seagoing vessel. Such a system must operate in numerous environmental conditions, in peace time, under attack by any number of weapon system, with dynamically varying resource availabilities, in a range of operational scenarios from dry-docked to sinking in flames. A complex hierarchy of models can be created to represent system behavior combinations at different levels, including the status of the structural hardware, software, communications, manpower, etc.

In sum, in comparison with other modeling techniques, simulation is the only solution methodology which enables us to model a complex system design at any level of detail desired. Examples of dynamic aspects of system behavior which require simulation as a basis for assessment of design quality are included in Figure 6.6.

- military vehicle (or system) mission success capability affected by:
  - personnel availability/attrition (natural and through failure-event loss)
  - personnel skill level (learning curve due to experience, task dynamics)
  - personnel performance affected by task duration, mental stress, fatigue
  - superstructure integrity/survivability under various damage conditions
  - superstructure integrity/survivability under various environmental conditions
  - communications reliability (clarity, veracity, security level enforcement)
  - logistical support/resupply under various hostile scenarios
  - system on-site maintainability (personnel, equipment, backups)
  - system survivability probabilities according to:
    - system cloaking capability
      - adaptability to dynamically varying visual environments
      - adaptability to dynamically varying acoustic environments
      - adaptability to dynamically varying thermal contrast environments
      - adaptability to dynamically varying electromagnetic environments
    - defensive systems capability
      - simultaneous intruder tracking (computer resource limitation)
      - dynamic and real time threat detection, discernment, and prioritizing
      - counter-threat ordnance re-supply (scenario dependent)
    - system maneuverability (varies by operational status)
      - propulsion capability/degradation
      - climb/dive capability/degradation
      - turn/stop capability/degradation
      - maximum sustained speed/maximum acceleration
- operator performance (e.g., in a nuclear power plant control room) affected by:
  - dynamic task loads due to operational mode (system start up, shut down, steady state)
  - dynamic task loads due to operational scenario (low manpower, core breach, toxic spill)
  - operator experience level (varies with personnel turnover)
  - operator management style (dynamically varying according to task/scenario)
  - human-software interface configuration/complexity
  - human-hardware interface configuration/complexity
  - variations in operator trust of automated system operation
  - inter-operator dynamics/communications/distractions
  - variations in on-line or on-site job aid quality
- communications and computer system performance affected by:
  - dynamic resource availability (destroyed, damaged, in maintenance, incompatible)
  - dynamic variances in availability of and demands for power supply
  - bandwidth capacity requirements varying by resource availability
  - bandwidth capacity demands varying by operational scenario (emergency, idle)
  - information retrieval response time varying by system load and operational mode
  - dynamic requirements for communications security according to operational scenario

Figure 6.6. Example Design Aspects Requiring Simulation.

6.5.2 Mapping Simulation Results to Indicator Scores

To incorporate simulation of system behavior into the design evaluation methodology, numerous simulation runs under a suite of operational scenarios, resource allotments, etc. are run. The results of experimentation with simulation models, aside from the visual animations of system behavior, may represent any of the following, in addition to others:

- response times
- thresholds
- break-even points
- throughputs
- bottlenecks
- resource utilizations
- resource availabilities
- resource drains
- process starvations
- durations
- single occurrences

While the results may represent many quantities, as above, in general they take one of relatively few forms. For example, the numerical output from experimentation with a simulation model may be discrete values (i.e., maximum, minimum, average, variance), probability distributions, or confidence intervals. Regardless of the native form on which they are produced, the results must be mapped to indicator scores in the standard range [0,100]. Simulation results can be mapped through a hybrid method, taking direct advantage of numerical results, yet requiring subjective interpretation by experts regarding the meaning, consequences, and acceptability of interval values. In this section, we describe a procedure for mapping simulation results to indicator scores.

6.5.2.1 Using Discrete Results from Simulation to Score Indicators

We describe in Section 6.2 the procedure for applying discrete values obtained from metrics to design quality indicator scoring. We provide several example mapping functions from a native range of computable scores to the range [0,100]. In some cases, the possible range for a value obtained through simulation is known or can be computed.

For example, the theoretical maximum utilization of a communications network can be computed using well-studied techniques, such as in [Bertsekas and Galagher 92]. Simulation results for minimum, mean, and maximum achievable values will fall within this theoretical range. For simulation-derived values of this nature, the same mapping technique as for metric-based scores can be applied to map values from the theoretical (or realizable) range [*a*, *b*] to [0,100]. The reader is referred to Section 6.2.2 for details.

In other cases, a discrete value can be obtained through simulation, but a theoretical range of values is either incalculable or impractical for use, e.g., [0, ∞]. In these cases, the expert evaluators with knowledge relating to the system characteristic being simulated use their judgment as a base in assigning a score. The value assigned would be either an interval value, applying the techniques from Section 6.3, or a linguistic value, applying the techniques from Section 6.4. The discriminating factors would be the same as discussed in Section 6.4. That is, whether or not a relatively narrow score interval is *believable*, and what size experience base can be used for comparison.

6.5.2.2 Using Interval Results from Simulation to Score Indicators

Common simulation results are an interval of values centered on an estimated mean value. We refer to the interval mean value as μ, and the interval extends η units to either side of μ. We use the symbol η to refer to the interval "half-width." Thus, the interval encompasses the values $[\mu-\eta, \mu+\eta]$. Associated with each interval is a *confidence level*, which expresses the probability that the actual mean value falls within the interval. As depicted in Figure 6.7, the higher the confidence level, the wider the confidence interval, and hence the wider is the interval. For example, a 95% confidence interval encompasses more possible values for the actual value, than does a 90% confidence interval. Put another way, one can intuitively be more certain that the true estimated value lies in a wider range, than in a more narrow range.

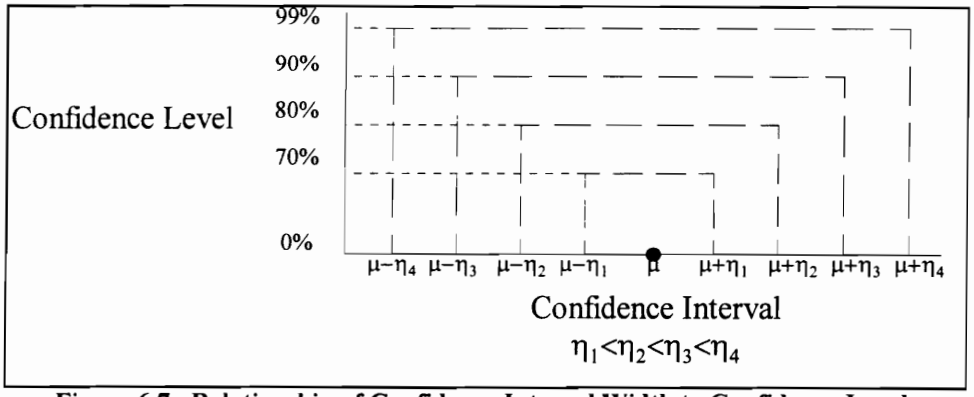


Figure 6.7. Relationship of Confidence Interval Width to Confidence Level.

Based on the simulation results, evaluators must select the confidence interval which, in their expert judgment, best represents the range of actual values which the system as designed could be expected to exhibit if developed. Contemporary statistical analysis texts, e.g., [Ott 93], recommend a confidence level in the range [.90, .95]. Confidence levels lower than 90% decrease the interval of possible values in which the true mean may be found. This increases the probability of assuming the true mean is not within the interval when it actually is. This is referred to in statistical analysis as a Type I error. On the other hand, confidence levels higher than 95% increase the interval of possible values in which the true mean may be found. This increases the probability of selecting a value in the interval which is not representative of the true mean. This is referred to in statistical analysis as a Type II error. To decrease the probability of committing either type of errors, it is recommended that confidence intervals be constructed with between 90% and 95% confidence level.

To determine the confidence interval to use, evaluators must consider that confidence intervals are arrived at by experimentation with a complex simulation model. Associated with the experimentations are possibly wide input variations due to numerous possible operational scenarios. Input data may be of course resolution. The degree of system component interaction may be over- or underestimated. These factors weigh significantly in the selection of a confidence interval. Expert evaluator judgment is thus critical to the selection of a realistic interval.

With the appropriate confidence interval (CI) selected, we require a mapping from scores within the CI to the standard range, [0,100]. The mapped value represents the *desirability* of the score. Score desirability is based on the proximity of the score to thresholds established in the system requirements specification. In the absence of a value explicitly specified in the requirements, a value derived from other specific requirements may be used. In the event this value is infeasible to obtain, we must rely on the judgment of appropriate experts, who offer a most desirable value (or interval). Hereafter, we will refer to the most desirable value(s) as the requirement. The desirability function then determines the acceptability of estimated values with respect to the requirement.

To effect the mapping, we construct or derive a function which maps any value to the range [0,100]. In this case, we choose 100 to be the most desirable score, and zero to be the least desirable. The reverse situation is easily accommodated. The desirability function should map to a score of 100, any value which meets or surpasses the requirement. Scores which do not meet the requirement are mapped to a value less than 100, according to the mapping function. An example of several mapping functions (a-d) centered on a requirement μ^* is depicted in Figure 6.8. Notice that depending on the mapping function employed, the mapped value for a single point in the interval can vary widely (see the solid dot in the figure). This places particular significance on the mapping function used. Note that functions based on intervals or one-tailed intervals may also be used. Examples of these function types are presented in Figures 6.9-6.11.

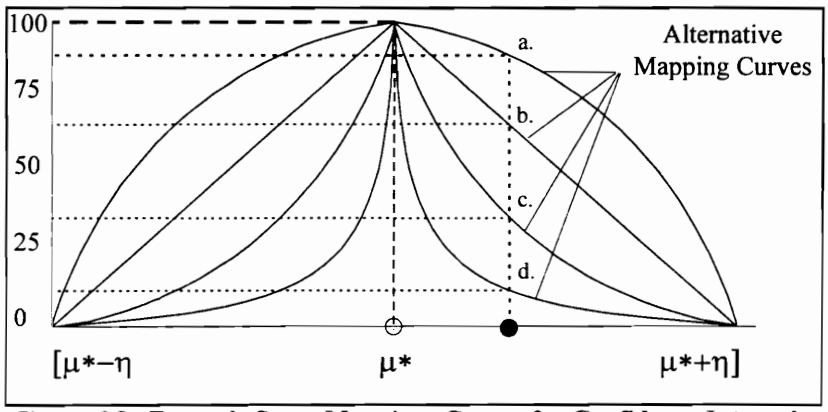


Figure 6.8. Example Score Mappings Curves for Confidence Intervals.

We pictorially illustrate how mapping functions are used in Figures 6.9-6.11. In the figures, we overlay different mapping function forms with a hypothetical CI, $[\mu-\eta,\mu+\eta]$. In each figure, the bounds of the CI are not coincident with the bounds of the requirement-based mapping function. Each figure depicts an offset, δ, of the estimated mean, floor, or ceiling from the required value. This is why a mapping is needed. In practice we may not expect the design to yield values exactly as specified in the requirements. However, we still require a means of arriving at a score in [0,100] for every score in the estimated interval. We show in Section 6.6 how depending on the *evaluation posture* taken, any value in the CI may be selected as the assumed true value.

In Figure 6.9, a trapezoidal function is shown. A trapezoidal mapping is needed when the requirement calls for an *interval* of most desirable values, centered on a mean

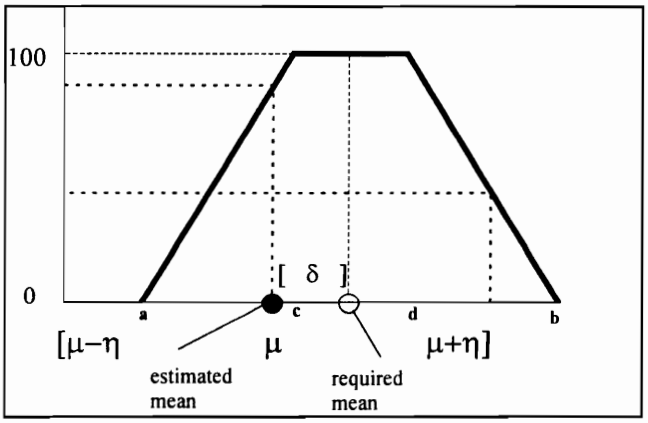


Figure 6.9. Example Trapezoidal Mapping Function.

value. Values outside the ideal interval are accepted with reduced desirability, as reflected in the slope of the trapezoid's legs. For the CI shown, the upper bound of the interval maps to a desirability score of near 50. The mean maps to a value near 90, and the lower bound to zero. By inspection, values just above the interval median map to the most desirable scores.

Mathematically, the mapping function for the trapezoidal map is as shown in Equation 6.6. In the equation, we make no assumption of symmetry with regard to the requirement function's variance about the mean. To be as general as possible, we refer to the four values on the abscissa which define the characteristic points of the trapezoid as **a**, **b**, **c**, and **d,** where **a** < **c** < **d** < **b**. On the abscissa, **a** is the lower bound, mapping to a score of zero. Respectively, **c** and **d** define the lower and upper bounds of the native scores which map to a score of 100. The abscissa value **b** marks the upper bound, which

$$s(x) = \begin{cases} 0, & x < a \\ 100 * \dfrac{x-a}{c-a}, & a \le x \le c \\ 100, & c \le x \le d \\ 100 * \dfrac{x-c}{b-c}, & d \le x \le b \\ 0, & x > b \end{cases} \tag{6.6}$$

maps to zero.

Figure 6.10 is a one-sided mapping function. This function type is needed when the requirement calls for a *ceiling*. Values below the ceiling are desirable with a score of 100. Values above the ceiling are accepted with less desirability, as depicted in the alternative mapping curves i-iii. Also, if no value above the ceiling is acceptable, then

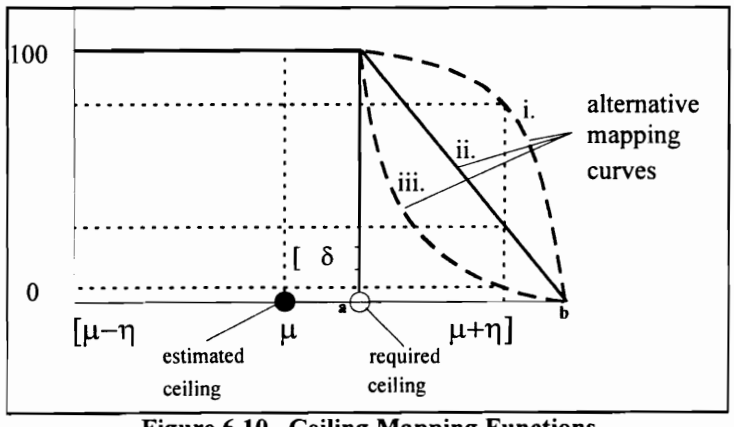


Figure 6.10. Ceiling Mapping Functions.

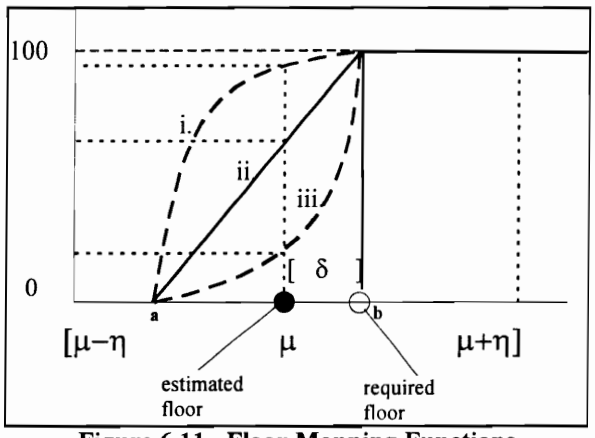


Figure 6.11. Floor Mapping Functions.

the mapping function resembles a step function, mapping all such values to a desirability of zero.

In Figure 6.10, the lower bound and the mean of the CI fall below the ceiling, so they both map to a score of 100. By inspection, the upper bound of the CI maps to desirability scores of approximately 80, 30, and 5 for alternative curves i, ii, and iii, respectively. Mathematically, the mapping function for the linear case (curve ii) is shown in Equation 6.7.

For a floor requirement, a set of example mappings are shown in Figure 6.11. In the figure, the upper bound of the CI is above the floor, and maps to a desirability score

$$s(x) = \begin{cases} 100, & x < a \\ 100 * \dfrac{b-x}{b-a}, & a \le x < b \\ 0, & x \ge b \end{cases} \tag{6.7}$$

of 100. The mean of the CI, μ, falls below the required floor. It maps to the approximate scores, 95, 70, and 20, using the mapping curves i, ii, and iii, respectively. The lower bound is below the minimum acceptable value according to any mapping shown. Thus, it maps to a score of zero. The actual mapping function for curve ii is Equation 6.8.

$$s(x) = \begin{cases} 0, & x < a \\ 100 * \dfrac{b-x}{b-a}, & a \le x < b \\ 100, & x \ge b \end{cases} \tag{6.8}$$

6.6. SCORE AGGREGATION

We now describe the process of aggregating scores assigned by the methods described in Sections 6.2-6.5. In general, we propose taking a linear combination of the weighted scores within an indicator sibling group. The resultant value is the aggregate score for the parent indicator. This method is recursed up the hierarchy to obtain an aggregate score for the root node. For cases where discrete scores have already been determined, as with metric-based scores, this combination process is straightforward. However, for interval and especially linguistic scores, the process requires additional actions, which are addressed in subsequent sections.

Additionally, reflected in both Figure 6.12 and Equation 6.9 is the possibility that

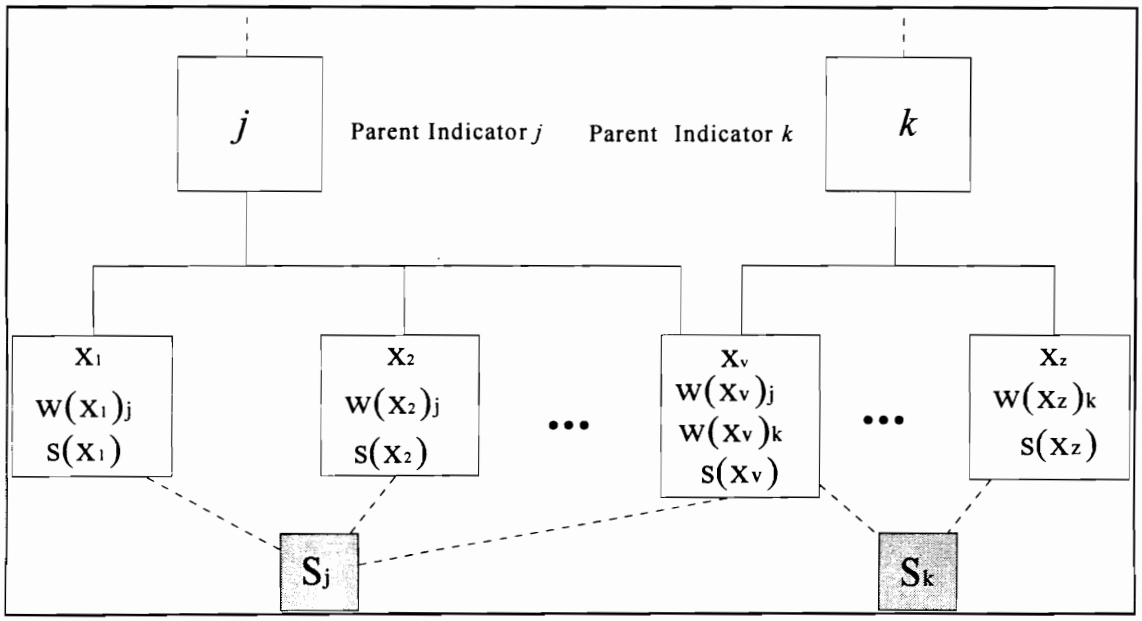


Figure 6.12. Aggregating Indicators through Weights and Scores.

a given indicator may be the child of more than one parent. From the discussion in Chapter 4 regarding the construction of the indicator hierarchy, this condition should be common in the hierarchy. A perhaps overlooked consequence is that an indicator is scored only once, but it is assigned a weight for each parent.

6.6.1 Aggregating Discrete Scores

The most simple score aggregation case is for an indicator sibling group in which each indicator has been assigned a discrete score in the range [0,100]. For "mixed" sibling groups, a combination of the techniques discussed in this and subsequent sections is required. From the techniques of Chapter 5, each indicator, x_i, has been assigned a weight, $w(x_i)_j$, reflecting its relative priority among all siblings which are child nodes of parent indicator j. Each indicator has also been assigned a score $s(x_i)$ which is a measure of its "performance" with respect to the design objectives. The aggregation function is as shown in Equation 6.9.

$$s(x_j) = \sum_{i=1}^{z} w(x_i)_j s(x_i) \tag{6.9}$$

6.6.2 Aggregating Interval Scores

In the foregoing, we describe three types of interval scores:

- direct assignment of intervals in the range [0,100];
- direct assignment of confidence intervals in the native score range; and
- implicit assignment of intervals in the range [0,100] by linguistic term.

In this section, we describe an aggregation procedure for indicators of the first two types. The aggregation of linguistic intervals is described in Section 6.6.4.

Making use of interval scores requires extracting a representative discrete value from the interval. This value represents the *essence* of the range, particular to the evaluation *posture* taken by the evaluating organization. Example postures include the search for worst case, best case, median case, or some case in between. This posturing

takes the form of *what-if* analysis, applying the essence extraction technique in Equation 6.10. The factor $k \in [0,1]$ is a measure of the degree of conservativeness applied to extracting the essence. Conservativeness here implies the intention of preferring a false rejection of a score (type I error) as opposed to a false acceptance (type II error).

$$essence(k,[a,b]) = (1-k)a + kb \tag{6.10}$$

For example, a value of k=0 would extract the lowest value in the range. A value of k=1, would extract the maximum value in the range. A value of k in between would extract essentially a weighted average of the extremes, with weight k applied to the maximum value, and 1-k applied to the minimum.

As an example, consider the interval scores from Figure 6.1, (p. 6-14) and reproduced in Table 6.7. Using a range of evaluation postures, reflected as values of k, the Table shows the corresponding weighted average interval essence scores in the last row.

Table 6.7. Example of Weighted Average Interval Essence Extraction.

					Essence		
Evaluator Emphasis Coefficient	**Lower Bound**	**Upper Bound**	**k=1**	**k=0.75**	**k=0.5**	**k=0.25**	**k=0**
0.25	60	80	80.00	75.00	70.00	65.00	60.00
0.37	65	75	75.00	72.50	70.00	67.50	65.00
0.18	61	72	72.00	69.25	66.50	63.75	61.00
0.20	65	84	84.00	79.25	74.50	69.75	65.00
Weighted Average Essence			**77.51**	**73.89**	**70.27**	**66.65**	**63.03**

The same essence extraction technique may also be applied to intervals which are not yet mapped to the range [0,100]. In this case, we perform the extraction using k on the native range of scores, i.e., the base of the native triangle when superimposed on the "ideal range." We then apply an appropriate mapping function, like the one in Equations 6.6-6.8. The result is a discrete value in [0,100]. After extracting the interval essences

appropriate to the evaluation posture, let us assume that all scores in a sibling group now have a discrete value. The technique from Section 6.6.1 is now directly applicable. If for some indicators this is not yet the case, then techniques discussed in subsequent sections can be used.

6.6.3 Aggregating Linguistic Scores

The final score form which must be dealt with in the aggregation phase is the linguistic term score. An indicator scored in this manner, e.g., as "acceptable" has associated with it a predefined range of values. The values were determined uniquely for the indicator according to the process described in Chapter 4. Actual assignment of the term(s) is discussed in Section 6.4.2.

These scores are combined exactly the same way as directly assigned intervals. A k is selected respective of an evaluation posture. Using k as in Equation 6.10, the interval is collapsed to a single value (its essence) in [0,100]. The only difference here is that the essence of the interval, while only a single value, may have a membership in two adjacent linguistic term scores, e.g., "Marginal" and "Acceptable."

As an example, we take the fuzzified intervals from Figure 6.5 and Tables 6.5 and 6.6. The applicable contents of Tables 6.5 and 6.6 appear in the first six columns of Table 6.8, reproduced here for reference. We also make use of the membership functions in Equations 6.3-6.5, as necessary. Applying a k factor of 0.5, interval essence scores are as in column 7 of the table. By comparison, using $k = 0.1$, the results in column 8 are obtained.

First, the essence scores from k=0.5 are combined taking a weighted average, using the evaluator emphasis coefficients. The resultant mean interval essence is computed as:
[(0.25*30)+(0.37*57.5)+(0.18*57.5)+(0.20*85.6)] = 56.2. From column 6 in Table 6.8, the resultant value falls into the range "Acceptable" and no others. Using Equation 6.3,

Table 6.8. Example of Collapsing Weighted Average Linguistic Interval Scores.

Evaluator	EEC	Score	Original [Min, Max]	δ	Fuzzified [Min, Max]	Essence (k = 0.5)	Essence (k = 0.1)
A	0.25	Marginal	(20, 40]	3	(17, 43]	**30**	**19.6**
B	0.37	Acceptable	(40, 75]	4.5	(35.5, 79.5]	**57.5**	**39.9**
C	0.18	Acceptable	(40, 75]	4.5	(35.5, 79.5]	**57.5**	**39.9**
D	0.20	Excellent	(75, 100]	3.75	(71.25, 100]	**85.6**	**74.1**

with a=36, b=80, and δ=4.5, the value 56.2 falls between a+δ (40+4.5=44.5) and b-δ (75-4.5=70.5). Its membership in "Acceptable" is then 1.0, by Equation 6.5.

For comparison, we take the essence scores from column 8 of the table, where k=0.1. Using the same technique as above, we obtain a weighted mean essence of 41.7. This value falls into two linguistic score ranges: "Marginal" and "Acceptable." To compute the membership in each fuzzy interval, we first need to know where in each interval the score falls. For "Marginal, "given the δ and original bounds for each interval, 41.7 falls in the upper tail of "Marginal," between b-δ (40-3=37) and b+ δ (40+3=43), so by Equation 6.3:

$\mu_{Marginal}(41.7) = (43-41.7)/6 = 1.3/6 = .22.$

For membership in "Acceptable," we observe 41.7 falls in the lower tail, between a-δ (40-4.5=35.5) and a+δ (40+4.5=44.5). Similarly, by Equation 6.3,

$\mu_{Acceptable}(41.7) = (41.7-35.5)/2= 5.2/6 = 0.87.$

Now that we have determined the "value" of the indicator under various evaluation postures, we need to determine how good it is. The expert evaluators are the ones who ultimately make the decision of the score's value. We discuss a knowledge-based framework for detecting and accounting for dual-membership score in Chapter 7.

6.6.4 Aggregating All Scores in an Indicator Hierarchy

Every base-level indicator in the hierarchy is scored based on one of the score types discussed in Sections 6.2-6.5. In sections 6.6.1-6.6.4, we describe the means of mapping any score to a discrete value, so Equation 6.9 may be applied to yield an

aggregate score for a parent node. The aggregate score for the entire hierarchy can be obtained by executing this procedure in a bottom-up fashion to the root node of the hierarchy.

This procedure is simply the mathematical mechanics of the scoring and aggregating process. The *interpretation* of the partial and final scores is of greater interest to design evaluation. Intuitively, in a hierarchy of non-trivial depth, the aggregate score at the root tends away from the maximum possible score, i.e., 100. This is clear from the simple observance that if any of the base-level indicators is scored less than 100, the root score could not possibly reach 100. This observation raises other issues regarding what results of the score aggregation are practical or even attainable. Specifically, what factors drive the maximum, minimum, median, etc. scores attainable for the hierarchy? Contributing factors described previously are summarized in Table

Table 6.9. Factors Affecting Aggregate Hierarchy Score.

Factor	Score Aspect Affected
Evaluation posture (k factor)	• Maximum score from interval • Minimum score from interval • Bias direction of intermediate score from interval
Evaluator emphasis weights	• Interval extremes • Discrete score arithmetic mean
Individual indicator weights	• Parent score arithmetic mean
Fuzzification factor	• Linguistic membership functions • Linguistic interval extremes • Linguistic interval overlap
Interval score range widths	• Maximum possible aggregate score • Minimum possible aggregate score • Bias degree of intermediate score from interval
Score mapping functions	• Bias direction and degree for discrete score from metrics • Bias direction and degree for mapped confidence intervals
Individual discrete scores	• Parent indicator score value
Simulation input/output resolution	• Confidence interval widths • Confidence interval mapping function widths
Metric resolution	• Measurement discriminability/sensitivity

6.9. Their relationships are graphically depicted in Figure 6.13.

6.7. SUMMARY AND CONCLUSIONS

This chapter provides descriptions and examples of a suite of methods applicable to deriving, computing, assigning, and combining scores for design quality indicators. Specific elaborated substeps include: (1) guidelines for determining acceptability of a metric or suite of metrics for assigning an indicator score; (2) methods for mapping native metric scores to a standard score; (3) guidelines for determining a credible confidence interval about simulated results as a basis for indicator scoring; (4) mapping simulation results, both discrete scores and confidence intervals, to a standard score; (5) assigning interval scores based on technical domain expert knowledge; (6) guidelines for discriminating between the choice of directly assigned relatively narrow interval scores and broader fuzzy linguistic scores; (7) creating fuzzy linguistic score intervals to serve as

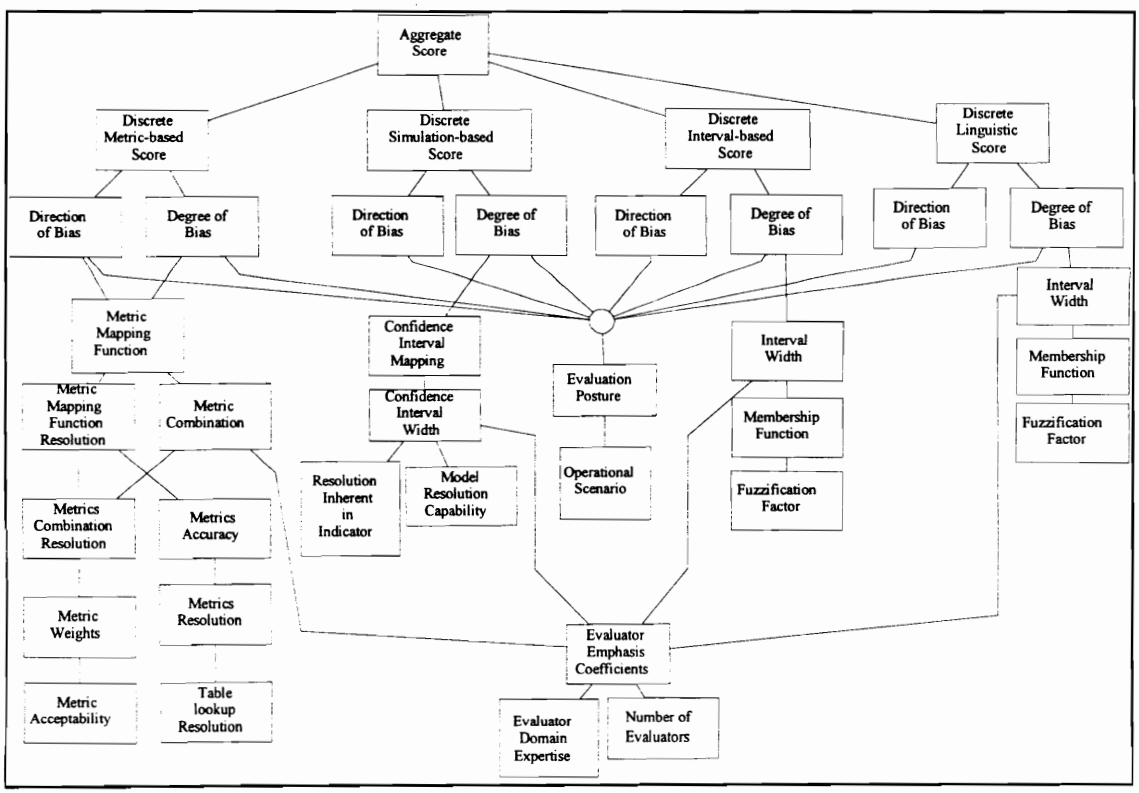


Figure 6.13. Dependence Relationships of Factors Affecting Aggregate Score.

a basis for operational domain expertise-based scores; (8) combining scores from multiple evaluators; and (9) aggregating indicator scores for an entire hierarchy.

We may draw several conclusions from the foregoing. First, any indicator can be assigned a score of essentially any native form. We require only a suitable mapping function from the native score to a form directly applicable to the methodology. Clearly, the precision of the mapped score is bounded from above by the precision with which a native score can be determined. Second, by using evaluator emphasis coefficients, scores from multiple evaluators may be combined to yield a single, weighted average score for an indicator. We note, however, that individual evaluator scores are retained for further analysis as desired. Finally, the evaluation posture taken when aggregating scores in the hierarchy makes an appreciable difference in the numerical outcome of the evaluation. By varying the k factor as a reflection of conservativeness in evaluation, bounds in the extremes of the overall score may be determined.

7. APPLICATION OF AN EXPERT KNOWLEDGE BASE TO DESIGN EVALUATION

7.1. INTRODUCTION

Assessing a complex system design's potential for meeting the requirements for which it was commissioned is the task of an independent evaluation organization. Given the complexity and size of such an assessment task, the organization requires a great deal of technically specific, and yet operationally practical expertise. To effectively apply that expertise to the conduct of a thorough, traceable, reproducible, and credible evaluation effort, a circumscribing evaluation framework is required. In Chapters 4-6 we describe in detail such a framework, comprising procedures, algorithms and techniques with which to effectively apply expertise to design evaluation.

In this chapter we present a final facet of the methodology. It serves as the glue that binds the procedures, algorithms and techniques into a cohesive, holistic design evaluation methodology. We herein describe the expert knowledge-related inputs to a knowledge base, which is provided as a component of the Knowledge-Based Evaluation of System Designs tool (KBESD). The KBESD is described in Chapters 3 and 4. This capability of the KBESD facilitates computer-assisted application of expert-generated, evaluation-oriented knowledge. Expertly employed, it contributes to a reliable, traceable, credible evaluation of a complex system design.

The chapter is organized as follows. Section 2 provides in separate major subsections a description of the application of expert knowledge to each of the subphases of: (1) indicator identification, (2) hierarchy construction, (3) indicator weighting, (4) indicator scoring, and (5) score aggregation and interpretation phases, respectively. Section 3 provides a summary of the chapter's content and draws appropriate conclusions.

7.2. APPLYING AN EXPERT KNOWLEDGE BASE TO DESIGN EVALUATION

Knowledge-based assistance is applicable to the entire methodology. In this section we describe examples and classes of its practical use for each phase of the methodology. To add further structure, we present knowledge-base contents from two perspectives. First, we motivate and describe the types of expert knowledge applicable to the particular phase. In so doing, we provide and describe contents of an appropriate expert knowledge rule base. We present examples of knowledge base facts and rules. In place of a specific syntax, as provided in, e.g., the Prolog rule-based language, we express rules in plain text. We use bold, italics, and other typographical devices to provide emphasis where needed. The knowledge-base examples provided are not applicable to any particular design class. They are examples and thought-provokers, intended to serve as templates. They are illustrative of the types of relationships which can be expressed and applied automatically using an expert knowledge base.

Second, in less detail, we enumerate other expressions of expert knowledge, in terms of interaction with and internal functions of the KBESD. They are categorized as prompters, triggers, alerters, and informers. We borrow these terms from operating systems terminology and adapt definitions from [Arthur, *et al.* 93 p. 58]. *Prompters* are opportunities for and the contents of initial or responsive inputs to the KBESD. They are generated by the KBESD as input panels, and inputs are provided by evaluators or knowledge-base authors. They are primarily facts, numbers, check-box entries, textual commentary, assertions, and rules. *Triggers* are behaviors performed automatically or on demand by the KBESD. They are computations, comparisons, and knowledge-based inferences. The results of triggers may not be revealed immediately, but may be simply the update of an internal variable. When immediately visible, they are referred to as *alerters*. Alerters signal the user of the KBESD that undesirable conditions specified in knowledge-base rules have occurred. Such conditions can occur as a result of user input, knowledge-based inferencing based on user input, or the results of triggers. Finally,

informers are our term for user query and subsequent display of KBESD contents, trigger results, or alerters which do not signal an undesirable condition.

Taken together, the two presentations of applicable expert knowledge provide a broad-based foundation for computer-assisted design evaluation. They are intended to prompt considerations and actions which ultimately enhance the results of each respective evaluation phase, and the methodology as a whole.

7.2.1 Identification of Evaluating Panel

Facts, either simple or complex, pertaining to each expert, must be supplied initially to the knowledge base. Example facts include the number of years each worked in a particular technical domain. These facts may be expressed as either numerical values (e.g. "5 years") or as subjective ratings (e.g., "very high" job knowledge). Rules relate these facts, alone or in combination, to conclusions which are applicable to this methodology phase.

7.2.1.1 Types of Knowledge Applicable to Evaluation Panel Identification

In Chapter 4 we introduce the concept of evaluator emphasis coefficients (EECs). They are derived when deemed necessary to objectively distinguish relative expertise among a panel of evaluators. When needed, EECs are computed for each evaluator, relative to the other evaluators on an individual indicator (or category) basis. In general, each evaluator's indicator score inputs are treated equally. However, when significant differences in expertise exist with respect to a particular indicator (or indicator class), and the input of multiple evaluators is still desired, greater emphasis should be given to the most qualified expert. Proportionately less emphasis is given to the others. The need for objectivity and prudence in applying these factors cannot be over-emphasized. See Chapter 4 for a detailed motivation of this purely mathematical device.

When evaluator emphasis, rules in the knowledge base can be applied to ensure the EECs are not severely disparate. Example knowledge types are listed as bulleted items in the remainder of this section.

- Expert knowledge can relate attributes of the background of an expert to acceptability of his *inclusion* as an evaluator for one or more particular domain categories.

In a large evaluation organization, potentially many experts in diverse fields may assess the value of a particular indicator. For purposes of traceability and evaluation results analysis, it is desirable to characterize and report the field(s) of expertise of each evaluator, and their degree of expertise in the field. Degree of expertise can be assessed, if only subjectively, by rating each evaluator with respect to a set of criteria. These are the same criteria used to determine EECs, for example:

* total years in the field (or related field)
* size of projects (lines of code, dollars, no. of components, complexity, criticality)
* years in design (as opposed to implementation, development, production, testing)
* years in development (coding, building, as opposed to design-level work)
* years in design evaluation (process evaluation, design measurement)
* years in product testing (code testing, hardware testing, interface evaluation)
* years in project management (higher-level than just design)
* thresholds for each of the attributes

Given an evaluator's rating with respect to these or similar attributes, several rule-based quality checks can be made. For example, to characterize the expertise of each evaluator in one or more fields, rules expressing relationships similar to the following are appropriate:

> IF for the domain under consideration, the expert's total year, project sizes, years in development, years in design evaluation, years in product testing, and years in project management EXCEED their respective *thresholds* THEN the expert's credibility as an evaluator for the domain is deemed ***exceptional.***

Other rules assign lesser ratings based on lesser degrees of attainment of the background attributes. The attributes may be considered individually or in some meaningful combination. Information gained from application of these rules could allow

reporting, by indicator, distributions of the scores offered by evaluators of each category of expertise. Additionally, checks can quickly be made to ensure that each evaluator who inputs a score for an indicator possesses a minimum degree of expertise. For example:

> IF evaluator qualification for the indicator is less than ***acceptable***, THEN raise appropriate alert.

It is reasonable that a single evaluator could be assessed to have expertise in several (related) domain categories, but perhaps with more or less "credibility" in each field. In these cases, the knowledge described in this section could be used to determine in which particular domains a given evaluator is likely to be the most valuable asset to the evaluation project.

- Expert knowledge can relate the evaluator emphasis coefficients (EECs) to some minimum, maximum, or variance thresholds.

In cases where experts have widely varying expertise in a particular domain, the EECs should vary commensurably. However, one consequence of this is that one evaluator could, by virtue of his/her EEC, have a *dominating* effect on the scoring of indicators. Recall, during the indicator scoring and aggregation phase (Chapter 6), each evaluator's EEC is applied to the score he/she assigns to the indicator, yielding a weighted arithmetic mean score. With widely disparate EECs, the mean score may amount to little different from the score assigned by the single dominating evaluator. This is not a primary concern, since it may well be that the dominating expert really does have the most expertise to bring to bear on a given indicator (or subdomain). However, it does bring into question the real contribution to the evaluation of the *other* evaluator's score inputs.

A consequence of perhaps greater concern is the *dwarfing* of an evaluator due to a particularly low EEC. As a consequence of a very low EEC, an evaluator's score may be counted almost negligibly in the evaluation. Recall, each evaluator has been selected by

virtue of the background attributes described earlier. We desire to guard against, then, the mitigation of his evaluative influence due to the EECs.

In either case, dominating or dwarfing, rules are needed which catch and, if necessary, recommend steps to correct the conditions. These rules would relate perhaps the ratio of each evaluator's EEC to the "expected value" of the EEC, namely the default EEC. If an evaluator is found to have this ratio below a given value, then a dwarfing condition may be occurring. If the ratio is above a given value, stated in the rule, then a domination condition may be detected. Values of each of the background attributes, the indicator itself, or even the particular operational situation being considered, may be included in the determination of the acceptability or resolution of a dwarfing or dominating condition. The ultimate objective is to incorporate the degree of input from each evaluator which enhances the overall credibility of the evaluation.

Given the EECs and appropriate *thresholds*, example rules for assessing EEC acceptability would express relationships such as:

> IF the relative EEC for an evaluator is BELOW the negligibility threshold, THEN signal an alert indicating this (dwarfing) condition exists.
>
> IF the relative EEC for an evaluator EXCEEDS the dominating threshold, THEN signal an alert indicating this dominating condition exists.
>
> IF the variance (or dispersion) of EECs for all evaluators in a domain EXCEEDS the variance threshold, THEN signal an alert indicating this condition exists.
>
> IF (some EEC-related violation) occurs, but (other overriding factors) are satisfactory, THEN negate discovery of the condition and/or suppress any otherwise applicable alerts.

7.2.1.2 Other Expressions of Knowledge Applicable to Evaluation Panel Identification

Other expressions of knowledge relevant to this phase of the methodology include: (1) entering factual and knowledge-based entries into the KBESD; (2) numerical computations performed by the KBESD; (3) firing of productions rules (as describe above) in the expert system shell; (4) alerting the evaluating organization of undesirable

conditions (defined in the rules); and (5) displaying factual or rule-based explanations of alert conditions.

Prompters

- Input domain specialties of each expert
- Input domain background attribute criteria to be used
- Input any weights which may be desired to distinguish between background attributes
- Input the value or rating for the background attributes for each expert
- Input the EECs for each evaluator in each domain (computed off-line using AHP)
- Input a threshold for EEC dwarfing
- Input a threshold for EEC dominating
- Input thresholds for each of the background attributes, or desirability ratings for a range of attribute values in the vicinity of the thresholds
- Input additional applicable rules (based on the examples given above)

Triggers

- Computation of variance of EECs
- Computation of "relative evaluator emphasis coefficient" REEC
- Firing of appropriate rules (based on the rule examples described above)

Alerters

- Notification that input information is incomplete for an evaluator
- Notification that the EEC variance is "abnormally" high/low
- Notification that an EEC dominating or dwarfing condition exists
- Notification that an evaluator is not rated above a minimum threshold for the indicator domain

Informers

- Request display of all information entered for each evaluator
- Request display of recomputed EECs given updated information
- Request explanation of EEC "dwarfing" or "domination" alert
- Request explanation of "EEC variance exceeds threshold" alert

7.2.2 Identification of Indicators

Expert knowledge in the form of facts and rules can be input into the KBESD knowledge base and applied to this phase of the methodology. In this section we describe some of the applicable knowledge types.

7.2.2.1 Types of Knowledge Applicable to Indicator Identification

Example knowledge types are listed as bulleted items in the remainder of this section.

- Expert knowledge can relate distinguishing an indicator as a "design killer" to a characterization of the operational mission of the designed system.

Individual indicator distinctives can be drawn respective of the operational scenarios under which the indicator is being evaluated. In the extreme case, under particular operational scenarios, a single indicator can be considered so critical as to "sack" the design if it is scored in its "rejection" region. Expert knowledge is required to make the determinations of if or when this condition is possible. Rules in the knowledge base expressing this relationship could be patterned after:

> IF *operational scenario* under which design is being evaluated is characterized as "system survival" AND (others applicable conditions) ... THEN rejection mode of <indicator> is "design critical."

Applications of rules of this type can be seen during the score aggregation phase. As a preview, consider the following rule:

> IF indicator score has membership greater than X in its rejection region, AND the rejection mode of the indicator is "design critical" THEN immediately signal appropriate alert.

- Expert knowledge can relate the operational scenario, domain dominance, or other factors to the minimum rejection region (or other region) size for an indicator.

This knowledge type concerns the actual or relative width of the rejection (or acceptance) regions for the score. Expert evaluator knowledge can be converted to knowledge base rules regarding this part of indicator identification. The rules either provide guidance for selecting appropriate interval widths, or for notifying evaluators that

interval widths are inappropriate. Factors included in the decision may include the operational scenario, criticality of the indicator, dominance of the indicator's major domain, or others.

Consider the case of a very critical indicator, as gauged by it's own weight, the weight of its parent or ancestry, and the operational scenario under which the design is being evaluated. If such an indicator has a very small rejection region, then the chances of failing to reject the indicator's score are appreciable. As a consequence, the false acceptance of the design could lead to a higher than anticipated probability of disaster in system operation. An example rule is of the form:

> IF the indicator's relative weight (local or global) exceeds (threshold) AND operational mode is "critical" AND (other conditions) THEN rejection region should be at least X% of the total score range.
>
> IF the indicator's relative weight is below some threshold AND the operational scenario is (less than "critical") THEN the minimum rejection region is no less than Y% of the total score range. (Y<X)
>
> IF the indicator's domain dominance factor is greater than D% THEN the minimum rejection region size is no less than Z%. (Y<Z<X)
>
> IF the indicator's rejection region size is below its minimum size THEN signal appropriate alert.

7.2.2.2 Other Expressions of Knowledge Applicable to Indicator Identification

Other expressions of knowledge relevant to this phase of the methodology include: (1) entering factual and knowledge-based entries into the KBESD; (2) numerical computations performed by the KBESD; (3) firing of productions rules (as describe above) in the expert system shell; (4) alerting the evaluating organization of undesirable conditions (defined in the rules); and (5) displaying factual or rule-based explanations of alert conditions.

Prompters

- Input relative domination of each of the major indicator domains

- Input indicator score desirability range boundaries. Either enter the numerical values, or select from a drop-down list of basic types, such as those in Figure 7.1.
- Input applicable rules into the knowledge base (patterned after the examples provided

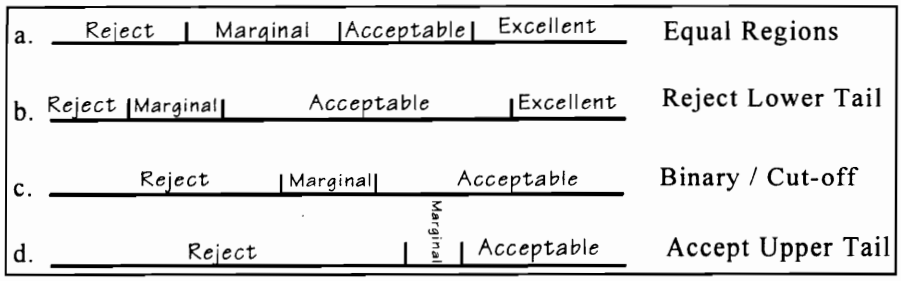


Figure 7.1. Example Score Desirability Region Distributions.

above)
- Input characterization of the operational scenario (e.g., critical, normal, idle, shutdown)

Triggers

- Computation of widths (and relative widths) of scores desirability ranges
- Comparison of relative desirability range widths to indicator relative weight
- Comparison of width of "below acceptable" ranges to "design killer" factor
- Firing of knowledge base rules relating to score desirability ranges, no-go region requirement, etc.

Alerters

- Notification that an indicator's desirability range widths are inconsistent with conditions specified in the rules (e.g., operational scenario, indicator weight, indicator domain dominance)
- Notification that an indicator has been designated as "design critical" if rejected

Informers

- Explanation of alerters, such as sequence of rules and conditions which triggered each of the alerters
- Display of re-evaluated results under different inputs (e.g., user responses to alerts)

7.2.3 Construction of Indicator Hierarchy

Knowledge-based entries applicable at this point in the methodology refer to *influence relationships* between indicators in the hierarchy. Recall from the methods in Chapter 4 that the indicator hierarchy construction generally proceeds in a top-down

fashion. A complex indicator, which is not directly scorable, is decomposed into a set of indicators, possibly also complex. The decomposition process proceeds depth first until the resultant indicators are directly scorable by the methods described in Chapter 6. The result of this decomposition process is a hierarchy, in which direct influence relationships are established between indicators as a consequence of the decomposition. The design qualities measured by the subindicators collectively give a measure of the parent quality. The extent of parent-child influence relationships are then completely defined and easily understood.

We recognize other types of indicator influence relationships. An example is the general positive relationship between cost indicators and performance indicators. That is, the better a system is to perform, the more it costs. Relationships such as this are observed between indicators which are not structurally related through the decomposition process. They are of particular importance to the design evaluation, however, since they provide insight, guidance, and checkups which cannot be captured explicitly and accounted for numerically in the hierarchy.

7.2.3.1 Types of Knowledge Applicable to Indicator Hierarchy Construction

In this section we discuss example rules pertinent to this design evaluation phase. We address both structural and non-structural indicator influence relationships. Example knowledge types are listed as bulleted items in the remainder of this section.

- Expert knowledge can relate attributes of non-structurally related indicators to augment design quality evaluation.

With indicators which exhibit this type of relationship, one type of knowledge required regards not only which indicators are related, but what aspect(s) and conditions relate them. That is, what does an expert know or need to know about relationships between indicators which gives indirect insight into the value of the design. In the obvious case of system performance and cost, it is the requirements *satisfiability score* of

the performance and the *absolute value* of the cost which are related. The distinction is, perhaps not quite so obvious. We do not expect an influence relationship between the *cost requirements satisfiability* and the *performance requirements satisfiability*. If the latter is high, then we hope the former is also high, but regardless, we should expect that the absolute cost will be proportional to the system performance, whether the cost satisfies our budget (i.e., requirements) or not.

Other related indicators or indicator categories are presented in Table 7.1. There may be many and various specific relationships, depending on the components and operational mission of the system. We offer illustrative examples applicable to a variety of complex systems.

Table 7.1. Examples of Non-Structural Indicator Influence Relationships.

(Degree) of Requirements Satisfiability for:	Related to (Degree) of Requirements Satisfiability of:
(Low) HHWI/HSI Complexity	• (High) Operator Training[1] • (Reduced) Operator Reliability[2]
(High) Computer Hardware/Software Performance	• (High) System Power Supply[3] • (Low) Physical System Dimensions • (Reduced) Operator Reliability[4]
(Low) Task/Mission Duration Minimization	• (High) Operator Accession • (Reduced) Operator Reliability[5]
(Low) External Threat Avoidance	• (High) Superstructure Maintainability
(High) Production Output	• (High) Logistical Support
(Low) Task Complexity	• (Reduced) Operator Reliability[6] • (High) Decision Aid Effectiveness[7]
(High) Software System Response Time	• (Reduced) Software Gernericity

- Expert knowledge can provide checkups regarding the acceptability of an indicator's level of decomposition.

---

[1] [Rouse 91].
[2] [Jubis 90; Woods 90a,90b; Hollnagel 92; Brown, *et al.* 91; O'Hara 92; Hewett 90].
[3] [Rostek 94].
[4] [Lee 92; Muir 87].
[5] [Becker, *et al.* 91].
[6] [Kirkpatrick 90].
[7] [Hallbert, *et al.* 92; Barnett 90; Ryder, *et al.* 90; Weiland *et al.* 92].

From Chapter 4, the result of the indicator decomposition process is a hierarchy of indicators. Borrowing from the family tree terminology, indicators in the hierarchy which share the same *parent* indicator are called *siblings*. The more complex the parent indicator, the more siblings (children) may be required to measure the design aspect represented by the parent. Less complex indicators may require little further decomposition, or may be scored directly by one of the methods described in Chapter 6. The direct consequence of this logical process is that the number of siblings from a parent indicator should be related to the complexity of the design aspect it is identified to measure.

Indicators representing very complex design aspects could be decomposed into many siblings. Overlooking the complexity of the design aspect measured by an indicator could lead to too few children. There are drawbacks to each of these conditions. A condition of potentially too many siblings can result from the over-decomposition of a single indicator due to: (1) the inclusion of indicators which belong in a lower level decomposition, or (2) the inclusion of indicators which really do not relate to the parent concept. Given that technical domain, operational domain, and systems engineering experts perform the indicator decomposition, we expect the occurrences of the latter situation can be considered negligible. Situation (1) is explained pictorially in Figure 7.2.

The primary facet of the methodology where too many indicators can negatively impact the evaluation is in the indicator weighting phase. Within a sibling group, each indicator is assigned a weight of criticality relative to its siblings. Chapter 5 provides a detailed description of the application of the Analytic Hierarchy Process for determining these weights. As the number of siblings increases beyond five, the likelihood of introducing inconsistency in determining the indicator weights increases. See Section 5.3.4 for a detailed discussion of this phenomenon. Inconsistency in weight assignments is at best a computational annoyance, but at worst, an incorrect representation of the

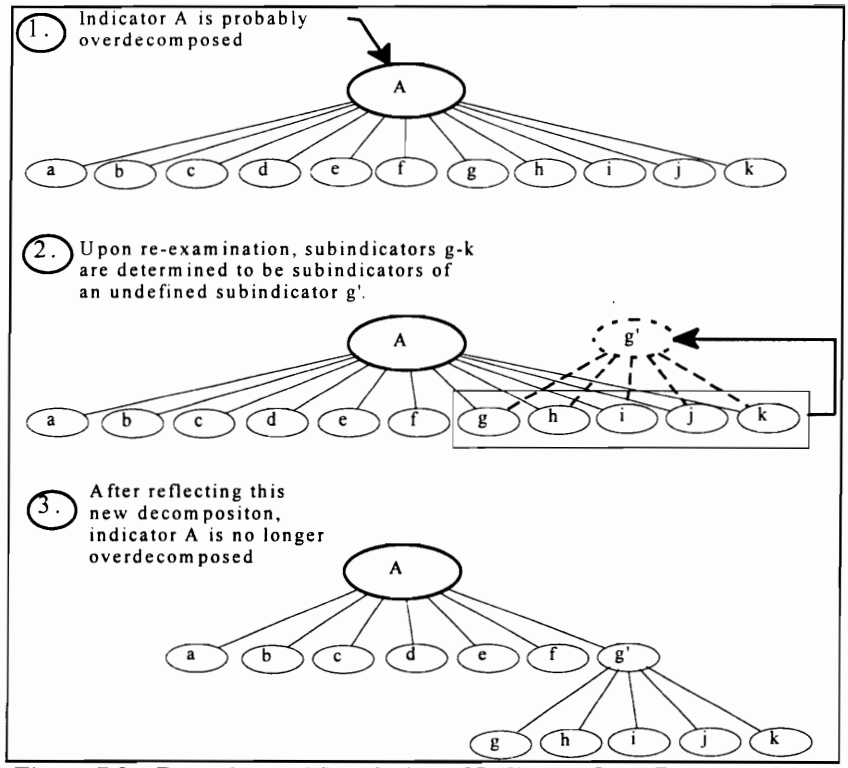


Figure 7.2. Detection and Resolution of Indicator Over-Decomposition.

addressing excessively large sibling groups. We address indicator weights from a different perspective in the next major section.

Too few sibling indicators can result primarily from failure to recognize the complexity or immeasurability of a parent indicator. An undesirable consequence is that not enough sub-elements of the measured quantity are considered in the determination of the indicator's score. This is not unlike evaluating the design of a uranium-powered automobile according to performance, reliability, and producibility, but ignoring supportability. One could conclude the design is valid based on the criteria examined. Ultimately, however, very few such cars would ever perform the operational mission for which they were designed and produced.

Expert knowledge applied to an examination of the hierarchy can identify these decomposition-related conditions, or the potential for them. Catching these conditions and either (1) applying expert knowledge to explain, accept, or dismiss them, or (2)

and either (1) applying expert knowledge to explain, accept, or dismiss them, or (2) forcing critical re-examination of them, can preclude potential corruption or even invalidation of an evaluation, in part or in total. Example knowledge-base entries applicable to this phase of design evaluation express relationships such as:

> IF the number of children of an indicator is greater than (small threshold) AND the number of children of an indicator is greater than (large threshold), THEN the number of children are considered ***normal***.
>
> IF the number of children is less than (small threshold) THEN indicator is under-decomposed.
>
> IF the number of children is greater than (large threshold) THEN indicator is over-decomposed.
>
> IF indicator is over-decomposed AND any of its children indicators are under-decomposed, THEN signal ***potential premature decomposition*** alert.
>
> IF indicator is under-decomposed AND any of its children indicators are over-decomposed, THEN signal ***potential insufficient decomposition*** alert.

7.2.3.2 Other Expressions of Knowledge Applicable to Indicator Hierarchy Construction

Other expressions of knowledge relevant to this phase of the methodology include: (1) entering factual and knowledge-based entries into the KBESD; (2) numerical computations performed by the KBESD; (3) firing of productions rules (as describe above) in the expert system shell; (4) alerting the evaluating organization of undesirable conditions (defined in the rules); and (5) displaying factual or rule-based explanations of alert conditions.

Prompters

- Input indicator name
- Input indicator definition, source, etc.
- Input indicator life cycle phase
- Input indicator major domain(s) (hardware, software, humanware, system, etc.)
- Input indicator score source (metric, simulation, etc.)

- Input score-based attributes of the non-structural relationship to other indicators, such as:
    * proportional scores
    * inverse scores
    * score finds ceiling by other's score
    * score finds floor by other's score
    * maximum delta between scores
    * linguistically equal
    * linguistically inverse
    * similar confidence intervals
- Input rules which relate non-structurally related indicators by the above attributes
- Input rules which dictate non-structurally related indicator major domain compatibility
- Input thresholds for number of children (sibling) indicators (e.g., < X or > Y) or conditions in which each condition is more or less tolerable, such as:
    * relative weight of parent indicator
    * number of siblings of parent indicator
    * score source type
    * indicator depth
    * domain dominance of indicator major domain category
    * whether or not an indicator is defined as a null-weight placeholder node

Triggers

- Check for indicator cycles in the hierarchy
- Check for indicator duplication by name
- Check for missing indicator identification information (life cycle phase, domain applicability)
- Compare indicator major domain compatibility with parent, ancestry
- Compare indicator major domain compatibility with non-structurally related indicators
- Compute number of sibling indicators
- Compute indicator depth
- Compare score attributes for non-structurally related indicators
- By rules, validate sibling count against applicable factors listed above

Alerters

- Notification of missing indicator identification information
- Notification of duplicate indicator name
- Notification of possible cycle in hierarchy
- Notification of indicator over- or under-decomposition
- Notification of indicator major domain incompatibility with ancestry

- Notification of indicator major domain incompatibility with non-structurally related indicators
- Notification of score attribute-related violations due to non-structurally related indicators
- Notification of weight attribute-related violations due to non-structurally related indicators

Informers

- Search for and display an indicator by name
- Search for and display an indicator by source
- Display of non-structurally related indicators and (value of) criteria for relationship
- Explanation of conditions triggering alerts regarding indicator decomposition levels
- Explanation of conditions triggering alerts regarding non-structurally related indicators

7.2.4 Determination of Indicator Weights

As described in Chapter 5, the Analytic Hierarchy Process (AHP) requires the direct application of expert knowledge during the pairwise comparison step. The facts relevant to this methodology phase include the indicator weights themselves (actual and relative), and appropriate thresholds for inconsistency, minimum and maximum weights. Knowledge-based rules relate these facts to interpret and assess the credibility of the resultant weights.

7.2.4.1 Types of Knowledge Applicable to Indicator Weighting

From a mathematical standpoint, the impact of the numerical indicator weights on the results of the design evaluation is directly traceable. From a logical standpoint, the impact is not so easily discerned, and requires expert knowledge as guidance. In this section we describe knowledge types which potentially improve the indicator weighting results. We propose application of expert knowledge to both the mathematical and logical dimensions.

- Expert knowledge can relate degree of judgment inconsistency to sibling group size, parent indicator weight, operational scenario, or other factors.

The AHP provides several alternative algorithms for determining the inconsistency of judgments as reflected in pairwise comparisons of indicators. According to T. L. Saaty [80, 94], who developed the AHP, an inconsistency ratio of less than 0.1 is acceptable. We propose in Section 5.4 a set of criteria by which we more crisply define the acceptability of inconsistencies. We couch these and other criteria within a knowledge-based framework for addressing acceptability of inconsistency.

Our thesis in Chapter 5 is that under special circumstances, inconsistency above the 0.1 threshold may be tolerable, while inconsistency under 0.1 may not be. Special circumstances include a relatively low weight of the parent of the "inconsistent" sibling group. This factor determines the relatively low global criticality of any child indicator, obviating the need for concern. Another circumstance involves the size of the sibling group itself. If there are many indicators, then we tolerate expectedly more inconsistency in human judgment. Other circumstances may apply to specific indicators during the evaluation of a specific complex system design. As illustrative examples, we propose the following rules.

> IF operational scenario is characterized as "mission critical" THEN max *desirable* inconsistency ratio is (some value below 0.1) AND max *tolerable* inconsistency ratio is (some delta higher than max desirable, but less than the default).
>
> IF inconsistency ratio is greater than (max desirable) but less than (max tolerable) AND parent relative weight less than (min desirable), THEN inconsistency ratio is ***acceptable.***
>
> IF inconsistency ratio greater than (max desirable) but less than (max tolerable) AND number of siblings is "high," THEN inconsistency ratio is ***acceptable.***
>
> IF inconsistency ratio greater than (max desirable) but less than (max tolerable) AND number of siblings is "normal," THEN inconsistency ratio is ***unacceptable*** with cause "relative to sibling count."
>
> IF inconsistency ratio greater than (max desirable) but less than (max tolerable) AND parent relative weight greater than (min desirable), THEN

inconsistency ratio acceptability is ***unacceptable*** with cause "high parent weight."

IF inconsistency ratio greater than (max tolerable) THEN inconsistency ratio acceptability is ***unacceptable*** with cause "excessive inconsistency ratio."

IF inconsistency ratio is ***unacceptable*** THEN raise appropriate alert.

- Expert knowledge can relate acceptability of weights to the presence of a dominating or dwarfed indicator

Even if the inconsistency ratio for an indicator sibling group is deemed acceptable according to the foregoing rules, there are still potentially undesirable circumstances we desire to catch. We consider the two special cases of dominating indicators, and dwarfed indicators. In the first case, the mathematical implication is that the score of a *single* indicator essentially determines the entire aggregate score of the parent. The logical implication is that the dominating indicator may better serve the evaluation by being moved out of the sibling group. This condition may occur when an indicator is over-decomposed (as in Figure 7.2).

In the second case, the mathematical implication is that the dwarfed indicator's weight is negligible. The logical implication is that the indicator may actually not be a legitimate contributor to measuring the design quality represented by the parent. In this case, it should be removed from the sibling list. The remaining alternative is to re-accomplish the pairwise comparison step with more careful attention paid to the dwarfed indicator(s).

For both conditions, expert determinations of the definitions of "dominating" and "dwarfed" are required. These definitions may themselves depend on the operational scenario, the relative weight of the parent indicator, or other factors. The numerical foundation for both concepts is the "relative" weight of an indicator. Relative weight is defined as the weight divided by the *expected* weight of the indicator. The expected

weight is the reciprocal of the size of the sibling groups. More details of this concept appear in Chapter 5. Illustrative knowledge-based rules express relationships patterned after the following:

> IF an indicator's relative weight exceeds the dominating threshold, THEN tag the indicator as ***dominating*** with probability X.
>
> IF an indicator is dominating AND its parent indicator's relative score is less than (threshold), THEN decrease the dominating probability to (a lesser value than X).
>
> IF an indicator is dominating AND its parent indicator's relative score is greater than (threshold), THEN increase the dominating probability to (a greater value than X).
>
> IF an indicator's relative weight is below the dwarfing threshold, THEN tag the indicator as ***dwarfed*** with probability X.
>
> IF an indicator is dwarfed AND its parent indicator's relative score is less than (threshold, e.g., 1.0), THEN decrease the dwarfed probability to (a lesser value than X).
>
> IF an indicator is dwarfed AND its parent indicator's relative score is greater than (threshold, e.g., 1.0), THEN increase the dwarfed probability to (a greater value than X).
>
> IF an indicator is dominating with probability exceeding (threshold) then signal **potentially dominant indicator** alert.
>
> IF an indicator is potentially dwarfed with probability exceeding (threshold) then signal **potentially dwarfed indicator** alert.

7.2.4.2 Other Expressions of Knowledge Applicable to Indicator Weighting

Other expressions of knowledge relevant to this phase of the methodology include: (1) entering factual and knowledge-based entries into the KBESD; (2) numerical computations performed by the KBESD; (3) firing of productions rules (as describe above) in the expert system shell; (4) alerting the evaluating organization of undesirable conditions (defined in the rules); and (5) displaying factual or rule-based explanations of alert conditions.

Prompters

- Indicator weight with respect to each parent
- Input threshold for relative weight dwarfing
- Input threshold for relative weight dominance
- Input rules for determining weight dominance/dwarfing
- Input inconsistency ratio for sibling group
- Input inconsistency acceptability threshold (overrides default)

Triggers

- Computation of relative weight as weight*n
- Comparison of relative weight with appropriate thresholds
- Computation of inconsistency acceptability
- Comparison of inconsistency acceptability with threshold
- Computation of global weight as recursive global weight of parent
- Computation of weight-related attributes of non-structurally related indicator

Alerters

- Notification that an indicator is potentially dwarfed with a high probability
- Notification that an indicator is potentially dominating with a high probability
- Notification that inconsistency is unacceptable for the sibling group

Informers

- Display weights for all sibling indicators
- Display weights of an indicator with respect to all parents
- Display relative weight of an indicator
- Explanation of conditions which triggered inconsistency acceptability alert
- Explanation of conditions which triggered weight dwarfing alert
- Explanation of conditions which triggered weight dominance alert
- Display of evaluator rationale for overriding defaults
- Display of evaluator notations regarding weight inputs (facilitated by *Expert Choice*)

7.2.5 Determination of Indicator Scores

During this phase of the methodology, there are several areas in which expert knowledge can be applied to enhance the credibility of the evaluation. From the logical perspective, the *acceptability* of scores with respect to tolerances, thresholds, variances, etc., is important. These relationships can be represented by knowledge-based rules. In this section, we motivate knowledge-based analysis of indicator scoring for the scoring types described in Chapter 6.

7.2.5.1 Types of Knowledge Applicable to Metrics-Based Scoring

Expert knowledge-based reasoning, implicitly or explicitly, is involved in guiding or evaluating the results of metric-based scoring. In this section we describe knowledge types applicable to this phase of the methodology. Knowledge types are listed as bulleted items in the remainder of this section.

- Expert knowledge can be used to relate the applicable acceptability criteria for discriminating between candidate metrics to the operational scenario under which an indicator's score is to be metric-based.

When more than one metric may serve as a basis for an indicator's score, we desire an objective approach to determining the most appropriate metric. In Chapter 6 we suggest criteria for discriminating between candidate metrics. Expert knowledge of both the metrics and the operational scenario in which the indicator is being evaluated, can be applied to increase the credibility of the discrimination process.

We use an example to illustrate the knowledge-types applicable for discriminating between metrics. We take the temperature reading from a thermometer as an example class of metric. A variety of thermometer types exist: those used in industry, cooking, and medicine. Thermometers of each type measure "temperature," but each are expected to operate in a particular environment, receive inputs within a particular range, and yield results of a particular degree of accuracy. Though countless thermometer types exist, every (expert) mother knows which type is most appropriate for determining the temperature of a baby.

Operational scenario is then clearly a factor. It determines which acceptability criteria may come into play, and possibly with what relative importance. Consider the "operational scenario" of encountering a child who appears to have a high fever. In a public school, there exists a requirement that the thermometer be sterile, germ-free, aseptic, etc. If the available thermometer fails to meet these criteria, the thermometer is not usable. That is, this metric is unsuitable. At home, however, no federal requirements

apply. A quick swipe under running tap water and a wipe on a hand towel yields a suitable thermometer. As for accuracy, in a hospital in a pediatric ward, digital precision to a tenth of a degree may be required. However, at home, if the digital thermometer has a dead battery, the less reliable mercurial type will do.

We do not wish to take the thermometer analogy to the extreme. However, it serves well to illustrate the relationship of operational scenario to metric discriminability factors. We acknowledge a metric selection process, in whole or part, may be quite formal, somewhat informal, or completely unnecessary. Driving factors include the availability of more than one applicable metric, organization policy, or expert preference. However, under the requirement for traceability of *all* decisions pertinent to design evaluation, formalization and documentation of all but trivial decisions cannot be viewed as superfluous. One form formalization can take is that of production rules in a knowledge base, as we describe throughout this chapter. Illustrative rules relevant to this aspect of metric-based scoring include:

> IF the operational scenario is characterized as "critical" THEN all metric acceptability criteria apply with maximum emphasis factors (weights).
>
> IF the operational scenario is characterized as "cautious" THEN
> metric acceptability criterion 1 applies with emphasis factor W_1 AND,
> metric acceptability criterion 2 applies with emphasis factor W_2 AND, ...
> metric acceptability criterion N applies with emphasis factor W_n.
>
> IF the operational scenario is characterized as "routine" THEN
> metric acceptability criterion 1 applies with emphasis factor W_1 AND,
> metric acceptability criterion 2 applies with emphasis factor W_2 AND, ...
> metric acceptability criterion N applies with emphasis factor W_n.
>
> IF the operational scenario is characterized as "critical" THEN all metric acceptability criteria must be attained by the metric with a minimum rating of "good."
>
> IF the operational scenario is characterized as "cautious" THEN
> metric acceptability criterion 1 must be attained by the metric with a minimum rating of (rating) AND ... (similarly for other criteria).

- Expert knowledge can relate the requirement for calibration, scaling, or otherwise "fudging" a metric's native values to external factors.

The value yielded by a metric is subject to some interpretation, depending on, e.g., the resolution of the input data. Interpretation may take the form of calibration, scaling, adjusting, etc. Interpretation becomes important especially in the presence of other factors, such as the gravity of the operational scenario, or presence or absence of other (un)desirable conditions.

We take again the household thermometer reading analogy to a metric. A temperature taken in an adult's mouth yields an acceptability accurate measure of body temperature. However, in a different "operational scenario," e.g., taking the temperature of an infant, an oral measurement is not practical. In this scenario, an underarm or rectal reading is taken. According to this author's pediatrician, two additional factors must be considered. If the skin is cold and clammy when the underarm reading is taken, an accurate reading requires *adding* one degree to the measured result. If the skin is flushed, however, one degree should be *subtracted* from the reading.

The point of the analogy is clear. With some metrics, the operational scenario and presence of other factors must be taken into consideration in the interpretation of the metric's native value. In the cases where relevant expert knowledge can or must be made explicit, knowledge-based facts and rules must be created by experts in the respective indicator domains. Applicable rules are then applied when interpreting metric-based results in the course of a design evaluation. Example rules may take the form of these examples, taken directly from the thermometer analogy:

> IF measurement scenario is for taking human body temperature, THEN the measurement device should be a medical thermometer.
>
> IF patient is an adult, THEN measurement should be taken by mouth.
>
> IF patient is an infant, THEN measurement should be taken under the arm.

IF body temperature is measured under the arm AND skin is clammy,
THEN add one degree to the thermometer reading.

IF body temperature is measured under the arm AND skin is flushed,
THEN subtract one degree from the thermometer reading.

- Expert knowledge can relate the operational scenarios intended for the designed system to the acceptability of native scores yielded by (the same or different) metrics, and from one or more evaluators.

This knowledge-based application to the use of metrics again involves interpretation of the metric results. It is applied to provide guidance regarding the acceptability of the actual results yielded by the metric. What metric reading is "good" or "bad?" How does the acceptability of the metric's native values vary as the reading ranges from its worst possible to its best possible value? Experts in their respective fields have this knowledge regarding metrics they use and operational scenarios they encounter. Represented explicitly in a knowledge base, such knowledge can be used to enhance the results of the design evaluation.

To illustrate, we borrow a final time from the thermometer analogy. The metric is "body temperature" and the measurement is obtained as degrees Fahrenheit from a household thermometer. An example native reading may be 101. Expert knowledge is required to interpret this value as bad, and how bad, or good, and how good. Assuming the reading is accurate or appropriate calibrations or adjustments have been considered, the operational scenario may factor heavily in the reading's interpretation. Again, according to this author's pediatrician, if the patient is a child, a reading of 101 is considered "elevated" but not "alarming." If the patient is an adult, this temperature is considered "high" and "cause for concern." If the patient has been participating in heavy physical activity, or recently working in a heated environment, a temperature of 101 may be considered "normal." If the temperature is accompanied by certain other conditions, such as unconsciousness, etc., it may fall into yet another category.

The point to be drawn is that links exist between factors such as the operational scenario and the acceptability of a metric's reading. Experts are aware of these links, or can derive them. The experts are then in a position to create knowledge-base entities which are automatically incorporated in the design evaluation. Example rules taken from this analogy may serve as templates for similar rules.

> IF measurement device is medical thermometer AND application is measurement of human body temperature, THEN minimum expected value is X.
>
> IF measurement device is medical thermometer AND application is measurement of human body temperature, THEN maximum expected value is Y.
>
> IF the measurement scenario involves a fevered adult AND the patient's activity level is "any" AND temperature reading is between 98.6 and 99.5 THEN body temperature is ***normal*** and cause for concern is ***low***. (acceptability is high)
>
> IF the measurement scenario involves a fevered adult AND the patient's activity level is "low" AND temperature reading is between 99.6 and 101 THEN body temperature is ***elevated*** and cause for concern is ***moderate***. (acceptability is marginal)
>
> IF the measurement scenario involves a fevered adult AND the patient's activity level is "strenuous" AND temperature reading is between 99.6 and 101 THEN body temperature is ***elevated*** and cause for concern is ***low***. (acceptability is marginal)
>
> IF the measurement scenario involves a fevered child AND the patient's activity level is "low" AND temperature reading is greater than 102 THEN body temperature is ***elevated*** and cause for concern is ***high***. (acceptability is low)

7.2.5.2 Other Expressions of Knowledge Applicable to Metrics-Based Scoring

Other expressions of knowledge relevant to this phase of the methodology include: (1) entering factual and knowledge-based entries into the KBESD; (2) numerical computations performed by the KBESD; (3) firing of productions rules (as describe above) in the expert system shell; (4) alerting the evaluating organization of undesirable

conditions (defined in the rules); and (5) displaying factual or rule-based explanations of alert conditions.

Prompters

- Input metric acceptability criteria
- Input metric acceptability criteria threshold (in total, or per criteria)
- Input scores for metric with respect to each criteria (subjective, linguistic)
- Input rules for determining "acceptability" of metric with respect to how it stacks up against the acceptability criteria (as in the examples above)
- Input rules for determining [conditions for, nature of, and degree of] any [calibrations of or adjustments to] the native metric score (as in the examples above)
- Input rules for determining interpretation of the native metric score (as in the examples above)
- Input minimum expected computed value of metric
- Input maximum expected computed value of metric
- Input selected mapping function (from options) of native scores to [0,100]
- Input native score computed by metric (or combined metric)
- Input fuzzy membership functions which determine mapped score membership in one or more desirability ranges (if desired)

Triggers

- Compute metric acceptability for each of the acceptability criteria
- Compare acceptability to any thresholds established for each criteria
- Compute overall metric acceptability
- Compare overall metric acceptability to any thresholds
- By rules determine metric acceptability with respect to criteria thresholds
- By rules determine metric calibration or adjustment requirements
- By rules determine metric score interpretation
- By rules determine metric acceptability with respect to indicator weight
- Compute mapping of indicator score to [0,100]
- Compute aggregate mapped score for metric if multiple evaluators have used the metric
- Compute desirability range fuzzification
- Determine desirability range in which score falls
- Determine membership of mapped score in fuzzy desirability ranges (if used)
- By rules determine acceptability of metric desirability memberships

Alerters

- Notification of unacceptability of metric based on the criteria used

- Notification of unacceptability of metric based on the operational scenario or other specified conditions
- Notification of calibrations or adjustments required/recommended for using the metric under the operational scenario
- Notification of the interpretation of the native indicator score based on the operational scenario or other specified conditions
- Notification of unacceptably high membership in a low desirability range
- Notification of unacceptably low membership in a high desirability range

Informers

- Display explanation of rules/facts which led to metric unacceptability alert
- Display explanation of rules/facts which led to metric score desirability membership alert
- Display indicator score computed under alternative score mapping function
- Display indicator memberships computed under a fuzzification factor
- Search for a metric by category from a metrics database
- Search for a metric by source from A metrics database
- Search for a metric by date from A metrics database
- Search for a metric by author from A metrics database

7.2.5.3 Types of Knowledge Applicable to Simulation-Based Scoring

Chapter 6 contains a description of the application of simulation-based modeling to scoring system design quality indicators. A great deal of expert knowledge is applied both explicitly and implicitly in the development and application of the simulation models. In this section, we motivate and illustrate by examples the types of expert knowledge applicable to analysis and interpretation of simulation-derived measures.

- Expert knowledge can relate the appropriate confidence level for a confidence interval about a simulated value to numerous factors including:
    - granularity of the input data
    - subjectivity/objectivity of the input data
    - granularity of the model
    - level of abstraction modeled
    - characterization of operational scenario(s) being modeled
    - probability of encountering the operational scenario(s)

Chapter 6 describes the application of confidence intervals in this methodology. They serve as the primary means of deriving an indicator score from experimentation

with simulation models. In Chapter 6, we recommend a confidence level (CL) between 90% and 95%, suggesting that expert knowledge evaluation scenario drive the determination. In this section, we discuss the formalization of expert knowledge of relevant factors which can be used to determining an appropriate CL. Forcing explicit knowledge representation prevents "gaming" to obtain a desired score. It also makes possible the degree of decision traceability we require in the methodology. Example rules of this type include the ones below. Note that these rules may become quite complex depending on the conditions which can or must be related to determining the appropriate confidence level.

> IF the operational scenario is characterized as "normal," THEN minimum CL is X%.
>
> IF the operational scenario is characterized as "critical," THEN minimum CL is Y% (Y>X).
>
> IF the relative weight of the indicator is less than (min desirable), THEN minimum CL is Z%.
>
> IF the relative weight of the indicator is greater than (max desirable), THEN minimum CL is Q% (Q>Z).
>
> IF the subjective reliability of the input data is "low" THEN minimum CL is V%.
>
> IF the subjective reliability of the input data is "high" THEN minimum CL is W% (W<V).

Other applicable types of expert knowledge are included in these general descriptions.

- Expert knowledge can relate the (maximum, minimum, thresholds of a) score of an indicator to the probability of occurrence of events observed during system simulation, such as:
    * key equipment in inoperable status
    * software in unreliable state

    * response time from key systems
    * unavailability of minimum no. of operators
    * key (command) personnel removed from duty
    * throughput below a given threshold
    * capacity reduced to below a minimum threshold
    * detection of hazardous material/vapor/etc. in operational setting

- Expert knowledge can relate the (maximum, minimum, thresholds of a) score of an indicator to the duration of specific events (as above) observed during system simulation
- Expert knowledge can relate the (maximum, minimum, thresholds of a) score of an indicator to the probability of occurrence of the duration of specific events (as above) observed during system simulation

7.2.5.4 Other Expressions of Knowledge Applicable to Simulation-Based Scoring

Other expressions of knowledge relevant to this phase of the methodology include: (1) entering factual and knowledge-based entries into the KBESD; (2) numerical computations performed by the KBESD; (3) firing of productions rules (as describe above) in the expert system shell; (4) alerting the evaluating organization of undesirable conditions (defined in the rules); and (5) displaying factual or rule-based explanations of alert conditions.

Prompters

- Input simulation-derived crisp score mapping function (possibly from options)
- Input rules for determining the appropriate confidence level (as described above)
- Input rules for determining the acceptability of the possible range of output scores (as described above)
- Input the simulation-derived confidence interval obtained for an estimated value
- Input confidence level derived by production rules (as above)
- Input associated probabilities of durations of occurrence of key events such as listed above
- Input rules for determining acceptability of probability of durations of undesirable events
- Input rules for determining what probabilities of occurrences of "binary" events could make an indicator score "rejection" or "design killer"

- Input simulated occurrences of binary events and the probabilities with which they occur

Triggers

- Convert simulated native crisp score to standard range score
- Convert simulated native confidence intervals to standard interval
- By rules determine the most acceptable CL based on (factors listed above)
- By rules determine if durations for key events have exceeded acceptable threshold lengths with unacceptably high probabilities
- By rules determine if binary events have occurred with unacceptably high probabilities

Alerters

- Notification that selected maximum CL is inconsistent with applicable determining factors
- Notification that the probability associated with undesirable event duration is unacceptable
- Notification that the probability associated with an occurring binary event is unacceptable
- Notification that the selected CL is inconsistent with the resolution/reliability of the data used to determine the CL
- Notification that the selected CL is inconsistent with the weight (local, global, local-global) of the indicator or its domain dominance value

Informers

- Display mapped score using alternative mapping function
- Display mapped intervals using alternative CL
- Display mapped intervals using alternative mapping function
- Display operational scenario from which simulated results were derived
- Display dynamic system behavior, respective of a particular simulated value at a selected point in simulation
- Display conditions and rules which triggered an undesirable event condition

7.2.5.5 Types of Knowledge Applicable to Technical Domain Knowledge-Based Scoring

Chapter 6 describes the process of applying technical domain expertise to scoring of design quality indicators. The general representation of such a score is an *interval.* The quantitative score itself notwithstanding, there are several facets of the *process* of interval scoring which involve either the explicit or implicit application of expert knowledge. Facets of particular interest include the absolute interval width, both for

individual scores and in the aggregate and relative differences in interval scores from each evaluator. We desire formalization, representation, and computer-based application of knowledge pertaining to each facet. These forms of expert knowledge become significant contributors to the traceability and verifiability of the results of this portion of the methodology. Specific knowledge types are listed as bulleted items in the remainder of this section.

- Expert knowledge can relate the minimum credible interval width to factors such as:
    - "measurability" of the indicator
    - experience base "sample size" from which similar indicators have been encountered by the evaluator
    - expertise level of the evaluator
    - operational scenario under which the indicator is being evaluated

In Chapter 6 we introduce the issue of credibility of a subjective interval score. The thrust of the discussion is that as the interval score for a subjective quality approaches zero (a discrete score), the credibility of the score diminishes. In effect, the more subjective the design quality being measured, the less precise the score can be. Several factors (as listed above) can be brought to bear on this issue. In fact, to some degree, they *must* be brought to bear for the sake of the credibility of the evaluation. Formalization of expert knowledge regarding interval score credibility can be represented as (perhaps complex) rules expressing relationships as in the following examples.

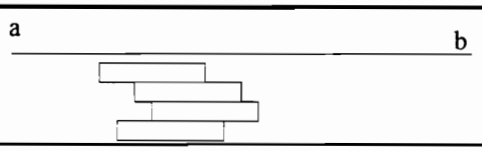
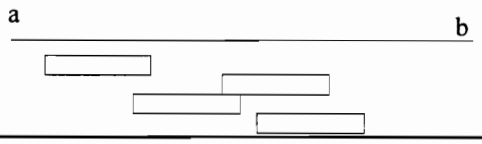
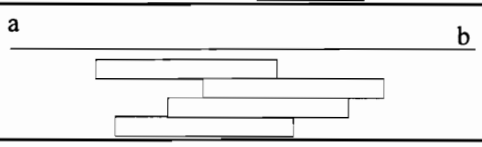
> IF indicator measurability is "high" AND (evaluation experience base is "large" OR evaluator expertise is "high") THEN minimum credible interval width $\geq$ X.
>
> IF indicator measurability is "low" AND (evaluation experience base is "large" OR evaluator expertise is "high") THEN minimum credible interval width $\geq$ Y (Y>X).
>
> IF operational scenario is characterized as "mission critical" AND (permutations of values for evaluation experience base and expertise level) THEN minimum credible interval width $\geq$ (appropriate value).

Other similar rules would broaden or possibly narrow the minimum acceptable interval. The rules should reflect tradeoffs between the contributing factors, such as letting high expertise override lower values of the other factors. Doing so keeps the interval reasonably narrow.

- Expert knowledge can determine the desirability of the *dispersion* or variance of the individual interval scores.

The dispersion we refer to is the difference between the least lower bound and the greatest upper bound of the union of the interval scores provided by a group of evaluators. If this dispersion is very large, especially relative to the entire possible score range, then the essence (see Section 6.6.2) of the aggregate interval will not be very representative of the indicator's "true" value. This is particularly true when relatively narrow intervals which were originally assigned. We depict possible dispersion desirability scenarios in Table 7.2. Example rules derived from this type of expert knowledge can be patterned after these examples:

Table 7.2. Desirabilities of Interval Width Combinations.

Dispersion vs. Interval Width	Interval Width vs. Score Range	Graphical Representation	Desirability
Low	Low		High
High	Low		Medium
Low	High		Medium
High	High		Low

IF dispersion vs. average interval width is "low" AND interval width vs. possible score range is "low" THEN acceptability of score dispersion is "high."

IF dispersion vs. average interval width is "low" AND interval width vs. possible score range is "high" THEN acceptability of score dispersion is "medium."

IF dispersion vs. average interval width is "high" AND interval width vs. possible score range is "low" THEN acceptability of score dispersion is "medium."

IF dispersion vs. average interval width is "high" AND interval width vs. possible score range is "high" THEN acceptability of score dispersion is "low."

- Expert knowledge can be used to detect and determine the desirability of *deviations* of individual interval scores from the others provided for that indicator.

In this scenario, we are interested in detecting and an interval score from an evaluator which is deviates "significantly" from the scores from the other evaluators. As an example, observe in the second row of Table 7.2, the first interval score is significantly lower than the other three scores. Making this determination is most applicable after all scores have been entered, at which point a "deviant" score is more detectable. Rules based on this genre of expert knowledge need not address the numerical deviation only — they may be tempered by factors such as:

* the evaluator expertise or experience
* the local or global weight of the indicator
* the desirability subrange in which the worst-case *essence* score would fall
* the domain dominance of the indicator's major domain
* the number of evaluators scoring on the indicator
* the evaluator emphasis coefficient of the "deviant" evaluator

Considering these and possibly other factors, example relationships are presented below. Evaluators populate the knowledge-base with as many as needed and conceivably more complex versions of these illustrative examples.

IF evaluator score deviation is less than X (value or percent) AND the relative EEC of the evaluator is less than 1.0, THEN (take no action).

IF evaluator score deviation is greater than X AND evaluator's EEC is greater than 1.0, THEN mark evaluator's score as ***potentially deviant***.

IF the worst case desirability range possible due to the evaluator's score using a *k* value > (some threshold, e.g., 0.5) is less than "acceptable," THEN mark the evaluator's score as ***significantly deviant***.

IF an evaluator's score is potentially deviant, THEN raise appropriate alert.

IF an evaluator's score is significantly deviant, THEN raise appropriate alert.

7.2.5.6 Other Expressions of Knowledge Applicable to Technical Domain Knowledge-Based Scoring

Other expressions of knowledge relevant to this phase of the methodology include: (1) entering factual and knowledge-based entries into the KBESD; (2) numerical computations performed by the KBESD; (3) firing of productions rules (as describe above) in the expert system shell; (4) alerting the evaluating organization of undesirable conditions (defined in the rules); and (5) displaying factual or rule-based explanations of alert conditions.

Prompters

- Input interval score from each evaluator
- Input individual range *width* threshold for an individual evaluator's score (this is a measure of how narrow and still credible an interval can be expected from an evaluator)
- Input assessment of the subjective "measurability" of the indicator (very low, moderate, high very high)
- Input assessment of the experience base sample size (from which similar indicators have been encountered by an individual evaluator, or the organization as a whole). Could be values like (very low, low, moderate, high, very high)
- Input values for the subjective expertise measure of an evaluator (moderate, high, very high)
- Input range width threshold for the aggregate score. This value has to take into account the possible dispersion of the individual interval scores. It could be significantly wider than the individual range widths.

Triggers

- By rules determine acceptable minimum interval widths
- Compute width of individual intervals
- By rules determine acceptability of individual interval widths
- Compute total range of individual intervals
- Compute weighted average of intervals
- Compute range of average intervals
- Compute dispersion of all entered intervals taking into account:
* individual bounds weighted by evaluator emphasis coefficients
* extraction of the essence of each interval
- By rules determine the desirability of the interval dispersion, taking into account the factors described in the "prompters" section
- By rules detect and determine acceptability of individual interval scores which deviate substantially from the other scores
- Compute the minimum *k* factor for which the intervals yield an "acceptable" or better interval essence

Alerters

- Notification that an individual evaluator's range is inappropriately wide or narrow
- Notification that an individual evaluator's range is much lower or higher than others
- Notification that the averaged interval's width is excessively wide or narrow
- Notification that the averaged interval falls largely below "acceptable"
- Notification that the dispersion of ranges is excessively wide, especially considering EECs

Informers

- Request re-evaluation of alerted condition with change to EEC
- Request re-evaluation of alerted condition with change to desirability ranges
- Request display of all original score ranges
- Request display of evaluator emphasis coefficients
- Request display of desirability ranges
- Request display of other evaluators' rationale for input scores
- Request display of score advice from expert database

7.2.5.7 Types of Knowledge Applicable to Operational Domain Knowledge-Based Scoring

Expert knowledge applied to this type of indicator scoring primarily relates to the fuzzy linguistic scores and their membership functions. We provide a brief review this type of score. Chapter 4 describes the motivation for and process of assigning subranges within the standard score range [0,100]. A qualitative *desirability* label, e.g., "rejection"

or "acceptable," is associated with each subrange (see Figure 7.4). Chapter 6 motivates the reasoning for using fuzzy linguistic scores for expert knowledge, based on the relatively low measurability of indicators of this type. Fuzzy membership functions can be derived from the original desirability subranges, or individually defined. Expert knowledge is the chief determining factor no matter which version of the membership functions are used. Example knowledge types are listed as bulleted items in the remainder of this section.

- Expert knowledge can be used to assign, or determine the acceptability of the degree of overlap between two adjacent fuzzy linguistic score intervals.

Example overlap between fuzzy linguistic scores is pictured in Figure 7.3. The degree of overlap should be related to the desired degree of *discriminability* between scores. Discriminability becomes particularly important when not all evaluators agree on the same score. To see why, consider an example. If one evaluator assigns a linguistic score lower than the rest, then the weighted average score will be lower than if all evaluators gave the same score. Under the influence of a "dissenting" score, we desire some form of "situational awareness." That is, we desire to know the proximity of the weighted average score to other score boundaries.

Assume that the weighted average score for an indicator is as depicted in Figure 7.3. The score is represented by a black dot on the abscissa of the graph. The ordinate represents membership in a fuzzy interval. In Figure 7.3a, with no overlapping intervals, the score has full membership in the fuzzy linguistic score "acceptable," and membership in no other fuzzy linguistic score. There is little indication that the score is very nearly having zero membership in the "acceptable" range. Such discrete boundaries are not realistic, however, when dealing with scores involving the amount of subjectivity which indicators of the present class require.

In Figure 7.3b, some necessary fuzzification has been applied to the intervals, using the formulae of Section 6.4 in Chapter 6. In this more realistic case, we are aware that the score has a non-zero membership (0.15) in a below-acceptable desirability range. Under the substantial fuzzification applied in Figure 7.3c, the same score has appreciable membership in both "acceptable" and "marginal." In this case, however, our ability to convincingly discriminate between the two desirability ranges is lost. Our point is the following: some fuzzification is needed, but both too little and too much, undesirably skew the evaluation. Too little is undesirable due to the subjective nature of the scores required for indicators of this type. Too much fuzzification is undesirable due to the need to discriminate, by degree of membership, just how desirable the assigned score is.

Taking note of the power of fuzzification to assign "meaning" to a score, we have two primary goals regarding its use. Foremost, we seek to harness the power of fuzzy intervals to enhance the credibility and reliability of the results of the evaluation. Our thesis is that the discriminability offered by fuzzy score memberships is a crucial capability. Secondly, we seek to avoid "gaming" the evaluation to achieve desirable

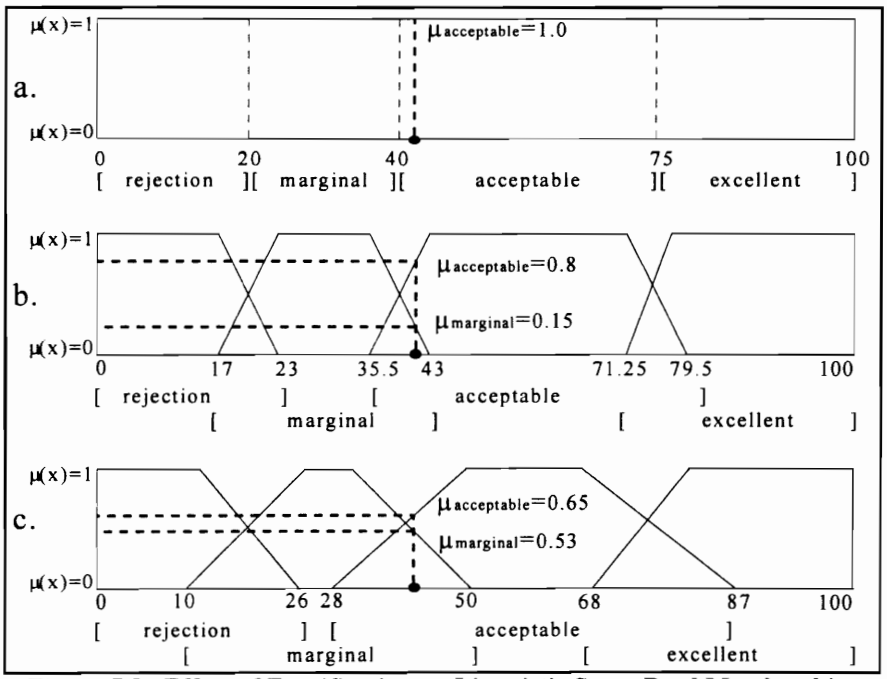


Figure 7.3. Effect of Fuzzification on Linguistic Score Dual Membership.

results. The tradeoff is not clear cut, and we ultimately rely on the knowledge of experts in the applicable domains to make the decisions. Those decisions can be assisted by formalizing the information experts bring to bear on the decisions, and applying them in an expert system knowledge base. Knowledge-base entries may include many and possibly complex combinations of instances of the following rule examples.

> IF operational scenario is characterized as "critical" THEN minimum fuzzification factor is X AND maximum fuzzification factor is Y (X<Y).
>
> IF operational scenario is characterized as "normal" THEN minimum fuzzification factor is X' and maximum fuzzification factor is Y' (X'<Y', X'<X, Y'<Y).
>
> IF indicator score dispersion is "high" THEN fuzzification factor is (near maximum acceptable value).
>
> IF indicator score dispersion is "low" THEN fuzzification factor is (near minimum acceptable value).

- Expert knowledge can be used to determine the acceptability of the dispersion of scores as assigned by each evaluator.

This topic is treated similarly as with technical domain knowledge-based scoring. It becomes especially crucial for scores of the present class, since we are already dealing with relatively wide score intervals. That is, the ratios presented in Table 7.2, especially the ratio of the score interval to the possible range of scores, will be high. The dispersion of scores will be more pronounced as well, since score intervals are fixed. A score of "marginal" and a score of "acceptable" will by definition be largely offset one from another. No such guaranteed offset is associated with freely-defined interval scores of the previous type.

To address this situation, the rules applied with respect to technical domain knowledge-based score intervals may be still be used. However, with a slightly greater tolerance for wide score dispersions and deviations, since they are a byproduct of the

scoring method, itself. The method itself, please recall, is driven by the subjectivity inherent in the concepts being measured. In extreme cases, *any* deviation of linguistic scores should be alerted, given the mathematical impact it has on the aggregate score.

7.2.5.8 Other Expressions of Knowledge Applicable to Operational Domain Knowledge-Based Scoring

Other expressions of knowledge relevant to this phase of the methodology include: (1) entering factual and knowledge-based entries into the KBESD; (2) numerical computations performed by the KBESD; (3) firing of productions rules (as describe above) in the expert system shell; (4) alerting the evaluating organization of undesirable conditions (defined in the rules); and (5) displaying factual or rule-based explanations of alert conditions.

Prompters

- Input linguistic score, e.g., "rejection," "marginal," "acceptable," "excellent," or others as required
- Input fuzzification factor as a percentage (e.g., 15% is entered as 0.15)
- Input scenario-dependent description, which is used as a basis for characterization of the operational scenario in which the indicator is being evaluated
- Respond to alerts as required with rationalization for scores or values entered
- Input applicable rules as describe above

Triggers

- Compute fuzzy membership functions from fuzzification factor and desirability ranges
- Compute dispersion of all entered intervals
- Compute weighted average of intervals
- Compute best-case, worst-case, etc. membership of averaged interval in linguistic ranges

Alerters

- Notification that an individual evaluator's range is much lower or higher than others
- Notification that the dispersion of ranges is excessively wide
- Notification that the averaged interval's width is excessively wide or narrow
- Notification that the averaged interval has a best case score falling below "acceptable"

Informers

- Request re-evaluation of alerted condition with change to EEC

- Request re-evaluation of alerted condition with change to fuzzification factor
- Request display of all original score ranges
- Request display of EECs
- Request display of desirability ranges
- Request display of other evaluators' input rationale in response to an alert

7.2.6 Aggregation of Indicator Scores

We describe the process of aggregating indicator scores in Chapter 6. The mechanics of the process involve several steps, which we abbreviate here for review. (1) if required, determine desirability mapping function from native scores to the range [0,100] (metrics-based and simulation-based scoring); (2) using the evaluator emphasis coefficients, combine scores from multiple evaluators to a single score for the indicator; (3) if required, based on the evaluation posture taken, collapse the interval scores to yield a discrete point, representing the essence of the interval; and (4) proceeding from the leaf indicators in the hierarchy to the root, combine the scores of sibling indicators using their relative weights as coefficients in a linear combination function.

Interpreting the results of the previous steps requires both the implicit and explicit application of expert knowledge. We seek to disclose as much implicit knowledge as possible. This knowledge along with the explicit knowledge-types applied, is then formalized as facts and rules in the KBESD's expert system shell. At the appropriate points in the score aggregation process, these facts and rules are examined and fired to provide feedback and checkups on the intermediate and final results.

7.2.6.1 Types of Knowledge Applicable to Indicator Score Aggregation

Expert understanding of the consequences of score desirability, proportions, thresholds, correlations, etc., are extremely important to interpreting the results of this phase of the methodology. We offer in this section bulleted items which express types of expert knowledge required.

- Expert knowledge can be used to determine the appropriate evaluation posture taken during score aggregation time. The posture is reflected in a k factor, used to collapse interval scores.

The first, and perhaps most critical decision to be made when computing aggregate scores is the determination of the evaluation posture to be taken. The evaluation posture is described in Section 6.6 of Chapter 6. The evaluation project management may make the determination, driven by the degree of safety they deem is appropriate for evaluating the design. The overwhelming influencing factor is likely the operational scenario. Specific considerations include the gravity, mission criticality, risk to human life, risk to system survivability, financial gamble, etc. The evaluation posture is then a reflection of the perceived probability of avoiding a potentially catastrophic type I error. That is, based on expert evaluation of the design, in light of the operational scenarios considered, the desire is to minimize the probability that the system as designed could fail to perform to stated operational requirements. The evaluating organization reduces that probability by taking a commensurably conservative evaluation posture.

The representation of this evaluation posture is the k factor, $k \in [0,1]$, (see Section 6.6). The application of this value is in the collapse of interval scores. For an interval $[a,b]$, a discrete value may be obtained from the interval by taking the value $(1-k)a+kb$ (from Equation 6.10). Thus, $k = 0$ yields the interval minimum, a, and $k = 1$, yields the interval maximum, b. One can see that as k ranges from 0 to 1, any value in the range from a to b is possible. This allows for what-if type of analysis of the design.

Expert knowledge is applied in the determination of the k value which provides the desired probabilistic assurance. Rules which express this knowledge may take be patterned after and combined into perhaps complex versions of the following examples.

> IF system failure could result in the loss of human life with probability greater than P_1, THEN minimum acceptable k is K_1.
>
> IF system failure could generate D or greater dollars in capital loss with probability greater than P_2, THEN minimum acceptable k is K_2.

(other similar rules, relating other intolerable circumstances and their probabilities of occurrence to a minimum *k* value).

Similar rules can be generated which are not targeted at complete system failure, but at smaller, yet undesirable consequences. For example:

> IF software maintainability costs could exceed C% of budgeted value in the first two years of the system life cycle, THEN minimum acceptable *k* is K_s.

Alternatively, regarding this topic, we note that the *k* value may not be directly assignable as suggested by the above rules. It may be arrived at iteratively, beginning with a selection of either 0 or 1. Then let the KBESD generate the results, with attendant alert messages, until the most severe indicator failures have been eliminated. The knowledge-based issue may revert to determining, based on the knowledge of the design qualities being scored, the acceptability of the *k* value arrived at by this procedure. That is, experts may determine that the *k* value for which the design can safely be considered acceptable eliminates too many realistic operational scenarios, which have a high probability of occurring. This may be the easier of the two *k*-related assessments for the experts to make. Knowledge-based entries to this effect may be patterned after this example.

> IF the minimum computed acceptable *k* to avoid catastrophic system failure with probability P is greater than K, THEN consider that the design is not acceptable.
>
> IF the minimum acceptable *k* to place the indicator in its acceptable range is greater than K, then score the indicator as "rejection."

- Expert knowledge can be used to determine acceptability of membership of an indicator score in each desirability region in which it falls.

Any set of score desirability ranges may be fuzzified using techniques from Chapter 6. We describe advantages to this technique above. The decisions required as a

result of applying the fuzzification technique involve determining the acceptability of the possible memberships each discretized indicator score can take. Evaluators generate rules for dual membership thresholds with regard to numerous factors, including, but not limited to the following:

* all indicators evaluated under a given operational scenario characterization
* all indicators in a given major domain
* all linguistic score-based indicators
* all indicators with a relative weight above some threshold

Example rules which express this knowledge are provided below.

> IF operational scenario is characterized as "critical" THEN membership greater than X in any "below-acceptable" desirability range is unacceptable.
>
> IF operational scenario is characterized as "critical" THEN membership greater than Y in any desirability range above "marginal" is acceptable.
>
> IF an indicator's major domain has a relative domain dominance factor of greater than (threshold), THEN membership greater than X in any "below-acceptable" desirability range is unacceptable.
>
> IF an indicator's relative weight is greater than (threshold) THEN membership greater than X in any "below-acceptable" desirability range is unacceptable.
>
> IF an indicator is scored in its rejection region AND it has been designated as design critical, THEN immediately signal an appropriate alert.

Additionally, combinations of these or other conditions may be associated with a dual membership threshold.

7.2.6.2 Other Expressions of Knowledge Applicable to Indicator Score Aggregation

Prompters

- Input *k* factor as a measure of evaluative posture
- Input adjusted fuzzification factor (if desired)
- Input level of refinement of triggers to fire (indicator, subhierarchy, etc.)
- Input target overall score (max/min)

- Input max/min tolerable memberships in linguistic ranges

Triggers

- Collapse interval scores by applying k factor
- Compute membership of discretized interval scores in linguistic set
- Compare all scores to desirability ranges
- Compare memberships in linguistic ranges to thresholds
- Determine min/max *k* value which will yield target range
- Determine the top n (or overall prioritize) indicators whose score would have to be increased to improve the overall score
- Determine the top n (or overall prioritize) indicators whose weights would have to be increased/lowered to improve the overall score
- Determine the overall score delta per unit change in k
- Determine the overall score delta per unit change in v (nu)
- Determine the overall score if "negligible" weighted indicators are removed (proportionally distributing remaining weight)
- Prioritize the contributors to a "design killer" contributor

Alerters

- Notification of which indicators fall below the "acceptable" range
- Notification of unacceptable membership of a score in an undesirable range
- Notification of any "design killer" conditions
- Notification of alert conditions

Informers

- Total and flexible report generation capability (Kiviat diagrams)
- Display how score would change if k factor were optimum
- Display how score would change if most offending scores were changed
- Display how score would change if most over-weighted indicator was changed
- Query of all entries, including input rationale
- Query of knowledge-based entries applied to any given indicator/alerter

7.3. SUMMARY AND CONCLUSIONS

In this chapter we provide motivation, examples, and details of the application of a computer-assisted knowledge-base to a methodology for complex system design evaluation. Beginning with the indicator identification phase as we describe in Chapter 4, we progress through indicator hierarchy construction, indicator weighting (from Chapter 5), and finally indicator scoring, score aggregation, and score interpretation (from Chapter 6).

For each phase in the methodology, we describe several types of knowledge applicable to the phase. Where practical, we provide example or template knowledge-base entries, after which numerous others may be fashioned for a particular system design evaluation scenario. Additionally, without detailed coverage, we identify a sizable complement of actual computer-based uses of the knowledge as we progress from one evaluation phase to the next. In practice, that use of expert knowledge is manifest through interaction with the KBESD.

During the entire evaluation process, much data is input, many facts in the knowledge base examined, many computations done, many comparisons made, and many rules fired. To the extent that potentially undesirable conditions are expressed as facts and related by rules in the knowledge base, computer-aided expressions of expert knowledge can be applied to greatly enhance the credibility of the evaluation. Specifically, formalizing expert knowledge and decisions mitigates the potential side effects of the caprices of subjectivity, which is prevalent in evaluation of designs of the class presently addressed. Applying expert-defined knowledge-base elements throughout each phase of the evaluation, promotes consistent, thorough, and if so applied, constant checks on the reliability of the intermediate and cumulative results of the evaluation.

CHAPTER 8. INDEPENDENT ASSESSMENT OF THE METHODOLOGY

8.1. INTRODUCTION

In this chapter we describe the process and results of conducting an independent assessment of the methodology described in Chapters 3-7. The methodology embodies techniques or procedures which have individually been validated in the literature, e.g., hierarchical decomposition, fuzzy mathematics, interval mathematics, and the Analytic Hierarchy Process. However, the circumscribing methodology, a combination of extant techniques and original contribution, has not been validated as a single, integrated evaluation approach. Consistent with the thrust of the methodology itself, we require an independent assessment of the methodology. To the end of assessing the credibility, usability, and acceptability of the methodology, we undertook the actions described herein.

Formal validation of the methodology would require the analysis of the results obtained by applying the methodology to a statistically suitably-sized sample of existing complex system designs. The sample of designs would need to include both similar and dissimilar designs. However, given (1) the general immediate inaccessibility of a sampling of such designs, (2) the extended duration of the design phase of many complex systems, and (3) the overall limited scope of the one-person effort, a more feasible assessment avenue was pursued in the present research.

In brief, we presented the complete methodology to a panel of experts in the systems, hardware, and software engineering disciplines. We solicited their expert assessment of the methodology with respect to both their own design and design evaluation-related work, and other related work of which they may have experience or knowledge. The results of their assessment have been analyzed and are presented herein.

The remainder of the chapter flows as follows. Section 8.2 describes the actual phases of conducting the independent expert assessment of the methodology. Section 8.3

presents the results of the assessment. Section 8.4 closes the chapter and draws appropriate conclusions.

8.2. CONDUCT OF THE METHODOLOGY ASSESSMENT

The methodology assessment was independently administered by the sponsor of this research project, the Engineering of Complex Systems (ECS) project at the Naval Surface Warfare Center, Dahlgren Division (NSWCDD), in Dahlgren, Virginia. We describe the phases of the assessment process in the following sections.

8.2.1 Development of Methodology Assessment Questionnaire

To facilitate written, documented assessment of the methodology, we created a feedback questionnaire. The questionnaire is provided in Appendix C. The questionnaire consisted of three sections, focusing on three key dimensions of the methodology.

8.2.1.1 Types of Questions

The first part of the questionnaire addresses the degree to which the methodology fulfills the stated objectives under which it was developed. The objectives are stated in Chapter 1, Section 1.3. For each of the 10 objectives, each evaluator is asked "How well does the methodology fulfill its stated Objective X?" The objective is then stated.

The second part seeks assessment of the methodology with respect to the method employed. The six methods addressed are: (1) Indicator-based evaluation, (2) Analytic Hierarchy Process, (3) Fuzzy arithmetic, (4) Expert knowledge-based evaluation, (5) Visual Simulation Environment, and (6) Kiviat diagrams. For each respective method, the evaluators are asked "How much does this method contribute to the achievement of the overall objectives of the methodology?"

The third part requests feedback on the methodology with respect to its features. The four features addressed are: (1) the integration of the methods from Part II, (2) the indicator template hierarchy, (3) the hierarchy assessment questionnaire, and (4) the documentation of the methodology.

A fourth part requests individual evaluator background information, to be provided so as to retain evaluator anonymity. Our intent is to attempt to draw rough distinctions between evaluators based on education, professional background, and work experience in the systems design/engineering field. This information is used to establish evaluator emphasis coefficients, as described in Chapter 4 and employed in Chapter 6.

8.2.1.2 Evaluator Response Formats

For all questions, we offered an assessment format as depicted in Figure 8.1. The format is similar to that described in the methodology, throughout Chapter 6. Scores can range from 0 to 100, and can be expressed as a discrete score, a directly assigned interval, or a linguistic term, implying a fixed interval.

For questions in Part I, responses are requested as in Figure 8.1a. For questions in Part II, responses are requested as in Figure 8.1b. For questions in Part III, responses are requested as in Figure 8.1a. For Part IV, responses are free form.

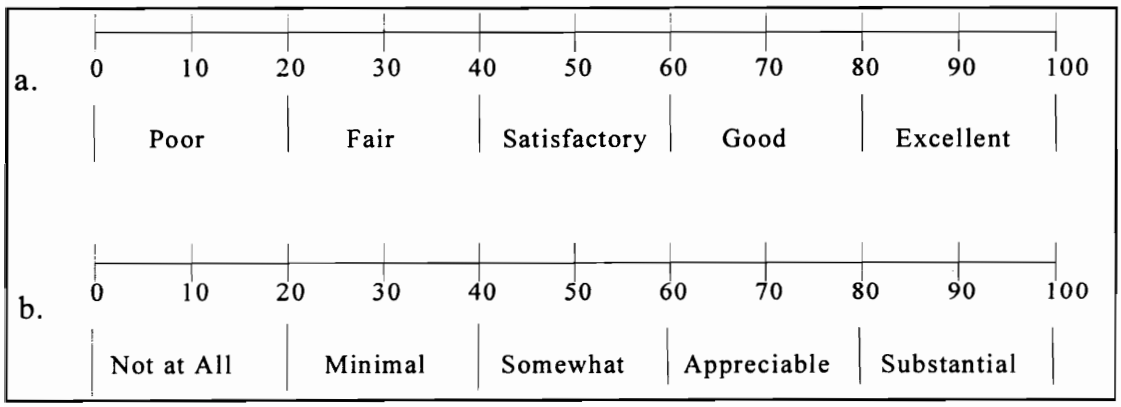


Figure 8.1. Evaluator Response Formats.

8.2.2 Expert Panel Identification

To remove any conceivable bias on the part of the research organization our sponsor, (NSWCDD ECS Program) was tasked to recruit an expert panel to assess the methodology. The sponsor initially compiled a list of over 80 potential evaluators, composed of engineering practitioners and researchers from the Department of Defense, other Federal Government agencies (e.g., NASA), private industry, and academia. Most

of these people have been affiliated with the NSWCDD in systems and software engineering related projects.

Our sponsor invited all potential evaluators they identified to serve on the expert panel, and to attend a seminar at NSWCDD. At the seminar, the methodology was presented in a question and answer forum. Each invitee was provided an executive summary of the research project and instructions for participating in the evaluation. Each evaluator was to provide his or her own funding for the seminar, including travel, meals, and lodging, if necessary. Responses from invitees who volunteered participate in the methodology assessment were returned to our sponsor's point of contact by a specified deadline. Ultimately, a total of 24 volunteer evaluators accepted the invitation to participate in the assessment process. A final list of expert panel members was compiled by our sponsor and forwarded to our research group at Virginia Tech.

8.2.3 Distribution of Methodology Documentation

Each evaluator was sent a copy of the documentation, which is the exact contents of Chapters 1-7 of this document, along with a cover letter from our sponsor, and one from us. Evaluators were instructed to review carefully the problem statement and research objectives stated in Chapter 1, and the high-level, pictorial methodology overview provided in Chapter 3. Chapter 2 was provided as background information. Chapters 4-7 were offered as references for additional detail on the separate major phases of the methodology. Each evaluator received the documentation at least two weeks prior to the seminar.

8.2.4 Presentation of Methodology to the Assessment Panel

The methodology assessment seminar was conducted at the Naval Surface Warfare Center, Dahlgren Division, in Dahlgren, Virginia, on October 30, 1995. Thirteen of the evaluators were present at the seminar.

Just before the presentation, we distributed the methodology assessment questionnaires to the evaluators. We instructed them to familiarize themselves with its

contents. We did so primarily to keep the specific topics to be assessed fresh on their minds, and afford opportunities for evaluators to ask questions specific to the questionnaire's content as the presentation unfolded. Our intent was to reduce relevant information recall demands on evaluators when then they actually completed the questionnaire.

We presented the complete methodology in three separate sessions. The first session laid background for the design evaluation "knowledge gap" problem, for which the methodology is a proposed solution. In this session, we also introduced the indicator-based nature, and detailed the indicator-related topics presented in Chapter 4.

In the second session, we detailed the mathematical portions of the methodology. We demonstrated the concepts of the AHP through an example hierarchy, discussing both the concept of judgment inconsistency, and our Lightweight Factor method for tolerating inconsistency (Chapter 5). We also described the four scoring methods, native score mapping, and score aggregation process, as presented in Chapter 6.

The third session was reserved for presentation of the knowledge-based portion of the methodology, and instructions for completing the methodology assessment questionnaire. Question and answer dialogues were conducted throughout the sessions. One-on-one discussions also took place before, between, and after the presentation sessions, as initiated by individual seminar attendees. At the completion of the presentation, additional questions and concerns from the floor were addressed.

8.3. THE METHODOLOGY ASSESSMENT RESULTS

Verbally, feedback was positive and supportive of the research. Questions posed during the presentation sessions served as a catalyst and forum for clarifying individual and group concerns. Overall, the evaluation panel commended the results of the research, and were impressed with the breadth and degree of detail included in a single dissertation effort.

Formally, assessment results were obtained through the assessment questionnaire, given in Appendix C. Only ten of 24 expert panel members were able to complete and return the questionnaire to our sponsor. The amount of information contained in 276 pages describing the methodology was found by some evaluators to be overwhelming. We believe that the evaluators who did not return the questionnaire could not complete it within the approximately twenty-day time period they were given, due to their busy schedules. In some cases, a few questions on a questionnaire were not given responses by evaluators. We elected to retain the responses which were provided. Since the evaluations were completed in anonymity, we had no means of recovering missing responses.

In the sections to follow, we provide the score results from the ten questionnaires returned, including those minimally incomplete. We adopt a standard presentation format for each question. The results are graphically displayed by using Kiviat diagrams. In each diagram, the individual responses are plotted along radii of the graph, with higher scores plotted away from the origin.

Since interval scores are allowed, we provide the minimum, mean, and maximum scores. Each score was obtained as follows. For responses which were a single discrete value, this value is treated as the min, mean, and max. For responses which were a directly assigned interval, we take the min and max directly from the interval. For linguistic responses, we take the min and max values which define the intervals (see Figure 8.1). In each interval case, the mean value is taken as the midpoint, defined as [(min+max)/2].

Finally, for each major grouping of responses, we provide aggregate results, again presented as Kiviat diagrams. For these figures, the individual response items (questions) are plotted along the circumference of the diagram, and aggregate min, mean, and max scores are shown.

8.2.5 Determining Evaluator Emphasis Coefficients

Consistent with the methodology's directions in Chapter 4, we desire a means of discriminating, by field of expertise, educational background, and length of career, among the evaluators who assessed the methodology. We have selected the criteria depicted below. By applying the AHP, we obtain relative criticalities of each major expertise discrimination indicator and its children indicators, as shown in Figure 8.2.

Indicator	Weight	Sub-Indicator	Weight	Definition
ED. DEGREE	.169			Educational Degree
		PhD	.696	Holds a BS as highest degree
		MS	.229	Holds an MS as highest degree
		BS	.075	Holds a PhD as highest degree
ED. FIELD	.055			Educational field of specialty
		Comp. Sci.	.071	Computer Science
		Math	.130	Mathematics or Applied Mathematics
		Statistics/ Ops. Res.	.311	Statistics or Operations Research
		Systems Engineering	.488	Systems Engineering
YEARS IN PROFESSION	.300			Amount of time working in a field relevant to complex systems
		20+	.579	More than 20 yrs work experience
		10-20	.289	15-20 yrs of work experience
		5-10	.093	5-10 yrs work experience
		< 5	.040	Less than 5 years work experience
WORK FIELD	.389			Occupational field of endeavor
		Systems Integration	.509	Dominant experience in systems integration
		Systems Simulation	.214	Dominant experience in system simulation
		Software	.063	Dominant experience in software-related work
		Hardware	.032	Dominant experience in hardware issues
		Acquisitions	.182	Dominant work experience in systems acquisition
WORK CAPACITY	.088			Capacity in which they spend most of their work
		Design	.111	Primarily functions in component design work
		Develop	.049	Primarily functions in product development
		Evaluate	.453	Primarily functions as design or product evaluator
		Project Mgmt	.386	Primarily functions as project management

Figure 8.2. Indicators and Their Weights for EEC Factor Priority Determination.

Using a ranking scheme provided by the AHP software tool used, we rated each evaluator by selecting the most appropriate subindicator from each indicator group. The determinations were made based on the responses each evaluator gave for Questions 21 and 22 in the questionnaire. In cases where information was not sufficient to make a completely defensible categorization, we extrapolated the information based on what could be reasonably discerned from what was given. The ratings and results are provided in Table 8.1.

8.2.6 Results from Part I, Fulfillment of Objectives

Part I sought feedback regarding the fulfillment of the objectives for the methodology. The scores for the objectives considered together, place the aggregate assessment of the "degree to which the methodology satisfies its objectives" solidly in the fourth highest category (of five). The averaged raw and weighted min, mean, and max for each of the ten objectives are depicted in Tables 8.2 and 8.3, respectively. Figures 8.3 and 8.4 depict these results graphically. In the figure, each objective is numbered along the circumference, and scores increase outward along each radial.

For each individual objective, we interpret the scores in the discussion below. Our discussion linguistically interprets the scores according to the scales defined in Figure 8.1, and offers additional commentary as needed.

Table 8.1. Evaluator Ratings and EECs.

Evaluator	Ed. Degree	Ed. Field	Yrs. Worked	Work Field	Work Capacity	EEC
1	PHD	COMPSCI	20+	SOFTWARE	EVALUATE	**0.098**
2	BS	MATH	5-10	SOFTWARE	EVALUATE	**0.035**
3	MS	COMPSCI	20+	SIMULATE	DEVELOP	**0.086**
4	PHD	STAT/OR	20+	ACQUISIT	PROJMGMT	**0.115**
5	MS	SYSENGIN	5-10	SYSINTEG	DESIGN	**0.091**
6	BS	COMPSCI	20+	SOFTWARE	EVALUATE	**0.074**
7	MS	MATH	20+	SYSINTEG	EVALUATE	**0.135**
8	PHD	STAT/OR	20+	SYSINTEG	EVALUATE	**0.157**
9	MS	MATH	20+	SOFTWARE	DEVELOP	**0.068**
10	MS	MATH	20+	SOFTWARE	DEVELOP	**0.068**
11	PHD	MATH	< 5	SIMULATE	EVALUATE	**0.073**

Table 8.2. Average Raw Scores for the Objectives.

Objective	Min	Mean	Max
1	62.3	67.3	72.3
2	53.2	59.5	65.9
3	67.9	74.3	80.6
4	63.8	71.5	79.3
5	77.1	83.7	90.3
6	73.8	79.8	85.8
7	63.2	69.8	76.4
8	55.5	61.8	68.2
9	58.2	65.7	73.2
10	64.1	70.4	76.6

Table 8.3. Weighted Average Scores for the Objectives.

Objective	Min	Mean	Max
1	63.1	69.4	75.7
2	54.5	61.4	68.3
3	66.1	73.3	80.5
4	62.9	71.2	79.4
5	77.6	85.5	93.3
6	72.2	79.3	86.5
7	63.8	71.6	79.4
8	55.9	63.4	71.0
9	59.1	66.9	74.7
10	62.0	69.4	76.9

Objective 1: How well does the methodology fulfill its stated Objective 1?

The methodology should facilitate the measurement and evaluation of a complex system design's dynamic (i.e., time-critical, performance critical, mission-critical) characteristics.

The scores for this objective fall within the fourth highest category, with a linguistic value of *good*. At least satisfactory achievement of this objective is crucial to the ultimate value of the methodology. The VSE is the primary component of the methodology which allows measurement and evaluation of time-critical system design elements. Technical and domain expertise provides assessment of the performance-related aspects. Operational domain expertise is the best source for mission-related assessments of design utility.

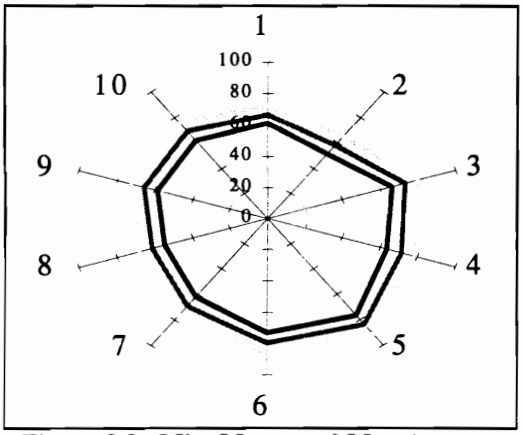


Figure 8.3. Min, Mean, and Max Average Raw Scores for Each Objective.

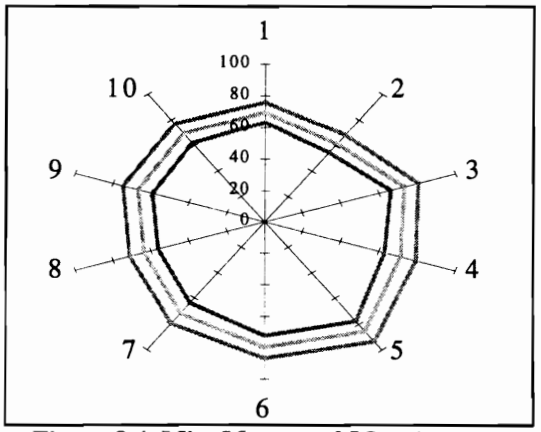


Figure 8.4. Min, Mean, and Max Average Weighted Scores for Each Objective.

Objective 2: How well does the methodology fulfill its stated Objective 2?

The methodology should facilitate the measurement and evaluation of a complex system design's real-time characteristics.

The scores for this objective straddle the border of the third and fourth highest categories, with a linguistic value between *satisfactory* and *good.* Satisfaction of this objective is related to the perceived value of the VSE, since real-time element measurement is based on a designed system's projected behavior. The VSE itself received a middle grade. Thus, a grade for this objective tending toward the center of the acceptability scale is consistent with the assessment of the tool which is intended to provide this capability. Further analysis of the VSE's rating is deferred until Part II.

Objective 3: How well does the methodology fulfill its stated Objective 3?

The methodology should facilitate the integrated measurement and evaluation of a complex system design with respect to its hardware, software, and humanware components, and their interfaces.

The scores fall solidly in the fourth highest category, with a linguistic value of *good.* Given that this is the bedrock concept for the entire indicator-based nature of the methodology, an above average score here is critical. In fact, it is one of the three highest scored objectives. We may have expected an even higher aggregate score on this particular objective. However, analysis of the individual scores from each evaluator revealed an overall reluctance to score on *either* extreme.

Objective 4: How well does the methodology fulfill its stated Objective 4?

The methodology should facilitate the qualitative and quantitative evaluation of a complex system design.

The scores fall solidly in the fourth highest category, with a linguistic value *good.* This score is quite expected. With the incorporation of both qualitatively and quantitatively scored indicators, taking advantage of interval and fuzzy linguistic scoring, this objective was solidly provided for in the methodology.

Objective 5: How well does the methodology fulfill its stated Objective 5?

The methodology should be generically applicable for a broad spectrum of complex system designs.

With the bare exception of the average minimum score, these scores fall solidly in the highest category, with a linguistic value *excellent*. In fact, applying the EECs actually bolsters the scores. This is the most highly scored objective of the set. This particular objective, among those of described by Verma [91] is another of the *sine qua non* for the methodology. High marks here reflect our effort to maintain a high degree of genericity for the methodology. We effected this through the generic hierarchy template and the broadly applicable mathematical techniques.

Objective 6: How well does the methodology fulfill its stated Objective 6?
The methodology should lend itself to the incorporation of computer-aided assistance.

These scores fall solidly in the upper half of the fourth highest category, with a linguistic value *good*. The average maximum score is well within the *excellent* range. The KBESD is the primary source for ensuring this objective is met. As discussed in Chapters 3 and 7, the KBESD is a software workhorse, providing computer assistance to nearly every phase of the methodology. As discussed in Chapter 5, AHP software tools such as *Expert Choice* facilitate all phases of conducting the determination of indicator relative criticality. High marks on this objective were certainly expected.

Objective 7: How well does the methodology fulfill its stated Objective 7?
The methodology should possess inherent, piece-wise credibility.

These scores fall solidly in the fourth highest category, with a linguistic value *good*. These marks were no less than expected. We were careful to incorporate proven methods and techniques from the literature, such as the AHP, Fuzzy Mathematics, and knowledge-based assistance.

Objective 8: How well does the methodology fulfill its stated Objective 8?
The methodology should be easily usable by a design evaluation organization.

These scores straddle the third and fourth highest category, with linguistic values between *satisfactory* and *good*. We attribute a slight downward turn in the otherwise category four scoring trend to several factors. First, the majority of evaluating personnel

had no hands-on experience with the KBESD, the user-friendly software centerpiece of the methodology. We are confident that greater familiarization with this tool, especially as its functionality and user interface mature, will alleviate the initial usability concerns revealed here.

Second, applying the techniques of the AHP and Fuzzy mathematics, is a new experience, if not a new consideration for most of the evaluators. One notable exception is an evaluator whose doctorate and career work focus on machine learning and artificial intelligence.

To summarize the second factor, while the techniques themselves are not unnecessarily burdensome, especially with software support and simplifying approaches described in the literature (e.g., [Harker 87a,b; Wedley 93]), general inexperience with the techniques naturally invokes some initial resistance.

Third, like the KBESD, the VSE is not a familiar tool to the evaluators outside the NSWC ECS group. The same reasoning as for the KBESD clearly accounts for reduced perceived ease of use of the methodology.

Objective 9: How well does the methodology fulfill its stated Objective 9?

The methodology should be applicable for the preliminary and detailed design phases of the system engineering process.

With the exception of the average minimum score, these scores fall primarily in the fourth highest category, with a linguistic value of *good*. Upon applying the EECs, all scores receive a *good* rating. This comes as no surprise. We attribute this above average mark to the inclusion in the methodology of an indicator decomposition approach, and of both precise metrics and imprecise intervals. Having done so, we allow for the incorporation of as much (or as little) detail as can be credibly included in a given design evaluation task.

Objective 10: How well does the methodology fulfill its stated Objective 10?

The methodology should promote independent system design evaluation to prevent developer's bias.

These scores fall in the fourth highest category, with a linguistic value of *good.* Chapter 3 motivates the need for an *independent* evaluation of complex system designs. From the outset, the methodology is intended to be applied by an organization which possesses the capability to obtain the evaluative expertise it needs, To that organization, we supply techniques and software tools. So equipped, an independent evaluating organization can conduct an evaluation free from any pressures associated with intra-organizational allegiances.

8.2.7 Results from Part II, Contribution of Individual Methods

Table 8.4 and Figure 8.5 present the individual values for and each of the six methods. Table 8.5 and Figure 8.6 depict the results after applying the EECs derived as discussed in Section 8.3.1. As in Section 8.3.2, these scores represent the average of the scores from all evaluators. Both the raw and weighted score sets place the aggregate score for all methods solidly in the fourth highest category, *good.* The two lowest scored methods are the Visual Simulation Environment and the Kiviat diagram. The results for the individual methods are discussed below.

Method 1: Indicator-Based Evaluation Method

How much does the use of this method contribute to the achievement of the overall objectives of the methodology?

This method received the highest aggregate score of the six methods listed in the assessment questionnaire, and the second highest scored aspect of the methodology. Linguistically, an aggregate score high in the *appreciable* category is obtained. The

Table 8.4. Average Raw Min, Mean, and Max Scores for each Method.

Method	Min	Mean	Max
1	73.7	80.2	86.7
2	65.2	71.7	78.2
3	63.0	68.8	74.5
4	64.5	70.8	77.0
5	51.0	57.8	64.5
6	48.0	55.0	62.0

Table 8.5. Weighted Average Min, Mean, and Max Score for each Method.

Method	Min	Mean	Max
1	73.2	80.6	87.9
2	63.9	71.3	78.6
3	64.9	71.4	78.0
4	65.2	72.3	79.4
5	48.0	55.6	63.3
6	48.8	56.3	63.9

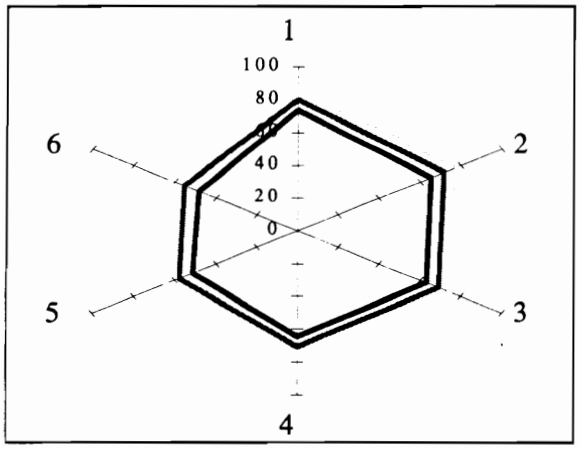


Figure 8.5. Min, Mean, and Max Average Scores for Each Method.

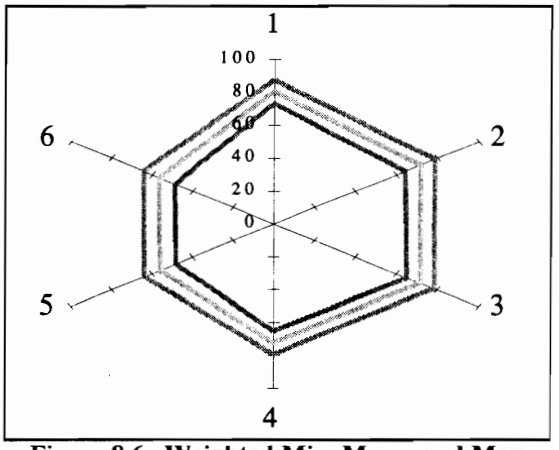


Figure 8.6. Weighted Min, Mean, and Max Average Scores for Each Method.

indicator-based approach promotes decomposition, division of labor, and pinpointing of the appropriate level of expertise for indicator evaluation. These are foundational to the methodology.

Method 2: Analytic Hierarchy Process Method

How much does the use of this method contribute to the achievement of the overall objectives of the methodology?

This indicator received an overall rating of *appreciable*. We specifically selected the AHP as the primary method of obtaining indicator weights of criticality because of its wide acceptance in the literature, and applicability to a large number of problems. Additionally, numerous short cut techniques and software tools have become available to make the AHP a credible, traceable, mathematically sound procedure to account for differences in importance and preference among otherwise equal elements.

Method 3: Fuzzy Arithmetic Method

How much does the use of this method contribute to the achievement of the overall objectives of the methodology?

Like the AHP, this mathematically-oriented technique was also viewed as a valued contributor to the achievement of the overall objectives. Its scores, only slightly lower than those for the AHP, earn it a linguistic score of *appreciable*, as well. We find as a result of analyzing the background information provided by each evaluator, the

common trait of a strong background in mathematics. Consequently, we are pleased and not surprised to find a large agreement of the value added by this particular method.

Method 4: Expert Knowledge-Based Method

How much does the use of this method contribute to the achievement of the overall objectives of the methodology?

This method was also very well received, with scores falling squarely in the fourth highest of five categories. Chapter 7 presents a broadly scoped, example-oriented, and solid case for the motivation and potential application of a computer-assisted expert knowledge-based component of the methodology. The overall score of *appreciable* affirms our conviction that this approach meets some of the needs for computer assistance in a design evaluation methodology, as discussed by Verma [91].

Method 5: Visual Simulation Method

How much does the use of this method contribute to the achievement of the overall objectives of the methodology?

The aggregate scores for this method trail those of methods 1-4. The mean score for the VSE places it just above center in the diagram of Figure 8.1b, yielding a score between *somewhat and appreciable*. As we explained in our analyses of the results for objectives 2 and 8, we believe limited hands-on experience with the VSE itself, and limited application of the VSE to problems of the scale addressed by the methodology are largely responsible for a less than enthusiastic rating.

The VSE is currently under beta-version development. We are encouraged by the acceleration of VSE development, and the fact that the VSE is slated for testbed use in some of the NSWC divisions. We greatly anticipate that the tool in a more mature form will be very well received and in short order embraced by the design evaluation community.

Method 6: Kiviat Diagram Method

How much does the use of this method contribute to the achievement of the overall objectives of the methodology?

We do not have a great deal of insight as to which this method received the lowest score of the 20 elements addressed in the present methodology assessment endeavor. We charged each evaluator to provide constructive criticism of any facet of the methodology which they scored low. For the several scores which fell below 50 on the evaluation questionnaires, no such feedback was provided. Since the average score for this facet is still in the high end of the *somewhat* category, we can proceed with the confidence that this method is not considered detrimental to the overall methodology.

8.2.8 Results from Part III, Evaluation of the Methodology's Features

In part III, four particular features of the methodology are assessed. The aggregate scores for the set of features fall within the *good* category, slightly below the midpoint of the region. The scores for each feature, averaged over all evaluators, are presented in Table 8.6 and Figure 8.7. The scores with the EECs applied are presented in Table 8.7 and Figure 8.8.

Feature 1: How well are the methods in Part II integrated within the methodology?

Scores for this feature place it in the *good* category. The KBESD is the software centerpiece of the methodology, which is intended to provide an integrating framework for the techniques described in Chapter 6. At present, the AHP functionality is not included in the KBESD, but we have recommended a useful software tool which is readily available and relatively inexpensive. If the methodology is executed from the methods in Chapter 4, sequentially through the methods in Chapter 6, the integration through pipelining is evident. Finally, the whole knowledge-based feature, implemented in the KBESD is a significant integrating force. Rules and facts regarding every phase of

Table 8.6. Average Raw Min, Mean, and Max Scores for each Method.

Feature	Min	Mean	Max
1	65.2	71.3	77.4
2	53.3	60.0	66.7
3	52.6	58.9	65.3
4	68.6	74.4	80.2

Table 8.7. Average Weighted Min, Mean, and Max Scores for each Method.

Feature	Min	Mean	Max
1	59.5	65.9	72.3
2	49.4	56.9	64.4
3	46.5	53.8	61.0
4	70.7	77.2	83.6

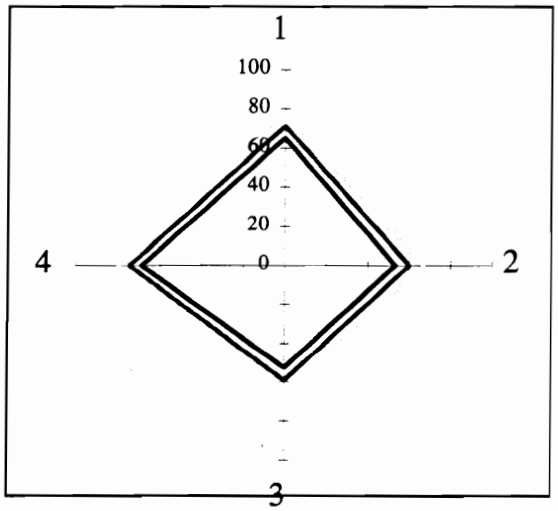


Figure 8.7. Min, Mean, and Max Average Scores for Each Feature.

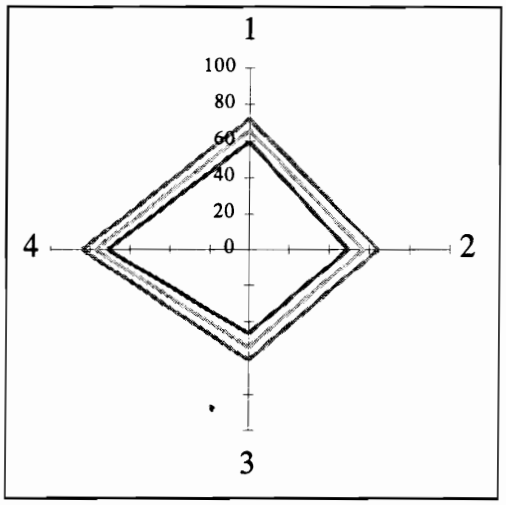


Figure 8.8. Min, Mean, and Max Weighted Average Scores for Each Feature.

the methodology can be accounted for in the knowledge-based aspect of the methodology.

Feature 2: How good is the example system design quality indicator hierarchy in Appendix A?

This feature received an aggregate score in the upper half of the *satisfactory* category. Since the Appendix was over 18 pages, and used a very small typesize, we can agree with the evaluators that the Appendix as presented is not particularly appealing. One evaluator commented that the reader easily loses the exact degree of intention across a page boundary. The evaluator recommended we augment this feature to include a "generation" number for each level of indenture. In printed form, this added feature would indeed be valuable. We have included the suggestion in Chapter 9, as a recommendation for future work.

Unfortunately, the KBESD version of the hierarchy was not available to the readers. In the KBESD form, each indicator has an inspector which contains additional information. Additionally, the graphical hierarchy browser serves very well to maintain user orientation of depth in the hierarchy. While screen dumps of portions of the

hierarchy are available in the document presented to the evaluators, the question addressed the printed version.

Feature 3: How well does the questionnaire in Appendix B assess the evaluative credibility of indicator hierarchy?

Aggregate scores for this indicator place it in the upper half of the *satisfactory* category. We firmly believe in the concept of hierarchy assessment, and are pleased that this method for assessing it is positively received. Since the score for this feature trailed other methodology elements, we will consult with the sponsor regarding enhancing this feature.

Feature 4. How well is the methodology documented?

This question refers to the printed version of the methodology each evaluator received. Scores for this indicator were, with one glaring exception, very high. One evaluator offered a discrete score of 35, which dragged the average score down 5 points. Without this anomalous entry, this feature has the third highest score among all 20 questions. Still, the average score is in the very high end of the *good* category. We provided examples, table, figures, and equations. We specifically included the storyboard-based pictorial overview of the methodology in Chapter 3 for the purpose of providing a top-down structured approach, so as to decrease the complexity of the document.

8.3. CONCLUSIONS

The aggregate mean scores for all ten objectives, the six methods, and the four features were respectively 70, 68, and 68. Each of these scores falls solidly in the fourth highest category, linguistically labeled *good* or *appreciable*. Applying the EECs altered the scores by less than five percent, yielding 72, 68, 65, respectively. Where aggregate scores for individual items appear to indicate a general lower acceptance by the evaluators, we have offered explanations for likely causes.

In retrospect, we see the potential for a measurably higher assessment of the methodology had the VSE and KBESD software environments been available to the evaluators during a presentation of the methodology. That this was needed is clear from our analyses of the methodology components which these tools primarily affect, i.e., Objectives 2 (system dynamic behavior) and 8 (methodology ease of application), Method 5 (the VSE), and Feature 2 (template indicator hierarchy).

9. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE RESEARCH

9.1. CONCLUSIONS

The methodology described was funded in part by the Engineering of Complex Systems research program of the Office of Naval Research. In a larger sense, however, it is an answer to the systems engineering community's call for new approaches to design evaluation. The objective of both groups is assurance of system design utility and quality earlier in the system design life cycle.

To answer the call, we reviewed the literature for approaches, methods, and indicators currently applied to performance and dependability of most disciplines intersected by modern complex systems. These disciplines include software, hardware, hardware-software interaction, human workload, human-machine, and human-software interaction. We also reviewed the literature for the application of performance and reliability prediction approaches in a wide range of systems, including: the space station, combat aircraft, surface seagoing vessels, submarines, tanks, rocket booster systems, nuclear power plant control rooms, high speed communications switches, large-scale off-shore oil rigs, robotics-controlled manufacturing, toxic waste removal, and corporate work places.

Our findings from this research are largely documented as a collection of generic design and utility indicators, which we intend to serve primarily as a template and launching point for evaluating any complex system. The template hierarchy of indicators also serves to punctuate and further define the complex and critical nature of the systems whose designs our methodology is used to evaluate.

A methodology such as ours which assists in the evaluation of designs for so broad yet complex an array of systems, must both incorporate and account for many quantitatively and qualitatively expressed assessments. Further, the earlier in the system

design cycle these assessments are applied, they may be heavily based on imprecise human judgment. None the less, organizations which require, fund, and eventually operate these systems, demand mathematically sound, reproducible, and traceable documentation of all design evaluation decisions. Evaluators subsequently require methods and techniques which are not cumbersome to apply, allow flexibility in inputs, and lend themselves to computer assistance and a large degree of self-documentation.

We developed a methodology to address these needs. The indicator-based approach facilitates complexity reduction through division of labor across the evaluating panel. It incorporates hierarchical decomposition of design utility indicators from the systems and operational level, to inter-domain experts, then to technical domain experts and metricians. The Analytic Hierarchy Process accounts for human judgment and its inherent inconsistency. Metrics take advantage of what is known and widely applied in each system-related discipline. System simulation promotes understanding of the dynamic and behavioral related aspects of the designed system. Interval judgments and fuzzy linguistic scoring techniques allow controlled, credible accounting for what is not or, so early in the system life cycle, cannot be known with any precision. Finally, incorporation of a computer-assisted expert knowledge base provides automated input and result checks, resulting in alerts to the evaluators. This facet of the methodology is a primary source of complexity management and assessment reliability, in the face of potentially large indicator hierarchies.

Taken together, these methods define a methodology to be applied by an organized, appropriately staffed panel of experts from all the disciplines embodied in the designed system. Applied in this way, the methodology can be used to provide system sponsors with expert assessment and feedback regarding the utility potential of a system design.

To determine how useful the methodology can be, a panel of experts from industry, the DOD, federal government, and academia evaluated the applicability and usefulness of the methodology. The panel pointed out directions for improvement and

expansion, while commending the effort and the product. Overall they assessed the methodology as satisfying or exceeding their base-level desiderata for such a methodology. We can state then that the research effort resulted in a sound foundational methodology for complex system design evaluation.

One of the initial goals for this research was the comprehensive definition and elaboration of such a methodology as presented herein. Some debate focused on the relative merits of providing significant depth-wise elaboration of a few methods versus comprehensive breadth-wise coverage of the complete methodology. The resultant methodology is largely a manifestation of the latter, with deference to the former.

In conclusion, the methodology has been defined by the scope of coverage, depth of content, and order of application detailed in this document. Where coverage may appear narrow, depth shallow, or order out of sync with practiced techniques, we recommend future research and development efforts in Section 9.2 to yield appropriate enhancements. Springboarding from independent expert affirmation of its potential as a credible design evaluation approach, we anticipate significant ultimate contribution to the design synthesis and assessment field and literature.

9.2. RECOMMENDATIONS FOR FUTURE RESEARCH

In this section, we suggest future research and development activities which will ultimately enhance the methodology and its successful employment.

As presented in Chapter 2, we conducted a significant survey of the literature, from individual complex system domains, to inter-domain subjects, to operational systems, to design evaluation techniques. The results of the technique-based research are primarily reflected in the methodology itself. The indicator hierarchy we have developed for the KBESD is the primary outgrowth of the indicator-related research. Not all of the indicators uncovered in the literature have been explicitly defined and positioned in the hierarchy. The overwhelming causes are: (1) the absence of an actual systems-level perspective on which to base a comprehensive top-down decomposition, and (2) the

abundance in the literature of many lower-level, domain-specific indicators, for which no appropriate intermediate links to the system-level indicators have been generated or are available.

To extend or augment the existing hierarchy and to increase its value as a device for design evaluation, we recommend the following actions be pursued in future dedicated research.

- Establish an initial systems-level perspective on an existing complex system design or requirements specification.
- Proceed with indicator decomposition based on interviews with or in legion with systems engineers and/or domain experts who are working on or have worked on a real project.
- Identify key non-structural relationships known, if not otherwise formally expressed in practice.
- Consult systems engineers and domain experts as a source for intermediate indicators which allow incorporation of existing bottom-up indicator relationships.
- Based on inputs from experts at the methodology assessment seminar, in consultation with practicing design evaluators, define and incorporate System Design Goals as a subhierarchy.

The KBESD is the software centerpiece of the methodology, as described in Chapter 3. This single piece of software bears the weighty burdens of indicator storage and retrieval, score computation, and eventually the knowledge-based inferencing functions. To enhance the readiness of the KBESD for its increasing role in the methodology, we recommend the following new or continued efforts.

- Inclusion of the expert system shell to facilitate user interface to the knowledge base (underway as an independent study project).
- Inclusion of the capability to generate knowledge base facts from user entries in the indicator inspector.
- Inclusion of the capability to generate knowledge base rules from "prompter" panels, e.g., for threshold values, fuzzification factors, etc., which apply to an indicator.
- Enhance the user interface to facilitate a user-friendly prune and graft capability.
- In the inspector panel, include the capability to build automatic links between literature sources abbreviated in the literature to a full bibliography.
- Include a capability to merge existing hierarchies while retaining common links in both.

- Include an insert capability to paste in a hierarchy from another file to a given node in an existing hierarchy.
- Include a capability to import a file from e.g., a Dbase, Access, Paradox, or FoxPro format.
- As recommended by a member of the methodology assessment panel, include a "generations" capability in the report generator, to identify which level of indenture applies to an indicator in the report.

We describe in Chapter 8 the motivation for an initial subjective, qualitative evaluation of the methodology. An appropriate follow up research effort is the direct application of the methodology to many design evaluation problems of varying, but non-trivial scales. The goals of the research would be to identify specific enhancements to improve the methodology.

As a result of our presentation of the methodology to experts in the complex systems design evaluation field, a representative from the NASA organization responsible for design evaluation of the International Space Station expressed an interest in applying the methodology to the Station's design evaluation efforts. Discussions are ongoing with our sponsor and NASA for potential application of the methodology to a real-world problem.

As a follow up effort to the present work, the sponsor has expressed a desire for extending the methodology to earlier phases in the system life cycle. This amounts to an ability to apply the methodology closer toward the requirements phase, to assist in *design synthesis*. Such a methodology would serve to close the gap between the present work and conceptual design evaluation [Verma 94].

A second follow up direction for the current work is in providing a greater elaboration of its knowledge-based aspect. Research should be conducted to (1) establish knowledge requirements, (2) perform knowledge acquisition from design evaluation experts, and (3) carry out the knowledge engineering required to map the elaborated knowledge elements into a fully integrated expert knowledge base within the KBESD.

BIBLIOGRAPHY

Aas, E.J., K. Klingsheim and T. Steen (1992), "Quantifying Design Quality: A Model and Design Experiments," In ***Proceedings of EURO ASIC '92***, 172-177.

Aas, J.H., K. Brathen, E. Nordo and O.O. Orpen (1989), "Man-Machine Interface in a Submarine Command and Weapon Control System: Features and Design Experiences," In ***IFAC Proceedings Series***, 3, 189-194.

Abbot, R.K. and H. Garcia-Molina (1990), "Scheduling I/O Requests with Deadlines: A Performance Evaluation," In ***Proceedings of the IEEE Real-Time Systems Symposium***, D. Locke (Ed.), 113-124.

Aczel, J. and T. Saaty (1983), "Procedures for Synthesizing Ratio Judgments," ***Journal of Mathematical Psychology 27***, 93-102.

Al-Janabi, A. and E. Aspinwall (1993), "An Evaluation of Software Design using the DEMETER Tool, ***Software Engineering Journal 6***, 319-324.

Allen, C.D. (1995), "Succeeding as a Clandestine Change Agent," ***Communications of the ACM 38***, 5, 81-86.

Alqadi, R. and P. Ramanathan (1993), "Architectural Synthesis of Mission-Critical Computing Systems," In ***Proceedings of the Complex Systems Engineering Synthesis and Assessment Technology Workshop (CSESAW '93)***, S.L. Howell and W. Farr (Eds.), Naval Surface Warfare Center Dahlgren Division, Dahlgren, VA, 185-192.

Amalberti, R. (1992), "Safety in Process-Control: An Operator-Centered Point of View," ***Reliability Engineering and System Safety 38***, 99-108.

Andert, E. and L. Peters (1993), "An Intelligent Real-Time System Assessment Tool," In ***Proceedings of the Complex Systems Engineering Synthesis and Assessment Technology Workshop (CSESAW '93)***, S.L. Howell and W. Farr (Eds.), Naval Surface Warfare Center Dahlgren Division, Dahlgren, VA, 193-197.

Apostolou, B. and J.M. Hassell (1993), "An Empirical Examination of the Sensitivity of the Analytical Hierarchy Process to Departures from Recommended Consistency Ratios," ***Mathematical and Computer Modelling 17***, 4/5, 163-170.

Arakawa, M. and H. Yamakawa (1993), "Study on Structural Optimum Design Based on Qualitative Sensitivities," ***Transactions of the Japan Society of Mechanical Engineers, Part C 59***, 558, 114-121.

Arlat, J., Kanoun, K. and J.C. Laprie (1990), "Dependability Modeling and Evaluation of Software Fault-Tolerant Systems," ***IEEE Transactions on Computers 39***, 4, 504-513.

Arthur, J.D., O. Balci and R.E. Nance (1993), "Tomahawk Weapon Control System. Knowledge-Based Support System for Software Evolution: Establishing Software Development Process Control Through Process and Product Quality Indicators," ***Final Report, SBIR Contract No. N60921-92-C-A344***, Simulation and Software Engineering, Inc., Blacksburg, VA.

Arthur, J.D. and R.E. Nance (1991), "A Framework for Assessing the Adequacy and Effectiveness of Software Development Methodologies, Technical Report SRC-91-005, Systems Research Center, Virginia Tech, Blacksburg, VA.

Arthur, J.D., R.E. Nance and S.M. Henry (1987), "A Procedural Approach to Evaluating Software Development Methodologies: the Foundation, Technical Report TR 86-24, Department of Computer Science, Virginia Tech, Blacksburg, VA.

Ashlund, S. and D. Hix (1992), "Ideal. a Software Tool to Evaluate Interface Usability," In ***Proceedings of the Human Factors Society 36***, 1, 414-417.

Ayani, R. (1993), "Parallel Simulation," In ***Performance Evaluation of Computer and Communications Systems: Joint Tutorial Papers of Performance '93 and Sigmetrics '93 Proceedings***, L. Donatiello and R. Nelson (Eds.), Springer-Verlag, Inc., New York, 1-20.

Ayyub, B.M. and Z.A. Eldukair (1989), "Safety Assessment Methodology for Construction Operations," In ***Proceedings of the 5th International Conference on Structural Safety and Reliability (ICOSSAR 89)***, A. Ang, M. Shinozuka, G. Schueller (Eds), 771-777.

Balci, O., A.I. Bertelrud, C.M. Esterbrook, and R.E. Nance (1995), "A Picture-Based Object-Oriented Visual Simulation Environment," In ***Proceedings of the 1995 Winter Simulation Conference***, IEEE, Piscataway, New Jersey.

Balci, O. and R.E. Nance (1987), "Simulation Model Development Environments: A Research Prototype," ***Journal of the Operational Research Society 38***, 8, 753-763.

Balci, O. and R.E. Nance (1985), "Formulated Problem Verification as an Explicit Requirement of Model Credibility," ***Simulation 45***, 2, 76-86.

Balci, O., D. DeVaux and R. Nance (1993), "Measurement and Evaluation of Complex Navy System Designs," In ***Proceedings of the Complex Systems Engineering Synthesis and Assessment Technology Workshop (CSESAW '93)***, S.L. Howell and W. Farr (Eds.), Naval Surface Warfare Center Dahlgren Division, Dahlgren, VA, 126-140.

Ballas, J.A., C.L. Heitmeyer and M.A. Perez (1992), "Evaluating Two Aspects of Direct Manipulation in Advanced Cockpits," In ***ACM Conference on Human Factors in Computing Systems - CHI '92***, 127-134.

Banai-Kashani, R. (1989), "New Method for Site Suitability Analysis: The Analtic Hierarchy Process," ***Environmental Management 13***, 6, 685-693.

Banks, J., J. Carson and B. Nelson (1995) ***Discrete Event System Simulation,*** Prentice Hall, Inc., Englewood Cliffs, N.J.

Banks, W.W., E.E. Schultz, Jr. and E. Crane (1988), "The Assessment of Human/Computer Performance: A Case for Connectivity," In ***Proceedings of the Hawaii International Conference on System Science 21***, 725-731.

Banks, W.W., E.E. Schultz, Jr. and E. Crane (1988), "The Assessment of Human/Computer Performance: A Case for Connectivity," In ***Proceedings of the Hawaii International Conference on System Science 21***, 725-731.

Barker, S. (1994), "Personal correspondence.

Barnett, B.J. (1990), "Aiding Type and Format Compatibility for Decision Aid Interface Design," In ***Proceedings of the Human Factors Society 34***, 1, 1552-1556.

Barnett, B.J., C.J. Arbak, J.L. Olson and L.C. Walrath (1992), "A Framework for Design Traceability," In ***Proceedings of the Human Factors Society 36***, 1, 2-6.

Basak, I. (1993), "Incorporating Within-Pair Order Effects in the Analytic Hierarchy Process," ***Mathematical and Computer Modelling 17***, 4/5, 83-92.

Basak, I. and T.L. Saaty (1993), "Group Decision Making Using the Analytic Hierarchy Process," ***Mathematical and Computer Modelling 17***, 4/5, 101-109.

Bea, R. and W. Moore (1994), "Reliability Based Evaluations of Human and Organization Errors in Reassessment and Requalification of Platforms," In ***Proceedings of the 13th International ASME Offshore Mechanics and Arctic Engineering Conference, 1994, OMAE II, Safety and Reliability***, C.S. Soares (Ed.), 211-226.

Becker, A.B., J.S. Warm, W.N. Dember and P.A. Hancock (1991), "Effects of Feedback on Perceived Workload in Vigilance Performance," In ***Proceedings of the Human Factors Society 35***, 2, 1491-1494.

Beounes, C., et al. (1993), "SURF-2: A Program for Dependability Evaluation of Complex Hardware and Software Systems," In ***Digest of Papers - International Symposium on Fault- Tolerant Computing***, IEEE Computer Society Press, Los Alamitos, CA, 668-673.

Beyer, H.R. and K. Holtzblatt (1995), "Apprenticing With the Customer," ***Communications of the ACM 38***, 5, 45-52.

Bhargava, H. (1995), World Wide Web page at URL http://bhargava.as.nps.navy.mil/www/hbk/research/dss.html

Bittner, A.C. Jr (1990), "Human Factors Measurement: Nature, Problems and Strengthening," In ***Proceedings of the Human Factors Society 34th Annual Meeting 34***, 1253-1257.

Blackman, H.S. (1990), "Modeling the Influence of Errors of Commission on Success Probability," In ***Proceedings of the Human Factors Society 35th Annual Meeting 35***, 1085-1087.

Blackwell, J.S. and D.L. Cuomo (1991), "Evaluation of a Proposed Space and Missile Warning Symbology Standard for Graphical Displays," In ***Proceedings of the Human Factors Society 35***, 1, 102-106.

Blanchard, B.S. and W.J. Fabrycky (1990) ***Systems Engineering and Analysis***, Second Edition., Prentice Hall, Inc., Englewood Cliffs, N.J.

Blankmeyer, E. (1987), "Approaches to Consistency Adjustment," ***Journal of Optimization Theory and Applications 54***, 3, 479-488.

Boender, C., J.G. de Graan and F.A. Lootsma (1989), "Multi-Criteria Decision Analysis with Fuzzy Pairwise Comparisons," ***Fuzzy Sets and Systems 29***, 133-143.

Boender, C., J.G. de Graan and F.A. Lootsma (1989), "Multi-Criteria Decision Analysis with Fuzzy Pairwise Comparisons," ***Fuzzy Sets and Systems 29***, 133-143.

Bowen, J. and V. Stavridou (1993), "Safety-Critical Systems, Formal Methods and Standards," ***Software Engineering Journal 4***, 189-209.

Boy, G.A. (1987), "Operator Assistant Systems," ***International Journal of Man-Machine Studies 27***, 5-6, 541-554.

Bradlow, H.S. (1990), "Performance Measures for Real-Time Continuous Bit-stream Oriented Services: Application to Packet Reassembly," ***Computer Networks and ISDN Systems 20***, 15-26.

Brehm, E., R. Goettge and F. McCaleb (1992), "START/ES -- An Expert System Tool for System Performance and Reliability Analysis," In ***Proceedings of the Complex Systems Engineering Synthesis and Assessment Technology Workshop (CSESAW '92)***, S.L. Howell and P.Q. Hwang (Eds.), Naval Surface Warfare Center Dahlgren Division, Dahlgren, VA, 303-318.

Brehm, E.W. (1994), "System Dependability Assessment Tool," In ***Proceedings of the Complex Systems Engineering Synthesis and Technology Assessment Workshop -- CSESAW '94***, S.L. Howell, M. McCoy (Eds.), 51-56.

Brown, C.B. and J.T.P. Yao (1983), "Fuzzy Sets and Structural Engineering," ***Journal of Structural Engineering 109***, 5, 1211-1225.

Brown, C.E., S.J. Swierenga and A.R. Wellens (1991), "Social Psychological Metaphors for Human-Computer System Design," In ***IEEE Proceedings of the National Aerospace and Electronics Conference***, 2, 793-799.

Brun-Cottan, F. and P. Wall (1995), "Using Video to Re-Present the User," ***Communications of the ACM 38***, 5, 61-70.

Buckley, J.J. (1985), "Ranking Alternatives Using Fuzzy Numbers," ***Fuzzy Sets and Systems 15***, 21-31.

Buckley, J.J. (1985), "Ranking Alternatives Using Fuzzy Numbers," ***Fuzzy Sets and Systems 15***, 21-31.

Butler, R.W. and G.B. Finelli (1993), "The Infeasibility of Quantifying the Reliability of Life-Critical Real-Time Software," In ***IEEE Transactions on Software Engineering 19***, 1, 3-12.

Byers, J.C., J.L. Harbour and C.A. Wilhelmsen (1991), "HSYS. a Computerized Methodology for Analyzing Human Performance in Complex Operational Settings," In ***Proceedings of the Human Factors Society 35***, 2, 1173.

Caccamise, D., C. Somers and A. Sebok (1994), "Optimizing Human Reliability: Mock-up and Simulation Techniques in Waste Management," In ***Proceedings of the 4th Annual Conference on High-Level Radioactive Waste Management***, B. Cole (Ed.), American Nuclear Society Press, 1338-1341.

Cacciabue, P.C. (1992), "Cognitive Modelling: A Fundamental Issue for Human Reliability Assessment Methodology," ***Reliability Engineering and System Safety 38***, 91-97.

Callahan, K.P., C.C. Baker, T.B. Malone and F.D. Pierce (1990), "Application of Human Engineering to a Shipboard Damage Control Console," In ***Proceedings of the Human Factors Society 34th Annual Meeting 34***, 1158-1162.

Card, D.N. (1992), "Designing Software for Producibility," ***Journal of Systems and Software 17***, 219-225.

Card, D.N. and R.L. Glass (1990) ***Measuring Software Design Quality***, Prentice-Hall Publishing Company, Englewood Cliffs, NJ.

Card, D.N. and W.W. Agresti (1988), "Measuring Software Design Complexity," ***Journal of Systems and Software 8***, 185-197.

Carter, R.J. and J.A. Wachtel (1992), "Advanced Control Room Design Review Guidelines. Merging Old and New," In ***Proceedings of the Human Factors Society 36***, 1, 423-427.

Chang, T.C., K. Hasegawa and C.W. Ibbs (1991), "The Effects of Membership Function on Fuzzy Reasoning," ***Fuzzy Sets and Systems 44***, 169-186.

Charniak, E. and D. McDermott (1985) ***Introduction to Artificial Intelligence***, Addison-Wesley, Reading, MA.

Chen, J-Y. and J-F. Lu (1993), "A New Metric for Object-Oriented Design," ***Information and Software Technology 35***, 4, 232-240.

Chimera, R. (1992), "Value Bars: An Information Visualization Tool for Multi-Attribute Listings," In ***Proceedings of ACM CHI '92***, 293-294.

Chockie, A. and K. Bjorkelo (1992), "Effective Maintenance Practices to Manage System Aging," In ***Proceedings of the Annual Reliability and Maintainability Symposium***, R. Evans (Ed.), IEEE Press, 166-170.

Choi, D.K. (1992), "Integrated System Evaluation," In ***Proceedings of the Complex Systems Engineering Synthesis and Assessment Technology Workshop (CSESAW '92)***, S.L. Howell and P.Q. Hwang (Eds.), Naval Surface Warfare Center Dahlgren Division, Dahlgren, VA, 575-579.

Chou, K.C. and J. Yuan (1993), "Fuzzy-Bayesian Approach to Reliability of Existing Structures," ***Journal of Structural Engineering 119***, 11, 3276-3290.

Chu, R.W., P.M. Jones and C.M. Mitchell (1991), "GT-POCC: Visualization and Animation of a Complex Dynamic System," In ***Proceedings of the IEEE International Conference on Systems, Man and Cybernetics***, 2, 1289-1294.

Clark, D.F. and A. Kandel (1991), "HALO: A Fuzzy Programming Language," ***Fuzzy Sets and Systems 44***, 2, 199-208.

Clements, P.C., C.L. Heitmeyer, B.G. Labaw and A.T. Rose (1993), "MT: A Toolset for Specifying and Analyzing Real-Time Systems," In ***Proceedings of the 1993 Real-Time Systems Symposium***, S. Davidson, I. Lee and F. Jahanian (Eds.), IEEE Press, Los Alamitos, CA, 12-22.

Cormen, T.H., C.E. Leiserson and R.L. Rivest (1992) ***Introduction to Algorithms***

Courtney, R. and D. Gustafson (1993), "Shotgun Correlations in Software Measures," ***Software Engineering Journal 1***, 5-13.

Crawford, J.L. (1992), "Intelligent Graphic Interface Project: Operator Interfaces for the Year 2000," In ***Proceedings of the Human Factors Society 36th Annual Meeting 36***, 1, 465-469.

Cristian, F. (1993), "Automatic Reconfiguration in the Presence of Failures," ***Software Engineering Journal 2***, 53-60.

Cui, H. and Y. Zao (1991), "Fuzzy Synthetic Judgment of Correlating Parameter of Fighter Design," ***Chinese Journal of Aeronautics 4***, 1, 20-25.

Cuomo, D.L. and A.P. Rizzuto (1990), "Methodology for Determining the Human Role in the Strategic Defense System Command Center," In ***Proceedings of the Human Factors Society 34***, 1, 1148-1152.

Cuomo, D.L. and C.D. Bowen (1992), "Stages of User Activity Model As a Basis for User-System Interface Evaluations," In ***Proceedings of the Human Factors Society 36***, 2, 1254-1258.

Czogala, E. (1984), "An Introduction to Probabilistic L-Valued Logic," ***Fuzzy Sets and Systems 13***, 179-185.

Dadkhah, K.M. and F. Zahedi (1993), "A Mathematical Treatment of Inconsistency in the Analytical Hierarchy Process," ***Mathematical and Computer Modelling 17***, 4/5, 111-122.

DeTurck, D.M. (1987), "An Approach to Consistency in the Analytic Hierarchy Process," ***Mathematical Modelling 9***, 3-5, 345-352.

Debenham, J.K. (1989) ***Knowledge Systems Design***, Prentice Hall, Inc., Englewood Cliffs, N.J.

Delen, G. and D. Rijsenbrij (1992), "The Specification, Engineering and Measurement of Information Systems Quality," ***Journal of Systems and Software 17***, 205-217.

de Mantaras, R.L. and L. Godo (1993), "From Fuzzy Logic to Fuzzy Truth-Valued Logic for Expert Systems: A Survey," In ***Proceedings of the IEEE International Conference on Fuzzy Systems***, 750-755.

Dembicki, E. and T. Chi (1991), "Approximation Approach for Personal Estimation in Safety Analysis of Existing Structures," ***Structural Safety 10***, 4, 327-335.

Derrick, E.J. (1992), "A Visual Simulation Support Environment Based on a Multifaceted Conceptual Framework, Ph.D. Dissertation, Department of Computer Science, Virginia Tech, Blacksburg, VA.

Dhingra, A.K. and H. Moskowitz (1991), "Application of Fuzzy Theories to Multiple Objective Decision Making in System Design," ***European Journal of Operational Research 53***, 3, 348-361.

Dimenna, R.A. and T.K. Larson (1989), "Ranking Significant Phenomena in Physical Systems," ***Transactions of the American Nuclear Society 60***, 755-758.

Diteman, M.L. and L.A. Stauffer (1993), "Testing of a Relative Comparison Method for Evaluating Design Concept Alternatives," ***Design Theory and Methodology***, ASME DE-53, 149-155.

Dong, Z. (1987), "Evaluating Design Alternatives and Fuzzy Operations," In ***Proceedings of the 1987 International Conference on Engineering Design - International Congress on Planning and Design Theory 1***, 322-329.

Dong,W., W.L. Chiang, H.C. Shah and F.S. Wong (1989), "Failure Possibility of Existing Buildings," ***Civil Engineering Systems 6***, 4, 170-179.

Dougherty, E.M., Jr. (1990), "Human Reliability Analysis - Where Shouldst Thou Turn?," ***Reliability Engineering and System Safety 29***, 283-299.

Douligeris, C. and I. Pereira (1992), "An Analytical Hierarchy Process Approach to the Analysis of Quality in Telecommunication Systems," In ***Proceedings of IEEE Global Telecommunications Conference (GLOBECOM '92), "3***, 1684-1688.

Driscoll, K. and K. Hoyme (1992), "The Airplane Information Management System: An Integrated Real-Time Flight-Deck Control System," In ***Proceedings of the 1992 Real-Times System Symposium***, IEEE Press, Los Alamitos, CA, 267-270.

Dubois, D. (1992), "Processing Vague Queries in Man-Machine Systems: A Fuzzy Set Approach," In ***Proceedings of the IFAC Symposium on Theory and Application of Digital Control***, 413-416.

Dubois,D.,H. Prade and J.P. Rossazza (1991), "Vagueness, Typicality and Uncertainty in Class Hierarchies," ***International Journal of Intelligent Systems 6***, 2, 167-183.

Duffy, J.B., J.W. Lehner and B. Pannell (1993), "Evaluation of the National Launch System as a Booster for the HL-20," ***Journal of Spacecraft and Rockets 30***, 5, 622-627.

Dugan, J.B. (1990), "On Measurement and Modeling of Computer Systems Dependability: A Dialog Among Experts," ***IEEE Transactions on Reliability 39***, 4, 506-510.

Dugan, J.B. and F.A. Patterson-Hine (1993), "Simple Models of Fault-Tolerant Software," In ***Proceedings of the Annual Reliability and Maintainability Symposium***, R. Evans (Ed.), IEEE Press, 354-359.

Embrey, D.E. SET(19 Quantitative and Qualitative Prediction of Human Error in Safety Assessments," In ***Proceedings of the Symposium on Major Hazards Onshore and Offshore: Institute of Chemical Engineers Symposium Series 130***, 329-350.

Embrey, D.E. (1992), "Incorporating Management and Organisational Factors Into Probabilistic Safety Assessment," ***Reliability Engineering & System Safety 38***, 1-2, 199-208.

Enderwick, T.P. (1990), "Some Pragmatic Issues of Measurement," In ***Proceedings of the Human Factors Society 34th Annual Meeting 34***, 1248-1252.

Evanco, W.M. (1993), "A Software Metrics Integration Framework," In ***Proceedings of the Complex Systems Engineering Synthesis and Assessment Technology Workshop (CSESAW '93)***, S.L. Howell and W. Farr (Eds.), Naval Surface Warfare Center Dahlgren Division, Dahlgren, VA, 112-125.

Fabrycky, W.J. (1994), "Modeling and Indirect Experimentation in System Design Evaluation," ***Journal of NCOSE 1***, 1, July-September, 133-144.

Farr, W. and A. Ashton (1992), "Developing a Metrics Assessment Program for the SLBM Software Development Division," In ***Proceedings of the Complex Systems Engineering Synthesis and Assessment Technology Workshop (CSESAW '92)***, S.L. Howell and P.Q. Hwang (Eds.), Naval Surface Warfare Center Dahlgren Division, Dahlgren, VA, 139-146.

Ferreirinha, P., V. Hubka and W.E. Eder (1993), "Early Cost Calculation: Reliable Calculation, Not Just Estimation," ***Design for Manufacturability DE-52***, P.J. Guichelaar (Ed.), "97-104.

Finnie, G.R., G.E. Wittig and D.I. Petkov (1993), "Prioritizing Software Development Productivity Factors Using the Analytic Hierarchy Process," ***Journal of Systems and Software 22***, 2 129-139.

Flach, J.M. and K.B. Bennett (1992), "Graphical Interfaces to Complex Systems. Separating the Wheat from the Chaff," In ***Proceedings of the Human Factors Society 36th Annual Meeting 36***, 1, 470-474.

Forman, E.H. (1987), "Relative vs. Absolute Worth," ***Mathematical Modelling 9***, 3-5, 195-202.

Forman, E.H. (1993), "Facts and Fictions about the Analytic Hierarchy Process," ***Mathematical and Computer Modelling 17***, 4/5, 19-26.

Frakes, W.B. and R. Baeza-Yates (1992) ***Information Retrieval: Data Structures and Algorithms***, Prentice-Hall, Englewood Cliffs, NJ.

Franklin, P.H. (1993), "Software Reliability Prediction in a Multiple-Processor Environment," In ***Proceedings of the Annual Reliability and Maintainability Symposium***, R. Evans (Ed.), IEEE Press, 348-353.

Fukuda, S. (1989), "Qualitative Design for Safety and Reliability," In ***Proceedings of the 5th International Conference on Structural Safety and Reliability (ICOSSAR 89)***, A. Ang, M. Shinozuka, G. Schueller (Eds), 1763-1769.

Furuta, K. and S. Kondo (1992), "Impact of Group Organization on Human Reliability," ***Reliability Engineering and System Safety 38***, 193-198.

Gandelli, A. and V. Piuri (1991), "A User-Friendly Measurement System: Design and Implementation of a Flexible High-Level Interface for a Parallel Architecture," In ***IEEE Instrumentation and Measurement Technology Conference***, A.R. Howland (Ed.), IEEE Press, Los Alamitos, CA, 483-488.

Gao, L. and E. Fernandez (1994), "Accelerating Conversations for Fault-Tolerant Concurrent Software," In ***SOUTHEASTCON '94***, O.A. Mohammed (Ed.), 334-338.

Gawron, V.J., L. Cipriano, E. Fleishman, F. Hegge, E. Lehman, D. Meister and J. Reising (1990), "Guide for Human Performance Measurements," In ***Proceedings of the Human Factors Society 34***, 2, 1238-1240.

Gertman, D.I. (1993), "Representing Cognitive Activities and Errors in HRA Trees," ***Reliability Engineering and System Safety 39***, 25-34.

Glass, R.L. (1992), "The Link between Software Quality and Software Maintenance," ***Journal of Systems and Software 19***, 1-2.

Golden, B.L. and Q. Wang (1989), "An Alternative Measure of Consistency," In ***The Analytic Hierarchy Process -- Applications and Studies***, Golden, Wasil and Harker (Eds.), Springer-Verlag, New York.

Goodman, P. (1993) ***Practical Implementation fo Software Metrics***, McGraw-Hill Book Co., England.

Greenwell, M. (1988) ***Knowledge Engineering for Expert Systems***, Ellis Horwood, Ltd., Chichester, England.

Guichelaar, P.F. (Ed.), "(1993), "Design for Manufacturability - 1993," ***Design for Manufacturability - 1993 American Society of Mechanical Engineers, Design Engineering Division (Publication), DE 52***.

Gupta, M.M. (1988) ***Approximate Reasoning in Expert Systems***, M.M. Gupta (Ed.), SOLE.

Gustafson, J., D. Rover. S. Elbert and M. Carter (1991), "The Design of a Scalable, Fixed-Time Computer Benchmark," ***Journal of Parallel and Distributed Computing 12***, 388-401.

Götz, N., V. Herzog and M. Rettelbach (1993), "Multiprocessor and Distributed Systems Design: The Integration of Functional Specification and Performance Using Stochastic Process Algebras," In ***Performance Evaluation of Computer and Communications Systems: Joint Tutorial Papers of Performance '93 and Sigmetrics '93 Proceedings***, L. Donatiello and R. Nelson (Eds.), Springer-Verlag, Inc., New York, 121-146.

Hadipriono, F.C. (1989), "Damage Assessment Based on Repairability Criterion," In ***Proceedings of the 5th International Conference on Structural Safety and Reliability (ICOSSAR 89)***, A. Ang, M. Shinozuka, G. Schueller (Eds), 2123-2130.

Hadipriono, F.C. and T.J. Ross (1991), "A Rule-based Fuzzy Logic Deduction Technique for Damage Assessment of Protective Structures," ***Fuzzy Sets and Systems 44***, 459-468.

Hahler, B., S. Dahl, R. Laughery, J. Lockett and B. Thein (1991), "CREWCUT - A Tool for Modeling the Effects of High Workload on Human PerformanceIn ***Proceedings of the Human Factors Society 35***, 2, 1210-1214.

Hahn, H.A. and F.K. Houghton (1994), "Impact of Staffing Parameters on Operational Reliability," In ***Proceedings of the 4th Annual International Conference on High Level Radioactive Waste Management***, 1342-1347.

Halang, W.A. (1992), "Real-Time Systems: Another Perspective," ***Journal of Systems and Software 18***, 101-108.

Halang, W.A. and A.D. Stoyenko (1991) ***Constructing Predictable Real-Time Systems***, Kluwer Academic Publishers, Boston, MA.

Hallbert, B.P., M.A. Rodriguez, J.L. Harbour, D.J. Caccamise and J.W. Keller (1992), "Development of Job Performance Aids to Increase Human Performance Reliability: A Case Study in the Evaluation of Human Factors Principles," In ***Proceedings of the Human Factors Society 36th Annual Meeting 36***, 1138-1142.

Halligan, R.J. (1993), "Requirements Metrics: The Basis of Informed Requirements Engineering Management," In ***Proceedings of the Complex Systems Engineering Synthesis and Assessment Technology Workshop (CSESAW '93)***, S.L. Howell and W. Farr (Eds.), Naval Surface Warfare Center Dahlgren Division, Dahlgren, VA, 9-14.

Hammer, D.K. (1994), "The Development of Large and Complex Software Systems: A Purely Technical Issue?," ***Software Systems Engineering PD-59***, ASME Press, 9-17.

Harbour, J.L. and S.G. Hill (1990), "Using HSYS in the Analysis of Human-System Interactions. Examples From the Offshore Petroleum Industry," In ***Proceedings of the Human Factors Society 34***, 1, 1190-1194.

Harker, P.T. (1987a), "Alternative Modes of Questioning in the Analytic Hierarchy Process," ***Mathematical Modelling 9***, 3-5, 353-360.

Harker, P.T. (1987b), "Incomplete Pairwise Comparisons in the Analytic Hierarchy Process," In ***Mathematical Modelling 9***, 11, 837-848.

Harker, P.T. and L.G. Vargas (1987), "The Theory of Ratio Scale Estimation: Saaty's Analytic Hierarchy Process," ***Management Science 33***, 11, 1383-1403.

Harrichunder, R. (1994), "The Virginia Tech Visual Simulation Environment Learning Support System: VTVSE-LSS, M.S. Thesis, Department of Computer Science, Virginia Tech, Blacksburg, VA.

Harrison, P.G. (1993), "Response Time Distributions in Queueing Network Models," In ***Performance Evaluation of Computer and Communications Systems: Joint Tutorial Papers of Performance '93 and Sigmetrics '93 Proceedings***, L. Donatiello and R. Nelson (Eds.), Springer-Verlag, Inc., New York, 147-164.

Hart, A. (1992) ***Knowledge Acquisition for Expert Systems***, McGraw-Hill, Inc., New York, NY.

Hart, S.G. and L.E. Staveland (1988), "Development of the NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research," In ***Human Mental Workload***, P.A. Hancock and N. Meshkati (Eds.), North-Holland, 139-183.

Hawksley, J.L. (1992), "Risk Analysis in Safety Reports Required by the Seveso Directive," ***Reliability Engineering and System Safety 35***, 193-199.

Heinecke, A.M. (1993), "Software Ergonomics for Real-Time Systems," In ***Proceedings of the Vienna Conference on Human Computer Interaction, VCHCI '93***, T. Grechenig and M. Tscheligi (Eds.), Springer-Verlag, Inc., New York.

Hellerstein, J.L. (1993), "An Introduction to Modeling Dynamic Behavior with Time Series Analysis," In ***Performance Evaluation of Computer and Communications Systems: Joint Tutorial Papers of Performance '93 and Sigmetrics '93 Proceedings***, L. Donatiello and R. Nelson (Eds.), Springer-Verlag, Inc., New York, 203-223.

Henneman, R.L. and W.B. Rouse (1986), "On Measuring the Complexity of Monitoring and Controlling Large-Scale Systems," ***IEEE Transactions on Systems, Man, and Cybernetics 16***, 2, 193-207.

Henry, S. and C. Selig (1990), "Predicting Source Code Complexity at the Design Stage," ***IEEE Software 3***, 36-44.

Henry, S.M. and D. Kafura (1984), "The Evaluation of Software Systems' Structure Using Qnantitative Software Metrics," ***Software Practice and Experience 14***, 6, 561-573.

Hetzel, B. (1993) ***Making Software Measurement Work: Building an Effective Measurement Program***, QED Publishing Group, Boston.

Hewett, T.T. (1990), "Interface Design Considered As Failure Analysis," In ***Proceedings of the Human Factors Society 34***, 1, 325-328.

Hill, M.R. and M. Joseph (1994), "Automated Timing Analysis of Real-Time Programs," ***Software Engineering Journal 5***, 221-227.

Hill, S.G., J.L. Harbour and C. Sullivan (1990), "Examining Human-System Interactions. the HSYS Methodology," In ***Proceedings of the Human Factors Society 34***, 2, 1186-1189.

Hingle, G.C. and M.M. Tanik (1994), "Interface-Oriented Performance Specification for Computational Systems," ***Software Systems Engineering PD-59***, ASME Press, 69-76.

Hollnagel, E. (1992), "The Reliability of Man-Machine Interaction," ***Reliability Engineering and System Safety 38***, 81-89.

Holtzblatt, K. and H.R. Beyer (1995), "Requirements Gathering: The Human Factor," ***Communications of the ACM 38***, 5, 31-32.

Hopgood, A.A. (1993) ***Knowledge-Based Systems for Engineers and Scientists***, CRC Press, Boca Raton, FL.

Hopgood, A.A. (1989), "An Inference Mechanism for Selection and its Application to Polymers," ***Artificial Intelligence and Engineering 4***, 4, 197-203.

Howell, F.W., R. Williams and R.N. Ibbett (1994), "Hierarchical Architecture Design and Simulation Environment," In ***Proceedings of the 2nd Annual Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems - MASCOTS '94***, V. Madisetti, et al. (Eds.), IEEE Computer Society Press, Los Alamitos, CA, 363-366.

Howell, S.L., C.M. Nguyen and P.Q. Hwang (1992), "Design Structuring and Allocation Optimization (DeStinAtiOn)," In ***Proceedings of the 25th Hawaii International Conference on System Sciences***, II, B.D. Shriver, (Ed.), IEEE Computer Society Press, Los Alamitos, CA, 517-528.

Howell, S.L., C.M. Nguyen and P.Q. Hwang (1991), "System Design Structuring and Allocation Optimization (DeStinAtiOn)," In ***Proceedings of the 1992 Complex Systems Engineering Synthesis and Assessment Technology Workshop (CSESAW '91)***, C.M. Nguyen (Ed.), Naval Surface Warfare Center, Dahlgren Division, Silver Spring, MD, 117-128.

Howell, S.L., P.Q. Hwang and C.M. Nguyen (1990), "Expert Design Advisor, Technical Report NAVSWC TR 90-46, October 1990, Naval Surface Warface Center, Dahlgren Division, Silver Spring, MD.

Hughes, W.R. (1993), "Consistent Utility and Probability Assessment Using AHP Methodology," ***Mathematical and Computer Modelling 17***, 4/5, 171-177.

Hugue, M.M. (1994), "Myths about Dependability," In ***Proceedings of the Complex Systems Engineering Synthesis and Technology Assessment Workshop -- CSESAW '94***, S.L. Howell, M. McCoy (Eds.), 69-71.

Hugue, M.M., M. Casey, G. Wood and M. Edwards (1994), "RECAP: A Requirements Capture Tool for Large Complex Systems," In ***Proceedings of the Complex Systems Engineering Synthesis and Technology Assessment Workshop -- CSESAW '94***, S.L. Howell, M. McCoy (Eds.), 39-44.

Hugue, M.M., N. Suri and C.J. Walter (1993), "A Dependable System Perspective," In ***Proceedings of the Complex Systems Engineering Synthesis and Assessment Technology Workshop (CSESAW '93)***, S.L. Howell and W. Farr (Eds.), Naval Surface Warfare Center Dahlgren Division, Dahlgren, VA, 207-213.

Hutchings, A.F. and S.T. Knox (1995), "Creating Products Customers Demand," ***Communications of the ACM 38***, 5, 72-80.

Hyde, R.S. (1989), "A Measure of Design Quality, M.S. Thesis, College of Graduate Studies, University of Idaho, Moscow, ID.

Hyde, R.S. and L.A. Stauffer (1990), "The Comparison of the Reliability of Three Psychometric Scales for Measuring Design Quality," In ***Proceedings of the ASME Conference on Design Theory and Methodology DTM 90***, 349-354.

Hyman, J.M., A.A. Lazar and G. Pacifici (1991), "Real-Time Scheduling with Quality of Service Constraints," ***IEEE Journal on Selected Areas in Communication, 9***, 7, 1052-1063.

Iqbal, N. and G.L. Kinzel (1992), "Approach to Represent Imprecision in Interactive Design Using Fuzzy Set Theory," In ***Computers in Engineering: Proceedings of the International Computers in Engineering Conference and Exhibit 1***, 9-17.

Islam, S. and H. Ammar (1992), "Performability of Integrated Software-Hardware Components of Real-Time Parallel & Distributed Systems," ***IEEE Transactions on Reliability 41***, 2, 352-362.

Jahanian, F. (1993), "Fault-Tolerance in Embedded Real-Time Systems," In ***Hardware and Software Architectures for Fault Tolerance: Experiences and Perspectives***, M. Banâtre and P.A. Lee (Eds.), Springer-Verlag, Inc., Berlin, 237-249.

Jeffords, R.D. (1994), "Formal Requirements Specification Languages for Real-Time Systems: Expressibility Issues," In ***Proceedings of the Complex Systems Engineering Synthesis and Technology Assessment Workshop -- CSESAW '94***, S.L. Howell, M. McCoy (Eds.), 30-37.

Jeffords, R.D. and B.G. Labaw (1992), "Organizing Top-Level System Requirements," In ***Proceedings of the Complex Systems Engineering Synthesis and Assessment Technology Workshop (CSESAW '92)***, S.L. Howell and P.Q. Hwang (Eds.), Naval Surface Warfare Center Dahlgren Division, Dahlgren, VA, 581-588.

Jensen, R.E. and T.E. Hicks (1993), "Ordinal Data AHP Analysis: A Proposed Coefficient of Consistency and a Nonparametric Test," ***Mathematical and Computer Modelling 17***, 4/5, 135-150.

Jin-ping, O. and W. Guang-yuan (1989), "Fuzzy Random Dynamical Reliability Analysis of Aseismic Structures," In ***Proceedings of the 5th International Conference on Structural Safety and Reliability (ICOSSAR 89)*** A. Ang, M. Shinozuka, G. Schueller (Eds), 1775-1778.

Johannesen, L. and D.D. Woods (1991), "Human Interaction With Intelligent Systems: Trends, Problems and New Directions," In ***Proceedings of the IEEE International Conference on Systems, Man and Cybernetics***, 2, 1337-1341.

Johnson, B. (1992), "TreeViz: Treemap Visualization of Hierarchically Structured Information," In ***Proceedings of ACM CHI '92***, 369-370.

Jones, D.R., V. Murthy and J. Blanchard (1992), "Quality and Reliability Assessment of Hardware and Software During the Total Product Life Cycle," ***Quality and Reliability Engineering International 8***, 477-483.

Jones, P.H. and D.W. Biers (1992), "Relative Contribution of Training and Interface Design to Mental Model Assimilation," In ***Proceedings of the Human Factors Society 36***, 1, 453-457.

Jones, P.M. and C.M. Mitchell (1991a), "Human-Computer Cooperative Problem Solving in Satellite Ground Control," In ***Proceedings of the Human Factors Society 35***, 1, 408-412.

Jones, P.M. and C.M. Mitchell (1991b), "Human-Machine Cooperative Interaction in the Supervisory Control of a Complex Dynamic System," In ***Proceedings of the IEEE International Conference on Systems, Man and Cybernetics***, 2, 1301-1306.

Jones, P.M. and C.M. Mitchell (1991), "A Mechanism for Knowledge-Based Reminding and Advice-Giving in the Supervisory Control of a Complex Dynamic System," In ***Proceedings of the IEEE International Conference on Systems, Man and Cybernetics***, 2, 1295-1300.

Jubis, R.M.T. (1990), "Coding Effects on Performance in a Process Control Task with Uniparameter and Multiparameter Displays," ***Human Factors 32***, 3, 287-297.

Kamel, A.,A. Nazif, O. El Dessouki and N. Kamel (1990), "MCFS: A Multiple Criteria Reasoning Fuzzy Expert Systems Building Tool," In ***Proceedings of the IEEE Computer Society's International Computer Software & Applications Conference.***, 605-610.

Kandel, A. (1986), "

Kandel, A. (1992) ***Fuzzy Expert Systems***, CRC Press, Boca Raton, FL.

Kantowitz, B.H. (1990), "Can Cognitive Theory Guide Human Factors Measurement?," In ***Proceedings of the Human Factors Society 34th Annual Meeting 34***, 1258-1262.

Kapasouris, P., D. Serfaty, J.C. Deckert, J.G. Wohl and K.R. Pattipati (1991), "Resource Allocation and Performance Evaluation in Large Human-Machine Organizations," ***IEEE Transactions on Systems, Man and Cybernetics 21***, 3, 521-532.

Kaposi, A. and I. Pyle (1993), "Systems are Not Only Software," ***Software Engineering Journal 1***, 31-39.

Karangelen, N., J. Intintolo, H. Ngocdung and S. Howell (1993), "The Representation of Resources for Large-Sized and Complex Systems," In ***Proceedings of the Complex Systems Engineering Synthesis and Assessment Technology Workshop (CSESAW '93)***, S.L. Howell and W. Farr (Eds.), Naval Surface Warfare Center Dahlgren Division, Dahlgren, VA, 107-111.

Karangelen, N.E. (1992), "The Environmental Capture View: Addressing External Factors in Capture and Analysis of Large Scale Complex System Design," In ***Proceedings of the Complex Systems Engineering Synthesis and Assessment Technology Workshop (CSESAW '92)***, S.L. Howell and P.Q. Hwang (Eds.), Naval Surface Warfare Center Dahlgren Division, Dahlgren, VA, 235-248.

Karangelen, N.E., N.T. Hoang and S. Howell (1994), "Representing System Resources in Design and Analysis of Complex Systems," ***Software Systems Engineering PD-59***, ASME Press, 51-58.

Karunanithi, N., D. Whitley and Y.K. Malaiya (1992), "Prediction of Sotware Reliability using Connectionist Models," ***IEEE Transactions on Software Engineering 18***, 7, 563-573.

Kaufmann, A. and M. Gupta (1985) ***Introduction to Fuzzy Arithmetic***, Van Nostrand Reinhold Co., Inc., New York, NY.

Keene, S. and C. Lane (1993), "Reliability Growth of Fielded Software," In ***Proceedings of the Annual Reliability and Maintainability Symposium***, R. Evans (Ed.), IEEE Press, 360-365.

Keeney, R.L. and H. Raiffa (1993) ***Decisions with Multiple Objectives***, Cambridge University Press.

Keil, M. and E. Carmel (1995), "Customer-Developer Links in Software Development," ***Communications of the ACM 38***, 5, 33-44.

Khorramshahgol, R. and H. Azani (1988), "A Decision Support System for Effective Systems Analysis and Planning," ***Journal of Information & Optimization Sciences 9***, 1, 41-52.

Khorramshahgol, R., H. Azani and Y. Gousty (1988), "An Integrated Approach to Project Evaluation and Selection," ***IEEE Transactions on Engineering Management 35***, 4, 265-270.

Khoshgoftaar, T.M., J.C. Munson, B.B. Bhattacharya and G.D. Richardson (1992), "Predictive Modeling Techniques of Software Quality from Software Measures," ***IEEE Transactions on Software Engineering 18***, 11, 979-986.

Kim, K.H. and L.F. Bacellar (1993), "A Real-Time Object Model: A Step Toward an Integrated Methodology for Engineering Complex Dependable Systems," In ***Proceedings of the Complex Systems Engineering Synthesis and Assessment Technology Workshop (CSESAW '93)***, S.L. Howell and W. Farr (Eds.), Naval Surface Warfare Center Dahlgren Division, Dahlgren, VA, 56-64.

Kim, K.H., M. Denz, T. Lawrence, C. Nguyen and R. Scalzo (1994), "

Kirk, B.R. (1994), "Designing Systems with Objects, Processes and Modules," ***Microprocessors and Microsystems 18***, 3, 105-171.

Kirkpatrick, M., T.B. Malone, C.C. Heasly and C.C. Baker (1990), "Manpower, Personnel, Training and Safety (MPTS), "Simulation Tools. Network and Simulation for Workload Assessment and Modeling (SIMWAM)," In ***Proceedings of the Human Factors Society 34***, 2, 1219-1223.

Kirova, V. and W. Rossak (1994), "Some Thoughts on Architecture Engineering," ***Software Systems Engineering PD-59***, ASME Press, 59-68.

Kirwan, B. (1988), "A Comparative Evaluation of Five Human Reliability Assessment Techniques," In ***Human Factors and Decision Making***, B.A. Sayers, Ed. Elsevier, 1988, 87-109.

Kirwan, B. and N. James (1989), "The Development of a Human Reliability Assessment System for the Management of Human Error in Complex Systems," In ***Reliability '89 Pt 2***, 5A/2/1-5A/2/11.

Kitchenham, B. (1987), "Towards a Constructive Quality Model - Part I: Software Quality Modelling, Measurement and Prediction," ***Software Engineering Journal 2***, 4, 105-113.

Kitchenham, B., S. Linkman and D. Law (1994), "Critical Review of Quantative Assessment," ***Software Engineering Journal 6***, 43-53.

Kjaer, A. and K.H. Madsen (1995), "Participatory Analysis of Flexibility," ***Communications of the ACM 38***, 5, 53-60.

Klir, G. and T. Folger (1988) ***Fuzzy Sets, Uncertainty and Information***, Prentice Hall Publishing, Englewood Cliffs, NJ.

Klir, G.J. and B. Yuan (1995) ***Fuzzy Sets and Fuzzy Logic: Theory and Applications***, Prentice-Hall.

Knapp, B.G., S.A. Seven, F.A. Muckler and A. Akman (1991), "Abilities Evaluation Method for Military Intelligence Personnel," In ***Proceedings of the Human Factors Society 35***, 1, 1321-1325.

Knapp, R.K. and J.J. Vardaman (1991), "Response to an Automated Function Failure Cue: An Operational Measure of Complacency," In ***Proceedings of the Human Factors Society 35***, 1, 112-115.

Kohout, L.J. (1984), "Fuzzy Decision Making and Its Impact on the Design of Expert Systems," ***IEE Colloquium (Digest)***, 67, 5.1 - 5.4.

Kopetz, H. (1993), "The Systematic Design of Large Real-Time Systems or Interface Simplicity," In ***Hardware and Software Architectures for Fault Tolerance: Experiences and Perspectives***, M. Banâtre and P.A. Lee (Eds.), Springer-Verlag, Inc., Berlin, 250-262.

Kraslawski, A., T. Koiranen and L. Nystrom (1993), "Concurrent Engineering. Robust Design in Fuzzy Environment," ***Computers & Chemical Engineering, 17 Supplement***, Oct, 447-452.

Kreider, D.K. and R.E. Nance (1991), "Objectives, Principles and Attributes: A Structured Approach to Systems Engineering," In ***Naval Surface Warfare Center Technical Digest***, J.D. Sellers (Ed.), Technical Report NAVSWC/MP/91-44, 22-31.

Krishnamachari, R., S. Badde, J.K. Allen and F. Mistree (1993), "Simultaneous Kinematic and Dynamic Synthesis in the Preliminary Design of a Linkage," ***Advances in Design Automation American Society of Mechanical Engineers, Design Engineering Division (Publication), "DE 65***, 1, 277-280.

Kruse, R. (1991) ***Uncertainty and Vagueness in Knowledge-based Systems: Numerical Methods***, Springer-Verlag, NY.

Kubic, W.L. and F.P. Stein (1988), "Theory of Design Reliability Using Probability and Fuzzy Sets," ***AIChE Journal 34***, 4, 583-601.

Kuchcinski, K. (1993), "System Level Modelling and Analysis of Complex Digital Systems," ***Microprocessing and Microprogramming 37***, 91-94.

Lala, J.H. and R.E. Harper (1993), "Fault-Tolerance in Embedded Real-Time Systems: Importance and Treatment of Common Mode Failures," In ***Hardware and Software Architectures for Fault Tolerance: Experiences and Perspectives***, M. Banâtre and P.A. Lee (Eds.), Springer-Verlag, Inc., Berlin, 263-282.

Langen, M. and G. Rau (1990), "Interactive Colour Design of Interactive Graphical Displays Using a Prototyping Tool Based on Colour Metrics," ***Ergonomics 33***, 8, 1043-1054.

Laprie, J.C. (1992) ***Dependability: Basic Concepts and Terminology***, Springer-Verlag, New York.

Laprie, J.C., G. Le Lann, M. Morganti and J. Rushby (1993), "Panel Session on Limits in Dependability," In ***Digest of Papers - International Symposium on Fault- Tolerant Computing***, IEEE Computer Society Press, Los Alamitos, CA, 608-613.

Law, A.M. and M.G. McComas (1994), "Simulation of Communications NetworksIn ***Proceedings of the 1994 Winter Simulation Conference***, J.D. Tew, et al. (Eds.), 166-170.

LeMay, M. and J.R. Comstock, Jr. (1990), "Initial Test of a Normative Figure of Merit for the Quality of Overall Task Performance," In ***Proceedings of the Human Factors Society 34***, 1, 81-85.

Leclercq, P.R. (1992), "A Software Reliability Assessment Tool," In ***Proceedings of the Annual Reliability and Maintainability Symposium***, R. Evans (Ed.), IEEE Press, 294-298.

Lee, H. (1993), "Structured Methodology for Software Development Effort Prediction Using the Analytic Hierarchy Process," ***Journal of Systems and Software 21***, 2, 179-186.

Lee, J. and N. Moray (1992), "Trust, Control Strategies and Allocation of Function in Human-Machine Systems," ***Ergonomics 35***, 10, 1243-1270.

Lee, P.A. (1993), "Software Faults: the Remaining Problem in Fault Tolerant Systems?," In ***Hardware and Software Architectures for Fault Tolerance: Experiences and Perspectives***, M. Banâtre and P.A. Lee (Eds.), Springer-Verlag, Inc., Berlin, 171-181.

Lee, W.D. (1988), "Simulated Annealing Applied to Shipbuilding Design," ***Neural Networks 1***, 1 Supplement, 453.

Lee, Y.W. and B.H. Ahn (1991), "Static Valuation of Combat Force Potential By the Analytic Hierarchy Process," ***IEEE Transactions on Engineering Management 38***, 3, 237-244 .

Levendel, Y. (1993), "Fault Tolerance Cost Effectiveness," In ***Hardware and Software Architectures for Fault Tolerance: Experiences and Perspectives***, M. Banâtre and P.A. Lee (Eds.), Springer-Verlag, Inc., Berlin, 13-20.

Li, W. and S. Henry (1993), "Object-Oriented Metrics that Predict Maintainability," ***Journal of Systems Software 23***, 111-122.

Liang, G-S. and M-J. Wang (1993), "Evaluating Human Reliability using Fuzzy Relation," ***Microelectronics and Reliability 33***, 1, 63-80.

Liao, J. and P. Milgram (1991), "On Validating Human Performance Simulation Models," In ***Proceedings of the Human Factors Society 35th Annual Meeting 35***, 1260-1264.

Lim, K.Y. and J.B. Long (1992), "Method for (Recruiting), "Methods: Facilitating Human Factors Input to System Design," In ***Proceedings of the ACM Conference on Human Factors in Computing Systems - CHI '92***, 549-556.

Lim, K.Y., J.B. Long and N. Silcock (1992), "Integrating Human Factors with the Jackson System Development Method. An Illustrated Overview," ***Ergonomics 35***, 10, 1135-1161.

Lin, K-J and S.H. Son (1993), "Real-Time Databases for Complex Embedded Systems: Predictability and Serializability," In ***Proceedings of the Complex Systems Engineering Synthesis and Assessment Technology Workshop (CSESAW '93)***, S.L. Howell and W. Farr (Eds.), Naval Surface Warfare Center Dahlgren Division, Dahlgren, VA, 215-234.

Lisboa, J.J. (1990), "Nuclear Power Plant Availability and the Role of Human Factors Performance," ***IEEE Transactions on Nuclear Science 37***, 2, 980-986.

Litke, J. (1992), "A Method for the Assessment of System Designs," In ***Proceedings of the Complex Systems Engineering Synthesis and Assessment Technology Workshop (CSESAW '92)***, S.L. Howell and P.Q. Hwang (Eds.), Naval Surface Warfare Center Dahlgren Division, Dahlgren, VA, 155-170.

Liu, J.W.S., J.L. Redondo, Z. Deng, T.S. Tia, R. Bettati, A. Silberman, M. Storch, R. Ha and W.K. Shih (1993), "PERTS: A Prototyping Environment for Real-Time Systems," In ***Proceedings of the 1993 Real-Time Systems Symposium***, S. Davidson, I. Lee and F. Jahanian (Eds.), IEEE Press, Los Alamitos, CA, 184-188.

Lo, S.L.A., N.C. Hutchinson and S.T. Chanson (1993), "Architectural Considerations in the Design of Real-Time Kernels," In ***Proceedings of the IEEE Symposium on Real-Time Systems***, S. Davidson (Ed.), 138-147.

Locke, C.D. (1993), "Fault-Tolerant Applications Systems: A Requirements Perspective," In ***Hardware and Software Architectures for Fault Tolerance: Experiences and Perspectives***, M. Banâtre and P.A. Lee (Eds.), Springer-Verlag, Inc., Berlin, 21-25.

Lopez-Benitez, N. (1992), "Dependability Analysis of Distributed Computing Systems using Stochastic Petri Nets," In ***Proceedings of the 11th Symposium on Reliabile Distributed Systems***, T. Lawrence (Ed.), IEEE Press, Los Alamitos, CA, 85-92.

Lupina, N.V., A.N. Slepchenko, S.V. Ul'yanov and M.M. Shakhnazarov (1993), "Hybrid Expert System With Depth Representation of Knowledge for the Design and Diagnostics of Bioengineering Products," ***Journal of Computer and Systems Sciences International 31***, 3, 74-95.

Luque, E., R. Suppi and J. Sorribes (1992), "Designing Parallel Systems: A Performance Prediction Problem," ***Microprocessors and Microsystems 16***, 1, 25-35.

MacLean, A., R.M. Young, V.M.E. Bellotti and T.P. Moran (1991), "Questions, Options and Criteria: Elements of Design Space Analysis," ***Human-Computer Interaction 6***, 201-250.

Malaiya, Y., N. Karunanithi and P. Verma (1992), "Predictability of Software-Reliability Models," ***IEEE Transactions on Reliability 41***, 4, 539-546.

Malone, T.B., C.C. Heasly, M. Kirkpatrick, R.M. Perse, P.J. Vingelis and D.L. Welch (1992), "Human System Integration (HSI), "and Manprint Requirements and Tools," In ***Proceedings of the Human Factors Society 36***, 2, 1128-1132.

Masuda, T. (1990), "Hierarchical Sensitivity Analysis of Priority Used in Analytical Hierarchy Process," ***International Journal of Systems Science 21***, 2, 415-427.

McCabe, T.J. (1976), "A Complexity Measure," ***IEEE Transactions on Software Engineering SE-2***, 308-320.

McColl, R.B. and J.C. McKim, Jr. (1992), "Evaluating and Extending NPath as a Software Complexity Measure," ***Journal of Systems and Software 17***, 275-279.

McDermid, J. (1993), "Safety-Critical Software: a Vignette," ***Software Engineering Journal 1***, 2-3.

Meister, D. (1989) ***Conceptual Aspects of Human Factors***, Johns Hopkins University Press, London.

Meister, D. (1993), "Human Reliability Database and Future Systems," In ***Proceedings of the Annual Reliability and Maintainability Symposium***, R. Evans (Ed.), IEEE Press, 276-280.

Meister, D. (1990), "An Alternative Measurement Paradigm," In ***Proceedings of the Human Factors Society 34th Annual Meeting 34***, 1243-1247.

Meister, D., T. Enderwick, A. Bittner and B. Kantowitz (1990), "Human Factors Measurement: the Challenge," In ***Proceedings of the Human Factors Society 34th Annual Meeting 34***, 1241-1242.

Merkuryeva, G.V., V.Y. Savelyev and A.N. Borisov (1989), "Synthesis of New Technical Decisions Using Fuzzy Production Systems," ***Cybernetics and Systems 20***, 1, 89-97.

Merwin, D.H. and C.D. Wickens (1991), "2-D Vs. 3-D Display for Multidimensional Data Visualization. the Relationship Between Task Integrality and Display Proximity," In ***Proceedings of the Human Factors Society 35***, 1, 388-392.

Meshkati, N. (1988), "Toward Development of a Cohesive Model of Workload," In ***Human Mental Workload***, P.A. Hancock and N. Meshkati (Eds.), North-Holland, 305-314.

Midkiff, S.F. and W.J. Fabrycky (1991), "Concurrent Engineering Methods and Tools for Hardware/Software Allocation in the Early Design of Embedded Systems," In ***Proceedings of the International Conference on Concurrent Engineering and Electronic Design Automation***, S. Madhat (Ed.), 340-347.

Min, B.K. and S.H. Chang (1993), "Strategy and Tool Development for Nuclear Power Plant Design Synthesis With Measurement of Reliability and Simplicity," ***Reliability Engineering & System Safety 40***, 1, 69-76.

Mitchell, C.M. and K. Williams (1993), "Failure Experience of Programmable Logic Controllers Used in Emergency Shutdown Systems," ***Reliability Engineering and System Safety 39***, 329-331.

Mitchell, N.B. (1991), "Selecting Space Station Freedom Hardware," In ***Proceedings of the Human Factors Society 35th Annual Meeting 35***, 21317-1320.

Mitta, D.A. (1993), "Application of the Analytic Hierarchy Process. A Rank-ordering of Computer Interfaces," ***Human Factors 35***, 1, 141-157.

Modrick, J.A. (1992), "The Role of Complexity in Human Performance, Memory and Training," In ***Proceedings of the Human Factors Society 36***, 2, 1163-1165.

Modrick, J.A. (1992), "Review of Concepts and Approaches to Complexity," In ***Proceedings of the Human Factors Society 36***, 1166-1170.

Moller, K.H. and D.J. Paulish (1993) ***Software Metrics: A Practitioner's Guide to Improved Product Development***, Chapman & Hall Publishing Co., London.

Mon, D-L., C-H. Cheng and J-C. Lin (1994), "Evaluating Weapon System Using Fuzzy Analytic Hierarchy Process based on Entropy Weight," ***Fuzzy Sets and Systems 62***, 127-134.

Mooney, E.L., W.R. Taylor and M. Amunrud (1984), "Fuzzy Sets - Computer Applications for IE's," In ***Proceedings of the 1984 Annual International Industrial Engineering Conference***, 89-98.

More, H.B. and J. Wu (1994), "Throughput Improvement through Dynamic Load Balance," In ***SOUTHEASTCON '94***, O.E. Mohammed (Ed.), 339-342.

Moreno-Jimemez, J.M. and L.G. Vargas (1993), "A Probabilistic Study of Preference Structures in the Analytic Hierarchy Process with Interval Judgments," ***Mathematical and Computer Modelling 17***, 4/5, 73-81.

Mostert, D.N.J and S.H. von Solms (1993), "Computer Security, Safety and Resilience Requirements," In ***Proceedings of the Complex Systems Engineering Synthesis and Assessment Technology Workshop (CSESAW '93)***, S.L. Howell and W. Farr (Eds.), Naval Surface Warfare Center Dahlgren Division, Dahlgren, VA, 324-362.

Muckler, F.A. and S.A. Seven (1992), "Selecting Performance Measures: 'Objective' versus 'Subjective' Measurement," ***Human Factors 34***, 4, 441-455.

Muir, B.M. (1987), "Trust Between Humans and Machines and the Design of Decision Aids," ***International Journal of Man-Machine Studies 27***, 5-6, 527-539.

Muller-Glaser, K.D., J. Bortolazzi and Y. Tanurhan (1992), "Towards a Requirements Definition, Specification and System Design Environment," In ***European Design Automation Conference '92***, 238-243.

Munson, J.C. and T.M. Khoshgoftarr (1990), "Regression Modelling of Software Quality: Empirical Investigation," ***Information and Software Systems 32***, 2, 106-114.

Murphy, C.K. (1993), "Limits on the Analytic Hierarchy Process From Its Consistency Index," ***European Journal of Operational Research 65***, 1, 138-139.

Mustafa, M.A. and J.F. Al-Bahar (1991), "Project Risk Assessment Using the Analytic Hierarchy Process," ***IEEE Transactions on Engineering Management 38***, 1, 46-52.

Nallon, J. and M. Edwards (1994), "Implentation of NSWC Requirements Traceability Models," In ***Proceedings of the Complex Systems Engineering Synthesis and Technology Assessment Workshop -- CSESAW '94***, S.L. Howell, M. McCoy (Eds.), 15-21.

Negoita, C.V. (1985) ***Expert Systems and Fuzzy Systems***, Benjamin/Cummings Publishing Co., New York.

Nguyen, C.M. and S.L. Howell (1992), "System Design Factors," In ***Proceedings of the Complex Systems Engineering Synthesis and Assessment Technology Workshop (CSESAW '92)***, S.L. Howell and P.Q. Hwang (Eds.), Naval Surface Warfare Center Dahlgren Division, Dahlgren, VA, 147-154.

Nguyen, C.M. and S.L. Howell (1994), "Systems Design Factors: The Essential Ingredients of System Design, Technical Report NSWCDD/TR-92/268, Version 0.4, 18 March 1994, Advanced Systems Technology Group, Systems Research and Technology Department, Naval Surface Warfare Center, Dahlgren Division, Silver Spring, MD.

Ntuen, C.A. (1991), "Design Principles for Cognitive Based Human-Machine Interactions," In ***Proceedings of the IEEE International Conference on Systems, Man and Cybernetics***, 2, 1231-1236.

Ntuen, C.A., E.H. Park and R.E. Roberts (1991), "An Experiment in Human-Robot Interaction During Task Execution," In ***Proceedings of the IEEE International Conference on Systems, Man and Cybernetics***, 2, 1213-1218.

O'Hara, J.M. (1992), "Advanced Control Rooms and Crew Performance Issues: Implications for Human Reliability," ***IEEE Transactions on Nuclear Science 39***, 4, 919-923.

O'Hara, J.M. and Wachtel (1991), "Advanced Control Room Evaluation. General Approach and Rationale," In ***Proceedings of the Human Factors Society 35th Annual Meeting 35***, 1243-1247.

Oery, Z. (1993), "An Integrating Common Framework for Measuring Cognitive Software Complexity," ***Software Engineering Journal 5***, 263-272.

Ogle, D.M., K. Schwan and R. Snodgrass (1993), "Application-Dependent Dynamic Monitoring of Distributed and Parallel Systems," ***IEEE Transactions on Parallel and Distributed Systems 4***, 7, 762-778.

Okuma, A. (1985), "Software and 'Fuzzy' Logic Let Any Good Programmer Design An Expert System," ***Electronic Design 33***, 8, 173-184.

Otto, K.N. (1993), "Measurement Foundations for Design Engineering Methods," ***Design Theory and Methodology***, ASME DE-53, 157-165.

Otto, K.N. and E.K. Antonsson (1991), "Trade-Off Strategies in Engineering Design," ***Research in Design Engineering 3***, J.R. Dixon and S. Finger (Eds.), 87-103.

Otto, K.N. and E.K. Antonsson (1993), "The Method of Imprecision Compared to Utility Theory for Design Selection Problems," ***Design Theory and Methodology***, ASME DE-53, 167-173.

Paek, J.H., Y.W. Lee and T.R. Napier (1992), "Selection of Design/Build Proposal Using Fuzzy-Logic System," ***Journal of Construction Engineering and Management 118***, 2, 303-317.

Palmer, J.D. and R.P. Evans (1994), "An Integrated Semantic and Syntactic Framework for Requirements Traceability," In ***Proceedings of the Complex Systems Engineering Synthesis and Technology Assessment Workshop -- CSESAW '94***, S.L. Howell, M. McCoy (Eds.), 9-14.

Panzieri, F. and R. Davoli (1993), "Real-Time Systems: A Tutorial," In ***Performance Evaluation of Computer and Communications Systems: Joint Tutorial Papers of Performance '93 and Sigmetrics '93 Proceedings***, L. Donatiello and R. Nelson (Eds.), Springer-Verlag, Inc., New York, 435-462.

Papantonopoulos, S.A. and G. Salvendy (1993), "Decision Model for Cognitive Task Allocation," In ***Proceedings of the Human Factors and Ergonomics Society 37***, 1 392-396.

Paula, H.M., M.W. Roberts and R.E. Battle (1993), "Operational Failure Experience of Fault-Tolerant Digital Control Systems," ***Reliability Engineering & System Safety 39***, 3, 273-289.

Pawlowski, T.J. III and C.M. Mitchell (1991), "Direct Manipulation to Facilitate Supervisory Control and Intent Inferencing in Complex Dynamic Systems," In ***Proceedings of the IEEE International Conference on Systems, Man and Cybernetics***, 2, 1317-1321.

Pawlowski, T.J. and C.M. Mitchell (1991), "Direct Manipulation to Facilitate Supervisory Control and Intent Inferencing in Complex Dynamic Systems," In ***Proceedings of the IEEE International Conference on Systems, Man and Cybernetics***, 2, 1317-1321.

Pazirandeh, M. and O. McBryan (1993), "An Environment for Analysis of Parallel Systems (EAPS), ,"

Perkusich, A. and J.C. de Figueiredo (1994), "Embedding Fault-Tolerant Properties in the Design of Complex Software Systems," ***Journal of Systems and Software 25***, 23-27.

Peters, L.J. (1991), "Auditory Performance: A Model to Predict Task Performance as a Function of Auditory Workload. Overview," In ***Proceedings of the Human Factors Society 35***, 1, 609-613.

Peterson, T.F. (1984), "Criteria and Method for Specification and Evaluation of Distributed Digital Control Systems," ***IEEE Transactions on Power Apparatus and Systems PAS-103***, 9, 2414-2417.

Plocher, T., J.F. Lockett III, P. Kovach and J. Powers (1991), "CREWCUT - A New Tool for Predicting Human Performance in Conceptual SystemsIn ***Proceedings of the Human Factors Society 35***, 2, 1206-1209.

Pukite, J. and P.R. Pukite (1994), "Complex System Optimization," In ***Proceedings of the Complex Systems Engineering Synthesis and Technology Assessment Workshop -- CSESAW '94***, S.L. Howell, M. McCoy (Eds.), 131-135.

Ramesh, B. (1994), "Towards a Pre Requirements Specification Traceability Model," In ***Proceedings of the Complex Systems Engineering Synthesis and Technology Assessment Workshop -- CSESAW '94***, S.L. Howell, M. McCoy (Eds.), 1-8.

Ramesh, B. (1992), "A Study for the Development of Requirements Traceability Model," In ***Proceedings of the Complex Systems Engineering Synthesis and Assessment Technology Workshop (CSESAW '92)***, S.L. Howell and P.Q. Hwang (Eds.), Naval Surface Warfare Center Dahlgren Division, Dahlgren, VA, 575-579.

Rao, S.S., K. Sundararaju, B.G. Prakashi and C. Balakrishna (1992), "Multiobjective Fuzzy Optimization Techniques for Engineering Design," ***Computers and Structures 42***, 1, 37-44.

Rasmussen, J. and K.J.Vicente (1989), "Coping With Human Errors Through System Design: Implications for Ecological Interface Design," ***International Journal of Man-Machine Studies 31***, 5, 517-534.

Reed, D.A. (1993), "Performance Instrumentation Techniques for Parallel Systems," In ***Performance Evaluation of Computer and Communications Systems: Joint Tutorial Papers of Performance '93 and Sigmetrics '93 Proceedings***, L. Donatiello and R. Nelson (Eds.), Springer-Verlag, Inc., New York, 463-490.

Reichardt, J. (1993), "Preventive Software Engineering," In ***Proceedings of the 4th European Software Engineering Conference***, I. Sommerville and M. Paul (Eds.), Springer-Verlag, Inc., New York, 251-262.

Reid, G.B. and T.E. Nygren (1988), "The Subjective Workload Assessment Technique: A Scaling Procedure for Measuring Mental Workload," In ***Human Mental Workload***, P.A. Hancock and N. Meshkati (Eds.), North-Holland, 185-218.

Rombach, H.D. (1990), "Design Measurement: Some Lessons Learned," ***IEEE Software 3***, 17-25.

Rostek, P.M. (1994), "Power System Design for Massive Parallel Computer Systems," In ***Proceedings of the IEEE Applied Power Electronics Conf and Exposition 2***, 808-814.

Rotenstreich, S. (1994), "Toward Measuring Potential Coupling," ***Software Engineering Journal 2***, 83-90.

Rouse, W.B. (1991), "Conceptual Design of a Computational Environment for Analyzing Tradeoffs Training and Aiding," ***Information and Decision Technologies 17***, 4, 227-254.

Rouse, W.B. and J.M. Hammer (1991), "Assessing the Impact of Modeling Limits on Intelligent Systems," ***IEEE Transactions on Systems, Man and Cybernetics 21***, 6, 1549-1559.

Rouse, W.B., W.R. Cody and K.R. Boff (1991), "Human Factors of System Design: Understanding and Enhancing the Role of Human Factors Engineering," ***International Journal of Human Factors in Manufacturing 1***, 1, 87-104.

Rowe, A.L., J. French, K.J. Neville and D.R. Eddy (1992), "The Prediction of Cognitive Performance Degradations During Sustained Operations," In ***Proceedings of the Human Factors Society 36th Annual Meeting 36***, 1, 111-115.

Rowe, K.W. (1992), "Sitting on the Ergonomic Hot Seat," In ***Proceedings of the Human Factors Society 36***, 2, 984-986.

Rueb, M.J., M. Vidulich and J. Hassoun (1992), "Establishing Workload Acceptability. An Evaluation of a Proposed KC-135 Cockpit Redesign," In ***Proceedings of the Human Factors Society 36***, 1, 17-21.

Ruff, D.N. and R.K. Paasch (1993), "Consideration of Failure Diagnosis in Conceptual System Design of Mechanical Systems," ***Design Theory and Methodology DE-53***, ASME, 175-187.

Rundlet, N. and W.D. Miller (1994), "DOORS to the Digitized Battlefield: Managing Requirements Discovery and Traceability," In ***Proceedings of the Complex Systems Engineering Synthesis and Technology Assessment Workshop -- CSESAW '94***, S.L. Howell, M. McCoy (Eds.), 23-28.

Ryder, J., A. Zaklad, F. Glenn and W. Zachary (1990), "An Integrated Embedded Training and Decision Aiding Design Methodology," In ***Proceedings of the Human Factors Society Annual Meeting 34***, 1163-1166.

Saaty, J. (1993) ***Expert Choice User Manual***, Expert Choice, Inc., MaClean, VA.

Saaty, R.W. (1987), "The Analytical Hierarchy Process -- What it is and How it is Used," ***Mathematical Modelling 9***, 3-5, 161-176.

Saaty, T.L. (1984), "Measuring the Fuzziness of Sets," ***Journal of Cybernetics 4***, 4, 53-61.

Saaty, T.L. (1972), "An Eigenvalue Allocation Model for Prioritization and Planning, Energy Management and Policy Center, Univ. of Pennsylvania, Philadelphia, PA.

Saaty, T.L. (1987), "How to Handle Dependence with the Analytic Hierarchy Process," ***Mathematical Modelling 9***, 3-5, 369-376.

Saaty, T.L. (1990a)," ***Decision Making for Leaders***, RWS Publications, Pittsburgh, PA.

Saaty, T.L. (1980) ***The Analytic Hierarchy Process***, McGraw-Hill, Inc.

Saaty, T.L. (1990b)," ***Multicriteria Decision Making: The Analytic Hierarchy Process***, RWS Publications, Pittsuburg, PA.

Saaty, T.L. (1986), "Axiomatic Foundation of the Analytic Hierarchy Process," ***Management Science 32***, 7, 841-855.

Saaty, T.L. (1993), "What is Relative Measurement? The Ratio Scale Phantom," ***Mathematical and Computer Modelling 17***, 4/5, 1-12.

Saaty, T.L. (1994) ***Fundamentals of Decision Making and Priority Theory with the Analytic Hierarchy Process***, RWS Publications, Pittsburg, PA.

Saaty, T.L. and L.G. Vargas (1984), "Inconsistency and Rank Preservation," ***Journal of Mathematical Psychology 28***, 205-214.

Salo, A.A. (1993), "Inconsistency Analysis by Approximately Specified Priorities," ***Mathematical and Computer Modelling 17***, 4/5, 123-133.

Sanchez, E. and L.A. Zadeh (1987) ***Approximate Reasoning in Intelligent Systems, Decision and Control***, Pergamon Press, Oxford.

Sanders, M.S. and E.J. McCormick (1993) ***Human Factors in Engineering and Design***, Seventh Edition, McGraw-Hill, Inc.

Sanou, K., S.G. Romaniuk and L.O. Hall (1993), "Connectionist Production System With Approximate Matching Function," In ***Proceedings of the IEEE International Conference on Fuzzy Systems***, 415-421.

Sarno, K.J. and C.D. Wickens (1992), "Predictive Workload Models and Multiple-Task Performance," In ***Proceedings of the Human Factors Society 36th Annual Meeting 36***, 1, 12-16.

Schepers, J. (1994), "Five Axioms for Systems Integration," ***Software Systems Engineering PD-59***, ASME Press, 29-36.

Schlemmer, R.A. and K. VanNoppen (1994), "Integrated Process Support for Large and Complex Development Projects," ***Software Systems Engineering PD-59***, ASME Press, 19-27.

Schwartz, D. (1987) ***Concept Testing***, Published by the American Management Association, New York, NY.

Selcon, S. J., R.M. Taylor and E. Koritsas (1991), "Workload or Situational Awareness?. TLX vs. SART for Aerospace Systems Design Evaluation," In ***Proceedings of the Human Factors Society 35***, 1, 62-66.

Shabalin, A.N. (1992), "Generation of Models for Reliability Growth," In ***Proceedings of the Annual Reliability and Maintainability Symposium***, R. Evans (Ed.), IEEE Press, 299-302.

Shen, R., X. Meng and Y. Yan (1990), "Analytic Hierarchy Process Applied to Synthetically Evaluate the Labour Intensity of Jobs," ***Ergonomics 33***, 7, 867-874.

Shepperd, M. (1989), "Design Metrics: an Emperical Analysis," ***Software Engineering Journal 1990***, 1, 3-10.

Shields, J. and G. Silcock (1986), "Application of the Hierarchical Approach to Fire Safety," ***Fire Safety Journal 11***, 3, 235-242.

Singh, S.K. and R.B. Singh (1993), "Cost Analysis of a Man-Machine System Operating under Changing Operator Conditions," ***Microelectronics and Reliability 33***, 9, 1267-1274.

Smith, C.U. (1993), "Software Performance Engineering," In ***Performance Evaluation of Computer and Communications Systems: Joint Tutorial Papers of Performance '93 and Sigmetrics '93 Proceedings***, L. Donatiello and R. Nelson (Eds.), Springer-Verlag, Inc., New York, 509-536.

Smith, C.U. and L.G. Williams (1993), "Software Performance Engineering: A Case Study Including Performance Comparison with Design Alternatives," In ***IEEE Transactions on Software Engineering 19***, 7, 720-741.

Somaiya, S.R. (1993), "SENATE: A Software System for Evaluation of Simulation Results, M.S. Thesis, Department of Computer Science, Virginia Tech, Blacksburg, VA.

Srinivasan, V. and P.J. Bolster (1990), "Industrial Bond Rating Model Based on the Analytic Hierarchy Process," ***European Journal of Operational Research 48***, 1, 105-119.

Stankovic, J.A. (1988), "Misconceptions About Real-Time Computing," ***IEEE Computer 21***, 10, 10-19.

Stassen, H.G., G. Johannsen and N. Moray (1990), "Internal Representation," Internal Model, Human Performance Model and Mental Workload," ***Automatica 26***, 4, 811-820.

Stassen, H.G., J.J. Kok, R. van der Veldt and G. Heslinga (1986), "Modelling Human Operator Performance, Possibilities and Limitations," ***IFAC Proceedings Series***, 8, 141-146.

Staveland, L. (1991), "MIDAS TLM: Man-Machine Integrated Design and Analysis System Task Loading ," In ***Proceedings of the IEEE International Conference on Systems, Man and Cybernetics***, 2, 1219-1223.

Stavrianidis, P. (1992), "Reliability and Uncertainty Analysis of Hardware Failures of a Programmable Electronic System," ***Reliability Engineering and System Safety 39***, 309-324.

Steward, M.G. (1992), "Modelling Human Error Rates for Human Reliability Analysis of a Structural Design Task," ***Reliability Engineering and System Safety 36***, 171-180.

Stoll, C.N. and C.F. Zubas (1994), "Total Quality Management (TQM), "in a Non-Manufacturing EnvironmentIn ***Proceedings of the Annual Reliability and Maintainability Symposium***, 46-51.

Stoyenko, A.D., L.R. Welch, R.L. Harrison and H. Crisp (1993), "A Methodology for Complex Computer Systems Engineering," In ***Proceedings of the Complex Systems Engineering Synthesis and Assessment Technology Workshop (CSESAW '93)***, S.L. Howell and W. Farr (Eds.), Naval Surface Warfare Center Dahlgren Division, Dahlgren, VA, 65-73.

Strosnider, J.K. (1993), "Engineering Analysis of Real-Time Systems," In ***Proceedings of the Complex Systems Engineering Synthesis and Assessment Technology Workshop (CSESAW '93)***, S.L. Howell and W. Farr (Eds.), Naval Surface Warfare Center Dahlgren Division, Dahlgren, VA, 15-28.

Sullivan, C. and H.S. Blackman (1991), "Insights Into Pilot Situation Awareness Using Verbal Protocol Analysis," In ***Proceedings of the Human Factors Society 35***, 1, 57-61.

Taguchi, G. (1986), ***Introduction to Quality Engineering: Designing Quality into Products and Processes***, The Organization, Tokyo.

Talbert, M.L., O. Balci and R.E. Nance (1994), "Application of the Analytic Hierarchy Process to Complex System Design Evaluation," In ***Proceedings of the Complex Systems Engineering Synthesis and Technology Assessment Workshop -- CSESAW '94***, S.L. Howell, M. McCoy (Eds.), 145-155.

Tamai, T. (1993), "Current Practices in Software Processes for System Planning and Requirements Analysis," ***Information and Software Technology 35***, 6/7, 339-344.

Tarnoff, D.L. (1991), "A Decision Support Tool for Preliminary System Design, M.S. Thesis, Department of Electrical Engineering, Virginia Tech, Blacksburg, VA.

Tennant, D.V. (1993), "Avoiding Failure in Project Management," ***Advances in Instrumentation and Control 48***, 1, 675-686.

Terano, T., Y. Murayama and N. Akiyama (1983), "Human Reliability and Safety Evaluation of Man-Machine Systems," ***Automatica 19***, 719-722|

Thurston, D.L. (1991), "A Formal Method for Subjective Design Evaluation with Multiple Attributes," ***Research in Design Engineering 3***, J.R. Dixon and S. Finger (Eds.), 105-122.

Thurston, D.L. and A. Locascio (1992), "Multiattribute Design Optimization and Concurrent Engineering, Decision Systems Laboratory, Dept. of General Engineering, U. of Illinois, Urbana, IL, 61801.

Thurston, D.L. and J.V. Carnahan (1992), "Fuzzy Ratings and Utility Analysis in Preliminary Design Evaluation of Multiple Attributes," ***Journal of Mechanical Design - Transactions of the ASME, 114***, 4, 648-658.

Thurston, D.L. and J.V. Carnahan (1992), "Fuzzy Ratings and Utility Analysis in Preliminary Design Evaluation of Multiple Attributes," ***Transactions of the ASME***, 648-671.

Thurston, D.L. and T. Liu (1991), "Design Evaluation of Multiple Attributes under Uncertainty," ***International Journal of Systems Automation: Research and Applications (SARA), 1***, 143-159.

Thurston, D.L., J.V. Carnahan and T. Liu (1991), "Optimization of Design Utility," ***Design Theory and Methodology***, DE-31, ASME, 173-180.

Toshtzar, M. (1988), "Multi-criteria Decision Making Approach to Computer Software Evaluation: Application of the Analytical Hierarchy Process," ***Mathematical and Computer Modelling 11***, 276-281.

Towsley, D. (1993), "Providing Quality of Service in Packet-Switched Networks," In ***Performance Evaluation of Computer and Communications Systems: Joint Tutorial Papers of Performance '93 and Sigmetrics '93 Proceedings***, L. Donatiello and R. Nelson (Eds.), Springer-Verlag, Inc., New York, 560-586.

Triantaphyllou, E. and S.H. Mann (1990), "An Evuation of the Eigenvalue Approach for Determining the Membership Values in Fuzzy Sets," ***Fuzzy Sets and Systems 35***, 295-301.

Triggs, T.J. (1988), "Some Vehicle Design Factors That Influence Driver Visual Performance," ***International Journal of Vehicle Design 9***, 4-5, 542-547.

Turek, J. and D. Shasha (1992), "The Many Faces of Consensus in Distributed Systems," ***IEEE Computer 25***, 6, 8-17.

Tzafestas, S.G. and A.N. Veretsanopolous (1994) ***Fuzzy Reasoning in Information, Decision and Control Systems***, Kluwer Academic Press, Boston, MA.

Ujita, H. (1992), "Human Characteristics of Plant Operation and Man-Machine Interface," ***Reliability Engineering and System Safety 38***, 119-124.

Vadde, S., S. Swadi, N. Bhattacharya, F. Mistree and J.K. Allen (1992), "Design of An Aircraft Tire: A Study in Modeling Uncertainty," ***Advances in Design Automation - 1992 American Society of Mechanical Engineers, Design Engineering Division (Publication), DE 44***, 2, 315-325.

Vander Wiel, S.A. and L.G. Votta (1993), "Assessing Software Designs Using Capture-Recapture Methods," ***IEEE Transactions on Software Engineering 19***, 11, 1045-1054.

van Laarhoven, P. and W. Pedrycz (1983), "A Fuzzy Extension of Saaty's Priority Theory," ***Fuzzy Sets and Systems 11***, 229-241.

Verfurth, S.C., T. Govindaraj and C.M. Mitchell (1991), "OFMSPERT for the 727: An Investigation into Intent Inferencing on the Flight Deck," In ***Proceedings of the IEEE International Conference on Systems, Man and Cybernetics***, 2, 1311-1316.

Verma, D. and J. Knezevic (1994), "Conceptual System Design Evaluation: Handling the Uncertainty," In ***Proceedings of the International Logistics Congress***, J. Knezevic (Ed.), Exeter, UK, March 4, 1994, 118-131.

Verma, D. and W. Fabrycky (1995), "Development of a Fuzzy Requirements Matrix to Support Conceptual System Design," In ***International Conference on Engineering Design, ICED '95***, Praha, August 22-24, 1995

Vidulich, M.A. (1988), "Speed Responses and Dual-Task Performance: Better Time-Sharing Or Asymmetric Transfer? ," ***Human Factors 30***, 4, 517-529.

Visciola, M., A. Armando and S. Bagnara (1992), "Communication Patterns and Errors in Flight Simulation," ***Reliability Engineering and System Safety 36***, 253-259.

Voland, G. (1987), "Decision Strategies for Expert Systems Software in Engineering Design," In ***Proceedings of the 1987 International Conference on Engineering Design - International Congress on Planning and Design Theory 2***, 697-704.

von Altrock, C. (1995) ***Fuzzy Logic & Neuro Fuzzy Applications Explained***, Prentice-Hall, Inc., Englewood Cliffs, N.J.

Voung, L.T. and H. Thuan (1989), "Relational Database Extended By Application of Fuzzy Set Theory and Linguistic Variables," ***Computers and Artificial Intelligence 8***, 2, 153-168.

Waheed, A., B. Kronmüller, R. Sinha and D. Rover (1994), "A Toolkit for Advanced Performance Analysis," In ***Proceedings of the 2nd Annual Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems - MASCOTS '94***, V. Madisetti, et al. (Eds.), IEEE Computer Society Press, Los Alamitos, CA, 376-380.

Wang, L. and T. Raz (1991), "Analytical Hierarchy Process Based on Data Flow Diagram," ***Computers in Industrial Engineering 20***, 3, 355-365.

Webster, D.D. (1987), "Hardware, Software, Firmware Allocation of Function in Systems Development, M.S. Dissertation, Department of Industrial and Systems Engineering, Virginia Tech, Blacksburg, VA.

Webster (1987), ***Webster's Ninth New Collegiate Dictionary***, Merriam-Webster, Inc. Springfield, MA.

Wedley, W.C. (1993), "Consistency Prediction for Incomplete AHP Matrices," ***Mathematical and Computer Modelling 17***, 4/5, 151-161.

Wedley, W.C., B.Schoner and T.S. Tang (1993), "Starting Rules for Incomplete Comparisons in the Analytic Hierarchy Process," ***Mathematical and Computer Modelling 17***, 4/5, 93-100.

Weiland, M.Z., B. Cooke and B. Peterson (1992), "Designing and Implementing Decision Aids for a Complex Environment Using Goal Hierarchies," In ***Proceedings of the Human Factors Society 36th Annual Meeting 36***, 394-398.

Wein, A.S. and A. Sathaye (1990), "Validating Complex Computer System Availability Models," ***IEEE Transactions on Reliability 39***, 4, 468-479.

Weisgerber, S.A. and M.P. Kibbe (1990), "Targeting Decisions Using Multiple Imaging Sensors. Operator Performance and Calibration," In ***Proceedings of the Human Factors Society 34***, 1, 56-60.

Weiss, E.N. (1987), "Using the Analytical Hierarchy Process in a Dynamic Environment," ***Mathematical Modelling 9***, 3-5, 211-216.

Werntz, D.G. (1989), "Software Design With Fuzzy Requirements (A Case Study)," In ***Proceedings of the IEEE International Conference on Systems Engineering***, 201-203.

West, G.J., R.J. Eckenrode and P. C. Goodman (1991), "Investigation of Events Involving Human Performance," In ***Proceedings of the Human Factors Society 35***, 1, 655-658.

West, S.A. and L. Adelman (1990), "Incorporating Cognitive Psychology Into Interface Design: A Review of Books," ***Information and Decision Technologies 16***, 2, 107-115.

Westermann, W.E. and M.M. Tanik (1994), "Interface Architecture Analysis Mechanisms," ***Software Systems Engineering PD-59***, ASME Press, 37-44.

Whitaker, L.A., L. Peters and G. Garinther (1990), "Effects of Speech Intelligibility Among Bradley Fighting Vehicle Crew Members: SIMNET Performance and Subjective Workload," In ***Proceedings of the Human Factors Society 34***, 186-188.

Wimmer, K. (1994), "Modeling and Communicating Requirements of Large Systems," ***Software Systems Engineering PD-59***, ASME Press, 1-8.

Wise, J.A., V.D. Hopkin and P. Stager (Eds.), "(1993), ***Verfication and Validation of Complex Systems: Human Factors Issues***, NATO ASI Series F: Computer and Systems Sciences v. 110, Springer-Verlag, NY.

Wisner, A., F. Daniellou, L. Pinsky and J. Theureau (1989), "Process Supervision and Control. Design of Technical Systems and Organization. Training of Operators," In ***IFAC Proceedings Series***, 3, 33-40.

Wohl, J., D. Serfaty, E. Entin and R. James (1988), "Human Cognitive Performance in Antisubmarine Warfare: Situation Assessment and Data Fusion," ***IEEE Transactions on Systems, Man and Cybernetics 18***, 5, 777-785.

Wong, F.S., T.J. Ross and A.C. Boissonnade (1985), "Fuzzy Sets and Survivability Analysis of Protective Structures," In ***Proceedings of Workshop on Civil Engineering Applications of Fuzzy Sets***, 193-218.

Wood, K.L. and E.K. Antonsson (1990), "Modeling Imprecision and Uncertainty in Preliminary Engineering Design," ***Mechanism & Machine Theory 25***, 3, 305-324.

Wood, K.L. and E.K. Antonsson (1989), "Computations with Imprecise Parameters in Engineering Design: Background and Theory," ***IEEE Transactions on Mechanisms, Transmissions and Automation in Design 111***, Waldron, K.J. (Ed.), 616-625.

Woods, D.D. (1990a) On Taking Human Performance Seriously in Risk Analysis. Comments on Dougherty," ***Reliability Engineering and System Safety 29***, 3, 375-381.

Woods, D.D. (1990b) Risk and Human Performance. Measuring the Potential for Disaster," ***Reliability Engineering & System Safety 29***, 3, 387-405.

Woods, D.D. (1987), "Commentary: Cognitive Engineering in Complex and Dynamic Worlds," ***International Journal of Man-Machine Studies 27***, 5-6, 571-585.

Woods, D.D. (1991), "Representation Aiding: A Ten Year Retrospective," In ***Proceedings of the IEEE International Conference on Systems, Man and Cybernetics***, 2, 1173-1176.

Woods, D.D. and M.C. Eastman (1989), "Integrating Principles for Human-Computer Interaction Into the Design Process: Heterarchically Organized HCI Principles," In ***Proceedings of the IEEE International Conference on Systems, Man and Cybernetics***, 1, 29-34.

Woods, D.D. and R.I. Cook (1991), "Nosocomial Automation: Technology-Induced Complexity and Human Performance," In ***Proceedings of the IEEE International Conference on Systems, Man and Cybernetics***, 2, 1279-1282.

Woods, D.D., E.M. Roth and H.E. Pople, Jr. (1990), "Modeling Operator Performance in Emergencies," In ***Proceedings of the Human Factors Society 34***, 2, 1132-1136.

Woods, D.D., L. Johannesen and S.S. Potter (1992), "Sophistry of Guidelines: Revisiting Recipes for Color Use in Human-Computer Interface Design," In ***Proceedings of the Human Factors Society 36***, 1, 418-422.

Xia, G., P. Zeephongsekul and S. Kumar (1993) ***Microelectronics and Reliability 33***, 1, 81-86.

Xu, J. and D.L. Parnas (1993), "On Satisfying Timing Constraints in Hard-Real-Time Systems," ***IEEE Transactions on Software Engineering 19***, 1, 70-84.

Yager, R.R. (1978), "Fuzzy Decision Making Including Unequal Objectives," ***Fuzzy Sets and Systems 1***, 87-95.

Yamada, S., K. Tokuno and S. Osaki (1993), "Software Reliability Measurement in Imperfect Debugging Environment and its Application," ***Reliability Engineering and System Safety 40***, 139-147.

Yan, J. and G. Svedberg (1991), "Analytical Hierarchy Process (AHP), Model for Screening Working Fluids in Heat Engine Cycles," In ***Proceedings of the Intersociety Energy Conversion Engineering Conference 2***, 424-429.

Yang, S.C. and D. Verma (1994), "System Dependability Evaluation," In ***Proceedings of the International Logistics Congress***, J. Knezevic (Ed.), Exeter, UK, March 4, 1994, 152-162.

Yin, B.H. and J.W. Winchester (1978), "The Establishment and Use of Measures to Evaluate the Quality of Software DesignsIn ***Proceedings of the Software Quality and Assurance Workshop***, 45-52.

Yoshimura, S., T. Ohsuka and J.I. Itoh (1992), "Reliability Aspects of Man-Machine Interface," ***Reliability Engineering and System Safety 38***, 1-2, 125-134.

Yu, X. and D.A. Lamb (1995), "Metrics Applicable to Software Design," ***Annals of Software Engineering 1***, 23-42.

Zadeh, L.A. (1965), "Fuzzy Sets," ***Information and Control 8***, 338-353.

Zadeh, L.A. (1975), "The Concept of a Linguistic Variable and its Application to Approximate Reasoning," ***Information Science 8,*** 15-24.

Zadeh, L.A. (1983), "Commonsense Knowledge Representation Based on Fuzzy Logic," ***Computer 16***, 10, 61-65.

Zage, W., D. Zage and C. Wilburn (1995), "Avoiding Metric Monsters: A Design Metrics Approach," ***Annals of Software Engineering 1***, 1, 43-55.

Zahedi, F. (1986), "The Analytical Hierarchy Process: A Survey of the Method and its Applications," ***Interfaces 16***, 4, 96-108.

Zahedi, F. (1987), "A Utility Approach to the Analytical Hierarchy Process," ***Mathematical Modelling 9***, 3-5, 387-395.

Zahedi, F. (1990), "Method for Quantitative Evaluation of Expert Systems," ***European Journal of Operational Research 48***, 1, 136-147.

Zahedi, F. and N. Ashrafi (1991), "Software Reliability Allocation Based on Structure, Utility, Price and Cost," ***IEEE Transactions on Software Engineering 17***, 4, 345-356.

Zemel, T. and W. Rossak (1994), "The Role of Software Architectures in Mega-Systems Development," ***Software Systems Engineering PD-59***, ASME Press, 77-81.

Zhang, W.G. and R.A. Gaggioli (1992), "Multiobjective Optimization With Aid of Fuzzy-Set Concepts," In ***Proceedings of ECOS '92 International Symposium on Efficient Cost Optimization Simulation***, 255-267.

Zhou, H., K. Schwan and I.F. Akyildiz (1992), "Performance Effects of Information Sharing in a Distributed Multiprocessor Real-Time Scheduler," In ***Proceedings of the Real-Time System Symposium***, L. Sha, S. Davidson and I. Lee (Eds.), IEEE Press, Los Alamitos, CA, 46-55.

Zimmerman, H-J. (1991) ***Fuzzy Set Theory and its Applications***, Second (Revised) Edition, Kluwer Academic Publishing, Norwell, MA.

Zimmerman, H.J. and H.J. Sebastian (1993), "Optimization and Fuzziness in Problems of Design and Configuration," In ***Proceedings of the IEEE International Conference on Fuzzy Systems***, 1237-1240.

Zinser, K. and R.L. Henneman (1988), "Development and Evaluation of a Model of Human Performance in a Large-Scale System," ***IEEE Transactions on Systems, Man and Cybernetics 18***, 3, 367-375.

APPENDIX A. EXAMPLE SYSTEM DESIGN QUALITY INDICATOR HIERARCHY

In this appendix we present a sizable generic indicator hierarchy, described in Chapter 4. Figure A.1 is replete with multiple-parented indicators, reflecting the true multi-way nature of indicator influence relationships. Currently there are over 400 distinct indicators in the hierarchy. It is printed in nested indenture form, so multiply-related indicators are repeated. Definitions and sources for individual indicators are included in the electronically readable format, available for the NeXTSTEP platform. Contact the author for questions regarding access.

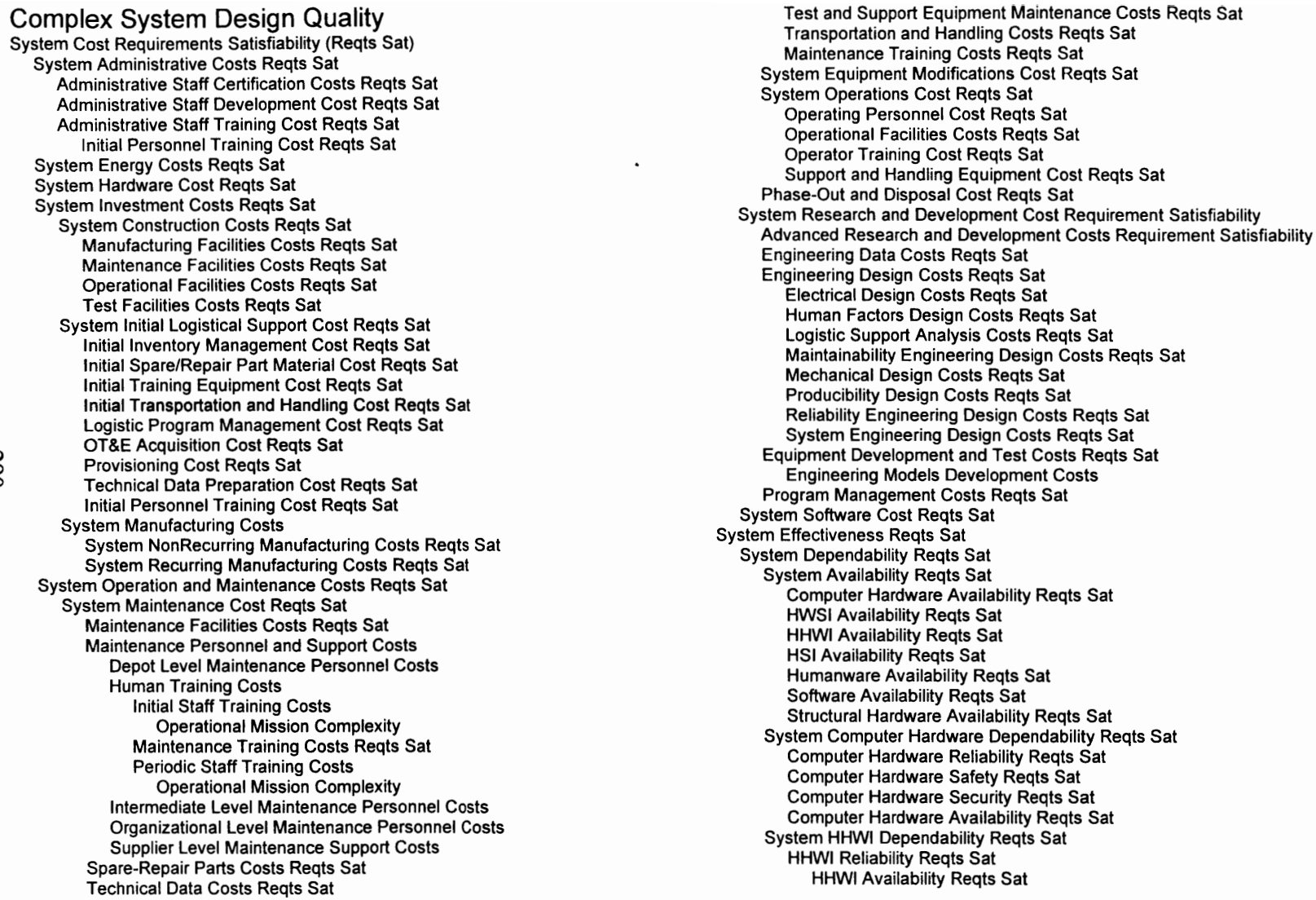


Figure A.1. Example System Design Quality Indicator Hierarchy.

- HHWI Equipment Fitness for Use Reqts Sat
- Human Operator Training Reqts Sat
- Job Aid Quality Reqts Sat
  - Job Aid Comprehensiveness
  - Job Aid Controllability
  - Job Aid Text Content Readibility
    - Job Aid Text Content Brevity
    - Job Aid Text Content Letter Size
      - Job Aid Text Content Simplicity
      - Job Aid Text Content Vocabulary Constraint
    - Job Aid Use of Icons
  - Job Aid Referability
  - Job Aid Content Understandability
  - Job Aid Usability
- Operational Control Environment (OCE) Quality Reqts Sat
  - OCE Ambient Air Quality Level Suitability
  - OCE Ambient Conditions Suitability
  - OCE Ambient Lighting Level Suitability
  - OCE Ambient Noise Level Suitability
  - OCE Ambient Temperature Level Suitability
  - OCE Ambient Wind Sutiability
  - OCE Vibration Sutiability
- HHWI Safety Reqts Sat
- HHWI Security Reqts Sat
- HHWI Availability Reqts Sat

System HSI Dependability Reqts Sat
- HSI Reliability Reqts Sat
- HSI Safety Reqts Sat
- HSI Security Reqts Sat
- HSI Availability Reqts Sat

System HWSI Dependability Reqts Sat
- HWSI Reliability Reqts Sat
  - HWSI Failure Avoidance Effectiveness
    - HWSI Common Mode Failure Avoidance Effectiveness
    - HWSI Use of COTS Components
    - HWSI Use of Design Diversity
    - HWSI Use of Integrated Formal Methods
    - HWSI Use of Reliability-Enhancing Design Methodologies
    - HWSI Use of Simplifying Abstractions
  - HWSI Failure Removal Effectiveness
    - HWSI Fault Injection Effectiveness
    - HWSI Testing Reqts Sat
  - HWSI Failure Tolerance Effectiveness
    - HWSI Common Mode Failure Detection Plans Effectiveness
    - HWSI Use of Exception Raising Mechanisms
    - HWSI Use of Run-Time Checks
    - HWSI Use of Watchdog Timers
  - HWSI Common Mode Failure Recovery Plans Effectiveness
    - HWSI Use of Checkpoints for Restart
    - HWSI Use of Exception Handlers
- HWSI Safety Reqts Sat
- HWSI Security Reqts Sat
- HWSI Availability Reqts Sat

System Humanware Dependability Reqts Sat
- Humanware Reliability Reqts Sat
  - Humanware Individual Reliability Reqts Sat
    - HHWI Error Prevention Reqts Sat
      - Humanware Cognitive Error Prevention Reqts Sat
        - Mental Fatigue
        - Mental Workload
        - Task Complexity
          - Procedural Complexity
      - Humanware Error Masking Capability Reqts Sat
      - Humanware Error Recovery Capability Reqts Sat
      - Humanware Physical Error Prevention Reqts Sat
        - Humanware Physical Fatigue Prevention Reqts Sat
          - Humanware Physical Workload Acceptability Reqts Sat
            - HHWI Complexity
            - HSI Complexity
          - Shift Length
        - Humanware Physical Workload Acceptability Reqts Sat
          - HHWI Complexity
          - HSI Complexity
        - Task Complexity
          - Procedural Complexity
      - Quality of Assistance
        - Job Aid Cognitive Compatibility
        - Job Aid Situational Compatibility
        - Job Aid Timeliness of Assistance
        - Job Aid Quality Reqts Sat
          - Job Aid Comprehensiveness
          - Job Aid Controllability
          - Job Aid Text Content Readibility
            - Job Aid Text Content Brevity
            - Job Aid Text Content Letter Size
            - Job Aid Text Content Simplicity
              - Job Aid Text Content Vocabulary Constraint
            - Job Aid Use of Icons

Figure A.1. Example System Design Quality Indicator Hierarchy (continued).

- Job Aid Referability
- Job Aid Content Understandability
- Job Aid Usability
- Quantity of Assistance
- HSI Error Prevention Reqts Sat
- Humanware Cognitive Error Prevention Reqts Sat
- Mental Fatigue
- Mental Workload
- Task Complexity
- Procedural Complexity
- Humanware Error Masking Capability Reqts Sat
- Humanware Error Recovery Capability Reqts Sat
- Humanware Physical Error Prevention Reqts Sat
- Humanware Physical Fatigue Prevention Reqts Sat
- Humanware Physical Workload Acceptability Reqts Sat
- HHWI Complexity
- HSI Complexity
- Shift Length
- Humanware Physical Workload Acceptability Reqts Sat
- HHWI Complexity
- HSI Complexity
- Task Complexity
- Procedural Complexity
- Quality of Assistance
- Job Aid Cognitive Compatibility
- Job Aid Situational Compatibility
- Job Aid Timeliness of Assistance
- Job Aid Quality Reqts Sat
- Job Aid Comprehensiveness
- Job Aid Controllability
- Job Aid Text Content Readibility
- Job Aid Text Content Brevity
- Job Aid Text Content Letter Size
- Job Aid Text Content Simplicity
- Job Aid Text Content Vocabulary Constraint
- Job Aid Use of Icons
- Job Aid Referability
- Job Aid Content Understandability
- Job Aid Usability
- Quantity of Assistance
- Humanware Work Group Reliability Reqts Sat
- Individual Competence
- Leadership Competence
- Leadership Style
- Team Cohesion
- Pairwise Member Cohesion
- Team Competence
- Quantity of Assistance
- Humanware Safety Reqts Sat
- Humanware Individual Safety Reqts Sat
- Humanware Work Group Safety Reqts Sat
- Humanware Security Reqts Sat
- Humanware Individual Security Reqts Sat
- Humanware Work Group Security Reqts Sat
- Humanware Availability Reqts Sat
- System Reliability Reqts Sat
- Software Reliability Reqts Sat
- Use of Hierarchical Decomposition
- Software Module Cohesion
- Software Module Completeness
- Software Module Complexity
- Software Module Conciseness
- Software Module Self-Descriptiveness
- Software Module Structural Simplicity
- Software Module Consistency
- Software Module Coupling
- Software Module Modifiability
- Software Module Traceability
- Use of Information Hiding
- Software Module Understandability
- Software Module Well-Defined Interface
- Software Module Cohesion
- Software Module Complexity
- Software Module Conciseness
- Software Module Self-Descriptiveness
- Software Module Structural Simplicity
- Software Module Coupling
- Software Module Modifiability
- Use of Stepwise Refinement
- Software Module Cohesion
- Software Module Completeness
- Software Module Complexity
- Software Module Conciseness
- Software Module Self-Descriptiveness
- Software Module Structural Simplicity
- Software Module Consistency
- Software Module Coupling
- Use of Structured Programming

Figure A.1. Example System Design Quality Indicator Hierarchy (continued).

- Software Module Accuracy
- Software Module Fault Tolerance
- Software Module Readability
- Software Module Complexity
  - Software Module Conciseness
  - Software Module Self-Descriptiveness
  - Software Module Structural Simplicity
- Software Module Consistency
- Software Module Understandability

Structural Hardware Reliability Reqts Sat

Humanware Reliability Reqts Sat

- Humanware Individual Reliability Reqts Sat
  - HHWI Error Prevention Reqts Sat
    - Humanware Cognitive Error Prevention Reqts Sat
      - Mental Fatigue
      - Mental Workload
      - Task Complexity
        - Procedural Complexity
    - Humanware Error Masking Capability Reqts Sat
    - Humanware Error Recovery Capability Reqts Sat
    - Humanware Physical Error Prevention Reqts Sat
      - Humanware Physical Fatigue Prevention Reqts Sat
        - Humanware Physical Workload Acceptability Reqts Sat
          - HHWI Complexity
          - HSI Complexity
        - Shift Length
      - Humanware Physical Workload Acceptability Reqts Sat
        - HHWI Complexity
        - HSI Complexity
      - Task Complexity
        - Procedural Complexity
    - Quality of Assistance
      - Job Aid Cognitive Compatibility
      - Job Aid Situational Compatibility
      - Job Aid Timeliness of Assistance
      - Job Aid Quality Reqts Sat
        - Job Aid Comprehensiveness
        - Job Aid Controllability
        - Job Aid Text Content Readibility
          - Job Aid Text Content Brevity
          - Job Aid Text Content Letter Size
          - Job Aid Text Content Simplicity
            - Job Aid Text Content Vocabulary Constraint
          - Job Aid Use of Icons
        - Job Aid Referability
        - Job Aid Content Understandability
        - Job Aid Usability
    - Quantity of Assistance
  - HSI Error Prevention Reqts Sat
    - Humanware Cognitive Error Prevention Reqts Sat
      - Mental Fatigue
      - Mental Workload
      - Task Complexity
        - Procedural Complexity
    - Humanware Error Masking Capability Reqts Sat
    - Humanware Error Recovery Capability Reqts Sat
    - Humanware Physical Error Prevention Reqts Sat
      - Humanware Physical Fatigue Prevention Reqts Sat
        - Humanware Physical Workload Acceptability Reqts Sat
          - HHWI Complexity
          - HSI Complexity
        - Shift Length
      - Humanware Physical Workload Acceptability Reqts Sat
        - HHWI Complexity
        - HSI Complexity
      - Task Complexity
        - Procedural Complexity
    - Quality of Assistance
      - Job Aid Cognitive Compatibility
      - Job Aid Situational Compatibility
      - Job Aid Timeliness of Assistance
      - Job Aid Quality Reqts Sat
        - Job Aid Comprehensiveness
        - Job Aid Controllability
        - Job Aid Text Content Readibility
          - Job Aid Text Content Brevity
          - Job Aid Text Content Letter Size
          - Job Aid Text Content Simplicity
            - Job Aid Text Content Vocabulary Constraint
          - Job Aid Use of Icons
        - Job Aid Referability
        - Job Aid Content Understandability
        - Job Aid Usability
    - Quantity of Assistance

Humanware Work Group Reliability Reqts Sat

- Individual Competence
- Leadership Competence
- Leadership Style

Figure A.1. Example System Design Quality Indicator Hierarchy (continued).

- Team Cohesion
  - Pairwise Member Cohesion
- Team Competence
- Quantity of Assistance

Computer Hardware Reliability Reqts Sat

HHWI Reliability Reqts Sat
- HHWI Availability Reqts Sat
- HHWI Equipment Fitness for Use Reqts Sat
- Human Operator Training Reqts Sat
- Job Aid Quality Reqts Sat
  - Job Aid Comprehensiveness
  - Job Aid Controllability
  - Job Aid Text Content Readibility
    - Job Aid Text Content Brevity
    - Job Aid Text Content Letter Size
    - Job Aid Text Content Simplicity
      - Job Aid Text Content Vocabulary Constraint
    - Job Aid Use of Icons
  - Job Aid Referability
  - Job Aid Content Understandability
  - Job Aid Usability
- Operational Control Environment (OCE) Quality Reqts Sat
  - OCE Ambient Air Quality Level Suitability
  - OCE Ambient Conditions Suitability
  - OCE Ambient Lighting Level Suitability
  - OCE Ambient Noise Level Suitability
  - OCE Ambient Temperature Level Suitability
  - OCE Ambient Wind Sutiability
  - OCE Vibration Sutiability

HSI Reliability Reqts Sat

HWSI Reliability Reqts Sat
- HWSI Failure Avoidance Effectiveness
  - HWSI Common Mode Failure Avoidance Effectiveness
  - HWSI Use of COTS Components
  - HWSI Use of Design Diversity
  - HWSI Use of Integrated Formal Methods
  - HWSI Use of Reliability-Enhancing Design Methodologies
  - HWSI Use of Simplifying Abstractions
- HWSI Failure Removal Effectiveness
  - HWSI Fault Injection Effectiveness
  - HWSI Testing Reqts Sat
- HWSI Failure Tolerance Effectiveness
  - HWSI Common Mode Failure Detection Plans Effectiveness
    - HWSI Use of Exception Raising Mechanisms
    - HWSI Use of Run-Time Checks
    - HWSI Use of Watchdog Timers
  - HWSI Common Mode Failure Recovery Plans Effectiveness
    - HWSI Use of Checkpoints for Restart
    - HWSI Use of Exception Handlers

System Safety Reqts Sat
- Software Safety Reqts Sat
- Structural Hardware Safety Reqts Sat
- Humanware Safety Reqts Sat
  - Humanware Individual Safety Reqts Sat
  - Humanware Work Group Safety Reqts Sat
- Computer Hardware Safety Reqts Sat
- HWSI Safety Reqts Sat
- HHWI Safety Reqts Sat
- HSI Safety Reqts Sat

System Security Reqts Sat
- Software Security Reqts Sat
  - Software Access Auditing Reqts Sat
  - Software Access Control Reqts Sat
- Structural Hardware Security Reqts Sat
- Humanware Security Reqts Sat
  - Humanware Individual Security Reqts Sat
  - Humanware Work Group Security Reqts Sat
- Computer Hardware Security Reqts Sat
- HWSI Security Reqts Sat
- HHWI Security Reqts Sat
- HSI Security Reqts Sat

System Software Dependability Reqts Sat
- Real-Time Software Correctness Reqts Sat
  - Use of Life Cycle Verification Techniques
    - Software Module Early Error Detection Capability
    - Software Module Visibility of Behavior
    - Software Module Traceability
  - Use of Hierarchical Decomposition
    - Software Module Cohesion
    - Software Module Completeness
    - Software Module Complexity
      - Software Module Conciseness
      - Software Module Self-Descriptiveness
      - Software Module Structural Simplicity
    - Software Module Consistency
    - Software Module Coupling
    - Software Module Modifiability
    - Software Module Traceability

Figure A.1. Example System Design Quality Indicator Hierarchy (continued).

- Use of Stepwise Refinement
  - Software Module Cohesion
  - Software Module Completeness
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Consistency
  - Software Module Coupling
- Use of Structured Programming
  - Software Module Accuracy
  - Software Module Fault Tolerance
  - Software Module Readability
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Consistency
  - Software Module Understandability
- Software Correctness Reqts Sat
- Use of Hierarchical Decomposition
  - Software Module Cohesion
  - Software Module Completeness
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Consistency
  - Software Module Coupling
  - Software Module Modifiability
  - Software Module Traceability
- Use of Life Cycle Verification Techniques
  - Software Module Early Error Detection Capability
  - Software Module Visibility of Behavior
  - Software Module Traceability
- Use of Stepwise Refinement
  - Software Module Cohesion
  - Software Module Completeness
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Consistency
  - Software Module Coupling
- Use of Structured Programming
  - Software Module Accuracy
  - Software Module Fault Tolerance
  - Software Module Readability
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Consistency
  - Software Module Understandability
- Software Availability Reqts Sat
- Software Reliability Reqts Sat
- Use of Hierarchical Decomposition
  - Software Module Cohesion
  - Software Module Completeness
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Consistency
  - Software Module Coupling
  - Software Module Modifiability
  - Software Module Traceability
- Use of Information Hiding
  - Software Module Understandability
  - Software Module Well-Defined Interface
  - Software Module Cohesion
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Coupling
  - Software Module Modifiability
- Use of Stepwise Refinement
  - Software Module Cohesion
  - Software Module Completeness
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Consistency
  - Software Module Coupling
- Use of Structured Programming
  - Software Module Accuracy

Figure A.1. Example System Design Quality Indicator Hierarchy (continued).

Software Module Fault Tolerance
Software Module Readability
Software Module Complexity
Software Module Conciseness
Software Module Self-Descriptiveness
Software Module Structural Simplicity
Software Module Consistency
Software Module Understandability
Software Safety Reqts Sat
Software Security Reqts Sat
Software Access Auditing Reqts Sat
Software Access Control Reqts Sat
System Structural Hardware Dependability Reqts Sat
Structural Hardware Availability Reqts Sat
Structural Hardware Reliability Reqts Sat
Structural Hardware Safety Reqts Sat
Structural Hardware Security Reqts Sat
System Survivability Reqts Sat
System Performance Reqts Sat
Computer Hardware Performance Reqts Sat
Computer Hardware Response Time
Computer IO Efficiency Reqts Sat
I/O Data Transfer Rate
I/O Disk Latency
I/O Seek Time
Hardware Hard Deadline Reqts Sat
Hardware Schedulability Reqts Sat
Parallel Computer Hardware Architecture Indicators
Parallel Computer Hardware Communication Processor Speed at Node
Parallel Computer Hardware Memory Storage at Node
Parallel Computer Hardware Network Protocol Efficiency
Parallel Computer Hardware Network Topology Efficiency
Parallel Computer Hardware No. Virtual Processors at Node
Parallel Computer Hardware Number of Nodes
Parallel Computer Hardware Processor Speed at Node
Parallel Computer Hardware Performance Reqts Sat
Parallel Computer Hardware Capacity Reqts Sat
Parallel Computer Hardware Cycles Per Second
Parallel Computer Hardware Instructions Per Second
Parallel Computer Hardware MFLOPS
Parallel Computer Hardware Peak MIPS
Parallel Computer Hardware Clock Rate
Parallel Computer Hardware Timing Constraint Reqts Sat
Parallel Computer Hardware Degree of Asynchronous Communication
Parallel Computer Hardware Clock Rate
Parallel Computer Hardware Communication Processor Speed at Node
Parallel Computer Hardware Cycles Per Second
Parallel Computer Hardware Instructions Per Second
Parallel Computer Hardware MFLOPS
Parallel Computer Hardware Peak MIPS
Parallel Computer Hardware Processor Speed at Node
Parallel Computer Hardware Execution Time Reqts Sat
Parallel Computer Hardware Communication Processor Speed at Node
Parallel Computer Hardware Cycles Per Second
Parallel Computer Hardware Instructions Per Second
Parallel Computer Hardware MFLOPS
Parallel Computer Hardware Peak MIPS
Parallel Computer Hardware Processor Speed at Node
Parallel Computer Hardware Processor Accuracy Reqts Sat
Parallel Computer Hardware Processor Precision Reqts Sat
Parallel Computer Hardware Response Time Requrements Satisfiability
Parallel Computer Hardware Clock Rate
Parallel Computer Hardware Speedup Reqts Sat
Parallel Computer Hardware Degree of Parallelism
Parallel Computer Hardware Number of Nodes
Parallel Computer Hardware Clock Rate
Parallel Computer Hardware Processor Speed at Node
Computer HWSI Performance Reqts Sat
Communications HWSI Performance Reqts Sat
Communications HWSI Quality of Service Reqts Sat
Communications HWSI Index of Dispersion
Communications HWSI Packet Delivery Probability Reqts Sat
Communications HWSI Transmission Connection Reqts Sat
Communication HWSI Connection Blockage Probability Reqts Sat
Communication HWSI Connection Setup Delay Reqts Sat
Communications HWSI Transmission Source Reqts Sat
Communications HWSI Transmission Path Propagation Delay Reqts Sat
Communications HWSI Transmission Receiver Buffer SpaceReqts Sat
Communications HWSI Transmission Source Buffer Space Reqts Sat

Figure A.1. Example System Design Quality Indicator Hierarchy (continued).

Communications HWSI Non-Real-Time Performance Reqts Sat
Communications HWSI End-to-End ReliabilityReqts Sat
Communications HWSI Packet Delay Reqts Sat
Communications HWSI Information Throughput Reqts Sat
Communications HWSI Real-Time Performance Reqts Sat
Communications HWSI Bounded Transmission Time Reqts Sat
Communications HWSI Jitter Time Bound Reqts Sat
Communications HWSI Packet Delivery Probability Reqts Sat
Parallel Computer Hardware Throughput Reqts Sat
Parallel Computer Hardware Inter-Processor Communication Speed
Parallel Computer Hardware Processor Speed at Node
Parallel Computer Hardware Processor Utilization
Parallel Computer Hardware Load Balance-Distribution
Effectiveness
Parallel Computer Hardware Task Allocation Strategy
Effectiveness
Parallel Computer Hardware Topology Suitability to Problem Size
Parallel Computer Hardware Processor Speed at Node
Software Task Schedulability Constraint Satisfiability
Dynamic Task Schedulability Constraint Satisfiability
Real-Time Software Hard Deadline Reqts Sat
Real-Time Software Task Precedence Constraints Satisfiability
Real-Time Software Task Release Time Constraints Satisfiability
Task Schedulability Multi-Programming Level
Task Schedulability Preemption Cost
Static Task Schedulability Constraint Satisfiability
Task Schedulability Multi-Programming Level
Task Schedulability Preemption Cost
Parallel Computer Hardware Performance Reqts Sat
Parallel Computer Hardware Capacity Reqts Sat
Parallel Computer Hardware Cycles Per Second
Parallel Computer Hardware Instructions Per Second
Parallel Computer Hardware MFLOPS
Parallel Computer Hardware Peak MIPS
Parallel Computer Hardware Clock Rate
Parallel Computer Hardware Timing Constraint Reqts Sat
Parallel Computer Hardware Degree of Asynchronous
Communication
Parallel Computer Hardware Clock Rate
Parallel Computer Hardware Communication Processor Speed at
Node
Parallel Computer Hardware Cycles Per Second
Parallel Computer Hardware Instructions Per Second
Parallel Computer Hardware MFLOPS
Parallel Computer Hardware Peak MIPS
Parallel Computer Hardware Processor Speed at Node
Parallel Computer Hardware Execution Time Reqts Sat
Parallel Computer Hardware Communication Processor Speed at
Node
Parallel Computer Hardware Cycles Per Second
Parallel Computer Hardware Instructions Per Second
Parallel Computer Hardware MFLOPS
Parallel Computer Hardware Peak MIPS
Parallel Computer Hardware Processor Speed at Node
Parallel Computer Hardware Processor Accuracy Reqts Sat
Parallel Computer Hardware Processor Precision Reqts Sat
Parallel Computer Hardware Response Time Requrements
Satisfiability
Parallel Computer Hardware Clock Rate
Parallel Computer Hardware Speedup Reqts Sat
Parallel Computer Hardware Degree of Parallelism
Parallel Computer Hardware Number of Nodes
Parallel Computer Hardware Clock Rate
Parallel Computer Hardware Processor Speed at Node
HSI Performance Reqts Sat
HSI Human Error Rate
HSI Interaction Timing Constraints Satisfiability
HSI Human Response Time
HSI Interface Response Time
HSI Interface Complexity
HSI Operator Training Adequacy
HSI Interface Error Rate
Humanware Performance Reqts Sat
Administrative Performance Reqts Sat
Operator Individual Performance Reqts Sat
Operator Cognitive Performance Reqts Sat
Operator Cognitive Abilities
Reasoning Skills
HHWI Complexity
Operator Task Stream Performance Endurance Reqts Sat
Operator Mental Fitness Quality
Operator Physical Fitness Quality
Humanware Physical Fatigue Prevention Reqts Sat
Humanware Physical Workload Acceptability Reqts Sat
HHWI Complexity
HSI Complexity
Shift Length
Operator Mental Performance Reqts Sat

Figure A.1. Example System Design Quality Indicator Hierarchy (continued).

- Mental Task Characteristics Acceptability
  - Information Complexity
  - Information Density Acceptability
  - Mental Task Complexity
  - Mental Task Criticality Acceptability
  - Mental Task Novelty
  - Mental Task Rate Acceptability
  - Task Required Turnaround Time Acceptability
  - Task Structure Acceptability
  - Operational Control Environment (OCE) Ambient Air Quality Level Suitability
    - OCE Ambient Conditions Suitability
    - OCE Ambient Lighting Level Suitability
    - OCE Ambient Noise Level Suitability
    - OCE Ambient Temperature Level Suitability
    - OCE Ambient Wind Sutiability
- Operator Mental Suitability to Task
  - Operator Arousal State Reqts Sat
  - Operator Cognitive Capabilities Acceptability
  - Operator Decision Style Acceptability
  - Operator Experience Reqts Sat
  - Operator Motivational State Acceptability
  - Operator Physical Abilities Acceptability
  - Operator Psychomotor Skills Acceptability
  - Operator Sensory Abilities Acceptability
  - Operator Training Acceptability
  - Personal Utility System Quality
- Task Performance Shaping Factors Acceptability
  - Objective Task Difficulty
  - Operator Fatigue Potential Acceptability
  - Operator Perceived Task Difficulty Acceptability
  - Operator Vigilance Maintenance Acceptability
  - Speed Stress Acceptability
- Operator Mental Fitness Quality
- Operator Physical Performance Reqts Sat
  - Physical Abilities
    - Audition
    - Communications Skills
    - Conceptual Skills
    - Gross Motor Skills
    - Psychomotor Skills
    - Speed Loaded Skills
    - Vision
- Operator Task Completion Rate Reqts Sat
  - Operator Skill-to-Task Suitability
    - Experience
    - Human Operator Training Reqts Sat
  - Task Accomplishment Resource Adequacy
  - Task Accomplishment Resources Adequacy
  - Task Minimimum Required TAT
  - Quality of Assistance
    - Job Aid Cognitive Compatibility
    - Job Aid Situational Compatibility
    - Job Aid Timeliness of Assistance
    - Job Aid Quality Reqts Sat
      - Job Aid Comprehensiveness
      - Job Aid Controllability
      - Job Aid Text Content Readibility
        - Job Aid Text Content Brevity
        - Job Aid Text Content Letter Size
        - Job Aid Text Content Simplicity
          - Job Aid Text Content Vocabulary Constraint
        - Job Aid Use of Icons
      - Job Aid Referability
      - Job Aid Content Understandability
      - Job Aid Usability
  - Task Complexity
    - Procedural Complexity
- Operator Team-Group Performance Reqts Sat
- Operator Task Completion Rate Reqts Sat
  - Operator Skill-to-Task Suitability
    - Experience
    - Human Operator Training Reqts Sat
  - Task Accomplishment Resource Adequacy
  - Task Accomplishment Resources Adequacy
  - Task Minimimum Required TAT
  - Quality of Assistance
    - Job Aid Cognitive Compatibility
    - Job Aid Situational Compatibility
    - Job Aid Timeliness of Assistance
    - Job Aid Quality Reqts Sat
      - Job Aid Comprehensiveness
      - Job Aid Controllability
      - Job Aid Text Content Readibility
        - Job Aid Text Content Brevity
        - Job Aid Text Content Letter Size
        - Job Aid Text Content Simplicity
          - Job Aid Text Content Vocabulary Constraint

Figure A.1. Example System Design Quality Indicator Hierarchy (continued).

- Job Aid Use of Icons
- Job Aid Referability
- Job Aid Content Understandability
- Job Aid Usability
- Task Complexity
- Procedural Complexity
- Operator Task Stream Performance Endurance Reqts Sat
- Operator Mental Fitness Quality
- Operator Physical Fitness Quality
- Humanware Physical Fatigue Prevention Reqts Sat
- Humanware Physical Workload Acceptability Reqts Sat
- HHWI Complexity
- HSI Complexity
- Shift Length

Software Performance Reqts Sat
- Real-Time Software Graceful Degradation Reqts Sat
- Real-Time Software Performance Engineering Reqts Sat
- Real-Time Software Feasibility Reqts Sat
- Real-Time Software Maintainability Reqts Sat
- Real-Time Software Simplicity Reqts Sat
- Real-Time Software Usability
- Real-Time Software User Preferability
- Real-Time Software Correctness Reqts Sat
- Use of Life Cycle Verification Techniques
- Software Module Early Error Detection Capability
- Software Module Visibility of Behavior
- Software Module Traceability
- Use of Hierarchical Decomposition
- Software Module Cohesion
- Software Module Completeness
- Software Module Complexity
- Software Module Conciseness
- Software Module Self-Descriptiveness
- Software Module Structural Simplicity
- Software Module Consistency
- Software Module Coupling
- Software Module Modifiability
- Software Module Traceability
- Use of Stepwise Refinement
- Software Module Cohesion
- Software Module Completeness
- Software Module Complexity
- Software Module Conciseness
- Software Module Self-Descriptiveness
- Software Module Structural Simplicity
- Software Module Consistency
- Software Module Coupling
- Use of Structured Programming
- Software Module Accuracy
- Software Module Fault Tolerance
- Software Module Readability
- Software Module Complexity
- Software Module Conciseness
- Software Module Self-Descriptiveness
- Software Module Structural Simplicity
- Software Module Consistency
- Software Module Understandability

Real-Time Software Workload Capacity Reqts Sat
- Real-Time Software Batch Performance Reqts Sat
- Real-Time Software Batch Data Volume Reqts Sati
- Real-Time Software On-Line Performance Capacity
- Real-Time Software Capacity for Concurrent Users Reqts Sat
- Real-Time Software On-Line Interarrival Rate Handling Reqts Sat
- Real-Time Software On-Line Request Arrival Rates Handling

Capability
- Real-Time Software Reactive Performance Reqts Sat

Software System Response Time Reqts Sat

Software System Throughput Reqts Sat

Software Correctness Reqts Sat
- Use of Hierarchical Decomposition
- Software Module Cohesion
- Software Module Completeness
- Software Module Complexity
- Software Module Conciseness
- Software Module Self-Descriptiveness
- Software Module Structural Simplicity
- Software Module Consistency
- Software Module Coupling
- Software Module Modifiability
- Software Module Traceability
- Use of Life Cycle Verification Techniques
- Software Module Early Error Detection Capability
- Software Module Visibility of Behavior
- Software Module Traceability
- Use of Stepwise Refinement
- Software Module Cohesion
- Software Module Completeness
- Software Module Complexity

Figure A.1. Example System Design Quality Indicator Hierarchy (continued).

- Software Module Conciseness
- Software Module Self-Descriptiveness
- Software Module Structural Simplicity
- Software Module Consistency
- Software Module Coupling
- Use of Structured Programming
- Software Module Accuracy
- Software Module Fault Tolerance
- Software Module Readability
- Software Module Complexity
  - Software Module Conciseness
  - Software Module Self-Descriptiveness
  - Software Module Structural Simplicity
- Software Module Consistency
- Software Module Understandability
- Software Task Schedulability Constraint Satisfiability
- Dynamic Task Schedulability Constraint Satisfiability
  - Real-Time Software Hard Deadline Reqts Sat
  - Real-Time Software Task Precedence Constraints Satisfiability
  - Real-Time Software Task Release Time Constraints Satisfiability
  - Task Schedulability Multi-Programming Level
  - Task Schedulability Preemption Cost
- Static Task Schedulability Constraint Satisfiability
  - Task Schedulability Multi-Programming Level
  - Task Schedulability Preemption Cost
- Structural Hardware Peformance Reqts Sat
- Hardware Bending Strength Constraint Satisfiability
- Hardware Buckling Resistance Constraint Satisfiability
- Hardware Deflection Under Load Reqts Sat
- Hardware Fatigue Resistence Reqts Sat
- Hardware Noise Tolerance Constraint Satisfiability
  - Potential Noise Exposure (dB, duration)
- Hardware Safety Reqts Sat
- Hardware Service Life Reqts Sat
  - Hardware Maintainability Reqts Sat
- Hardware Temperature Resistance Reqts Sat
- Hardware Torsional Stiffness Reqts Sat
- Hardware Vibration Tolerance Constraint Satisfiability
- Hardware Yield Resistance Constraint Satisfiability

System Life Cycle Applicability

- System Deployability Reqts Sat
  - System Operational Expendables Resupply Capability Quality
  - System Transportation to Operational Location Quality
- System Retirement Reqts Sat
- System Disposability Reqts Sat
- System Recyclability Reqts Sat

System Maintenance Reqts Sat

- Software Maintainability Reqts Sat
- Use of Concurrent Documentation
- Software Module Complexity
  - Software Module Conciseness
  - Software Module Self-Descriptiveness
  - Software Module Structural Simplicity
- Software Module Modifiability
- Software Module Readability
- Software Module Traceability
- Software Module Understandability
- Use of Functional Decomposition
- Software Module Expandability
- Software Module Cohesion
- Software Module Completeness
- Software Module Complexity
  - Software Module Conciseness
  - Software Module Self-Descriptiveness
  - Software Module Structural Simplicity
- Software Module Coupling
- Software Module Modifiability
- Software Module Understandability
- Use of Hierarchical Decomposition
- Software Module Cohesion
- Software Module Completeness
- Software Module Complexity
  - Software Module Conciseness
  - Software Module Self-Descriptiveness
  - Software Module Structural Simplicity
- Software Module Consistency
- Software Module Coupling
- Software Module Modifiability
- Software Module Traceability
- Use of Information Hiding
- Software Module Understandability
- Software Module Well-Defined Interface
- Software Module Cohesion
- Software Module Complexity
  - Software Module Conciseness
  - Software Module Self-Descriptiveness
  - Software Module Structural Simplicity
- Software Module Coupling

Figure A.1. Example System Design Quality Indicator Hierarchy (continued).

- Software Module Modifiability
- Use of Stepwise Refinement
  - Software Module Cohesion
  - Software Module Completeness
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Consistency
  - Software Module Coupling
- Use of Structured Programming
  - Software Module Accuracy
  - Software Module Fault Tolerance
  - Software Module Readability
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Consistency
  - Software Module Understandability
- System Computer Hardware Maintainability
- System Engineering Maintenance Plan Quality
- System HHWI Maintainability
- System HSI Maintainability
- System HWSI Maintainability
- System Maintenance Facility Quality
  - System Maintenance Facility Administrative Quality
  - System Maintenance Facility Operator Quality
  - System Maintenance Facility Supply Quality
- System Maintenance Support Quality
- System Structural Hardware Maintainability

System Modifiability Reqts Sat

System Producibility Reqts Sat

- System Computer Hardware Producibility Reqts Sat
- System HHWI Producibility Reqts Sat
  - Software Portability Reqts Sat
    - Use of Device Independence
    - Use of Operating System Independence
    - Use of Concurrent Documentation
      - Software Module Complexity
        - Software Module Conciseness
        - Software Module Self-Descriptiveness
        - Software Module Structural Simplicity
      - Software Module Modifiability
    - Software Module Readability
    - Software Module Traceability
    - Software Module Understandability
    - Use of Functional Decomposition
      - Software Module Expandability
      - Software Module Cohesion
      - Software Module Completeness
      - Software Module Complexity
        - Software Module Conciseness
        - Software Module Self-Descriptiveness
        - Software Module Structural Simplicity
      - Software Module Coupling
      - Software Module Modifiability
      - Software Module Understandability

Software Reusability Reqts Sat

- Use of Concurrent Documentation
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Modifiability
  - Software Module Readability
  - Software Module Traceability
  - Software Module Understandability
- Use of Functional Decomposition
  - Software Module Expandability
  - Software Module Cohesion
  - Software Module Completeness
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Coupling
  - Software Module Modifiability
  - Software Module Understandability
- Use of Hierarchical Decomposition
  - Software Module Cohesion
  - Software Module Completeness
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Consistency
  - Software Module Coupling

Figure A.1. Example System Design Quality Indicator Hierarchy (continued).

- Software Module Modifiability
- Software Module Traceability
- Use of Information Hiding
  - Software Module Understandability
  - Software Module Well-Defined Interface
  - Software Module Cohesion
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Coupling
  - Software Module Modifiability
- Software Correctness Reqts Sat
  - Use of Hierarchical Decomposition
    - Software Module Cohesion
    - Software Module Completeness
    - Software Module Complexity
      - Software Module Conciseness
      - Software Module Self-Descriptiveness
      - Software Module Structural Simplicity
    - Software Module Consistency
    - Software Module Coupling
    - Software Module Modifiability
    - Software Module Traceability
  - Use of Life Cycle Verification Techniques
    - Software Module Early Error Detection Capability
    - Software Module Visibility of Behavior
    - Software Module Traceability
  - Use of Stepwise Refinement
    - Software Module Cohesion
    - Software Module Completeness
    - Software Module Complexity
      - Software Module Conciseness
      - Software Module Self-Descriptiveness
      - Software Module Structural Simplicity
    - Software Module Consistency
    - Software Module Coupling
  - Use of Structured Programming
    - Software Module Accuracy
    - Software Module Fault Tolerance
    - Software Module Readability
    - Software Module Complexity
      - Software Module Conciseness
      - Software Module Self-Descriptiveness
      - Software Module Structural Simplicity
    - Software Module Consistency
    - Software Module Understandability
- Software Maintainability Reqts Sat
  - Use of Concurrent Documentation
    - Software Module Complexity
      - Software Module Conciseness
      - Software Module Self-Descriptiveness
      - Software Module Structural Simplicity
    - Software Module Modifiability
    - Software Module Readability
    - Software Module Traceability
    - Software Module Understandability
  - Use of Functional Decomposition
    - Software Module Expandability
    - Software Module Cohesion
    - Software Module Completeness
    - Software Module Complexity
      - Software Module Conciseness
      - Software Module Self-Descriptiveness
      - Software Module Structural Simplicity
    - Software Module Coupling
    - Software Module Modifiability
    - Software Module Understandability
  - Use of Hierarchical Decomposition
    - Software Module Cohesion
    - Software Module Completeness
    - Software Module Complexity
      - Software Module Conciseness
      - Software Module Self-Descriptiveness
      - Software Module Structural Simplicity
    - Software Module Consistency
    - Software Module Coupling
    - Software Module Modifiability
    - Software Module Traceability
  - Use of Information Hiding
    - Software Module Understandability
    - Software Module Well-Defined Interface
    - Software Module Cohesion
    - Software Module Complexity
      - Software Module Conciseness
      - Software Module Self-Descriptiveness
      - Software Module Structural Simplicity
    - Software Module Coupling

Figure A.1. Example System Design Quality Indicator Hierarchy (continued).

- Software Module Modifiability
- Use of Stepwise Refinement
  - Software Module Cohesion
  - Software Module Completeness
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Consistency
  - Software Module Coupling
- Use of Structured Programming
  - Software Module Accuracy
  - Software Module Fault Tolerance
  - Software Module Readability
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Consistency
  - Software Module Understandability

System HSI Producibility Reqts Sat

- Software Correctness Reqts Sat
  - Use of Hierarchical Decomposition
    - Software Module Cohesion
    - Software Module Completeness
    - Software Module Complexity
      - Software Module Conciseness
      - Software Module Self-Descriptiveness
      - Software Module Structural Simplicity
    - Software Module Consistency
    - Software Module Coupling
    - Software Module Modifiability
    - Software Module Traceability
  - Use of Life Cycle Verification Techniques
    - Software Module Early Error Detection Capability
    - Software Module Visibility of Behavior
    - Software Module Traceability
  - Use of Stepwise Refinement
    - Software Module Cohesion
    - Software Module Completeness
    - Software Module Complexity
      - Software Module Conciseness
      - Software Module Self-Descriptiveness
      - Software Module Structural Simplicity
    - Software Module Consistency
    - Software Module Coupling
  - Use of Structured Programming
    - Software Module Accuracy
    - Software Module Fault Tolerance
    - Software Module Readability
    - Software Module Complexity
      - Software Module Conciseness
      - Software Module Self-Descriptiveness
      - Software Module Structural Simplicity
    - Software Module Consistency
    - Software Module Understandability

Software Maintainability Reqts Sat

- Use of Concurrent Documentation
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Modifiability
  - Software Module Readability
  - Software Module Traceability
  - Software Module Understandability
- Use of Functional Decomposition
  - Software Module Expandability
  - Software Module Cohesion
  - Software Module Completeness
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Coupling
  - Software Module Modifiability
  - Software Module Understandability
- Use of Hierarchical Decomposition
  - Software Module Cohesion
  - Software Module Completeness
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Consistency
  - Software Module Coupling
  - Software Module Modifiability
  - Software Module Traceability

Figure A.1. Example System Design Quality Indicator Hierarchy (continued).

- Use of Information Hiding
  - Software Module Understandability
  - Software Module Well-Defined Interface
  - Software Module Cohesion
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Coupling
  - Software Module Modifiability
- Use of Stepwise Refinement
  - Software Module Cohesion
  - Software Module Completeness
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Consistency
  - Software Module Coupling
- Use of Structured Programming
  - Software Module Accuracy
  - Software Module Fault Tolerance
  - Software Module Readability
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Consistency
  - Software Module Understandability

Software Portability Reqts Sat
- Use of Device Independence
- Use of Operating System Independence
- Use of Concurrent Documentation
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Modifiability
  - Software Module Readability
  - Software Module Traceability
  - Software Module Understandability
- Use of Functional Decomposition
  - Software Module Expandability
  - Software Module Cohesion
  - Software Module Completeness
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Coupling
  - Software Module Modifiability
  - Software Module Understandability

Software Reusability Reqts Sat
- Use of Concurrent Documentation
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Modifiability
  - Software Module Readability
  - Software Module Traceability
  - Software Module Understandability
- Use of Functional Decomposition
  - Software Module Expandability
  - Software Module Cohesion
  - Software Module Completeness
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Coupling
  - Software Module Modifiability
  - Software Module Understandability
- Use of Hierarchical Decomposition
  - Software Module Cohesion
  - Software Module Completeness
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Consistency
  - Software Module Coupling
  - Software Module Modifiability
  - Software Module Traceability
- Use of Information Hiding
  - Software Module Understandability
  - Software Module Well-Defined Interface
  - Software Module Cohesion

Figure A.1. Example System Design Quality Indicator Hierarchy (continued).

- Software Module Complexity
  - Software Module Conciseness
  - Software Module Self-Descriptiveness
  - Software Module Structural Simplicity
- Software Module Coupling
- Software Module Modifiability

System HWSI Producibility Reqts Sat

System Humanware Producibility - Staffing- Reqts Sat

System Software Producibility Reqts Sat

- Software Adaptability-Flexibility Reqts Sat
  - Use of Concurrent Documentation
    - Software Module Complexity
      - Software Module Conciseness
      - Software Module Self-Descriptiveness
      - Software Module Structural Simplicity
    - Software Module Modifiability
    - Software Module Readability
    - Software Module Traceability
    - Software Module Understandability
  - Use of Functional Decomposition
    - Software Module Expandability
    - Software Module Cohesion
    - Software Module Completeness
    - Software Module Complexity
      - Software Module Conciseness
      - Software Module Self-Descriptiveness
      - Software Module Structural Simplicity
    - Software Module Coupling
    - Software Module Modifiability
    - Software Module Understandability
  - Use of Hierarchical Decomposition
    - Software Module Cohesion
    - Software Module Completeness
    - Software Module Complexity
      - Software Module Conciseness
      - Software Module Self-Descriptiveness
      - Software Module Structural Simplicity
    - Software Module Consistency
    - Software Module Coupling
    - Software Module Modifiability
    - Software Module Traceability
  - Use of Information Hiding
    - Software Module Understandability
    - Software Module Well-Defined Interface
    - Software Module Cohesion
    - Software Module Complexity
      - Software Module Conciseness
      - Software Module Self-Descriptiveness
      - Software Module Structural Simplicity
    - Software Module Coupling
    - Software Module Modifiability
  - Use of Stepwise Refinement
    - Software Module Cohesion
    - Software Module Completeness
    - Software Module Complexity
      - Software Module Conciseness
      - Software Module Self-Descriptiveness
      - Software Module Structural Simplicity
    - Software Module Consistency
    - Software Module Coupling
  - Use of Structured Programming
    - Software Module Accuracy
    - Software Module Fault Tolerance
    - Software Module Readability
    - Software Module Complexity
      - Software Module Conciseness
      - Software Module Self-Descriptiveness
      - Software Module Structural Simplicity
    - Software Module Consistency
    - Software Module Understandability

Software Component Quality

- Degree of Incorporation of Components of Known Capability
- Software Component Parts Capability Acceptabilty
- Software Components Operational Simplicity
- Software Number of Components Acceptability

Software Design Complexity

- Software Object-Oriented Design Complexity Acceptability
  - OOD Attribute Complexity Acceptability
  - OOD Class Coupling Acceptability
  - OOD Class Hierarchy Metric Acceptability
  - OOD Cohesion Acceptability
  - OOD Operation Argument Complexity Acceptability
  - OOD Operation Complexity Acceptability
  - OOD Operation Coupling Acceptability
    - Number of Operations accessed by other classes - Acceptabilty
    - Number of Operations accessing other classes - Acceptabilty

Figure A.1. Example System Design Quality Indicator Hierarchy (continued).

- Number of Operations co-operated with other classes - Acceptabilty
- OOD Overall Class Complexity Acceptability
- Software Structural Complexity Acceptability
  - Card and Agresti's Metric Acceptability
  - Henry and Kafura's Complexity Metric Acceptability
  - McCabe's Cyclomatic Complexity Metric Acceptability
- Software Design Traceability to Requirements
- Software Efficiency Reqts Sat
  - Software Execution Efficiency Reqts Sat
  - Software Resource Utilization Efficiency Reqts Sat
- Software Testability Reqts Sat
  - Use of Functional Decomposition
    - Software Module Expandability
    - Software Module Cohesion
    - Software Module Completeness
    - Software Module Complexity
      - Software Module Conciseness
      - Software Module Self-Descriptiveness
      - Software Module Structural Simplicity
    - Software Module Coupling
    - Software Module Modifiability
    - Software Module Understandability
  - Use of Hierarchical Decomposition
    - Software Module Cohesion
    - Software Module Completeness
    - Software Module Complexity
      - Software Module Conciseness
      - Software Module Self-Descriptiveness
      - Software Module Structural Simplicity
    - Software Module Consistency
    - Software Module Coupling
    - Software Module Modifiability
    - Software Module Traceability
  - Use of Life Cycle Verification Techniques
    - Software Module Early Error Detection Capability
    - Software Module Visibility of Behavior
    - Software Module Traceability
  - Use of Stepwise Refinement
    - Software Module Cohesion
    - Software Module Completeness
    - Software Module Complexity
      - Software Module Conciseness
      - Software Module Self-Descriptiveness
      - Software Module Structural Simplicity
    - Software Module Consistency
    - Software Module Coupling
  - Use of Structured Programming
    - Software Module Accuracy
    - Software Module Fault Tolerance
    - Software Module Readability
    - Software Module Complexity
      - Software Module Conciseness
      - Software Module Self-Descriptiveness
      - Software Module Structural Simplicity
    - Software Module Consistency
    - Software Module Understandability
- Software Correctness Reqts Sat
  - Use of Hierarchical Decomposition
    - Software Module Cohesion
    - Software Module Completeness
    - Software Module Complexity
      - Software Module Conciseness
      - Software Module Self-Descriptiveness
      - Software Module Structural Simplicity
    - Software Module Consistency
    - Software Module Coupling
    - Software Module Modifiability
    - Software Module Traceability
  - Use of Life Cycle Verification Techniques
    - Software Module Early Error Detection Capability
    - Software Module Visibility of Behavior
    - Software Module Traceability
  - Use of Stepwise Refinement
    - Software Module Cohesion
    - Software Module Completeness
    - Software Module Complexity
      - Software Module Conciseness
      - Software Module Self-Descriptiveness
      - Software Module Structural Simplicity
    - Software Module Consistency
    - Software Module Coupling
  - Use of Structured Programming
    - Software Module Accuracy
    - Software Module Fault Tolerance
    - Software Module Readability
    - Software Module Complexity
      - Software Module Conciseness

Figure A.1. Example System Design Quality Indicator Hierarchy (continued).

- Software Module Self-Descriptiveness
- Software Module Structural Simplicity
- Software Module Consistency
- Software Module Understandability

Software Portability Reqts Sat
- Use of Device Independence
- Use of Operating System Independence
- Use of Concurrent Documentation
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Modifiability
  - Software Module Readability
  - Software Module Traceability
  - Software Module Understandability
- Use of Functional Decomposition
  - Software Module Expandability
  - Software Module Cohesion
  - Software Module Completeness
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Coupling
  - Software Module Modifiability
  - Software Module Understandability

Software Reliability Reqts Sat
- Use of Hierarchical Decomposition
  - Software Module Cohesion
  - Software Module Completeness
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Consistency
  - Software Module Coupling
  - Software Module Modifiability
  - Software Module Traceability
- Use of Information Hiding
  - Software Module Understandability
  - Software Module Well-Defined Interface
  - Software Module Cohesion
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Coupling
  - Software Module Modifiability
- Use of Stepwise Refinement
  - Software Module Cohesion
  - Software Module Completeness
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Consistency
  - Software Module Coupling
- Use of Structured Programming
  - Software Module Accuracy
  - Software Module Fault Tolerance
  - Software Module Readability
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Consistency
  - Software Module Understandability

Software Reusability Reqts Sat
- Use of Concurrent Documentation
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Modifiability
  - Software Module Readability
  - Software Module Traceability
  - Software Module Understandability
- Use of Functional Decomposition
  - Software Module Expandability
  - Software Module Cohesion
  - Software Module Completeness
  - Software Module Complexity
    - Software Module Conciseness
    - Software Module Self-Descriptiveness
    - Software Module Structural Simplicity
  - Software Module Coupling
  - Software Module Modifiability

Figure A.1. Example System Design Quality Indicator Hierarchy (continued).

- Software Module Understandability
- Use of Hierarchical Decomposition
- Software Module Cohesion
- Software Module Completeness
- Software Module Complexity
  - Software Module Conciseness
  - Software Module Self-Descriptiveness
  - Software Module Structural Simplicity
- Software Module Consistency
- Software Module Coupling
- Software Module Modifiability
- Software Module Traceability
- Use of Information Hiding
- Software Module Understandability
- Software Module Well-Defined Interface
- Software Module Cohesion
- Software Module Complexity
  - Software Module Conciseness
  - Software Module Self-Descriptiveness
  - Software Module Structural Simplicity
- Software Module Coupling
- Software Module Modifiability

System Software Cost Reqts Sat

System Structural Hardware Producibility Reqts Sat

System Supportability Reqts Sat

System Physical Reqts Sat

Physical Size Reqts Sat

System Physical Area Reqts Sat

- System Physical Depth
- System Physical Height
- System Physical Length
- System Physical Width

System Physical Depth Reqts Sat

System Physical Height Requirements Satistiability

System Physical Length Reqts Sat

System Physical Volume Reqts Sat

- System Physical Depth
- System Physical Height
- System Physical Length
- System Physical Width

System Physical Width Reqts Sat

Physical Weight Reqts Sat

- System Physical Weight

System Energy Reqts Sat

- Electrical Energy Consumption Constraint Satisfiability
- Energy Dissipation Reqts Sat
- Fuel Energy Consumption Reqts Sat
- System Power Supply Reqts Sat
  - Power Supply System Availability
    - Power Supply System Maintainability
    - Power Supply System On-Line Repaceability
  - Power Supply System Capacity
  - Power Supply System Redundancy
  - Power Supply System Scalability

System Physical Portability Reqts Sat

- System Physical Volume Reqts Sat
  - System Physical Depth
  - System Physical Height
  - System Physical Length
  - System Physical Width
- System Physical Weight

System Physical Ruggability Reqts Sat

- Hardware Bending Strength Constraint Satisfiability
- Hardware Buckling Resistance Constraint Satisfiability
- Hardware Deflection Under Load Reqts Sat
- Hardware Fatigue Resistence Reqts Sat
- Hardware Noise Tolerance Constraint Satisfiability
  - Potential Noise Exposure (dB, duration)
- Hardware Safety Reqts Sat
- Hardware Service Life Reqts Sat
  - Hardware Maintainability Reqts Sat
- Hardware Temperature Resistance Reqts Sat
- Hardware Torsional Stiffness Reqts Sat
- Hardware Vibration Tolerance Constraint Satisfiability
- Hardware Yield Resistance Constraint Satisfiability

System Physical Survivability Reqts Sat

- Hardware Bending Strength Constraint Satisfiability
- Hardware Buckling Resistance Constraint Satisfiability
- Hardware Deflection Under Load Reqts Sat
- Hardware Fatigue Resistence Reqts Sat
- Hardware Hard Deadline Reqts Sat
- Hardware Maintainability Reqts Sat
- Hardware Noise Tolerance Constraint Satisfiability
  - Potential Noise Exposure (dB, duration)
- Hardware Safety Reqts Sat
- Hardware Service Life Reqts Sat
  - Hardware Maintainability Reqts Sat
- Hardware Temperature Resistance Reqts Sat

Figure A.1. Example System Design Quality Indicator Hierarchy (continued).

Hardware Torsional Stiffness Reqts Sat
Hardware Vibration Tolerance Constraint Satisfiability
Hardware Yield Resistance Constraint Satisfiability

Figure A.1. Example System Design Quality Indicator Hierarchy (continued).

APPENDIX B. QUESTIONNAIRE FOR ASSESSING EVALUATIVE CREDIBILITY OF INDICATOR HIERARCHY

In Chapter 4 we describe the process of constructing a hierarchy of design quality indicators to be used in evaluating the design. We also motivate a need for an independent assessment of the indicator hierarchy itself. This is to ensure the evaluating organization avoids a Type III error, i.e., the error of solving the wrong problem. The reader is referred to Chapter 4, Section 4.5.2 for details. A questionnaire for assessing the evaluative quality of a design evaluation hierarchy follows.

Design Objectives Traceability

1. Which of the statements below reflect the degree to which the design objectives, as reflected in the design and requirements specifications, are mapped to or represented by indicators in the hierarchy?

    a. The hierarchy provides a *very thorough* mapping of design evaluation indicators to design and requirements objectives. All design objectives are directly traceable to indicators in the hierarchy.

    b. The hierarchy provides a mapping of indicators for *most* of the design objectives. (e.g., all of the primary objectives from the design and requirements specifications are represented, but some of the lower priority objectives are not traceable to indicators in the hierarchy).

    c. The hierarchy provides a mapping of *many* of the design objectives to indicators, but some primary and lower priority objectives are not represented.

    d. The hierarchy does not reflect an adequate mapping of design evaluation indicators to design and requirements objectives. Many primary and secondary objectives are not directly traceable to indicators in the design evaluation hierarchy.

2. Which design objectives are not represented by indicators?

3. In your expert opinion, to what degree does any disparity between indicator-to-requirements coverage reflect:

a. inadequate requirements definition which had to be compensated for by additional indicators in the hierarchy

minuscule small moderate large overwhelming

b. insufficient recognition or understanding of the design objectives as reflected in the design and requirements specifications.

minuscule small moderate large overwhelming

c. other reasons for any disparity (please explain).

Life Cycle Coverage

4. Does the design evaluation hierarchy contain appropriate categories and amounts of indicators to evaluate the design's accounting for system life cycle phases, including:

a. Implementation/Production	Yes	No	(if "no" please explain)
b. Deployment	Yes	No	(if "no" please explain)
c. Maintenance	Yes	No	(if "no" please explain)
d. Logistical Support	Yes	No	(if "no" please explain)
e. Modification	Yes	No	(if "no" please explain)
f. Retirement	Yes	No	(if "no" please explain)

g. other (please specify)

Evaluation Coverage and Depth

5. Does the hierarchy of indicators for measuring the specific domain (a-f) of the designed system reflect sufficient *depth of decomposition* so that the base-level indicators are directly measurable either by metrics, simulation, technical domain expert judgment, or operational domain expert judgment?

a. Hardware	Yes	No	(if "no" please explain)
b. Software	Yes	No	(if "no" please explain)

c. Humanware Yes No (if "no" please explain)

d. Hardware-Software Interaction Yes No (if "no" please explain)

e. Human-Hardware Interaction Yes No (if "no" please explain)

f. Human-Software Interaction Yes No (if "no" please explain)

6. Are there other indicator (sub)domains which do not reflect sufficient depth of refinement, such that even a subjective evaluation of the design with respect to the indicator domain could not credibly be made?

Yes No

If you answered "Yes," which indicator (sub)domains are insufficiently refined?

7. Which (if any) individual indicators have not been sufficiently decomposed to allow a credible (even subjective) measurement to be obtained?

8. If in determining answers to questions 5-7 you found there were a substantial number of indicators with insufficient decomposition, what percent of the indicators do you feel are insufficiently decomposed?

a. 1-10% b. 11-25% c. 26-40% d. 41-60% e. over 60%

9. Is the breadth of coverage of indicators comprehensive enough to measure all essential elements of the specified domains (a-f) of the designed system?

a. Hardware Yes No (if "no" please explain)

b. Software Yes No (if "no" please explain)

c. Humanware Yes No (if "no" please explain)

d. Hardware-Software Interaction Yes No (if "no" please explain)

e. Human-Hardware Interaction Yes No (if "no" please explain)

f. Human-Software Interaction Yes No (if "no" please explain)

10. Are there other indicator (sub)domains which do not reflect sufficient breadth of coverage, such that essential elements required to assess the value of the domain are not included?

Yes No

11. If you answered "Yes," which indicator (sub)domains are not comprehensively covered?

12. Which (if any) individual indicators have not been decomposed with sufficient breadth to allow a credible measurement to be obtained?

13. If in determining answers to questions 9-12 you found there were a substantial number of indicators with insufficient breadth of coverage, what percent of the indicators do you feel are inadequately covered?

a. 1-10% b. 11-25% c. 26-40% d. 41-60% e. over 60%

Indicator Influence Relationship Representation

14. We define parent-child relationships between nodes in the indicator hierarchy as *indicator influence relationships*. One indicator may parent several child indicators, and an indicator may be the child of multiple parent indicators. Which of the following statements best states your findings regarding *completeness* of indicator influence relationships.

    a. The indicator influence relationships presented in the hierarchy reflect a complete representation of *all or most* relationships between respective indicators.

    b. The indicator influence relationships presented in the hierarchy reflect *many* of the true relationships, but a non-negligible number of true relationships are not represented.

    c. The indicator influence relationships presented in the hierarchy reflect *some* of the true relationships, but a significant number of true relationships are not represented.

    d. The indicator influence relationships presented in the hierarchy reflect *few* of the true relationships.

15. If you selected (a) above, please identify which indicator influence relationships you determined to be missing from the hierarchy.

16. If you selected (b) or (c) above, what number OR percent of indicator relationships do you determine to be missing from the hierarchy?

(number) _____ OR (percent)_____%

17. If you selected (a, b, or c) above, which (if any) of the relationships do you believe to be *critical* to the credibility of the evaluation?

18. If you selected (d) above, please indicate which indicator influence relationships were correct.

19. Which of the following statements best states your findings regarding *accuracy* of indicator influence relationships.

    a. *All* or *Most* of the indicator influence relationships presented in the hierarchy are true.

    b. *Many* of the indicator influence relationships presented in the hierarchy reflect true relationships, but a non-negligible number of the relationships are not true.

    c. *Some* of the indicator influence relationships presented in the hierarchy reflect true relationships, but a significant number of relationships are not true.

    d. *Few* of the indicator influence relationships presented in the hierarchy reflect true relationships.

20. If you selected (a) above, please indicate which indicator influence relationships you determined to be untrue based on the state of the art of engineering design and analysis.

21. If you selected (b) or (c) above, what number or percent of indicator relationships do you determine to be missing from the hierarchy?

(number) _____ OR (percent)_____%

22. If you selected (a, b, or c) above, which (if any) of the false relationships do you believe to be *critical* to the credibility of the evaluation?

23. If you selected (d) above, please indicate which indicator influence relationships were true.

24. Regarding dependencies between indicators within the same sibling group, which of the following statements best describes the condition of the hierarchy?

    a. There are *no* occurrences of dependencies between indicators within the same sibling group which need to be accounted for.

    b. There are a *few* occurrences of dependencies between indicators within the same sibling group which need to be accounted for. (Please describe.)

    c. There are *several* occurrences of dependencies between indicators within the same sibling group.

    d. There are *numerous* occurrences of dependencies between indicators within the same sibling group.

25. If you selected (b) above, please indicate which inter-sibling group dependencies are present between indicators in the hierarchy, but are not explicitly represented.

26. If you selected (c or d) above, what is the extent (number or percent) of indicator sibling groups in which there are indicator influence dependencies which are not explicitly represented?

    (number)_____ or (percent)_____%

Indicator Face Validation

27. For which (if any) indicators do you question *appropriateness* of its use for measuring the concept it is intended to measure? (i.e., it is the *wrong* indicator).

28. For which (if any) indicators do you question *capability* of its use for measuring the concept it is intended to measure? (i.e., it is *too weak* to give meaningful measure).

29. Do you find any indicators in the hierarchy which are inappropriate indicators of design quality for the particular stage in design evolution at which the design is being evaluated (e.g., indicators appropriate for detailed design but not preliminary design)?

Design Crippling Potential

30. Which indicator(s) presented in the hierarchy do you believe to be so critical to the design evaluation, that if they were scored below an acceptable level, they would be capable of invalidating the entire design?

31. Are there any indicators which are not accounted for in the hierarchy which could have the effect of invalidating the design if they were scored below an acceptable level? If so, please list or give references.

Overall Assessment

32. Which statements below best express your overall assessment of the design evaluation hierarchy?

    a. If realistic weights and scores are applied by qualified experts in the respective indicator domains, this indicator hierarchy *as presented* is sufficient to yield credible evaluative results.

    b. If realistic weights and scores are applied by qualified experts in the respective indicator domains, this indicator hierarchy *with corrections as indicated* is sufficient to yield credible evaluative results.

    c. Even if realistic weights and scores are applied by qualified experts in the respective domains, this indicator hierarchy with or without indicated corrections could not yield credible evaluative results.

33. If you selected (c) above, list the areas of the indicator hierarchy which contribute most to the poor assessment of the hierarchy (worst first).

34. Which of the areas you listed in question 33 have the greatest potential for improvement?

35. Other than the specific facets of the indicator hierarchy for which we have solicited feedback, are there other dimensions of the hierarchy which you believe weaken the potential for the hierarchy to yield credible evaluative results? If so, state.

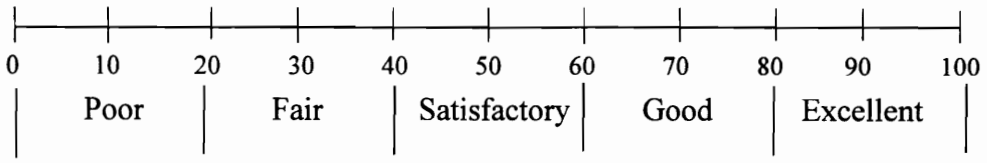
APPENDIX C. A QUESTIONNAIRE FOR INDEPENDENT ASSESSMENT OF THE METHODOLOGY

PART I: ASSESSMENT OF THE METHODOLOGY WITH RESPECT TO OBJECTIVES

1. How well does the methodology fulfill its stated Objective 1?

The methodology should facilitate the measurement and evaluation of a complex system design's dynamic (i.e., time-critical, performance critical, mission-critical) characteristics.

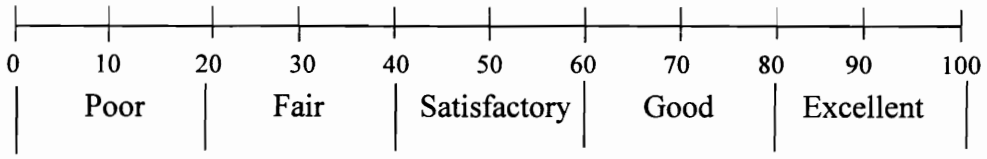
(Place an X on the scale below to assign a crisp score or mark an interval score or circle a word.)



2. How well does the methodology fulfill its stated Objective 2?

The methodology should facilitate the measurement and evaluation of a complex system design's real-time characteristics.

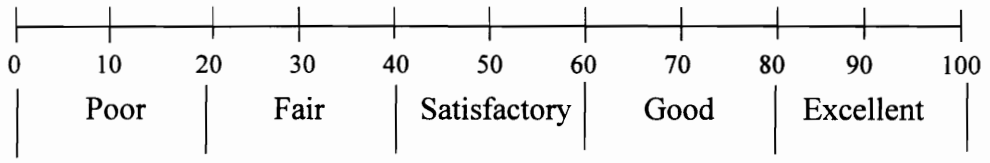
(Place an X on the scale below to assign a crisp score or mark an interval score or circle a word.)



3. How well does the methodology fulfill its stated Objective 3?

The methodology should facilitate the integrated measurement and evaluation of a complex system design with respect to its hardware, software, and humanware components, and their interfaces.

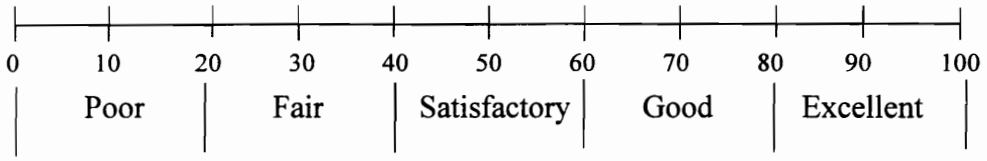
(Place an X on the scale below to assign a crisp score or mark an interval score or circle a word.)



4. How well does the methodology fulfill its stated Objective 4?

The methodology should facilitate the qualitative and quantitative evaluation of a complex system design.

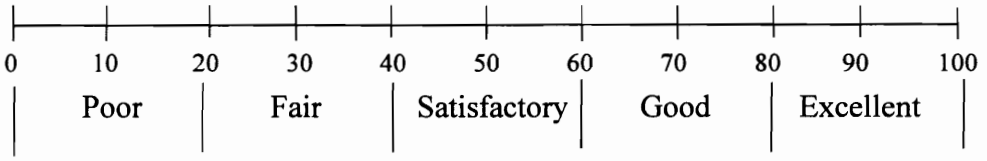
(Place an X on the scale below to assign a crisp score or mark an interval score or circle a word.)



5. How well does the methodology fulfill its stated Objective 5?

The methodology should be generically applicable for a broad spectrum of complex system designs.

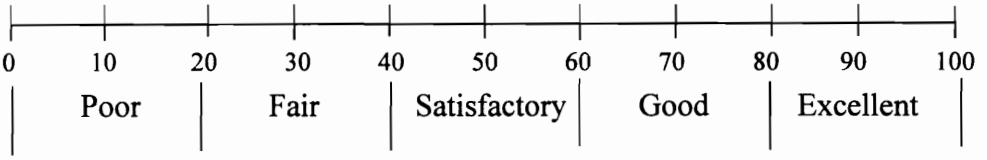
(Place an X on the scale below to assign a crisp score or mark an interval score or circle a word.)



6. How well does the methodology fulfill its stated Objective 6?

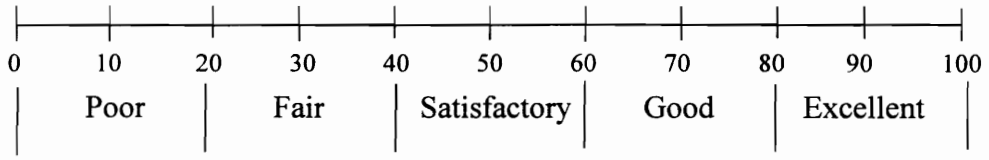
The methodology should lend itself to the incorporation of computer-aided assistance.

(Place an X on the scale below to assign a crisp score or mark an interval score or circle a word.)



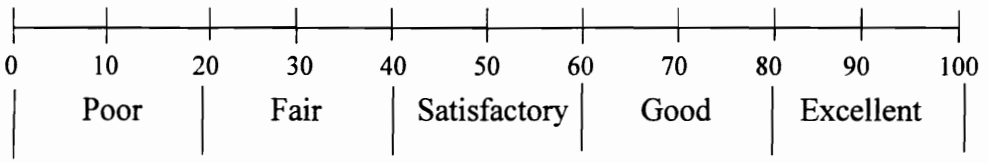
7. How well does the methodology fulfill its stated Objective 7?

The methodology should possess inherent, piece-wise credibility.
(Place an X on the scale below to assign a crisp score or mark an interval score or circle a word.)



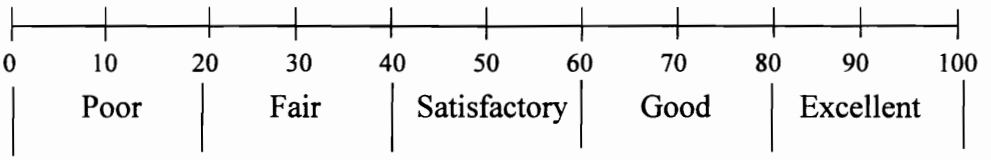
8. How well does the methodology fulfill its stated Objective 8?

The methodology should be easily usable by a design evaluation organization.
(Place an X on the scale below to assign a crisp score or mark an interval score or circle a word.)



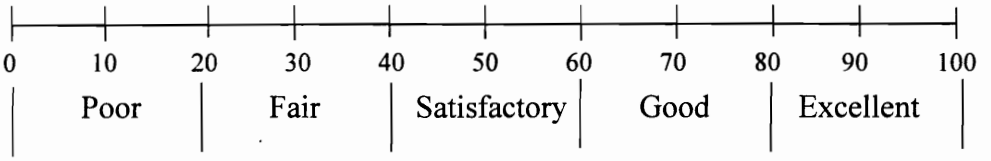
9. How well does the methodology fulfill its stated Objective 9?

The methodology should be applicable for the preliminary and detailed design phases of the system engineering process.
(Place an X on the scale below to assign a crisp score or mark an interval score or circle a word.)



10. How well does the methodology fulfill its stated Objective 10?

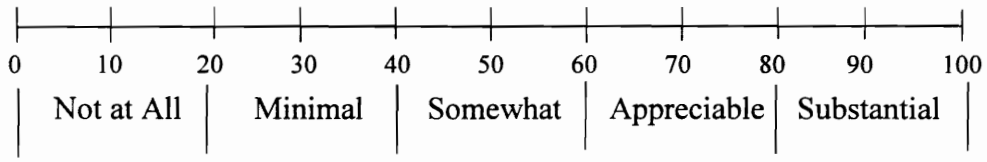
The methodology should promote independent system design evaluation to prevent developer's bias.
(Place an X on the scale below to assign a crisp score or mark an interval score or circle a word.)



PART II. ASSESSMENT OF THE METHODOLOGY WITH RESPECT TO ITS METHODS

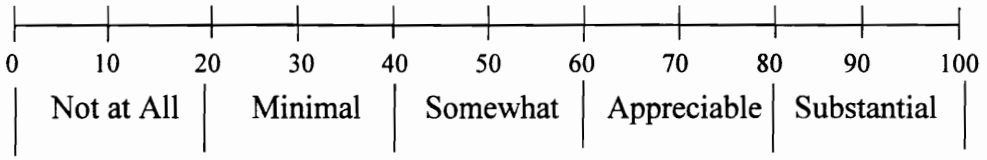
11. Indicator-Based Evaluation Method

How much does the use of this method contribute to the achievement of the overall objectives of the methodology?
(Place an X on the scale below to assign a crisp score or mark an interval score or circle a word.)



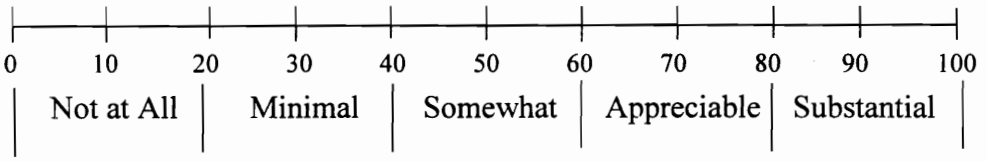
12. Analytic Hierarchy Process Method

How much does the use of this method contribute to the achievement of the overall objectives of the methodology?
(Place an X on the scale below to assign a crisp score or mark an interval score or circle a word.)



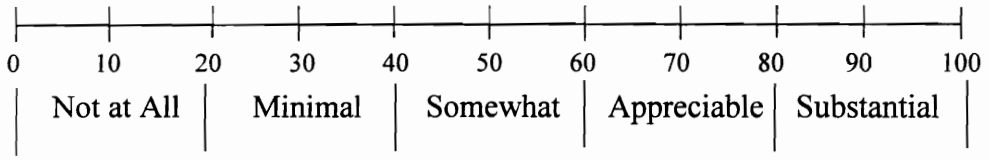
13. Fuzzy Arithmetic Method

How much does the use of this method contribute to the achievement of the overall objectives of the methodology?
(Place an X on the scale below to assign a crisp score or mark an interval score or circle a word.)



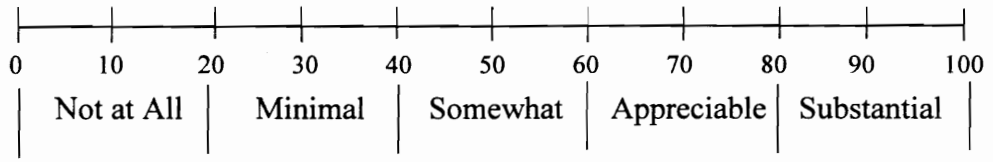
14. Expert Knowledge-Based Method

How much does the use of this method contribute to the achievement of the overall objectives of the methodology?
(Place an X on the scale below to assign a crisp score or mark an interval score or circle a word.)



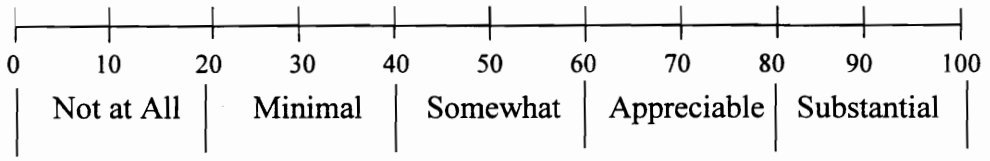
15. Visual Simulation Method

How much does the use of this method contribute to the achievement of the overall objectives of the methodology?
(Place an X on the scale below to assign a crisp score or mark an interval score or circle a word.)



16. Kiviat Diagram Method

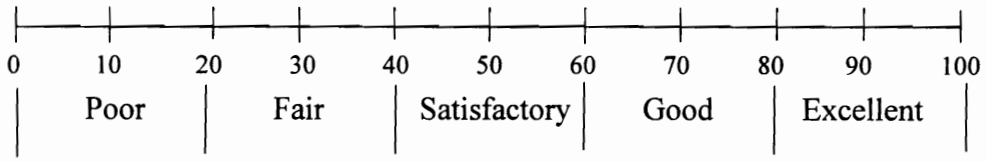
How much does the use of this method contribute to the achievement of the overall objectives of the methodology?
(Place an X on the scale below to assign a crisp score or mark an interval score or circle a word.)



PART III. ASSESSMENT OF THE METHODOLOGY WITH RESPECT TO ITS FEATURES

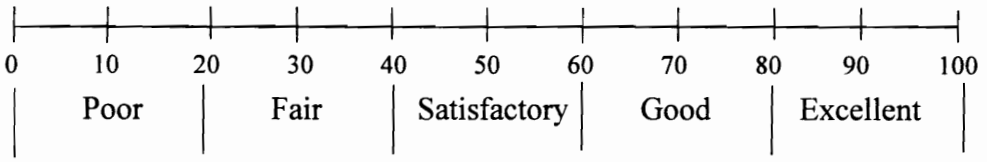
17. How well are the methods in Part II integrated within the methodology?

(Place an X on the scale below to assign a crisp score or mark an interval score or circle a word.)



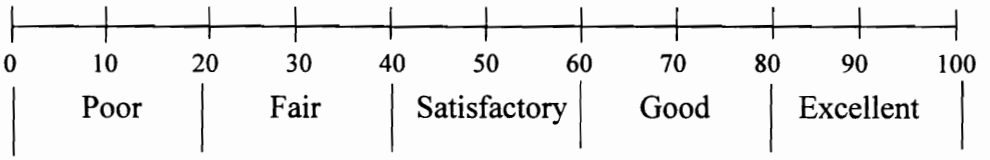
18. How good is the example system design quality indicator hierarchy in Appendix A?

(Place an X on the scale below to assign a crisp score or mark an interval score or circle a word.)



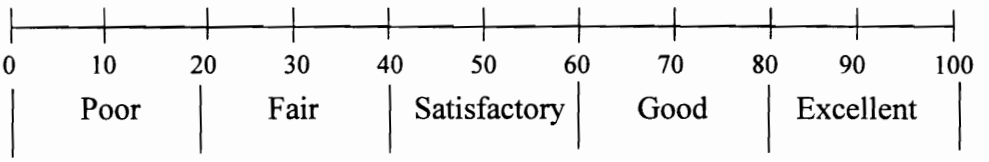
19. How well does the questionnaire in Appendix B assess the evaluative credibility of indicator hierarchy?

(Place an X on the scale below to assign a crisp score or mark an interval score or circle a word.)



20. How well is the methodology documented? (The document you received)

(Place an X on the scale below to assign a crisp score or mark an interval score or circle a word.)



PART IV. INFORMATION ABOUT THE EVALUATOR

21. Please indicate your educational background:

Degree (B.S., M.S., Ph.D.)	Degree Area	Year Obtained

22. Please attach your biography that details your professional experience. Do not identify yourself! <u>All of the information you provide must be unclassified.</u>

VITA

Michael Lane Talbert

Education

Ph.D.: Computer Science	Virginia Tech	1995
M.S.: Computer Information Systems	Air Force Institute of Technology	1988
B.S.: Meteorology	North Carolina State University	1985

Personal Data

Born: 23 Nov 62, Rock Hill, South Carolina

Married: Kristen Converse True of Portsmouth, Virginia

Children: Anna Catherine and Thomas Michael

Address: 3801 Chandonwood Ct., Charlotte, NC 28226

Phone: (704) 541-3118

Immediate Career Plans

Captain on active duty service in the U.S. Air Force regular forces. As of January 1996, he is an assistant professor of computer science on the faculty of the Department of Computer and Electrical Engineering at the Air Force Institute of Technology.

Society Memberships

Member of Tau Beta Pi, Upsilon Pi Epsilon, Phi Kappa Phi, Phi Eta Sigma, Alpha Lambda Delta, and the Association of Computing Machinery.