SOFTWARE METRICS:
SOFTWARE QUALITY METRICS FOR DISTRIBUTED SYSTEMS

by

Jonathan V. Post
Boeing Aerospace Company

ABSTRACT

Recent publication of numerous books and papers indicates the growing importance of Software Quality Metrics [1]. Studies at the Boeing Aerospace Company [2,3] have extended this field to cover Distributed Computer Systems. Emphasis is placed on studying Embedded computer systems, and on viewing them within a system life cycle [4]. The approach of J.A.McCall, et.al. [5,6], at General Electric was pursued and extended, maintaining the hierarchy of quality factors, criteria, and metrics [fig.1]. New software quality factors have been added, including Survivability, Expandability, and Evolvability [fig.2].

KEYWORDS

Software, Quality, Metrics, Distributed, Survivability, Life Cycle, Expandability, Evolvability, Virtuality

INTRODUCTION

What is a distributed computer system? Enslow [7] requires such a system to meet five criteria, while LeLann [8] requires it to be a collection of entities participating in system performance. Mauchley and Eckert built the first distributed computer, BINAC, circa 1947 , and acknowledged [9] that the structure of the human brain, with its two cerebral hemispheres, was a guiding design metaphor. Dr.Roger Sperry's Nobel Prize in Medicine was for experiments performed at Caltech which established that the human brain is a distributed computer [10]. We consider a distributed system to be formed by the interconnection of potentially autonomous systems to accomplish system functions cooperative-

ly.

There are several ways the term "distributed" may be interpreted. Data may be distributed, processors may be distributed, processes may be distributed, users may be distributed, communications may link geographically dispersed clusters of components, or some combination of these strategies may be imposed on system architecture. Each of these types of distributedness leads to design tradeoffs, and to qualitative distinctions between centralized and distributed systems. No single model allows analysis of all such tradeoffs; data is either specialized, anecdotal, or condensed to "lessons learned" or scenario form. The application of Software Quality Metrics should help to provide a unifying framework for all such distributed systems. As Norber Weiner first emphasized [11], it is possible to build a reliable system out of unreliable parts.

It will be increasingly important to understand distributed computer systems. Some of their characteristics will emerge more extensively in future configurations. One characteristic peculiar to distributed systems, and of importance in the 80's, is Geographic Dispersion. The extent to which computers within a distributed system can be physically displaced from each other, range from the centimeter to the multi-thousand-kilometer. Computers will indeed be "tightly-coupled" over intercontinental distances by fiber-optics technology currently under research. This technology complements that of the communications satellite. Interconnection of even a very small percentage of available computers will be able to form distributed systems of complexity beyond those of today, since by 1999 there will be on the order of one billion computers in the world [13].

QUALITY METRICS APPROACH

The approach chosen to evaluate distributed systems is the Software Quality Metrics methodology, which has been fruitfully applied to the study of a broad range of uniprocessor computers and embedded computer systems [1]. Since the 1970's, additional factors have been judged necessary in evaluating the performance of software and systems besides that of classic Reliability which was a factor closely identified with software and system quality. McCall and others [5,6] identified eleven software quality factors and developed a system of metrics to predict and assess the degree of presence of these factors. As shown in fig.1, each factor is composed of a number of criteria which are further broken down into quantitative metrics. The eleven Factors identified : Correctness, Reliability, Efficiency, Integrity, Usability, Maintainability, Testability, Flexability, Portability, Reusability, and Interoperability. The extension of this approach to distributed systems was introduced at last year's workshop by Robert W. Lawler of Boeing Aerospace Company [15]. The research conducted during the past year, as reported to RADC[2], has concentrated on identifying unique characteristics of distributed systems, and on

definition or redefinition of factors and criteria which can measure these characteristics. Three new software factors, four new system factors, twelve new software criteria , and two new system criteria have been described, and the factor of Testability has been generalized into the factor of Verifiability. Examples of these new factors and criteria are described below and in fig.2.

DISTRIBUTED SYSTEM CHARACTERISTICS

How do we approach the identification of the characteristics of distributed systems? Distributed System characteristics are identified and classified, along with rationales for the selection of Distributed Systems. 58 rationales are grouped into 9 reasons in fig.3 . The rationales given for selection of a distributed system over a uniprocessor system indicate the characteristics which people imagine distributed systems, as a whole, exhibit. No one system meets more than a fraction of these identifications, just as no system life cycle for a distributed system quite fits into the system life cycle models for uniprocessor systems. Instead, we find the distributed system to be distributed through time in a distributed life cycle of concurrent phases of Operation, Revision, and Transition [fig.4].

NEW QUALITY FACTORS

The main difference between software metrics for a distributed system and software metrics for a uniprocessor system is that the quality of software in a distributed environment depends upon the design and performance characteristics of the entire system. We therefore distinguish between Software Quality Factors and System Quality Factors, although these have impact upon each other. The quality factor of Survivability, for example, reflects system performance when one or more nodes or communication links become totally nonoperational. The concepts of Reliability and Redundancy in a uniprocessor are not broad enough to describe Survivability.

Survivability is a factor which measures the capability of a system to operate when one or more components are destroyed. For a non- distributed system, Survivability is not a very meaningful measure. A single unit computer, depending on the degree of hardening and the damage received in the tactical environment, will usually either continue to operate, or else be completely incapacitated. For a geographically dispersed system, it is desirable that damage or destruction of individual components shall allow the system to continue functioning, albeit at a somewhat lower level of performance. Survivability, then, might measure the likelihood of a distributed system to exhibit this "graceful degradation". The 5 criteria within the system quality factor of Survivability are Autonomy, Distributedness, Anomaly Management, Modularity, and Reconfigurability. (See fig. 2)

Distributed Systems also require metrics to evaluate the capaci-

ty of expanding and upgrading the system, so we have identified and defined the corresponding factors of Expandability and Evolveability. Expandability is the extent to which the system capability can be expanded to enhance current functions or to add new functions. The criteria within the factor of Expandability include: Virtuality, Generality, Modularity, Augmentability, Clarity, Specificity, and Simplicity. Evolvability is the extent to which the system performance could be enhanced by the incorporation of new technology. Criteria within Evolvability are Virtuality, Generality, Modularity, Clarity, Specificity, and Simplicity. In addition, we have defined four new system quality factors, Availability, Safety, Transportability, and Interchangeability.

NEW CRITERIA

Twelve new software criteria were identified during investigation of characteristics for distributed systems [2]. These criteria are: Compliance, Validity, Clarity, Specificity, Virtuality, Comprehensibility, Reconfigurability, Distributedness, Autonomy, Supportability, Augmentability, and Compatibility [fig.5]. In addition, two new system criteria were identified : Self-containedness (an attribute of Transportability) and Homogeneity (an attribute of Interchangeability). A majority of these system and software criteria are applicable to uniprocessors as well. The following brief discussion on one of the new software criteria, Virtuality, shows how the entire system, including the human users, needs to be measured to evaluate the system quality.

For Distributed Systems, there is a new criterion within the quality factor for Usability. We refer to this criterion as Virtuality. The structure of a distributed system can be quite complex, and it is not always desirable for the user to be appraised of this structure. The user may perceive the system in terms of a virtual architecture, and be shielded from knowing the actual internal representation and location of data.

Virtuality is a measure of the extent to which the system appears to the user as it is intended to appear to the user. The user (or users) of a system is not expected or intended to see the system's logical, topological, or physical structure. Instead, an abstract "virtual" system is designed. The "real" system supports, emulates, and embodies the designed appearance and "feel" of the virtual system.

Theodor H. Nelson [12] explains the relationship between Virtuality and other criteria such as Conceptual Simplicity, Machine Independance, and Network File Availability. "Our approach to computer design we call 'the design of virtuality.' By virtuality we mean the seeming of an object or system, its conceptual structure, its atmospherics and its feel.... What counts is effects, not techniques.... The design of an interactive computer environment, similarly, should not be based on particular

hardware, or a particular display device, or a programming technique.... the systems analysis for an interactive system should deal with the mental space of the user's experience."

Virtuality also measures the subjective component of the user interface. In the special case of flight training simulators [14], the "feel" of the system has long been regarded as crucial to Usability. "Feel" is evaluated by expert pilots (superusers). This goes beyond Human Engineering, which concentrates on one display/sensory modality at a time, or on total bits per second. "Feel", and therefore Virtuality, involves gestalt perception, with an emphasis on right-brain holistic activity. Virtuality, and the human brain, cannot be ignored when studying distributed systems.

NEW METRICS

During the next year of this research effort there will be a set of metrics developed within the criteria and factors discussed above. The existing metrics [6] will be added to, deleted, and modified in accordance with results to date. The work yet to be performed may be summarized as follows:

(1) Select Quality Metrics for Validation (Identify those metrics that will make the greatest contribution to validating the quality measurement framework previously developed);

(2)Develop Scenarios and Collect Data (Design the data collection methodologies and gather relevant data from Boeing Aerospace Company projects which use distributed embedded computer systems);

(3) Validate Metrics (Validation techniques consistent in concept and methodology with McCall, et.al. [6], but with multivariate regression analysis and other numerical analysis and correlation methods; conduct interviews with engineering and management personnel to supplement empirical data);

(4) Produce a Report and Handbook (Final Report to be published by RADC. A Handbook will be prepared that describes the step-by-step procedures required to implement the quality measurements for distributed systems).

SUMMARY

Software Quality Metrics may be applied to the evaluation of distributed computer systems. Exactly what constitutes a distributed system is disputed in the literature. They have been built in various configurations for thirty years, but the human brain shares some of the characteristics of these systems and provides a valuable model. The approach of McCall et.al., with factors, criteria, and metrics, has been extended. New factors and new criteria have been defined. New metrics will be devised and validated as the research described in this paper is continued.

BIBLIOGRAPHY

[1] Perlis, A.; Sayward, F.; Shaw, M.; "Software Metrics", MIT Press, 1981, includes 130 page annotated bibliography on Software Metrics, compiled by Mary Shaw

[2] Post, Jonathan, and Bowen, Thomas P. "Interim Report for Quality Metrics for Distributed Systems", Boeing Document D180-26748-1, November 1981, prepared for RADC under contract F30602-80-C-0330

[3] Henrick, John, "Performance Modeling of Distributed Computing Systems: A Literature Search", Boeing Document D182-1,0827-1 1 December 1981

[4] Post, Jonathan; "Software Systems Engineering", Boeing Document D180-25488-5, January 1980, prepared for ASD USAF under contract number F33657-76-C-0723

[5] McCall, J., Richards, P., Walters, G.; "Metrics for Software Quality Evaluation and Prediction", Proceedings of the NASA/ Goddard Second Summer Engineering Workshop, September 1977

[6] McCall, J.A., Matsumoto, M.T., "Software Quality Metrics Enhancements Final Report", prepared for RADC under contract number F-30602-78-C-0216

[7] Enslow, Philip. H., "What is a 'Distributed' Data Processing System?", Computer, January 1978, p.13-21

[8] LeLann, Gerard, "Distributed Systems -- Towards a Formal Approach," 1977 IFIP Congress Proceedings, p.155-160

[9] Mauchley, John; Personal communication, Philadelphia, PA, June 1978

[10] Sperry, Dr.Roger; Personal communication, Caltech, 1973

[11] Weiner, Norbert, "Cybernetics", MIT Press, 1947

[12] Nelson, Theodor H., "Replacing the Printed Word", Information Processing 80, Proceedings of the IFIPS Congress 1980, North-Holland Publishing Co.

[13] Post, Jonathan V., "Quintillabit: Parameters of a Hyperlarge Database", Proceedings of the 6th International Conference on Very Large Databases, Montreal, 1-3 October 1980

[14] Post, Jonathan V., "Software Development and Maintenance Facilities Guidebook", Boeing Document D180-25488-3, Sep. 1979, Prepared for USAF ASD, Contract F33657-76-C-0723

[15] Lawler,R.L., "Software Quality Tradeoff Measurement", Proceedings from the Fifth Annual Software Engineering Workshop, 24 Nov 1980, Goddard Space Flight Center, NASA, Greenbelt, Maryland

THE VIEWGRAPH MATERIALS
for the
J. POST PRESENTATION FOLLOW

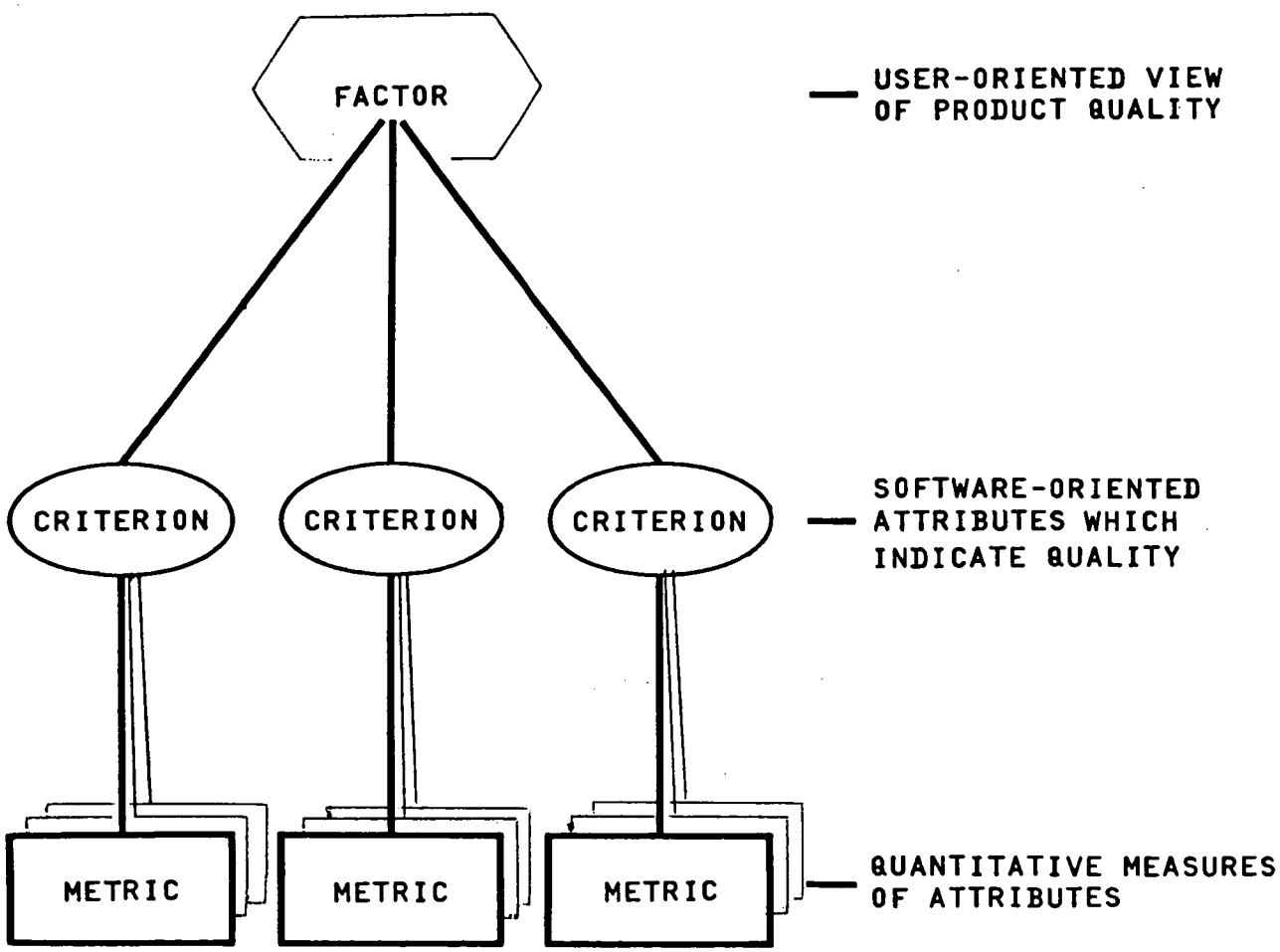
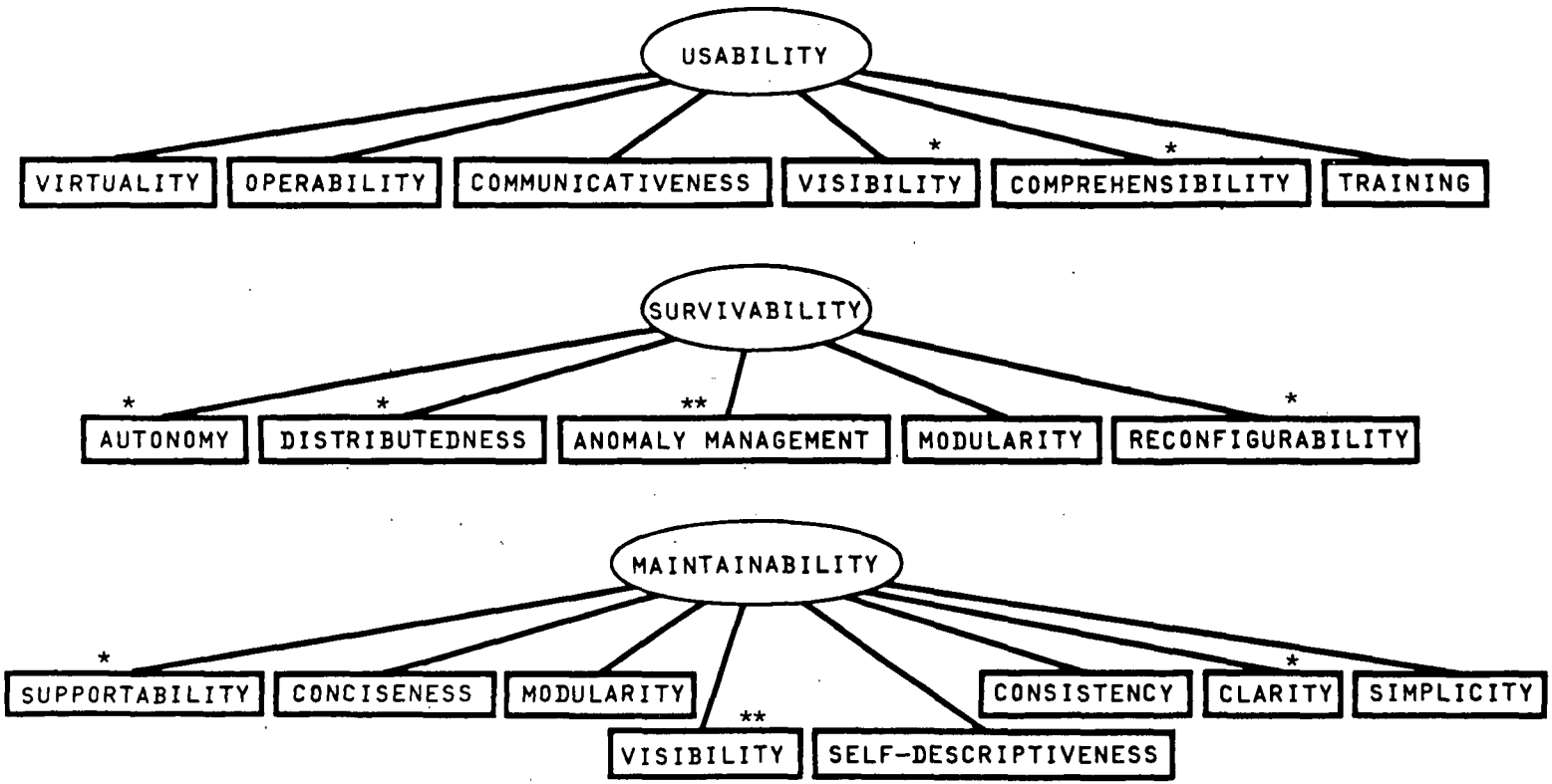


Figure 1 Software Quality Model

Figure 2 Relationship of Criteria to Software Quality Factors

* = New
** = Different



* = New
** = Different

Figure 2 Relationship of Criteria to Software Quality Factors

REASON NO.	REASONS FOR SELECTION OF DISTRIBUTED SYSTEMS	CORRECTNESS	MAINTAINABILITY	RELIABILITY	FLEXIBILITY	TESTABILITY	PORTABILITY	REUSEABILITY	EFFICIENCY	USEABILITY	INTEGRITY	INTEROPERABILITY	SURVIVABILITY
1	IMPROVE RESPONSE TIME												
	•CONCURRENCY OF DIAGNOSIS WITH NORMAL OPERATION	X	X	X		X							X
	•ENHANCED DATA PARALLELISM				X				X	X			
	•MINIMIZE MEMORY/PROCESSOR COMMUNICATION TIME								X				
	•ALLOW OPTIMAL PARTITIONING OF WORKLOAD				X			X		X		X	
	•LOAD LEVELING				X				X	X			
	•REAL-TIME COORDINATION OF MULTIPLE SUBSYSTEMS												
2	PROVIDE GREATER PROCESSING AND ACCESSING CAPABILITIES												
	•AUTOMATIC JOB SEGMENTING				X			X	X				
	•PARTITIONING OF FUNCTIONALITY	X	X		X		X		X			X	X
	•INCREASED VARIETY OF PROCESSING MODES				X					X		X	
	•RESOURCE UNIFORMITY	X	X			X	X	X					
	•SPECIALIZED HARDWARE: DATABASE MACHINE					X			X				
	•INTEROPERABILITY WITH EXISTING SYSTEMS						X	X				X	
3	REDUCE COST												
	•LOWER COST TO UPGRADE (EXPANDABILITY)			X	X			X					
	•LOCAL ADMINISTRATIVE APPROVAL OF COMPONENTS									X		X	
	•NEW TOPOLOGICAL CONFIGURATIONS ON DEMAND		X	X	X					X			X
	•LOWER INITIAL COST								X	X			
	•INCREASED PROCURABILITY									X			
	•INCREASED DEPLOYABILITY		X		X								X
	•LOWER TOTAL WEIGHT		X		X				X				
	•LOWER TOTAL POWER CONSUMPTION		X		X				X				X
	•NETOWRK TOPOLOGY OPTIMIZATION			X	X				X				X
	•RESOURCE SHARING			X	X			X		X		X	
4	REDUCE VULNERABILITY TO HARDWARE ERROR												
	•REDUNDANCY AT EACH NODE		X	X									X
	•TOLERANCE TO NODE FAILURE	X	X	X		X				X	X		X
	•TOLERANCE TO COMMUNICATIONS LINK FAILURE		X	X	X	X					X		X
	•CAPABILITY FOR ISOLATING FAILED COMPONENTS		X	X		X					X		X
	•DIAGNOSIS OF FAILURE TO LEAST REPLACEABLE UNIT		X	X		X							
	•REPAIR WITHOUT INTERRUPTION			X	X				X	X	X		X
5	REPLACE HARDWIRED LOGIC WITH MICROPROCESSOR												
	•RESOURCE UNIFORMITY	X	X			X	X	X					
	•RECONFIGURABILITY		X	X	X							X	X
	•MACHINE INDEPENDENCE	X					X	X				X	
	•DELAYED COMMITMENT TO SPECIFIC NODE HARDWARE		X		X							X	
	•MULTIPLICITY OF VENDORS		X	X	X							X	
	•RECONFIGURABILITY THROUGH LOW-COST HARDWARE				X				X	X			

Figure 3 Relationship Between Reasons, Rationales, and System Quality Factors (page 1 of 2)

REASON NO.	REASONS FOR SELECTION OF DISTRIBUTED SYSTEMS	CORRECTNESS	MAINTAINABILITY	RELIABILITY	FLEXIBILITY	TESTABILITY	PORTABILITY	REUSEABILITY	EFFICIENCY	USEABILITY	INTEGRITY	INTEROPERABILITY	SURVIVABILITY
6	IMPROVE THRUPUT												
	• DISTRIBUTE JOBS TO SEVERAL NODES CONCURRENTLY				X				X				X
	• EXPLOITATION OF UNIFORM INTERCHANGE MEDIA			X			X	X				X	
	• ENHANCED DATA PARALLELISM								X	X			
	• ENHANCED COMPUTATIONAL PARALLELISM				X				X				
	• OPTIMAL PARTITIONING OF WORKLOAD			X					X	X			
	• REDUCE LOAD ON HOST			X	X				X				X
	• DISTRIBUTED OPERATING SYSTEM				X								X
	• ELIMINATE MULTIPROGRAMMING		X			X			X	X			
7	IMPROVE SURVIVABILITY												
	◦ SECURITY ON HIERARCHICAL NETWORK									X	X		X
	◦ SYSTEM PROTECTION FROM OVERLOAD		X	X		X					X		X
	◦ BACKUP REDUNDANCY	X		X	X								X
	• RESTORATION/RECOVERY	X	X	X		X			X		X		X
	• ENDURANCE/HARDENING		X	X							X		X
8	IMPROVE SENSOR PERFORMANCE												
	• DISTRIBUTED SENSORS	X	X	X	X					X			X
	• DISTRIBUTED EFFECTORS			X	X					X			X
	• INTELLIGENT SENSOR CLUSTERS		X		X								
	• DEPLOYABLE SENSOR ARRAYS			X	X	X							X
	• CONCURRENT MULTI-SPECTRAL SCANNING			X	X				X				
9	IMPROVE GEOGRAPHIC DISPERSION												
	• USER DISTRIBUTION				X			X	X	X		X	
	• GATEWAY TO NATIONAL/INTERNATIONAL NETWORK						X			X	X	X	
	• GLOBAL C3I APPLICATIONS	X		X						X	X	X	X
	• SPACE SYSTEMS NETWORKS		X	X	X						X	X	X
	• NEED FOR MOBILE NODES		X	X	X					X			
	• NEED FOR DISTRIBUTED DATABASE MANAGEMENT	X						X	X	X	X		
	• ADAPTIVE ROUTING						X			X	X		X

Figure 3 Relationship Between Reasons, Rationales, and System Quality Factors (page 2 of 2)

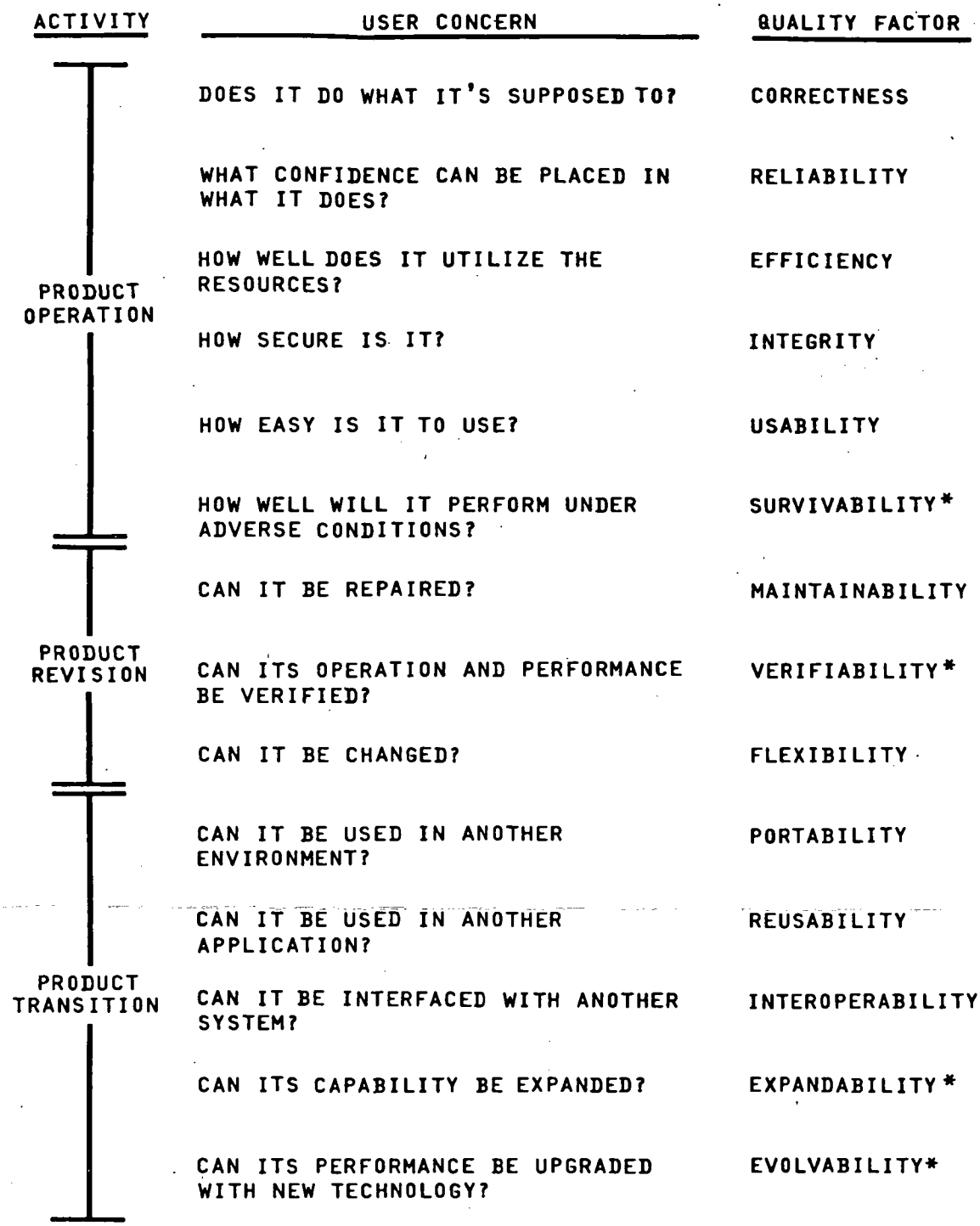


Figure 4 Quality Life Cycle Scheme

CRITERION	DEFINITION
• TRAINING	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE TRANSITION FROM CURRENT OPERATION OR PROVIDE INITIAL FAMILIARIZATION.
• COMMUNICATIVENESS	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE USEFUL INPUTS AND OUTPUTS WHICH CAN BE ASSIMILATED.
• OPERABILITY	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH DETERMINE OPERATIONS AND PROCEDURES CONCERNED WITH THE OPERATION OF THE SOFTWARE.
• MODULARITY	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE A STRUCTURE OF HIGHLY COHESIVE MODULES WITH OPTIMUM COUPLING.
• RECONFIGURABILITY*	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE FOR CONTINUITY OF SYSTEM OPERATION WHEN ONE OR MORE PROCESSORS, STORAGE UNITS, OR COMMUNICATIONS LINKS FAIL.
• DISTRIBUTEDNESS*	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH DETERMINE THE DEGREE TO WHICH SOFTWARE FUNCTIONS ARE GEOGRAPHICALLY OR LOGICALLY SEPARATED WITHIN THE SYSTEM.
• AUTONOMY*	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH DETERMINE ITS DEPENDENCY ON INTERFACES.
• CONCISENESS	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE FOR IMPLEMENTATION OF A FUNCTION WITH A MINIMUM AMOUNT OF CODE.
• SUPPORTABILITY*	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE FOR EASE IN CREATION OF NEW SOFTWARE VERSIONS (e.g., USE OF HOL, VERSION UPDATE SCHEME).
• SELF-DESCRIPTIVENESS	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE EXPLANATION OF THE IMPLEMENTATION OF A FUNCTION.
• GENERALITY	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE BREADTH TO THE FUNCTIONS PERFORMED.
• INDEPENDENCE**	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH DETERMINE ITS DEPENDENCY ON THE SOFTWARE ENVIRONMENT (COMPUTING SYSTEM, OPERATING SYSTEM, UTILITIES, INPUT/OUTPUT ROUTINES, LIBRARIES).
• AUGMENTABILITY*	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE EXPANSION CAPABILITY FOR FUNCTIONS AND DATA.
• COMPATIBILITY*	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE INTERFACE PROTOCOLS AND ROUTINES THAT ARE APPROPRIATE TO THE INTERFACE EQUIPMENT FEATURES AND CAPABILITIES.
• COMMONALITY**	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE FOR THE USE OF INTERFACE STANDARDS FOR PROTOCOLS, ROUTINES, AND DATA REPRESENTATIONS.

Figure 5 Software Quality Criteria Definitions

* = New
** = Different

CRITERION	DEFINITION
• TRACEABILITY	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE A THREAD OF ORIGIN FROM THE IMPLEMENTATION TO THE REQUIREMENTS WITH RESPECT TO THE SPECIFIED DEVELOPMENT ENVELOPE AND OPERATIONAL ENVIRONMENT.
• CONSISTENCY	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE FOR UNIFORM DESIGN AND IMPLEMENTATION TECHNIQUES AND NOTATION.
• COMPLETENESS	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE FULL IMPLEMENTATION OF THE FUNCTIONS REQUIRED.
• COMPLIANCE*	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROMOTE IMPLEMENTATIONS THAT CONFORM TO THE REQUIREMENTS.
• VALIDITY*	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH CONSTRAIN IMPLEMENTATIONS TO A RANGE OF ACCEPTABLE SOLUTIONS.
• CLARITY*	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE NON-AMBIGUOUS DESCRIPTIONS OF FUNCTIONS AND IMPLEMENTATIONS.
• SPECIFICITY*	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE SINGULARITY IN THE DEFINITION AND IMPLEMENTATION OF FUNCTIONS.
• SIMPLICITY	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE FOR THE DEFINITION AND IMPLEMENTATION OF FUNCTIONS IN THE MOST NON-COMPLEX AND UNDERSTANDABLE MANNER.
• ANOMALY MANAGEMENT**	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE FOR CONTINUITY OF OPERATIONS UNDER AND RECOVERY FROM NON-NOMINAL CONDITIONS.
• ACCURACY	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE THE REQUIRED PRECISION IN CALCULATIONS AND OUTPUTS.
• EFFECTIVENESS**	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE FOR MINIMUM UTILIZATION OF RESOURCES (PROCESSING TIME, STORAGE, OPERATOR TIME) IN PERFORMING FUNCTIONS.
• ACCESSIBILITY**	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE FOR CONTROL AND AUDIT OF ACCESS TO THE SOFTWARE AND DATA.
• VIRTUALITY*	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PRESENT A SYSTEM THAT DOES NOT REQUIRE USER KNOWLEDGE OF THE PHYSICAL CHARACTERISTICS (e.g., NUMBER OF PROCESSORS/DISKS, STORAGE LOCATIONS)
• VISIBILITY**	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE STATUS MONITORING OF THE DEVELOPMENT AND OPERATION (e.g., INSTRUMENTATION).
• COMPREHENSIBILITY*	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH ENHANCE UNDERSTANDING OF THE OPERATION OF THE SOFTWARE.

Figure 5 Software Quality Criteria Definitions

* = New
** = Different