A Taxonomy for the Evaluation of Computer Documentation

K. Todd Stevens, James D. Arthur and Richard E. Nance

TR 88-38

Technical Report SRC-88-008

A TAXONOMY FOR THE EVALUATION OF COMPUTER DOCUMENTATION

K. Todd Stevens
Department of Computer Science

James D. Arthur
Department of Computer Science

Richard E. Nance
Systems Research Center
and
Department of Computer Science

Virginia Polytechnic Institute
and State University
Blacksburg, VA 24061

15 June 1988

This research was supported in part by the Naval Surface Warfare Center under Contract No. N60921-83-G-A165-B038 through the Systems Research Center at Virginia Tech.

Abstract

The evaluation of software documentation is a key issue in the more general framework of evaluating products of a software development process. The research described in this report focuses on (1) a general taxonomy of document characteristics that support an assessment of documentation adequacy, and (2) the refining of the general evaluation taxonomy to a specific application, namely, the Automated Design Description System (ADDS).

Key Phrases: K.6.3. [Software Maintenance]: Software Management; [Document Assessment]

Table of Contents

List of Illustrations

1.0 Introduction

The recognition of the need to view software costs in the context of *a life cycle* helped to focus attention on the root problems with early software/hardware systems. Aided by the visibility attributed to *software maintenance*, computer scientists were able to explain the need for a disciplined development process, controlled through meticulous attention to configuration management and governed by an emerging set of fundamental principles. Among these principles is *Concurrent Documentation*, the recording of requirements, design, specification and implementation decisions *as they occur* with the commitment to convey purpose, content and clarity [TAUR77, pp. 32-33].

The failure to observe the concurrent documentation principle unfortunately is common in all application domains. However, in no domain is a higher price exacted for such failure than in that of *time-critical embedded systems*. The deficiencies in the development process for such systems visit huge costs on the maintenance process, and the familiar claim is that more than one-half the total life cycle costs are incurred during the maintenance phase [BOEB76, LIEB78]. Contributing to this substantial burden are several factors:

- a severe shortage of skilled software talent, especially lacking the experience so vital for maintenance tasks,
- the void in complementary methods and techniques for guidance and direction of the in-service support responsibilities, and
- the scarcity of tools for supporting maintenance activities intrinsic to complex embedded systems software.

1.1 Background

Seeking to overcome these burdensome factors, a group of software engineers at the Naval Surface Warfare Center (NSWC) in Dahlgren, Virginia has developed the Automated Design Description System (ADDS) to support the in-service engineering responsibility for the AEGIS Combat System software. This group, in response to the need for improved documentation of software deliverables, has adopted an approach based on *reverse engineering*.

Reverse engineering in the maintenance phase is achieved through the synthesis of documents based on the analysis of existing databases populated by the product (and possibly the process of) software development. With respect to ADDS, and the complementary interactive tool, Micro-ADDS, the database consists of the AEGIS combat system source code and the Program Performance Specifications (PPSs). In essence, ADDS generates specialized documents based on the structure and relational characteristics of the *extant* AEGIS software.

This report is the second in a three-phase examination of ADDS and its capabilities for supporting the maintenance requirements of the AEGIS Combat System. The division of the phases into tasks has taken the following form:

Task 1. Review current literature to determine current documentation standards as they relate to the user of a document, the document's scope and purpose. This effort includes (a) the identification of specific categories of documentation amenable to automated analysis, and (b) an investigation and formulation of quality indicators (DQIs) that can support the qualitative assessment of such document categories.

Task 2. Examine ADDS with respect to identifying document quality indicators. This task includes (a) an investigation of the enhanced ADDS analytical capabilities relative to DQIs, and (b) its potential for exploiting DQIs based on currently available data.

Task 3. Based on the DQI approach, the current capabilities of ADDS and its underlying database, explore the necessary requirements for identifying (and synthesizing) DQIs missing from the current input to and output from the ADDS system.

1.2 Synopsis

This report examines ADDS from both designer and user perspectives. Among the several conclusions addressed in greater detail in subsequent sections, the following are viewed as key:

- The formulation of a general theory of documentation analysis based on the Document Quality Indicator (DQI) concept
- The development of an Evaluation Taxonomy that relates document qualities to factors and document factors to quantifiers. The quality/factor/quantifier triple uniquely identifies each DQI and serves as a well-defined basis for document evaluation
- Based on characteristics of the in-service engineering activity, the tailoring of the general evaluation taxonomy to the ADDS domain so as to support the analysis of documents synthesized from ADDS
- The identification of DQIs not measurable in ADDS generated documents and suggestions as to how one might correct the deficiency.

2.0 The General Evaluation Taxonomy

A general taxonomy for the evaluation of computer documentation can be best described by a tree model. The general structure of the tree is described in this chapter, including discussions of the logic used in refinement of the design. The different levels are also described and defined.

The tree structure is described in detail in a top-down fashion. The nodes of the first level of the tree are the *Qualities*; the second level includes the *Factors*; and the third, or lowest, level includes the *Quantifiers*. Figure 1 on page 6 shows the entire taxonomy which is developed in the following sections.

2.1 The General Tree Structure for the Taxonomy

Intuitively, a Document Quality Indicator (DQI) is a variable where value can be determined through direct analysis of document characteristic and where evidential relationship to one or more *Quality/Factor* pairs is undeniable. DQIs provide the basis for assessing the quality of documentation. DQIs could be illustrated using a variety of models. One such model might be a simple list

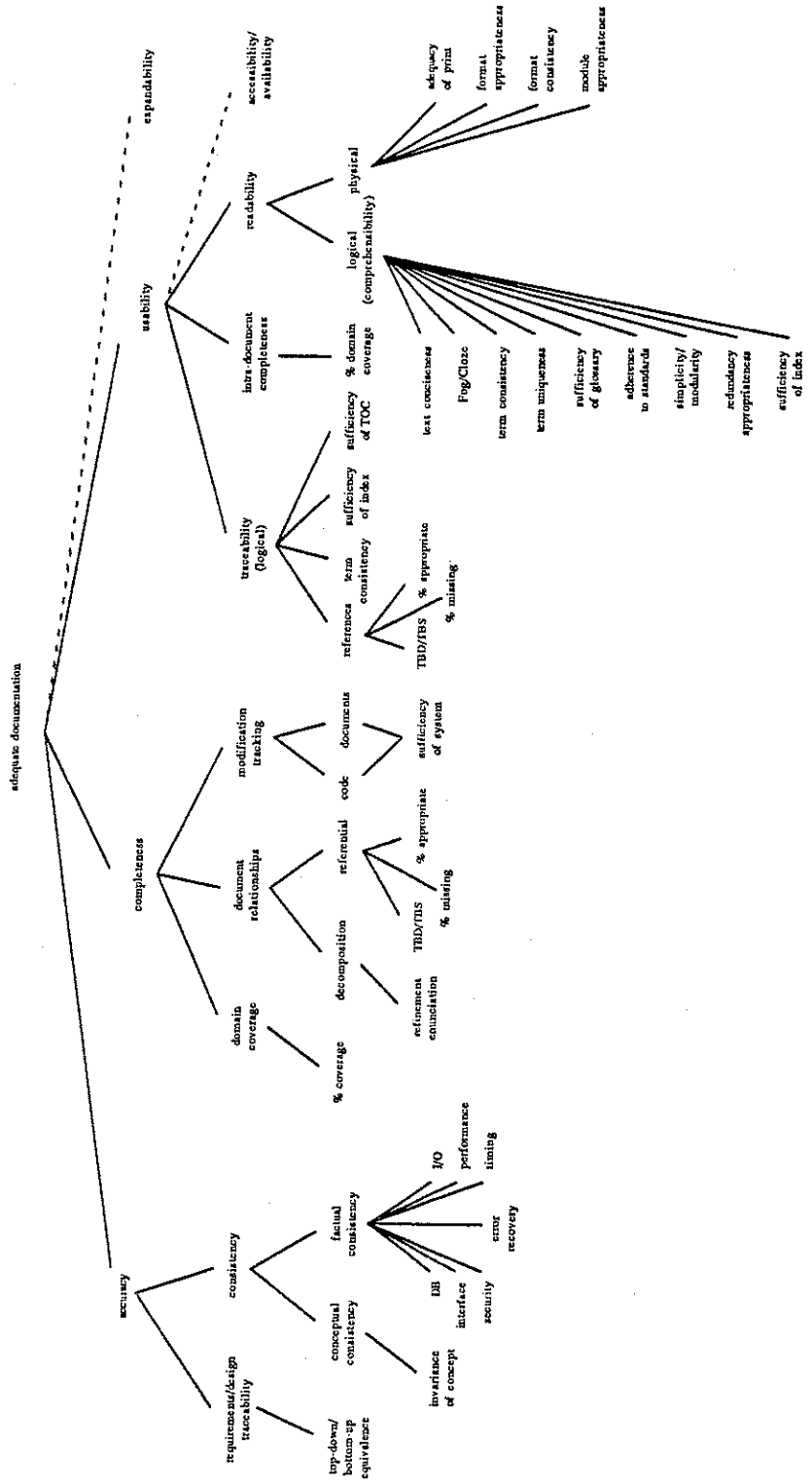


Figure 1. General Evaluation Taxonomy

of the measurable characteristics, to which weighted factors would be assigned. A top-down, tree-structured hierarchy, however, was chosen as being the most descriptive of the desired relationships among the contributors to document quality.

In response to the basic question, "What is looked for in 'good' documentation?", the main, high-level *Qualities* of good documentation can be defined as: Accuracy, Completeness, Usability and Expandability. While these *Qualities* are not actually measurable, they can be further broken down into characteristics which support their assessment. By applying a Divide & Conquer approach, each *Quality* characteristic is successively divided into "smaller" and less abstract characteristics. While, in the typical Divide & Conquer approach, the problem is actually divided into smaller and smaller segments, in the evaluation taxonomy each characteristic is not actually divided; it is partitioned into more precise, supportive elements. "Supportive," in this context, means that the smaller child characteristics help to measure part of the larger parent characteristic. For example, *Accuracy* supports Adequate Documentation. Adequate Documentation must be accurate, but *Accuracy* is only one characteristic that Adequate Documentation must exhibit; it must also exhibit *Completeness*, among others. The Divide & Conquer approach is applied recursively until the child characteristics are tangible and measurable.

The top-down hierarchical model promotes methodical, simple stepwise refinement. This simplicity makes the taxonomy easy to apply, understand, and extend.

2.1.1 Refinement Criteria

The preceding description of the tree structure states that general characteristics are divided into or supported by more tangible characteristics, where each branch of the tree structure reduces the level of abstraction. A branch from a given node-characteristic of the tree reduces the level of abstraction by either creating a sub-class or a supportive characteristic.

2.1.1.1 *Sub-class Characteristics*

Sub-class characteristics are a type of child-node, being specific instantiations of a more general parent-node characteristic. As such, they are more tangible or measurable. For example, a supporting characteristic of Completeness is Document Relationships. There are two types or classes of this characteristic: Decompositional and Referential (See Figure 2 on page 9). Although, in fact, the characteristic Document Relationships could be removed from the hierarchy and the two relationships, Decompositional Relationship and Referential Relationship, replace it, this would also remove the more general idea that document relationships are actually the more abstract characteristic that supports the characteristic above it. Therefore, the single-step refinement needs to be enforced in order to maintain the hierarchy of abstraction.

2.1.1.2 *Supportive Characteristics*

Supportive characteristics, being one step less abstract, allow a more general parent characteristic to be assessed. For example, Logical Traceability supports the Usability of the documentation (See Figure 3 on page 10). No sub-class relationship is identified in this case. However, the usefulness of a document clearly depends on its support of traceability: a user must be able to trace and follow the logic of the documentation in order to use it. Thus, Logical Traceability helps the assessment of Usability. Also, it can be seen that, unlike the example for sub-class characteristics (Figure 2 on page 9), the parent-node cannot be removed and replaced by the child-node characteristics.

A second example on a more tangible level is Term Consistency, which supports Logical Traceability (See Figure 4 on page 11). The use of the same term, when referring to the same concept, makes the documentation easier to understand. Obviously the reciprocal is also true: the use of different terms to represent the same concept makes it more difficult to understand. This clearly shows that the consistent use of terms supports Logical Traceability.

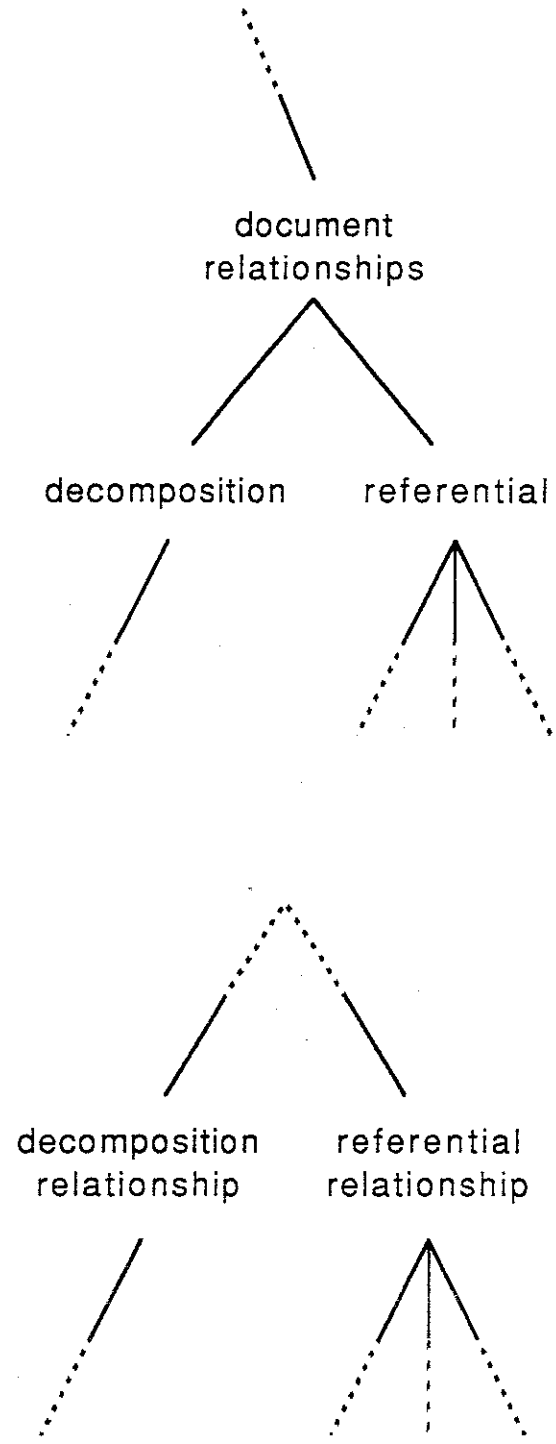


Figure 2. Example of Sub-class Characteristics

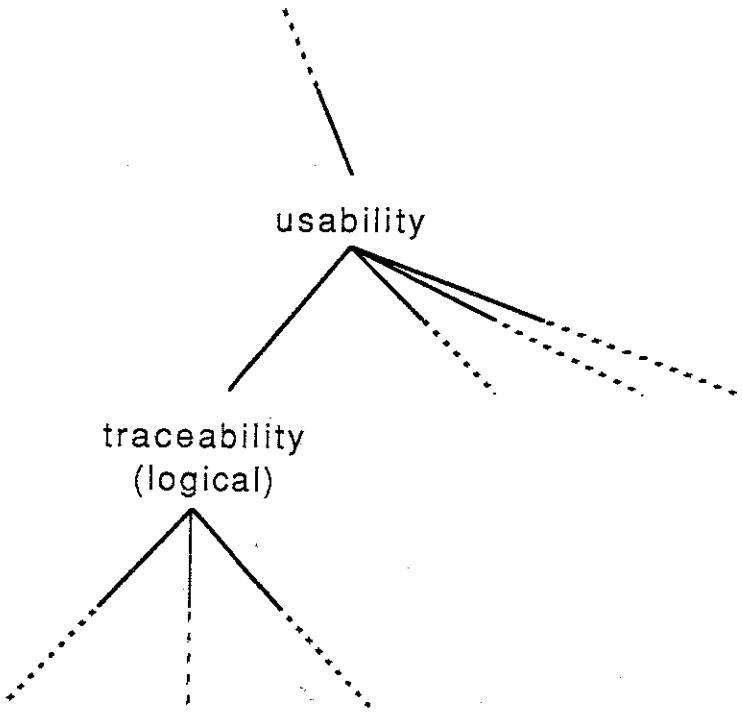


Figure 3. Example I of Supportive Characteristics

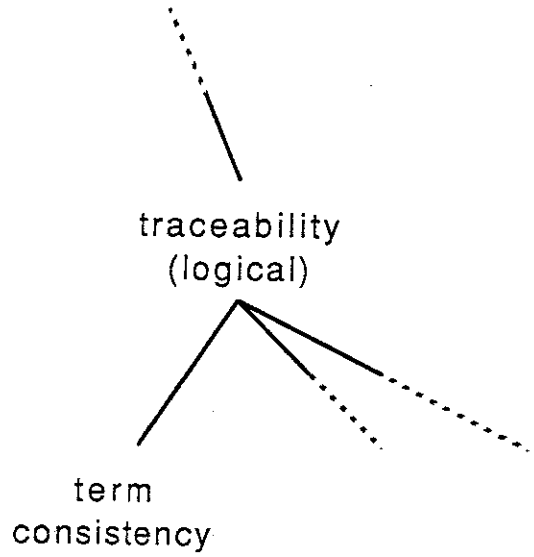


Figure 4. Example II of Supportive Characteristics

2.1.2 Levels of Characteristics

The tree produced by the top-down development approach is only a tree structure of characteristics. The structure can also be viewed as having three distinct, horizontal levels of logical abstraction: the *Quality* level, the *Factor* level, and the *Quantifier* level. The root node Adequate Documentation, that is, adequacy of the documentation, can only be assessed by its *Qualities*, each of the *Qualities* must be assessed by their *Factors* and, in turn, to assess the *Factors*, the *Quantifiers* must be assessed. These *Quantifiers* are tangible, measurable characteristics of documentation. A general diagram of the three-level structure is shown in Figure 5 on page 13.

2.1.2.1 Qualities

Qualities are the very abstract characteristics of Adequate Documentation, defined as essential characters of their parent-nodes. The very generality of the definition relates the abstractness of this level. In the taxonomy tree, *Qualities* are the nodes directly under and most closely tied to the even more abstract notion of the root node, Adequate Documentation.

To procure stepwise refinement, the documentation must be examined from the viewpoint of the user. This is the driving factor, because the sole purpose for the existence of documentation is to inform users. To derive characteristics that will allow the adequacy of the documentation to be assessed, the user's needs for the documentation and the characteristics required of the documentation should be examined. One such characteristic is Completeness. The user requires that the documentation contain every piece of necessary information. This is a fairly uncomplicated request, but one which may not be easy to fulfill or even to check. It is also a very abstract characteristic. How can it be determined if everything that the user needs is present? A *Quality*, then, is a very abstract, simple, and desirable characteristic.

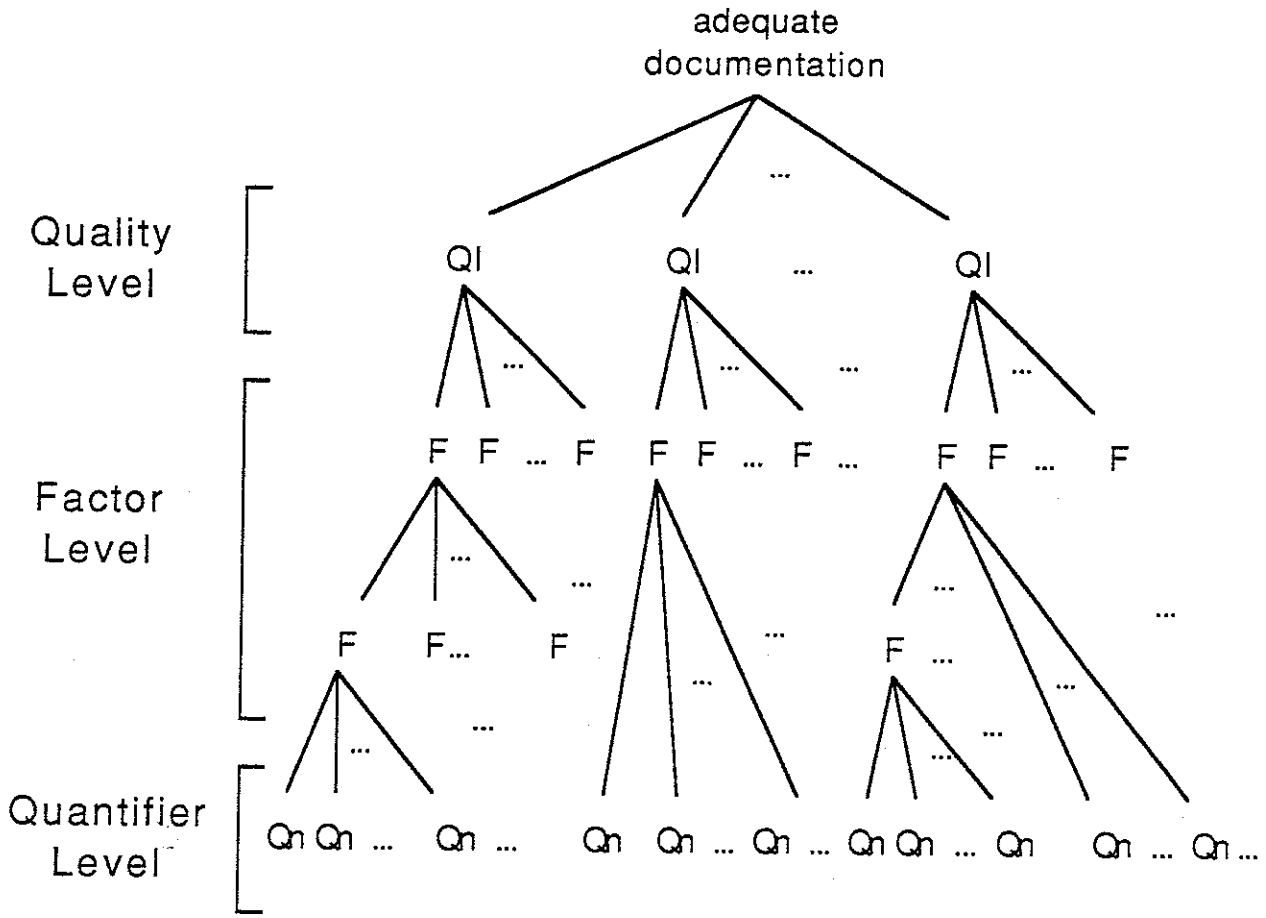


Figure 5. General Structure of Characteristics

2.1.2.2 *Factors*

Factors are intermediate characteristics, defined as an essential part which contributes to the production of a result. The term implies, by another usage, the idea of factorizing or breaking into parts the larger, more general qualities.

In the taxonomy tree, *Factors* are all of the intermediate characteristics between the extremely abstract *Qualities* and the measurable *Quantifiers*. They are directly derived from one of the *Qualities* and are less abstract. There are also multiple levels within the *Factor* level. The high-level *Factors* are characteristics that directly support a better understanding of one of the *Qualities*, while the secondary levels of *Factors* are all of the characteristics that lie between the top-level *Factors* and the *Quantifiers*. Most of the secondary *Factors* consist of the sub-class type of characteristic (Section 2.1.1.1). For example, two *Factors* of the *Quality* Completeness are Domain Coverage and Document Relationships. Both of these are less abstract characteristics that help to assess the *Quality* Completeness. As shown previously, Document Relationships has two sub-classes, Decompositional and Referential, which are secondary *Factors*. These are less abstract (i.e. more tangible), but are still not directly measurable.

2.1.2.3 *Quantifiers*

Quantifiers are the tangible, measurable characteristics of documentation. In the taxonomy tree, *Quantifiers* are at the bottom, i.e., the leaves. For example, a *Quantifier* associated with the *Factor* Domain Coverage is Percent Coverage. This is a measurable characteristic of documentation that is directly derived from the more abstract Domain Coverage. *Quantifiers* are directly derived from a *Factor* and support its assessment.

2.2 *Qualities*

As described above, *Qualities* are the most abstract characteristics in the tree hierarchy. Although abstract, they most closely convey the meaning of Adequate Documentation and are the primary characteristics a user is concerned with when dealing with documentation. When considering what makes documentation good, i.e. what constitutes Adequate Documentation, very general and simple characteristics are examined. These characteristics, the *Qualities*, are Accuracy, Completeness, Usability, and Expandability (See Figure 6 on page 16). First, the information in the documentation must be correct. Second, all of the information necessary must be present. Third, the people who need the information must be able to extract that information that they need from the documentation. Fourth, the documentation must be amenable to addition and modification.

The following sections give intuitive descriptions and formal definitions for all four of the *Qualities*. First, each section presents a general definition of the term that represents each of the *Qualities*. A complete description on an intuitive, less formal level is then given. Finally, a rigorous definition, within the context of Document Quality Indicators, is given for each.

2.2.1 Accuracy

The common definition for "accuracy" is the freedom from mistake or error; a synonym is "correctness." This definition is insufficient to convey the meaning of the accuracy of computer documentation. Within this context, Accuracy implies that the documentation must correctly reflect the actual state of the system it represents. Accuracy of the computer documentation includes both

- the intra-document consistency
- the consistency of the documentation with the executing computer programs.

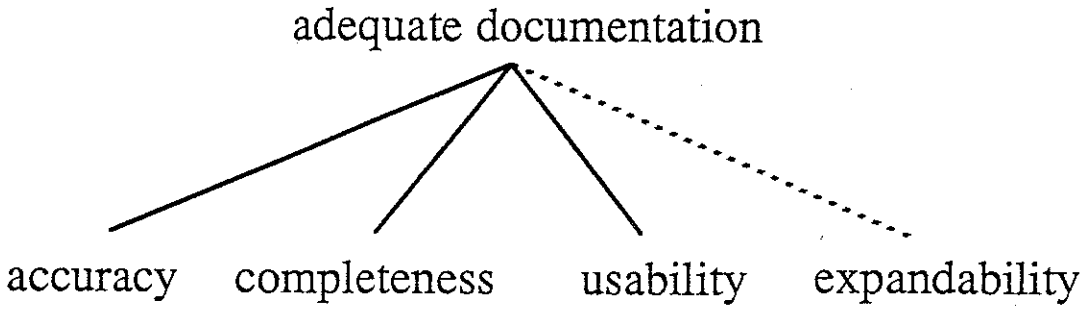


Figure 6. Decomposition into Qualities

When there *is* an inconsistency somewhere in the documentation, which of the versions is correct? This is relevant to both types of consistency listed above. The time at which the error was introduced may determine which version should be considered correct. To fully understand the points at which errors can be introduced, the distinction between the requirements and development documentation must be made. The key products of the development life cycle are

- requirements documents
- design documents
- code.

In the process of translating the information into these products, errors can be introduced when the requirements are being interpreted in design, and when the design is being interpreted and implemented as code. The requirements can be correct and the code and/or development documentation incorrect.

This, however, does not explain how the opposite type of error can occur: How can the code be correct, yet the development or requirements documentation be incorrect? This type of inconsistency is introduced during the maintenance phase of the Development Life Cycle. The code is maintained, problems are fixed, but the corresponding development and/or requirements documentation is not updated. As a result, some or all of the development documentation could be incorrect.

The next problem is how to assess Accuracy. One can not just "know" that all of the requirements are consistent with the development documentation and, in turn, with the code. The solution to this problem defines the next level in the hierarchical taxonomy tree, in the *Factors* of Accuracy section. Requirement and design decisions need to be traceable through the levels of documentation and code, and consistency needs to be maintained between these levels.

The full definition for **Accuracy** in the context of Document Quality Indicators is

> the consistency among the code and all of the documentation concerning the code, for all requirements. This implies that all requirements are traceable from requirements to design to code, where applicable.

2.2.2 Completeness

The standard definition for "completeness" is the possession of all necessary parts, elements, or steps, and its synonyms are "wholeness" and "integrity." This definition, unlike the one for accuracy, is basically satisfactory. For the purposes of computer documentation, a set of documentation is complete if all of the required information is present. The only problem lies in determining what is required or needed.

Two issues need to be considered to determine what documentation is necessary for a computer system. First, is the computer system itself; computer systems vary a great deal, so the documentation needed for them varies. Their differences notwithstanding, a standard exists for every type of computer system and these standards define what is required for the documentation completeness. Many such standards have been established [ANSI74, IEE185, POSA84, USDC76].

The definition for **Completeness** in the context of Document Quality Indicators is

> the existence of all documents required by a set of standards.

2.2.3 Usability

The dictionary definition for "usability" is the capability, convenience, or suitability of being employed. A synonym is "practicability." The two previous *Qualities* have defined extremely static characteristics of documentation, that establish both the *presence* and *correctness* of the necessary information. Unfortunately, even if the documentation is present and correct, the inability to extract the needed information can make the documentation useless. Usability assesses the amount of effort required for the user to understand the information. For example, part of assessing Usability is assessing the Logical Traceability of the documentation. How easy is it to find some item in the documentation? Furthermore, how easy is it to then trace the item through different parts of the documentation?

The definition for **Usability** in the context of Document Quality Indicators is

> the suitability of the documentation in terms of the ease of extracting needed information.

2.2.4 Expandability

The definition for "expandability" is the ability to increase an object's extent, number, volume, or scope. A synonym is "extensibility." The thrust of this *Quality* is the maintainability of the documentation. To what extent can the documentation be added to and modified?

The definition for **Expandability** in the context of Document Quality Indicators is

> the capability of the documentation to be modified in reaction to changes in the system. This *Quality* is assessed by the ease of modification.

Since this research considers only pre-defined static sets of documentation, Expandability is not considered any further.

2.3 Factors

Factors are characteristics which support assessment of one of the *Qualities* described in the previous section. This section describes each *Factor* individually and presents how each supports a designated *Quality*. The section is divided into three subsections; each subsection presents the *Factors* for one of the *Qualities*: Accuracy, Completeness, and Usability. Figure 7 on page 21 shows the addition of the *Factors* to the *Qualities* in the taxonomy.

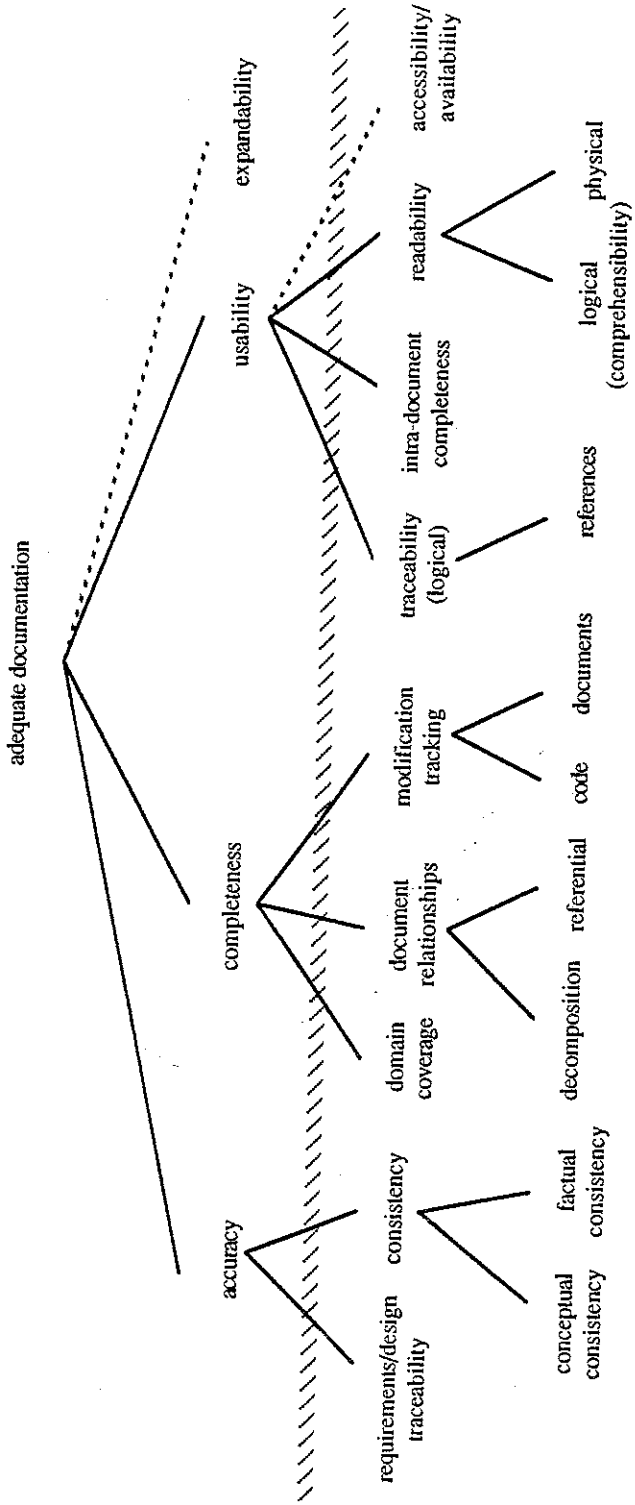


Figure 7. Decompositional to the Factor Level

2.3.1 Factors of Accuracy

In general, to demonstrate that documentation exhibits the *Quality* of Accuracy, one must show that the documentation is consistent with the deployed system. All versions of documentation, from requirements to high- and low-level design documents, need to be consistent with the code and exhibit consistency among themselves. In addition, one version of the documentation or code must be established as the correct version. In order to assess the consistency, the information items need to be linked in some manner. The information must be traceable from one version to the next. This traceability characteristic, along with two forms of consistency, are the *Factors* which support Accuracy.

In the following sections, each *Factor* of Accuracy is presented individually. Each is described in an intuitive fashion, then is defined with respect to DQIs. Accuracy is supported by two *Factors*: Requirements/Design Traceability and Consistency (see Figure 8 on page 23). Consistency is further divided into the secondary *Factors*, Conceptual and Factual. This section gives the descriptions and definitions for each of these *Factors*. Each intuitive description includes an example.

2.3.1.1. Requirements/Design Traceability

Requirements/Design Traceability provides the means for assessing whether the necessary links exist to evaluate Accuracy. As described in the description of Accuracy, traceability is necessary in order to check consistency. To achieve traceability, the requirements and design decisions must be enumerated as "atoms," with each requirement given as a single, indivisible entity. These atoms can then be traced and assessed for consistency.

In addition to the system requirements, new requirements are usually introduced during the design phase. These are called design requirements and are spawned by design decisions (See Figure 9 on page 25). For example, if a database is listed as a system requirement, then this need will gen-

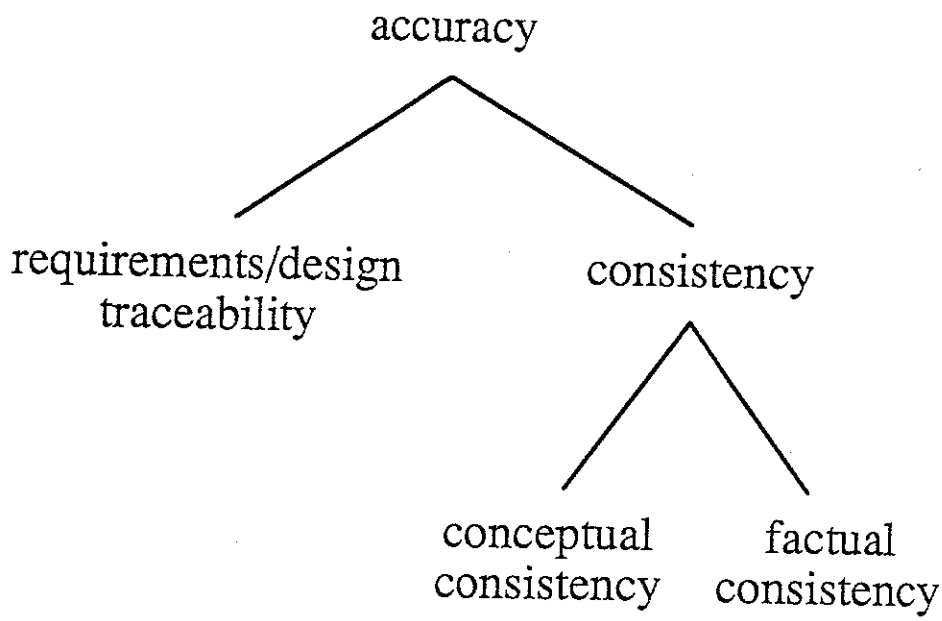


Figure 8. **Factors which Support Accuracy**

erate a set of design requirements (eg., where the database is to reside, its name, and so forth). The design decision, to incorporate a database, imposes additional requirements to be placed on the system. These design requirements must also be enumerated as atoms.

In another example, the system requirements for a particular system might include a requirement that specifications set forth in Standard 'X1' be met. This is a general requirement which affects the entire system. Since the standard has numerous individual specifications, it must be broken into atoms to assess traceability. Among such specifications might be one which states, "error messages must be logged in file 'ERROR FILE.'" Not only do the high- and low-level designs have to note this in some manner, the code *must* implement this requirement. The high- and low-level designs might, in fact, group the atoms together in the same general way that the system requirements do, such as by simply saying, "in accordance with Standard 'X1.'" Nonetheless, it is important for the evaluation that the requirements can be atomized, for two reasons:

- the requirements cannot be systematically assessed for consistency, unless the requirements can be treated as atoms
- the requirements can break apart at the source code level, in that only some of them are germane to each section of the code.

The definition for **Requirements/Design Traceability** in the context of Document Quality Indicators is

> the ability to track individual atoms of system requirements and design requirements from their introduction to the code which implements them.

The manner in which these requirements are actually assessed for traceability is described in the Quantifiers section for this Factor.

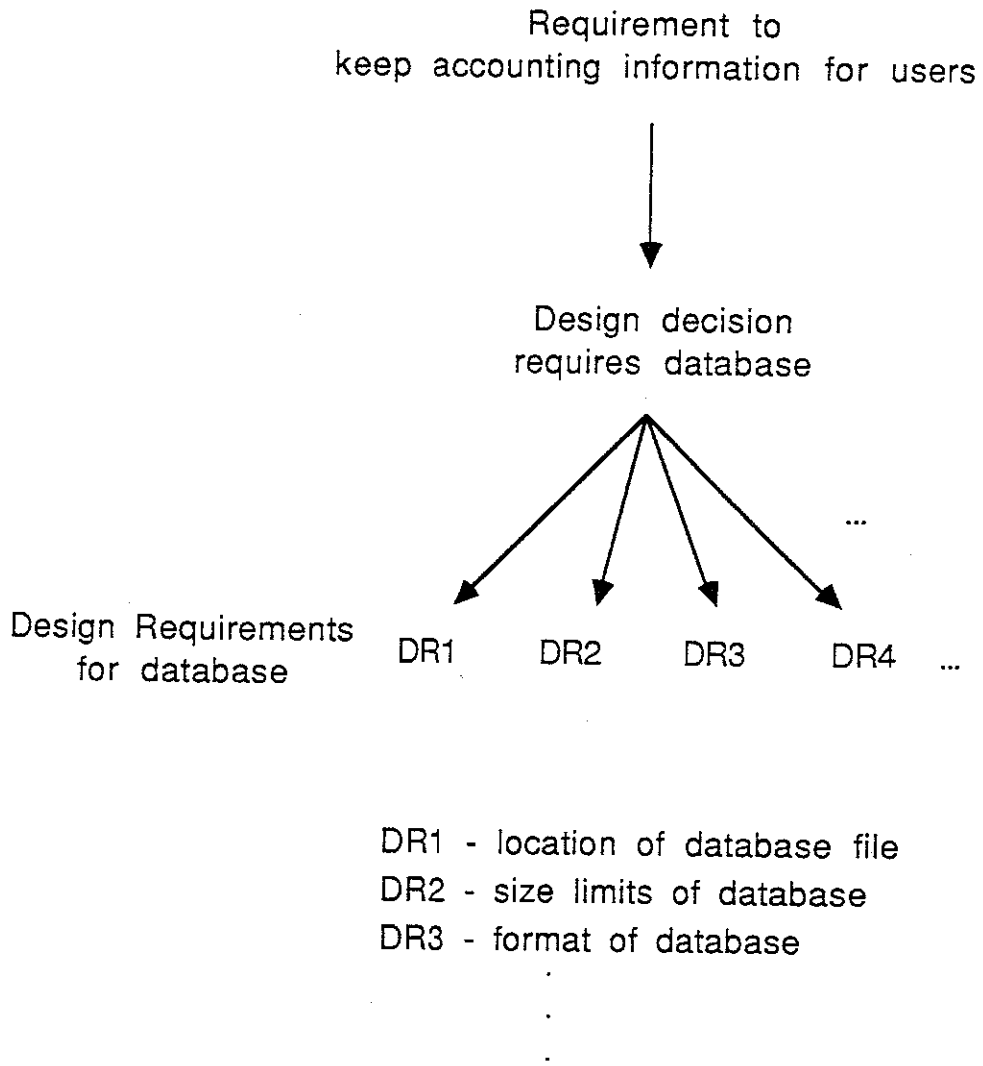


Figure 9. Creation of Design Requirements

2.3.1.2. Consistency

Consistency is the agreement or harmony demonstrated among separate items. For documentation, it is the statement of multiple occurrences of information in a congruent manner. The same idea must be expressed in a similar fashion or in a way that is not contradictory. Consistency has two facets: Conceptual and Factual. Conceptual Consistency means that an idea may be stated in different forms, but the forms must reflect the same idea. Factual Consistency means that statements of value, logical relationships, and definitive structure must remain invariant irrespective of their location and repetition within the documentation.

Consistency can be checked in both a horizontal and a vertical direction with respect to the Development Life Cycle Products. For the horizontal direction, all of the documents which are products of a given stage of the Life Cycle must be consistent. For the vertical direction, the products from different stages of the Life Cycle must be consistent. Factual Consistency must be checked in both directions, because facts are presented at various places in the documentation at each stage in the Life Cycle. Conceptual Consistency, on the other hand, is primarily a concern in the vertical direction; because the concept has the highest potential for change between stages of the Life Cycle.

The definition for **Consistency** in the context of Document Quality Indicators is

> the agreement or concurrence of all information in the documentation. For all information items in the documentation, all statements concerning each information item must agree.

2.3.1.2.1. Conceptual

Conceptual Consistency is the coherence or agreement of ideas. This characteristic assumes that the representation, i.e. the language of representation, may change in the transition from requirements to design and also from design to implementation. This change might stem from:

- the requirements language, specification language, design language, and implementation language can be very different
- the specific words or phrases may differ among the requirements, specifications, design, and code.

The representation change is generally a combination of the two sources above. Most often, the requirements, specifications, and design documents are written in English, but the implementation is written in a programming language (e.g. Pascal, FORTRAN, CMS-II, COBOL).

For example, a requirement states that titles are to be printed at the top of each report. High-level design documents would make the same statement. The low-level design documents would be more specific; they would state approximately or exactly the words for the report titles. The code, of course, would have to print the title using the implementation language.

The definition for **Conceptual Consistency** in the context of Document Quality Indicators is

> an idea representing an individual requirement atom which remains unchanged from requirements through design to implementation, regardless of the representation.

2.3.1.2.2. Factual

Factual Consistency is the agreement among factual expressions (statements, values, logical relationships, and definitive structures). Some of the requirements for a system are stated as simple numeric or name facts, such as that the system must handle 10 input lines. The design document would state 10 input lines, while the code would check a number-of-lines variable to make sure it is at least 10. As another example, the requirements state that error messages must go into file 'ERROR FILE.' The designs would make the same statement and the code would write errors to a file 'ERROR FILE.' The definitive facts, 10 and 'ERROR FILE,' can be checked directly for consistency.

The definition for **Factual Consistency** in the context of Document Quality Indicators is

> factual expressions remain the same from one level in the Development Life Cycle Documentation to the next, from requirements to code.

2.3.2 Factors of Completeness

In general, "completeness" has three facets: (1) existence of all of the necessary documents, (2) existence of relationships among the documents, and (3) capability for tracking changes to code and documentation. These facets are embodied in three high-level *Factors*: Domain Coverage, Document Relationships, and Modification Tracking. The last two of these *Factors* are further decomposed into secondary *Factors* (See Figure 10 on page 29). Document Relationships is divided into Decompositional and Referential relationships. Modification Tracking is divided into Code and Documentation secondary *Factors*.

Each *Factor* that supports Completeness, high-level and secondary, is described in an intuitive fashion and defined with respect to DQIs. Following the definitions is a section describing the reasoning behind the taxonomic decomposition.

2.3.2.1. Domain Coverage

Domain Coverage is a characteristic used when examining whether a set of elements for completeness. In general, a standard list of necessary items is envisioned and, if everything on the list is present, the set is judged complete; if not, the set is deficient. More specifically, Domain Coverage considers whether a set of documentations is complete, i.e. whether all of the necessary documents are present. Precisely determining which documents are necessary involves defining a *template* for the system. A template is a list of documents that should exist for given system, and which can

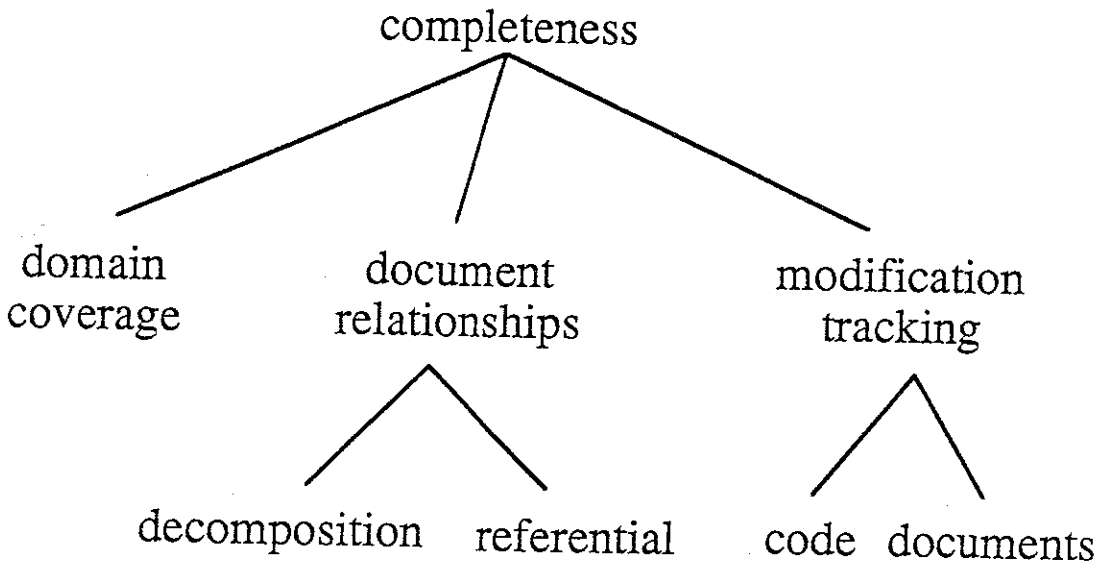


Figure 10. Factors which Support Completeness

be derived from a standard. It may be part of the system documentation itself or it may need to be "manually" defined because none exists.

The definition for **Domain Coverage** in the context of Document Quality Indicators is

> the presence of all documents as specified in a pre-defined template.

2.3.2.2. Document Relationships

Document Relationships are transparent components of Completeness. Unlike Domain Coverage, which is intuitive and tangible, the relationships among the documents are more difficult to envision. While the characteristic Document Relationships may not be as visible, it has been noted as an important aspect of documentation [HORE86a, HORE86b]. Not only must all of the documents exist, they must be linked together. A "graph model for the set of documents ... consists of a representation of the structure, content, and interrelationships of all the documents" [HORE86a]. "Each component of a document is related to the components of other documents and components within the document itself" [HORE86b]. The nodes of a graph model represent the documents themselves, the arcs represent the relationships; the arcs are as important as the documents because they bind together the isolated nodes, creating a solid unit.

Obviously, this characteristic is dependent upon the previous one, Domain Coverage. Nodes, or documents, must exist in order for there to be any relationship between them.

The definition for **Document Relationships** in the context of Document Quality Indicators is

> the structure among and within the individual documents of a set of documentation, which ties the documents together into a single, unified set.

There are two types of document relationships: Decompositional and Referential. The following sections describes and defines these two supporting *Factors*.

2.3.2.2.1. Document Relationships/Decompositional

The Decompositional relationship is the natural, hierarchical ranking of a set of documentation (see Figure 11 on page 32). It successively divides the nodes of the graphical representation into smaller and smaller nodes. "Smaller," in this context, means that the scope of each individual document is reduced. Correspondingly, as the scope of the individual documents shrink, the amount of detail for each document increases.

For example, Figure 11 on page 32 shows the decomposition of the documentation for a system. At the top of the hierarchy is an overview of the entire system. The second level is more specific; it consists of overviews of Subsystems A, B, and C. The third level consists of descriptions of the individual features of each subsystem, which are more detailed than the overviews. In the subsequent levels, the detail of the documentation increases until the decomposition is down to the level of describing the source code itself. Overall, the level of detail gradually grows from general at the top to very detailed at the bottom.

The definition for the **Decompositional Relationship** in the context of Document Quality Indicators is

> the existence of a definitive or obvious hierarchical ranking among the documents.

The graphic structure of the relationships is a tree: a single encompassing document at the top, which is successively decomposed into documents with smaller and smaller scopes. As the scope of each individual document decreases, the amount of detail increases.

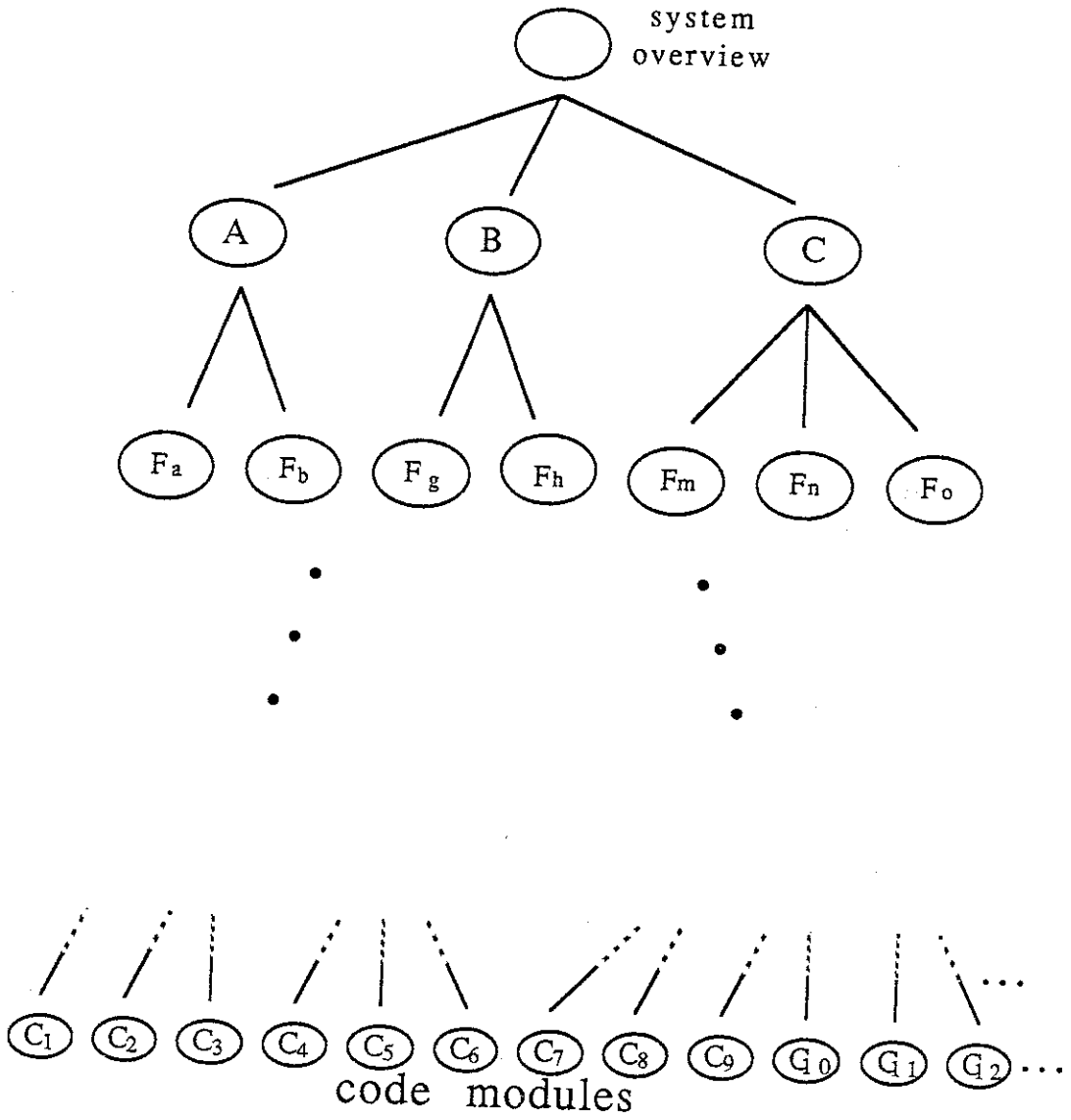


Figure 11. Decompositional Relationship Among Documents

2.3.2.2.2. Document Relationships/Referential

Referential Relationships are links between documents that are more tangible than simply the Decompositional Relationship. A Referential Relationship is a specific cited reference to another document or another section in the same document. One form of of the Decompositional Relationship can be mistaken for a Referential Relationship. For example, the reference, "See Users Command Language Manual for more information" is a reference which explicitly states a Decompositional Relationship. The cited manual has more detail on a given subject, which makes this example more an instance of a decomposition than of a cited reference. On the other hand, a reference stating "See *for* statement in the Users Command Language Manual" is a simple reference. Figure 12 on page 34 shows how Referential Relationships help to tie together the documents of a set.

The definition for the **Referential Relationship** in the context of Document Quality Indicators is

> the existence of appropriate references to other documents or to other parts of the same document.

2.3.2.3. Modification Tracking

Modification Tracking is the recording of changes made to files. Modification tracking systems record the changes and maintains a history of the changes. The history should contain several pieces of information:

- who made the change
- what exactly was changed
- when it was changed
- why it was changed

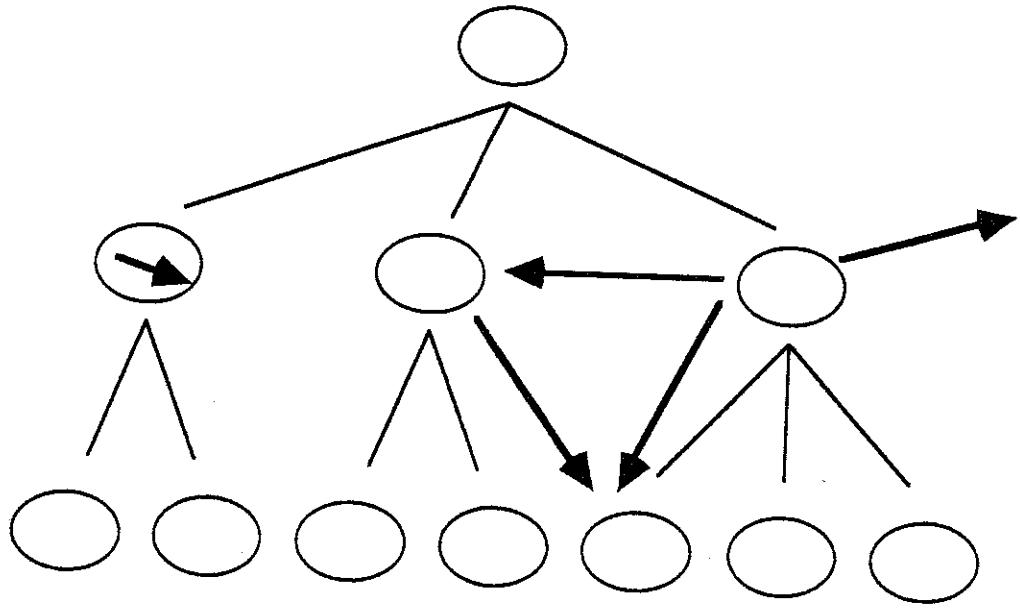


Figure 12. Referential Relationships

- what other files the change affected
- version numbers.

The system itself should have some of the following features:

- a friendly interface
- the general qualities associated with good software (e.g. fast, good error recovery, etc.)
- the ability to reconstruct previous versions of sets of associated files.

The documentation for a computer system should be able to reflect Modification Tracking information, not only for the code but also for the documentation. This implies that there should be two tracking methods, and possibly two separate systems might be necessary. Regardless of whether there is one system or two, the assessment should be separate for the code and for the documentation, since features that are good for one may not be good for the other. The requirements for the methods and their users need to be considered to determine whether two systems are needed. General features of each method are presented in the following sections.

The definition for **Modification Tracking** in the context of Document Quality Indicators is

> the existence of systems to record changes made to the code and documentation for a computer system.

2.3.2.3.1. Modification Tracking/Code

Modification Tracking of Code has all of the general features presented in the previous section. Its primary emphasis is on the changes themselves: What changed between the previous version and this version? Of secondary concern is the effects of the change on other code. For example, if an error was never noted before a certain modification was made, but the error occurred repeatedly

after the modification, then the code which was changed between the two versions may have introduced the error. The old code and new code should be examined side-by-side to discern how the modifications could have introduced the error.

This same problem includes an example for the secondary concern, mentioned above. The changes in a given portion of the code could change

- which code is affected by the given code
- the manner in which other code is affected.

In the first case, code could be added to the original, so that new code is called by the original. In the second, subroutine calls could be modified in such a way that the affected code is changed.

The definition for the **Modification Tracking/Code** in the context of Document Quality Indicators is

> the existence of a good system to manage and record the changes made to the source code of a system. The "goodness" of the system is determined by how well it meets the needs of its users.

2.3.2.3.2. Modification Tracking/Documentation

Modification Tracking of Documentation is very similar to Modification Tracking of Code. They differ only because of the inherent differences between documentation and code. As with Modification Tracking of Code, the most important information is the change itself. What exactly changed between versions? The secondary concern revolves around the effects changes have on other documentation and on the code described by the document. Although the importance of keeping track of the change itself is self-evident, the effects of a change are less straightforward, but

still important. If the effects are tracked, then the accuracy of the documentation can more easily be tracked and consistency maintained.

The definition for the **Modification Tracking/Documentation** in the context of Document Quality Indicators is

> the existence of a good system to manage and record the changes made to the documentation of a system.

2.3.3 Factors of Usability

Usability is the characteristic measuring the capability, convenience or suitability of employing the documentation. As presented previously (see section 2.2.3), this *Quality* assesses the ease of extracting information from the documentation. Assuming that some given information is in the documentation, how easy is it to find?

In the following sections, each high-level and secondary *Factor* which supports Usability, is described in an intuitive fashion. Then, each is briefly defined with respect to DQIs.

Usability is divided into four high-level *Factors*: Traceability, Intra-document Completeness, Readability, and Accessibility/Availability (See Figure 13 on page 38). Some of these *Factors* are further divided into secondary *Factors*. Traceability is supported by the secondary *Factor*, References, along with other *Quantifiers*. This branch of the taxonomy is described more fully in the following Traceability section. Another *Factor* that is supported by secondary *Factors* is Readability; it is divided into Logical Readability and Physical Readability. Each of these is described and defined in the following sections. Each intuitive description includes an example.

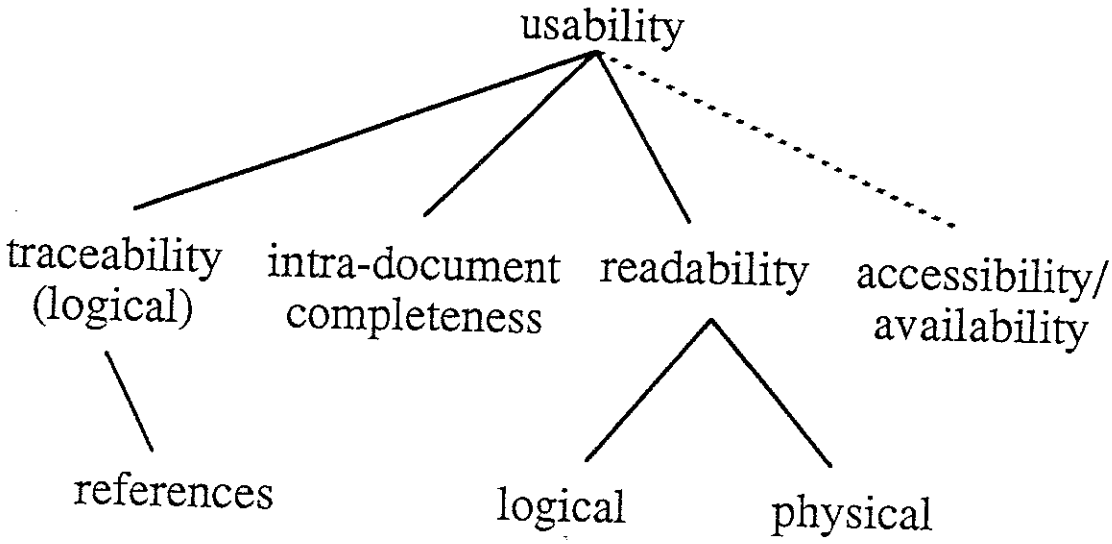


Figure 13. Factors which Support Usability

2.3.3.1. Traceability (Logical)

Traceability, in the form in which it refines Usability is called "Logical Traceability." Logical Traceability is the ability to track a given theme through the documentation. "Theme," in this case, means an idea or a particular word. Basically, if one tries to get some information out of the documentation and it is not all in one section of the documentation, how easy to find is all of the necessary or interesting parts? A user must be able to

- trace through the documentation
- find everything which he/she needs
- understand the ideas presented.

This is a unique portion of the taxonomy tree. Traceability is supported by one secondary *Factor* and directly by several additional *Quantifiers*. No other branch of the tree has this configuration; this places a secondary *Factor* on the same level, more or less, as some *Quantifiers* (See Figure 14 on page 40). All other branches of the tree have either all *Factors* or all *Quantifiers* supporting the parent characteristic.

The secondary *Factor*, References, is itself supported by two *Quantifiers*. These *Quantifiers* could be "pulled up" to directly support Traceability, but that action would destroy the hierarchical relation.

The definition for the **Logical Traceability** in the context of Document Quality Indicators is

> the ability to follow the logical train of thought in the documentation through all of the pertinent parts, regardless of whether the parts are contiguous or not.

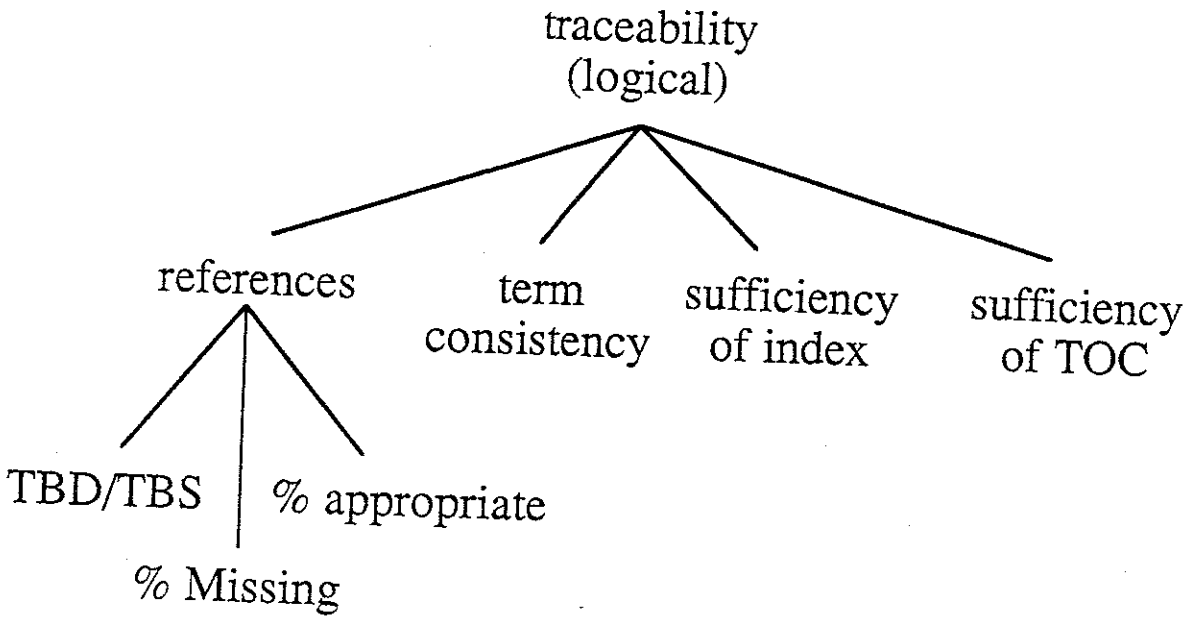


Figure 14. Taxonomy which Supports Traceability

2.3.3.1.1. Traceability/References

References, a refinement of Traceability, provides direct links to allow users to follow the logic of the documentation. At places in the text where the user needs to jump to another portion of the documentation, it is very helpful if the text states specifically where to go for more information. This characteristic, which binds the documents together, has the practical quality of allowing the user to trace through the documentation. This is similar to the Referential Relationship presented under Completeness, but here has a user-oriented emphasis.

If, for example, a user is investigating the "SCALE" command, it would be useful to have explicit references to related commands. It may be that the user will discover, upon reading the documentation for "SCALE," that it is not the needed command. If related commands are listed, they might help the user identify the required command. A reference can also help users by giving further detail. Although the additional text is not always necessary, it can benefit some novice users by providing additional information to clarify a problem.

The definition for the **Traceability/References** in the context of Document Quality Indicators is

> the existence of means to provide ties between separate portions of the text which can give more information about some item in the text. The additional information can either be more detail or related topics.

2.3.3.1.2. Traceability/Other Quantifiers

The other characteristics which, along with References, help to assess Logical Traceability are not gathered at the *Factor* level; as measurable characteristics they more properly belong on the *Quantifier* level. These *Quantifiers* are presented in sections 2.4.3.2.1 - 2.4.3.2.3.

2.3.3.2. Intra-document Completeness

Intra-document Completeness is similar to the larger Completeness of the entire documentation. The major difference is that, since this completeness deals only with single documents, it not only demands that all of the parts be present for each document, but that each *part* be complete. The *Quality* Completeness and its *Factor* Domain Coverage are concerned with the existence of all of the required, individual documents, but existence does not guarantee content. In other words, Domain Coverage does not assess the utility of the documents. Intra-document Completeness has two basic functions, to determine:

- that all of the necessary sections are present in each document
- that the sections have relevant content.

The first function is similar to Domain Coverage: each document or type of document has a defined set of sections, or components, which must be present for it to be complete. The definition of the necessary sections is dependent on the type of document in question. The second function, determining if the sections have relevant content, does not require that each section be carefully read and analyzed for accuracy and lucidity. Each document should be assessed to determine if all sections appear to have some content.

If, for example, the "Command Language User Manual" for a given system should have a table of contents, introduction, three sections, an index, and cross-references, the assessment of Domain Coverage should use a template to determine that everything is present. Then, each of the sections should be given a cursory evaluation for relevant content. The definition for *Intra-document Completeness* in the context of Document Quality Indicators is

> the existence of all required sections for all of the documents in the set of documentation for a system. These sections should not only exist, but should have relevant content.

2.3.3.3. Readability

Readability is one of the easiest characteristics to understand intuitively. In order for the documentations to be usable, the user must be able to read and understand it. Readability has two facets: Logical and Physical. Logical Readability is the characteristic of being comprehensible or understandable. Physical Readability concerns the ease with which the actual document can be read. How easy is it for the reader's eye to physically scan the text? In the following sections, each of these two facets are described individually.

The definition for the **Readability**, in general, in the context of Document Quality Indicators is

> the ease with which the documentation can be physically and intellectually comprehended.

2.3.3.3.1. Readability/Logical (Comprehensibility)

Logical Readability, also called "Comprehensibility," is a measure of how easy the documentation is to understand. Much research has been done in the area of establishing a level of difficulty of reading a given portion of text [DALE48, FLER48, KINJ75, SIDC82, TAYW53]. This research, however, only concerns textual or prose documentation. While it is helpful in this context, it does not assist in the evaluation of non-textual, or representational, portions of documentation. Additional methods of assessing non-prose documentation must be developed.

Some readability measures produce a number which indicates the grade level which is assumed for a given portion of text [DALE48, FLER48, KINJ75]. Others produce a measure of reading difficulty which can then be scaled to produce a grade level [TAYW53]. A level of User is chosen or assumed for the evaluation of the documentation and a reading grade level is associated with it. If the level produced by the reading formulae is less than or equal to the level of the User, then the text is assessed favorably.

The sections of a design description, for example, are usually in prose form. These can be evaluated for readability using various known formulae, as presented in the previous paragraph . Other documents, such as a Procedure Calling Hierarchy for the system, are given in source code and therefore cannot be evaluated by such formulae; other characteristics must be examined to measure the readability of sections of this type.

The definition for the **Readability/Logical** in the context of Document Quality Indicators is

> the ease with which the documentation, in general, can be comprehended.

2.3.3.3.2. Readability/Physical

Physical Readability concerns the format and design of the documentation and the degree to which it contributes to (or inhibits) comprehension by the user. Is the print legible? Does the format of the documentation assist and encourage the user, or is it visually confusing?

Documentation which has been typeset in very small type with narrow margins and little white space between the lines can be very difficult to read; no matter how lucid and useful the content of the prose may be, the difficulty presented by the physical layout of the documentation reduces its readability.

The definition for the **Readability/Physical** in the context of Document Quality Indicators is

> the appearance of the documentation, in terms of format and design, and its relation to the ease with which it can be used.

2.3.3.4. Accessibility/Availability

Accessibility/Availability is the characteristic concerned with the ability to use the documentation when it is needed. Documentation serves no purpose if it cannot be accessed. Two issues arise in respect to this characteristic: form and access. The form of the documentation can be hard-copy and on-line. Both forms are considered equivalent, although this is documentation-system dependent. In fact, the on-line version of the documentation may not be as complete as the hard-copy version, or the opposite can be true. The documentation can be so voluminous that all of it is never printed. Additionally, the on-line facilities might be incomplete.

Access to the documentation can be limited or restricted for the following reasons:

- the security classification of the documentation
- distributed development
- outside development of software
- the documentation simply does not exist, having been lost or even never written.

Availability/Accessibility can be an overriding factor when evaluating documentation. This research, however, assumes that the documentation is present, complete and accessible. Therefore, this characteristic is not considered further.

For the purposes of further research, the definition for the **Availability/Accessibility** in the context of Document Quality Indicators is

> the ease with which the user can obtain or view the documentation. This includes both hard-copy or on-line versions of the documentation.

2.4 *Quantifiers*

Quantifiers are measurable characteristics, which associate numeric values to their related *Factors*. Each *Factor* is supported by one or more *Quantifiers* and each *Quantifier*, in turn, can be measured in one or more ways.

This section is divided into three subsections, corresponding to the three *Qualities*. Under each *Quality*, the supporting *Factors* are listed, and for each of these *Factors*, the supporting *Quantifiers* are presented. Figure 15 on page 47 shows the addition of the *Quantifiers* to the taxonomy. Note that this is the lowest and final level of the taxonomy; therefore, this figure, Figure 15, is the same as Figure 1 on page 6.

2.4.1 Accuracy: Supporting Quantifiers

The *Quantifiers* which support each of the *Factors* of Accuracy are described in the following sections. For an illustration of the sub-tree structure, see Figure 16 on page 48.

2.4.1.1 Requirements/Design Traceability: Top-down/Bottom-up Equivalence

Top-down/Bottom-up Equivalence is the equality of the specified requirements to the implemented code. In order to meet a necessary-and-sufficient condition, the equality must be subject to examination in both a top-down and a bottom-up manner. The requirements must be "sufficiently" met by the implementation, and all of the code must be "necessary" to meet the requirements, i.e. there is no superfluous code.

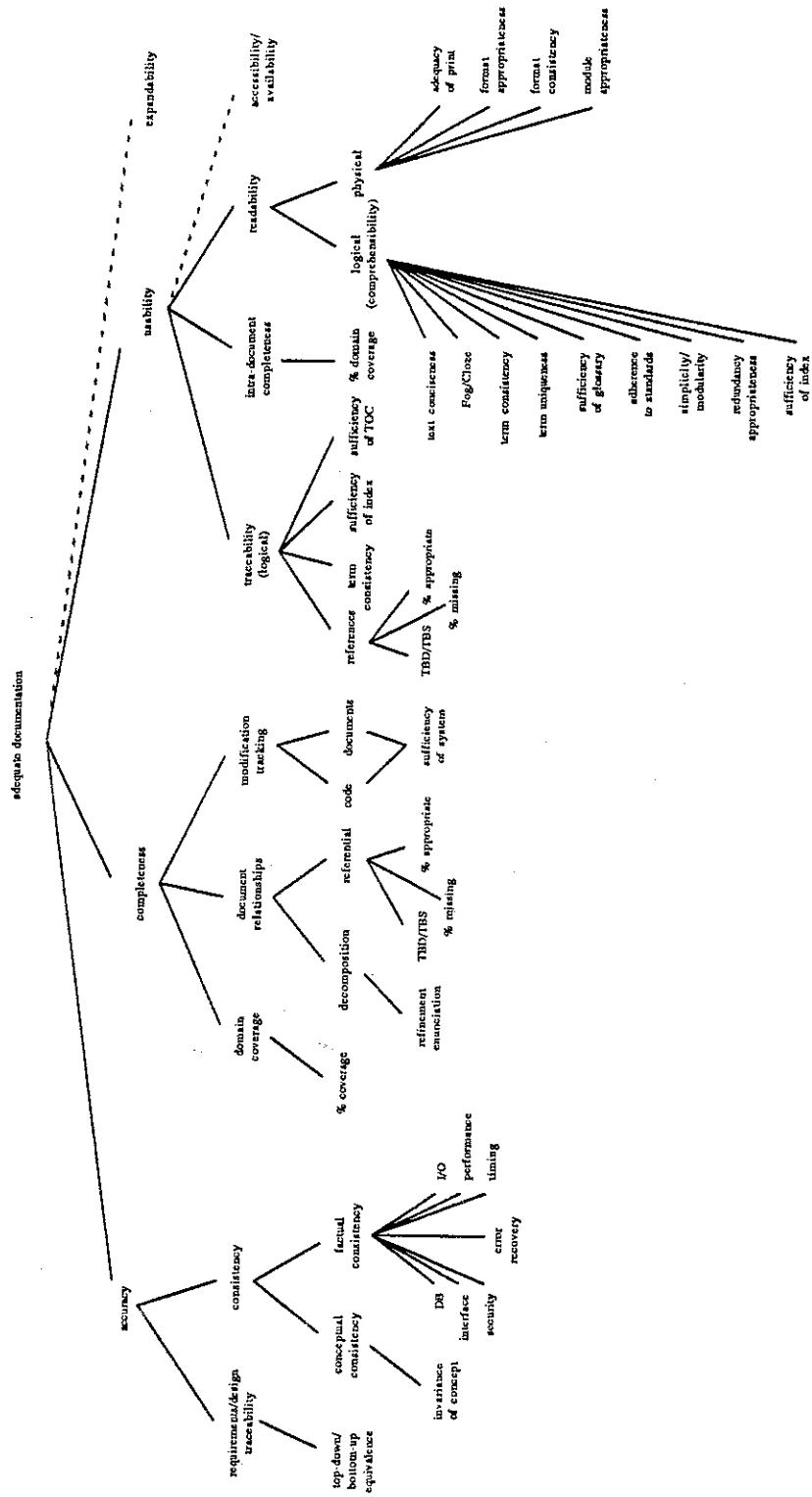


Figure 15. Decomposition to the Quantifier Level

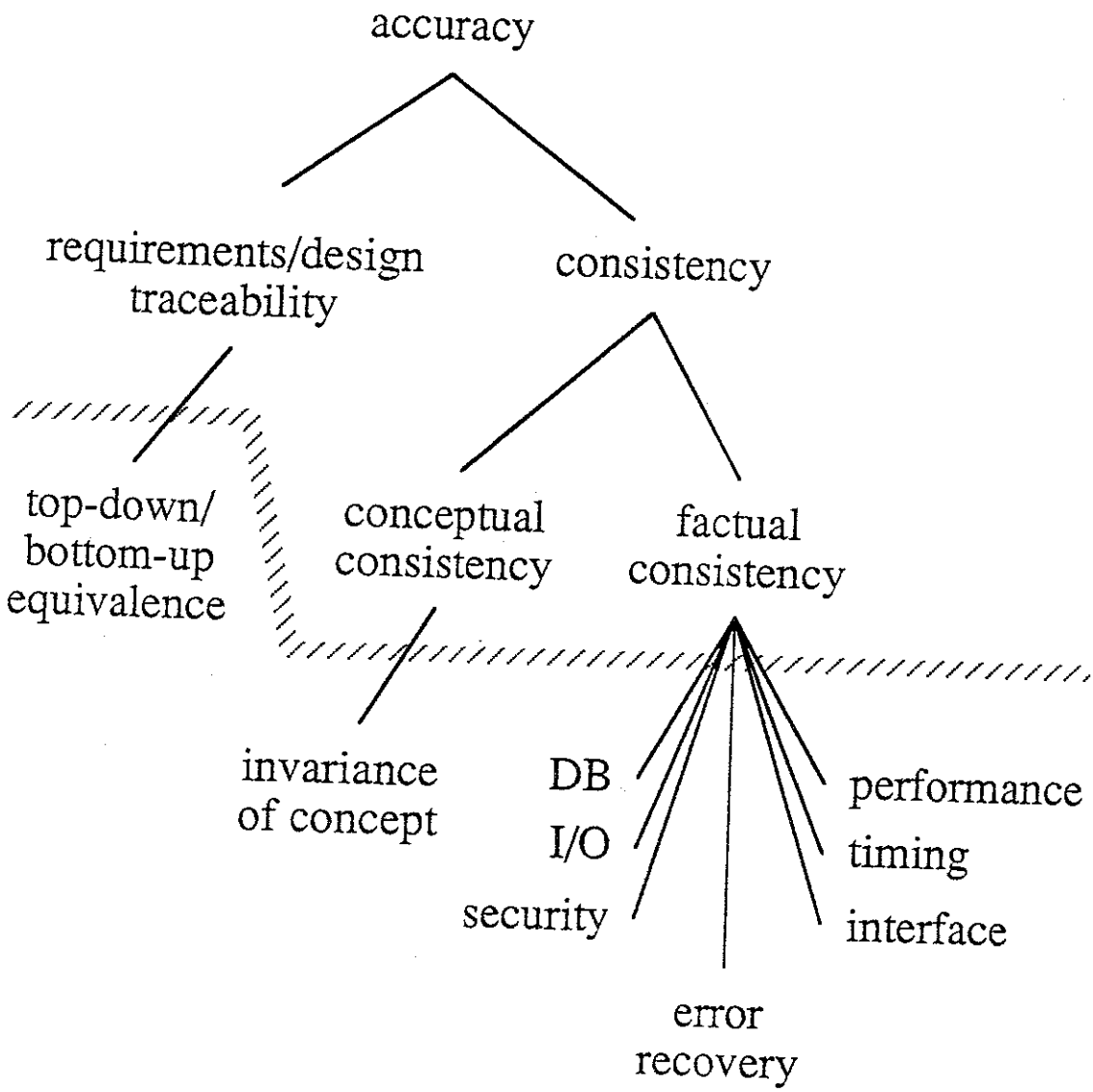


Figure 16. Quantifiers Which Support Accuracy

The top-down trace checks whether all of the requirements have been "sufficiently" met by the implementation. All of the requirements are enumerated as atoms (see section 2.3.1.1 on Requirements/Design Traceability), so that they can be divided and associated with separate features which are helping to meet or satisfy them. The separate features are then further divided, thereby dividing the requirements with which they are associated. The bottom-up trace checks whether all of the code modules are actually "necessary" or needed to meet a requirement.

This top-down/bottom-up tracking can best be accomplished by setting up matrices (See Figure 17 on page 50). One matrix presents the requirements related to the design modules; another presents the design modules related to the code modules. For the first, the requirements are enumerated across the top of the matrix and the design modules along the left side. For each requirement, an "X" is marked for the design modules that partially or completely fulfill that requirement. For the second matrix, the design modules from the first matrix are enumerated across the top and the code modules along the left side. Similar to the first matrix, an "X" is marked for the code modules which partially or completely implement each design module.

Once the matrices are filled in, they are used to check that all of the requirements are met and that all of the code modules are necessary. If a requirement column in the first matrix does not have at least one "X," this is evidence that no design module exists to implement it. Further, the requirements also depend on the design modules being implemented; each design in the second matrix which is not implemented in a code module fails to fulfill requirements. In this manner, the requirements can be checked to determine whether they are all *sufficiently* fulfilled.

Secondly, the rows of the matrices need to be examined. Any row which does not have at least one "X" in it is not needed; it does not help to fulfill a requirement. This examination allows one to determine whether a design module is necessary to meet requirements.

It should be noted that the above procedure not only shows how to examine for vertical traceability, i.e. traceability between requirements, design, and code, but also provides a means of

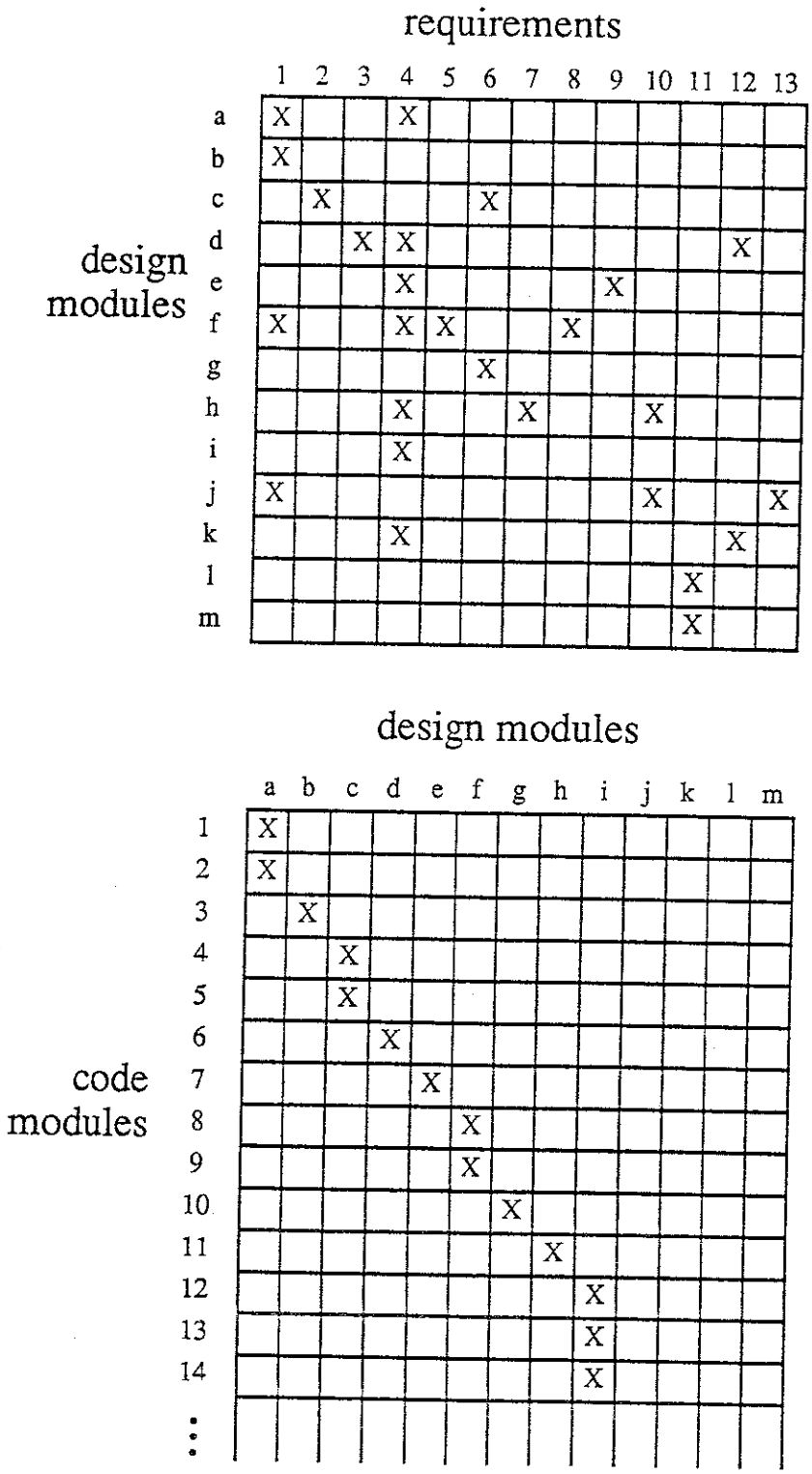


Figure 17. Requirements/Design and Design/Code Matricies

performing the same in a horizontal direction. Since a single requirement can effect more than one design (and thereby code), several design modules may exist to help meet that requirement. If each design module is traceable to the original requirement, then relationships can be traced through that requirement. Although these relationships are horizontal, in that the design modules are on the same level, they are not actually *traced* horizontally; instead, their relationships are traced indirectly using the vertical relationships in Top-down/Bottom-up Equivalence.

The above description of the matrices shows how to check the documentation for "sufficiently meeting the requirements" and "necessary to meet the requirements." Additionally, since the requirements must be made into atoms and the atoms are then "linked" from requirements to design to code, the atoms are traceable through the documentation and code. The Consistency of the atoms is dealt with by the next two *Quantifiers*.

2.4.1.2 Conceptual Consistency: Invariance of Concept

To achieve Conceptual Consistency, the ideas presented by the requirements, designs, and code must be invariant. The requirement atoms are linked together through these levels for traceability. Each of the atoms for all of the concepts must be compared to make sure that all present the same idea. The concern, here, is primarily with the consistency between requirements, design modules, and code, since concepts are most likely to become inconsistent from level to level. This assumes consistency within a single document. As explained above, consistency among the documents at a particular level will exist as a product of the consistency between the levels.

2.4.1.3 Factual Consistency

The assessment of Factual Consistency is subdivided into categories: performance specs, timing specs, database specs, interface specs, security specs, error recovery specs, and I/O specs. Each of

these categories has a set of facts or specifications which are assessed for their consistency. Again, the traceability links set up in the matrices are used to determine which items need to reflect consistency. The items themselves should be compared to decide factual consistency.

2.4.2 Completeness: Supporting Quantifiers

The *Quantifiers* which support each of the *Factors* of Completeness are described in the following sections. For an illustration of the sub-tree structure, see Figure 18 on page 53.

2.4.2.1 Domain Coverage: Percent Coverage

In the description of Domain Coverage, the concept of a template is presented. Percent Coverage is the proportion of the items in the template which exist to the total number items in the template.

2.4.2.2 Document Relationships/Decompositional: Refinement Enunciation

Refinement is enunciated in one of three forms: standard, implicit, and explicit. *Standard* enunciation is among a standard set of documents. *Implicit* enunciation concerns the existence of documents which refine the information in a section of a document but for which no defined relationships are explicitly stated. *Explicit* enunciation is the stated relationships among the documents. These relationships can be given in a list or as references from the higher-level to the refinement document.

To measure the standard enunciation, the template is checked and a number, which represents the completeness of the enunciation, is calculated. This number could be expressed as the percent of total relationships, for example. To measure the implicit enunciation, a bottom-up view is used.

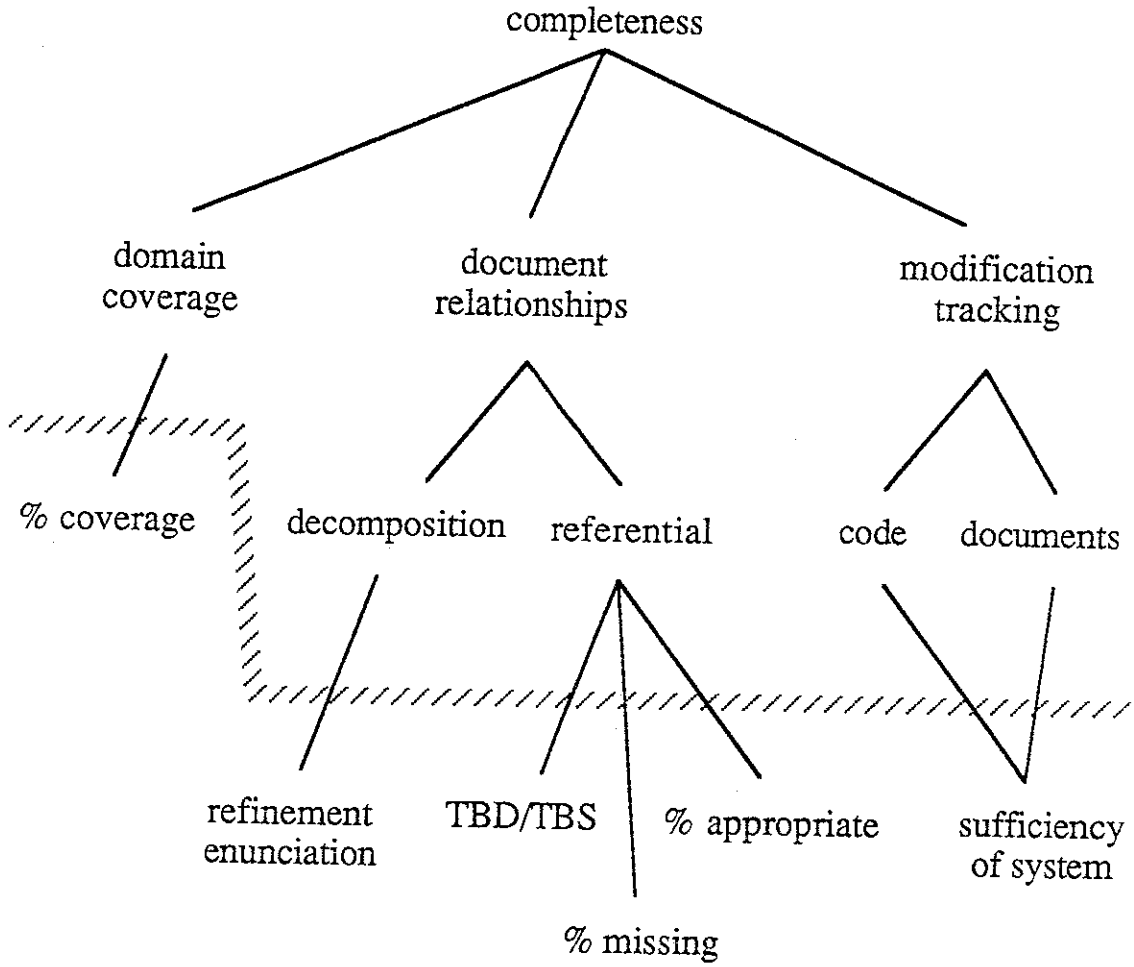


Figure 18. Quantifiers Which Support Completeness

Individual documents are examined for a section or subsection which presents some material which is the sole subject of another entire document. The document which contains the subsection is the parent node in the hierarchy; the corresponding document, concerned only with that subject, is the child node. Therefore, a document on word processing, which contains a sub-section on a spell-checking feature, is the parent node to a separate document which only discusses the spell-checker. The relationships between documents, when mapped, should produce a hierarchical tree structure. The degree to which such a structure can be produced should be translated into a refinement enunciation number. Finally, to measure explicit enunciation, all explicit references to refinement documents are extracted and the existence of those referenced documents are counted. The number of referenced documents should be translated into a percent of the total number of explicit references.

2.4.2.3 Document Relationships/Referential

Referential Relationships are supported by the assessment of the three *Quantifiers*: TBD/TBS, Percent Appropriate, and Percent Missing.

2.4.2.3.1 TBD/TBS

TBD/TBS is a count of the number of occurrences of "To Be Defined" and "To Be Specified" statements in the documentation. Frequently, during development of a system, documentation is written before the system is complete. Since information required for documentation is sometimes not available until completion of the system, the author of the document inserts one of these standard phrases, intending to replace the phrase as soon as the missing information becomes available.

This *Quantifier* also supports Traceability/References for Usability.

2.4.2.3.2 Percent Appropropriate

Percent Appropriate is the proportion of references which are useful or necessary to the total number of references.

This *Quantifier* also supports Traceability/References for Usability.

2.4.2.3.3 Percent Missing

Percent Missing is the proportion of missing references to the total number of references. A reference is considered missing if either

- the item referenced does not exist or
- a reference should be at a place in the text and is not there.

This *Quantifier* also supports Traceability/References for Usability.

2.4.2.4 Modification Tracking/Code: Sufficiency of System

Sufficiency of System is an evaluation of the computer system which maintains a history of the changes made to the source code. The assessment should account for the nature and characteristics of source code which are presented in section 2.3.2.3.1, on the *Factor* Modification Tracking/Code.

2.4.2.5 Modification Tracking/Documentation: Sufficiency of System

This *Quantifier* is similar to the one described in the previous section. The only difference concerns what it is tracking. This difference, however, affects the evaluation, since the nature and charac-

teristics of documentation are different than that of source code (see section 2.3.2.3.2 on the *Factor* Modification Tracking/Documentation).

2.4.3 Usability: Supporting Quantifiers

The *Quantifiers* which support each of the Factors of Usability are described in the following sections. For an illustration of the sub-tree structure, see Figure 19 on page 57.

2.4.3.1 Traceability/References

Several *Quantifiers* support the assessment of References, which in turn support Traceability. This characteristic is the same as for Referential Relationship, which supports Completeness, and so the *Quantifiers* are the same: TBD/TBS, Percent Appropriate, and Percent Missing. The way in which References aid Traceability has already been presented in section 2.3.3.1.1, but greater emphasis has been placed upon linking these *Quantifiers* to the *Quality* of Usability in this section.

2.4.3.1.1 TBD/TBS

Within the scope of Usability, the measure of TBD/TBS statements (see section 2.4.2.3.1) affects the serviceability of the documentation. The less complete a document is (i.e. the greater the number of TBD/TBS statements), the less useful that document will be.

2.4.3.1.2 Percent Appropriate

From the viewpoint of usability of the documentation, if a reference is inappropriate, then it is not useful.

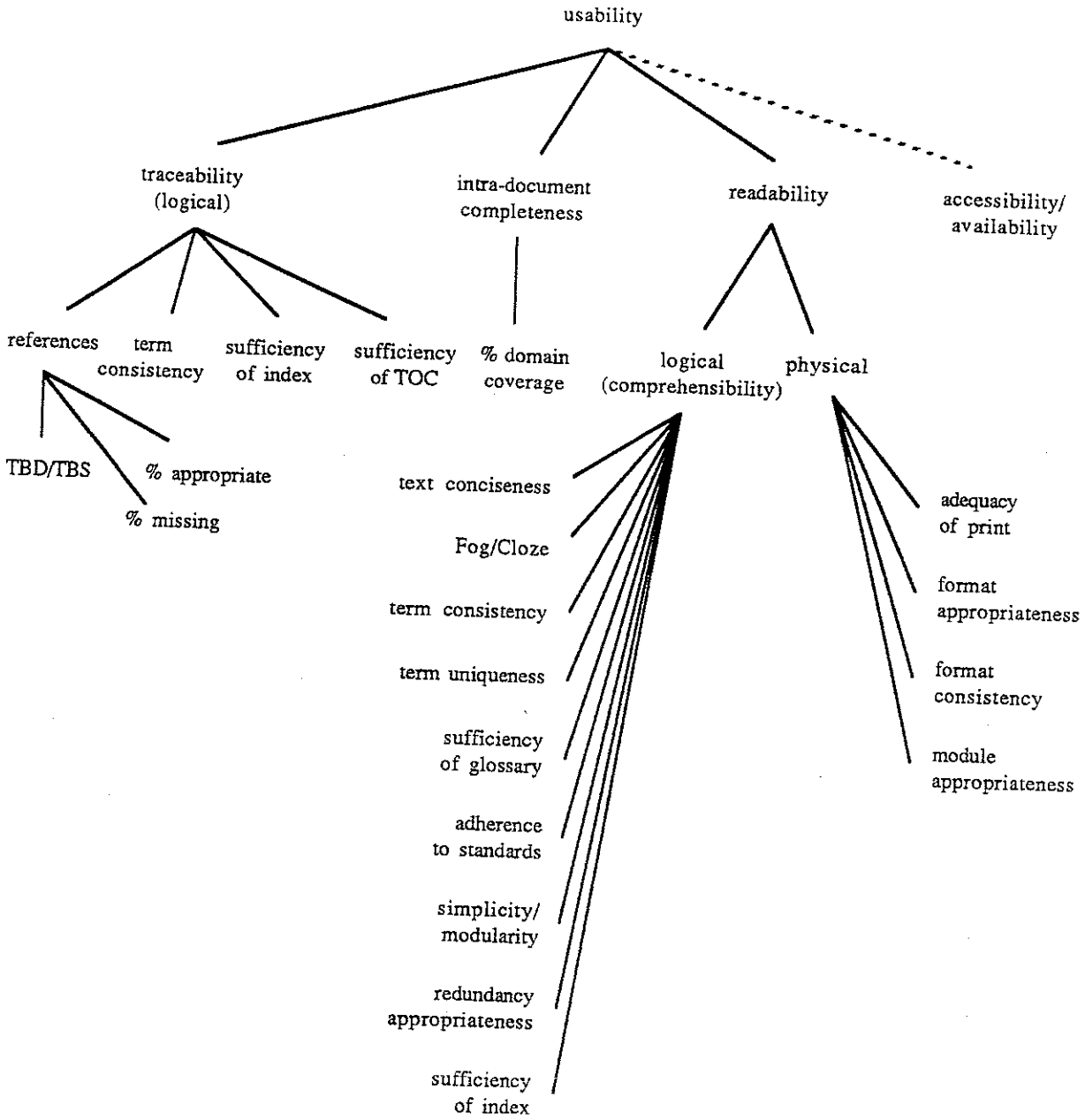


Figure 19. Quantifiers Which Support Usability

2.4.3.1.3 Percent Missing

The emphasis in this *Quantifier* is, again, on the Usability of the documentation. If a reference is missing, then, by definition, it is needed to make the documentation more understandable and thereby usable.

2.4.3.2 Traceability: Direct Support

The following *Quantifiers* directly support Traceability. (See Figure 19 on page 57.)

2.4.3.2.1 Term Consistency

Term Consistency is the use of the same word to describe the same concept. This aids Traceability because the user can search for occurrences of the same word when looking for related information.

To measure Term Consistency multiple samples must be taken for several terms. These terms can then be examined in context to determine whether they are used consistently. This process can be used to calculate a score, expressed as a percent, showing the number of times the terms are used consistently.

This *Quantifier* also supports Logical Readability for Usability.

2.4.3.2.2 Sufficiency of Index

A good index enables a user to find pertinent information quickly, allowing the user to go directly to a useful page. This *Quantifier* supports Traceability by assessing how easily users can find all occurrences of specific terms. It also can act as a "random reference," meaning that wherever a user is in the text, the index can be used to find additional information. Moreover, the index must not

only exist, it must be good; a poor index will not aid users in searches for terms. Therefore, the index must be evaluated to determine whether it is helpful in the search for terms.

A measure could consist of a set of questions for users. Questions that should be included should cover the following:

- Does an index exist?
- Is one needed?
- Is it sufficiently detailed (i.e. complete)?
- Are most terms specific to computer science or the given document?
- Is the format appropriate?
- Is the primary page reference high-lighted?
- Is it in the proper location in the document (e.g. in the back)?

The questions should be weighted and a score calculated. The score would be the value for this *Quantifier*.

This *Quantifier* also supports Logical Readability for Usability.

2.4.3.2.3 Sufficiency of Table of Contents

The Table of Contents (TOC) is useful for basically the same reason as an index. A good TOC allows the user to go directly to sections which are useful. Like an index, it must also be evaluated for its ability to aid the user, although the characteristics that make a TOC good are different than those which make an index good. An index is much more complex, in that it contains lists of words instead of a simple list of section headings. Another major difference is in the format of an index and a TOC. Part of the overall assessment is an evaluation of the format.

A measure could consist of a set of questions for users. Questions that should be included are the following:

- Does a TOC exist?
- Is one needed?
- Is it sufficiently detailed (i.e. complete)?
- Is the format appropriate?
- Does the format reflect the hierarchy of the sections?
- Is it in the proper location in the document (e.g. in the front)?

The questions should be weighted and a score calculated. The score would be the value for this *Quantifier*.

This *Quantifier* also supports Logical Readability for Usability.

2.4.3.3 Intra-document Completeness: Percent Domain Coverage

Percent Domain Coverage is the proportion of the existing sections and parts of individual documents to the total number of sections and parts called for in the template. "Existing sections and parts," in this context, means that they must contain reasonable information. This *Quantifier* cannot assess the total usefulness or completeness of the information in all of the sections, but a cursory examination of the contents of the sections should be conducted to ensure that the sections do not contain unrelated or nonsensical words or information.

2.4.3.4 Readability/Logical (Comprehensibility)

Logical Readability is supported by nine *Quantifiers*: Sufficiency of Index, Sufficiency of Glossary, Fog/cloze, Simplicity/Modularity, Redundancy Appropriateness, Term Consistency, Term Uniqueness, Adherence to Standards, and Text Conciseness.

2.4.3.4.1 Sufficiency of Index

This *Quantifier* is the same as described under Traceability for Usability. From the viewpoint of reading and using documentation, the index is required to access terms required for additional information.

2.4.3.4.2 Sufficiency of Glossary

This *Quantifier* is a measure of how well the glossary aids the user in reading and using the documentation. In the middle of reading a document, a user can turn to the glossary to get additional information or clarification for a particular term. A good glossary is easy to use and contains succinct definitions for each of its terms.

A measure could consist of a set of questions for users. Questions that should be included should cover the following:

- Does a glossary exist?
- Does it consist of only terms specific to this field or document?
- Are all of the necessary terms in the Glossary?
- Is the format appropriate?
- Is the Glossary in the proper location in the document (e.g. towards the back)?
- Are terms high-lighted?

- Are the terms ordered logically (e.g. alphabetically)?

The questions should be weighted and a score calculated. The score would be the value for this *Quantifier*.

2.4.3.4.3 Fog/cloze

This *Quantifier* is actually a collection of formulae or techniques which calculate the reading level of a given block of English text. The Fog Index calculates a reading level using a count of the number of words with greater than three syllables and the average average sentence length. Other formulas [DALE48, FLER48, KINJ75] calculate a reading level in similar ways, using average sentence length, average length of words in syllable, number of "uncommon" words, and so forth. The cloze procedure, on the other hand, is generally considered more reliable but less automatic [BREE81]. Typically, for the cloze procedure, every fifth word is removed from a passage of text. A reader is asked to fill in the blanks, using the remaining words and context.

Any one of these methods can be used individually or in concert with others. A combination of one of the Fog-type, automatic formulas and the cloze method would be best, since it would give two independent measures for this type of readability.

2.4.3.4.4 Simplicity/Modularity

Simplicity/Modularity is an assessment of the appropriateness of the divisions of the chapters, sections, subsections, paragraphs, figures, tables, and so forth. Each chapter, etc., should express only one central theme or idea; the documentation should be broken into simple modules.

To measure this *Quantifier*, the text must be sampled. The samples should be examined by users to determine if the samples express only one central idea or theme. A value could then be calculated as the percent of samples which are simple, i.e. contain one idea or theme.

2.4.3.4.5 Redundancy Appropriateness

Redundancy Appropriateness is an assessment of the suitability of the amount of repeated information or the average number of times information is repeated.

This *Quantifier* must be measured in a general way, using a manual technique. Users need to be questioned concerning whether

- the documentation is too voluminous
- some information could be referenced instead of being repeated
- previous knowledge should be assumed or not
- a word or phrase is overused
- the use of a standard form causes information to be repeated needlessly.

The questions should be weighted and a score calculated. The score would be the value for this *Quantifier*.

2.4.3.4.6 Term Uniqueness

This concept is the antithesis to Term Consistency; a word or term which can be defined in several different ways (including newly-coined definitions introduced by an author of documentation). Users can read the text more quickly and easily if they are not forced to determine which meaning, of several available, is being used for a term in the documentation. The meaning should be clear from the context or, if necessary, defined in the text.

To measure Term Uniqueness, multiple samples must be taken for several terms. These terms can then be manually checked to determined whether the definition is clear from the context in which it is used. This process can be used to calculate a score, expressed as a percent, showing the number of times that terms with multiple definitions are correctly used.

2.4.3.4.7 Term Consistency

This *Quantifier* is the same one as described under Traceability for Usability. The emphasis for this placement is, again, on the ease to the reader. If terms are used with a consistent meaning, it will make reading much faster and easier.

Regardless of the emphasis of the *Quantifier*, the measurement is the same. For a general procedure on how to measure this *Quantifier*, see Term Consistency under Traceability for Usability.

2.4.3.4.8 Adherence to Standards

Adherence to Standards simply allows the reader to know what to expect. If a set of standards has been established, then the documentation should adhere to these standards. This allows the user to know, in advance, what should be in the documentation, thereby making it easier for the user to understand the information which the documentation presents. This *Quantifier* is an assessment of the amount of consistent or proper use of the rules set down in the standards.

The measurement of this *Quantifier* can be accomplished by first determining whether standards exist. Then, assuming that they do exist, a percent is calculated for the number of samples which meet the standards.

2.4.3.4.9 Text Conciseness

Text Conciseness is an assessment of the concurrent completeness and briefness of the text in the documentation. It must be assessed with respect to the amount of text used to present the given information.

Manual evaluation is called for in this case. Currently, there is no way to automatically determine how much text is necessary to properly present a given concept. The text must be examined to determine whether part of the text could be removed and the relevant information still covered. Interestingly enough, the cloze concept presented earlier certainly appears to offer similar information but, unfortunately, no investigations of the adapting the cloze procedure to measure text conciseness are known.

2.4.3.5 Readability/Physical

Physical Readability is supported by the *Quantifiers* Format Appropriateness, Adequacy of Print, Format Consistency, and Module Appropriateness. Each of these is presented in the following sections.

2.4.3.5.1 Format Appropriateness

Format Appropriateness is used to assess the suitability of the presentation style or layout, i.e. the medium for the information. For example, numeric data should be presented in a table or graph. Even further, the type of graph - line, bar, or pie chart - varies with the type of data to be presented. Other information is best presented as prose. The format that is the easiest for the user to understand should be used.

Evaluating the appropriateness of the presentation format can be done using sampling. At the present time, the samples must be manually judged as to their appropriateness; in the future, it may be possible to automate this process.

2.4.3.5.2 Adequacy of Print

Adequacy of Print is a characteristic concerned with the ease with which the user can physically read a page of documentation. Aspects includes the quality of the reproduction, appropriateness of the font sizes, quality and appropriateness of paper, and appropriateness of font style. These aspects do effect the user's comfort and comprehension of the documentation; fuzzy print quality, poor paper, baroque typefaces and small print can make a document virtually impossible to use.

The assessment of the print can be done using sampling. The sample need to be examined and manually evaluated with respect to the characteristics listed above.

2.4.3.5.3 Format Consistency

Format Consistency is an assessment of the consistency with which formatting of the documentation is done. All tables, within and between documents, should have the same format; all indexes should have the same format. The same is true for all Tables of Contents, chapters, sections, and so forth. Consistent use of format prepares the user for easy assimilation of the information.

One way to measure this characteristic is through sampling. If standards have been established, then samples can be compared to the standard. Otherwise, several items of each format must be selected randomly and a consensus established. A set of important features must be enumerated, and a "standard" of a sort must be established for the purposes of the evaluation. The value for the Format consistency could be the percent of the items which meet the standard.

2.4.3.5.4 Module Appropriateness

Module Appropriateness is a measure of the suitability of the physical division of the chapters, sections, paragraphs, tables, figures, and so forth. Basically, this is a measure of the success of translating the Simplicity/Modularity of Logical Readability into a physical format. Does the physical layout of the documentation reflect the logical partitioning of the information? Is the documentation physically divided into chapter, sections, and so on in a way that is easy to use and understand? The divisions should, therefore, be based on what is easiest for the user to read.

3.0 An Application of the Assessment Procedure to ADDS Synthesized Documents

The previous section presents a detailed description of how one applies the DQI concept in assessing document quality. As presented, the assessment procedure is based on a taxonomic partitioning of desired document qualities, factors and quantifiers, and is designed to support the evaluation of general (or generic) set of documents. In concert with Tasks 2 and 3 of the Statement of Work, this section focuses on the application of that assessment procedure to documents generated by the Automated Design Description System (ADDS). In particular, this section describes

1. assumptions and related ramifications underlying
   - dictates of the Task statement, i.e. the evaluation framework,
   - the process of automatically synthesizing documentation, and
   - a document's intended User, Scope and Purpose,
2. the identification of DQIs that should be present in ADDS synthesized documents, and within this set
3. the identification of those DQIs for which no basis of measurement exists under the current ADDS approach to document synthesis.

Also discussed is the relationship between item (1) and items (2) and (3), i.e. the exploitation of results from (1) in providing an investigative foundation for (2) and (3).

3.1 Task Dictates Guiding The Framework for Evaluation

In relating the general evaluation taxonomy to documents generated by ADDS, the underlying tenet is that assumptions guiding document synthesis emphasize the importance of certain DQIs while de-emphasizing others. Assessing the extent to which each DQI contributes to the evaluation of documents generated by ADDS provides a basis for refining the evaluation. The first set of assumptions that provide refinement insights is associated with task dictates guiding the framework for the evaluation. More specifically, the Statement of Work stresses the analysis of documentation generated by ADDS. Restricting the assessment process to such documents implies that the quality of Expandability should be removed from consideration. That is, Expandability is not considered a crucial element in assessing the quality of documents generated by ADDS. The exclusion of this document quality is, in fact, natural because ADDS generated documents are not intended to be expanded; changes are introduced through the re-generation of documents.

A second element that can be excluded from consideration, also due to task dictates, is the quality factor of Accessibility. As described in Section 2, Accessibility is a significant factor in assessing the quality of document Usability. Nonetheless, general accessibility to ADDS documents by in-service engineers is assumed, and hence, should not play a role in the evaluation of such documents.

3.2 *Process Dictates Guiding The Framework for Evaluation*

ADDS supports a well defined process for the synthesis of documents tailored to the in-service engineering activity. This process includes the *building* of the PSL/PSA database and *translating* objects/relationships defined in the database to document format. Each document is automatically generated, and reflects a particular set of relationships among AEGIS system objects. The database of information from which the relationships are derived, is itself generated by an automatic process. Because synthesis of the information database and documents are founded on automated procedures, one must assume that the translation process by which ADDS generates documents is *accuracy preserving*. That is, all documents generated by ADDS are assumed to correctly reflect the system whose characteristics are captured in the ADDS (i.e. PSL/PSA) database. Based on this assumption, the document quality of Accuracy must also be removed from consideration in the evaluation of ADDS synthesized documents.

Additional evidence supporting the exclusion of Accuracy is the Task related dictate that only ADDS generated documents be considered in the evaluation. Implications of this directive is that only the product and not the process be considered in document assessment. If one assumes that the original source of information is correct, then accurately conveying that information in alternate forms is an exclusive function of the translation process. Hence, the directive to consider only the product in assessing the quality of an ADDS' document assumes as a bias the accuracy preserving assumption of the translation process mentioned above.

3.3 *The Effects of User, Scope and Purpose in Refining the Evaluation Taxonomy*

Figure 20 on page 73 depicts an initial refinement of the Evaluation Taxonomy based on (a) implications of the two Task related dictates mentioned above and (b) the "accuracy preserving" assumption of the ADDS translation process. When compared to the Taxonomy for general document evaluation, one observes that major portions of the tree have been eliminated, i.e. those portions denoting the document characteristics of Accuracy, Expandability and Accessibility. The motivation for excluding these document characteristics from an overall evaluation scheme, however, is based on pragmatic decisions independent of actual characteristics inherent to ADDS' synthesized documents. For example, removing Accuracy from assessment consideration is based on the assumption that the synthesis process is accuracy preserving. In effect, document accuracy is assumed. Based primarily on Task Statement directives, similar arguments can be given for the exclusion of Expandability and Accessibility. Although not stated explicitly, other singleton *Quantifiers* have also been removed from consideration due to implications of task-related dictates, e.g. Fog/cloze. They are specifically enumerated in Figure 22 on page 79.

In addition to the above, if one also considers the User of a document, the documents' Scope and its intended Purpose, then further refinement of the General Evaluation Taxonomy is possible. To fully understand how user, scope and purpose affects that taxonomy (as applied to documents generated by ADDS) one must first recognize the relationships among them. Intuitively, the user of a document is the person who consults that document for information leading to a better understanding the documents' subject. In the domain of systems engineering, that user can span a spectrum from high-level management to in-service engineers and "trench-level" programmers. The scope of a document can be viewed as its range or field of information. For example, a single document might describe all system components related to threat response. The scope of another

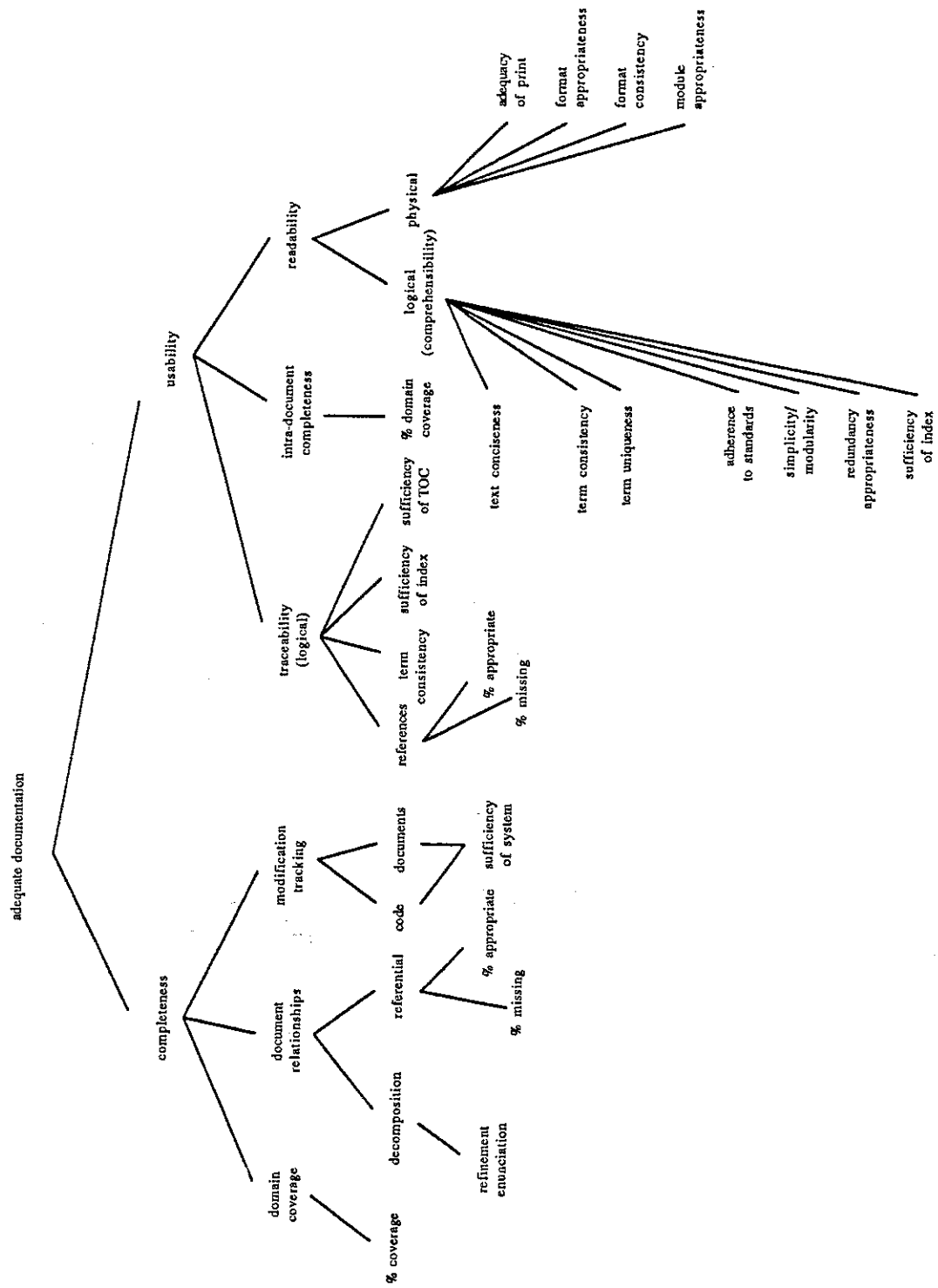


Figure 20. A Refinement Based on the Task Statement

document, however, might be restricted to only a single threat response component like the Harpoon system. Purpose, on the other hand, is a document attribute reflecting the level of detail at which the document is to present the information within its scope. For example, a high-level description of a document's subject matter is perceived as an overview. Hence, the purpose of such a document is to provide a high-level overview of the subject.

3.3.1 An Explication of User, Scope and Purpose

As mentioned above, the user of a document, a document's scope and its purpose are all related. More specifically, user and purpose are directly bound together, whereas scope remains somewhat independent of the other two. Consider, for example, a document's scope. If the scope is to provide information about a particular subject, the concepts presented will remain relatively invariant with respect to different users and purpose. The level of detail presented might change, however, but the subject matter, i.e. the document's scope, will remain constant. This observation leads to the following question:

What influences a document's level of detail?

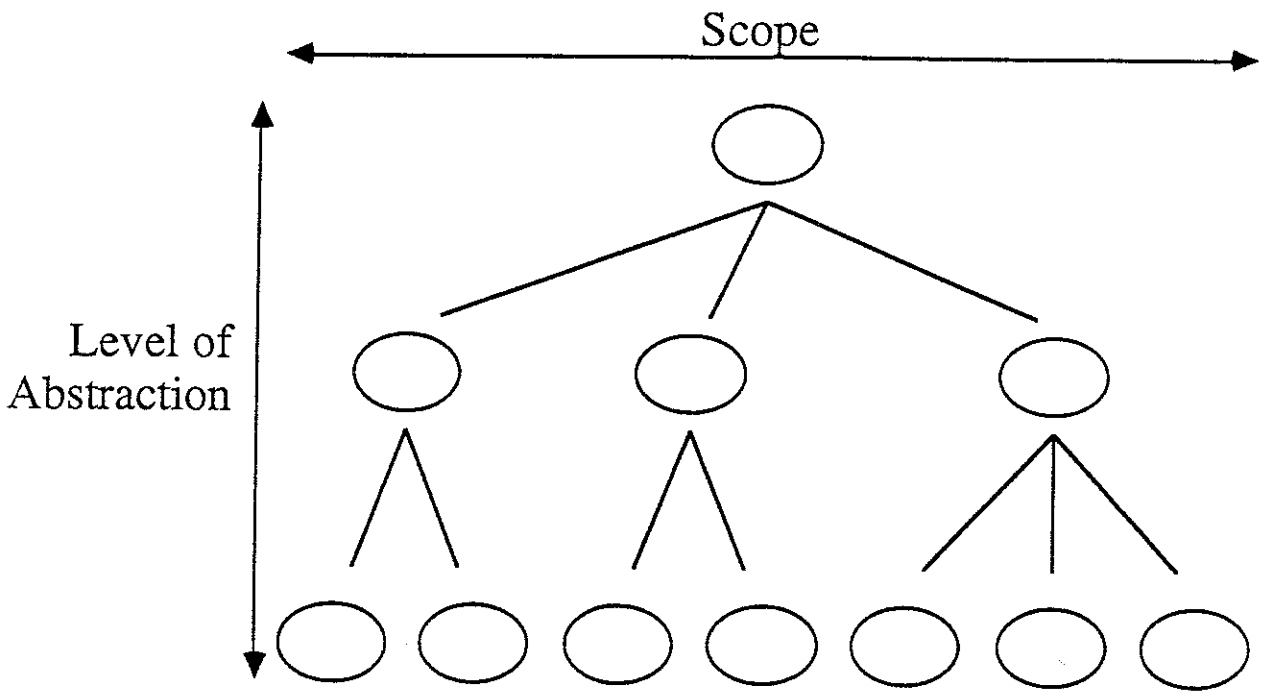
An investigation of this question reveals an important relationship between the user of a document and the purpose of that document, that is, the intended user of a document directly effects its purpose. As an example, consider the needs of a high-level manager. In presenting technical subject matter, a most appropriate document purpose is that of an overview. This statement is motivated by the fact that high-level management personnel more frequently require breadth of knowledge rather than technical depth to make decisions. As one moves down the management level continuum toward persons that work more closely with the subject of a document, e.g. a software system, a more detailed level of explanation is desired. In particular, if one considers in-service engineers addressing changes to a software system, then one expects their needs to to be oriented around documents that provide meticulously detailed system information. In general, one does not

expect a high-level manager to request a document filled with technical details nor an overview document to be deemed as valuable to an engineer.

Figure 21 on page 76 illustrates the relationships among user, scope and purpose. The horizontal arrow represents the the breadth of information included in the set(s) of documents. In effect, this breadth is the scope of the document(s). As Figure 21 on page 76implies, the scope of a document set is fixed. At a high level of abstraction, however, one expects to find a document providing a global view of the system where each component is individually outlined. Refining the level of abstraction, one then expects to find documents focussing on individual system components and tailored toward middle management. Moving toward more refined levels of abstraction, one eventually encounters document sets that provide specific details relating information from many perspectives. (See Figure 21 on page 76).

If, for example, one considers a set of documents presenting information from an engineering perspective, at the top of the document hierarchy, one would find an overview of the entire system and associated requirements. The next level might be the high-level design documents; then comes the low-level design documents, and so forth. At the bottom of the hierarchy one expects to find actual code modules. Note that the level of detail proceeds from general to specific (abstract to detailed) as one moves from the apex of the document hierarchy illustrated in Figure 21 on page 76 to its lower elements. Obvious also is the influence on user and purpose as one moves down the hierarchy. In particular,

- from a user's perspective, high-level management is most interested in the more abstract documentation that provides a general system overview; in-service personnel, however, are most interested in the detailed documents at the bottom of the hierarchy and needing only minimal access to the high-level documentation;



User: Related to Expected Level of Abstraction

Scope: Fixed (Entire System)

Purpose: Tied to User

Figure 21. The Relationships Among User/Scope/Purpose

- from the view of a document's purpose, as one moves down the document hierarchy one finds the purpose of each document is to present more detail about a more limited scope.

3.3.2 A Taxonomic Refinement Based on User, Scope and Purpose

As discussed above, document characteristics vary with their defined purpose and intended user. Although a scope may vary among documents, when considering a single set of documents for evaluation, the scope is fixed. In a effort to further refine the Evaluation Taxonomy based on an improved understanding of the interrelationships among documents well as its intended user and its purpose, we must address the following question:

> Relative to a document's user and purpose, what is significant about the in service engineering activity that either mandates the presence of or relaxes the need for particular document qualities, factors and quantifiers?

By considering the "conventional" in-service engineer and the attendent activities one realizes that

- a set of documents is needed whose scope focuses on the entire software system aboard an AEGIS ship, and that
- the presentation level must be sufficiently detailed to enable a understanding consistent with needs of a maintenance programmer.

Bearing in mind that our primary goal is the assessment of documents generated by ADDS, the additional document characteristics implied above does provide an insight for further refinement of the Evaluation Taxonomy. For example, most of the detail in documentation generated by ADDS consists of information concerning calling hierarchies, variable scope and definitions, and other data about the source code. These types of documents neither require nor need conventional document elements like indices, table of contents, or explicit references to other documents. Subsequently, such elements should not be considered in the evaluation of documents synthesized from

ADDS. Moreover, other quality factors must be consider in relation to such documents, e.g. domain coverage.

Applying the impact of User/Scope/Purpose to documents synthesized through ADDS provides an additional refinement to the Evaluation Taxonomy. In summarizing that impact, Figure 22 on page 79 outlines those document elements that

- are no longer appropriate for consideration in the evaluation of documents synthesized through ADDS, e.g. the existence of document decompositional relationships, or
- require a "re-thinking" as to how User/Scope/Purpose affects their contribution to the evaluation process, e.g. domain coverage, or
- are not affected at all by User/Scope/Purpose, e.g. term consistency.

Finally, Figure 23 on page 80 illustrates the newly refined Evaluation Taxonomy Tree that should be applied in the evaluation of documents generated from ADDS. For more precise statements as to why specific document elements have been removed from evaluation consideration, the authors refer the reader to [STEK88].

A Refinement Summary of the General Evaluation Taxonomy

Document Evaluation Element	Status	
Accuracy	Removed	(2)
Completeness		
Domain Coverage	Affected	(3)
Document Decomposition Relationship	Removed	(3)
Document Referential Relationship	Affected	(3)
Code Modification Tracking	Unaffected	
Document Modification Tracking	Removed	(3)
Usability		
Logical Traceability through		
References	Affected	(3)
Term Consistency	Unaffected	
Index Assessment	Removed	(3)
Table of Content Assessment	Removed	(3)
Inner-Document Completeness	Affected	(3)
Logical Readability through		
Text Conciseness	Unaffected	
Fog/Cloze	Removed	(1)
Term Consistency	Unaffected	
Term Uniqueness	Unaffected	
Glossary Assessment	Removed	(1)
Adherence to Standards	Unaffected	
Simplicity/Modularity	Unaffected	
Redundancy Appropriateness	Unaffected	
Index Assessment	Removed	(1)
Physical Readability through		
Assessment of Print	Unaffected	
Format Appropriateness	Unaffected	
Format Consistency	Unaffected	
Module Appropriateness	Unaffected	
Accessibility	Removed	(1)
Expandability	Removed	(1)

Refinement Criteria Inducing Status

(1) Task Statement Framework
(2) Process of Document Generation
(3) User/Scope/Purpose

Figure 22. A Refinement Summary of the General Evaluation Taxonomy

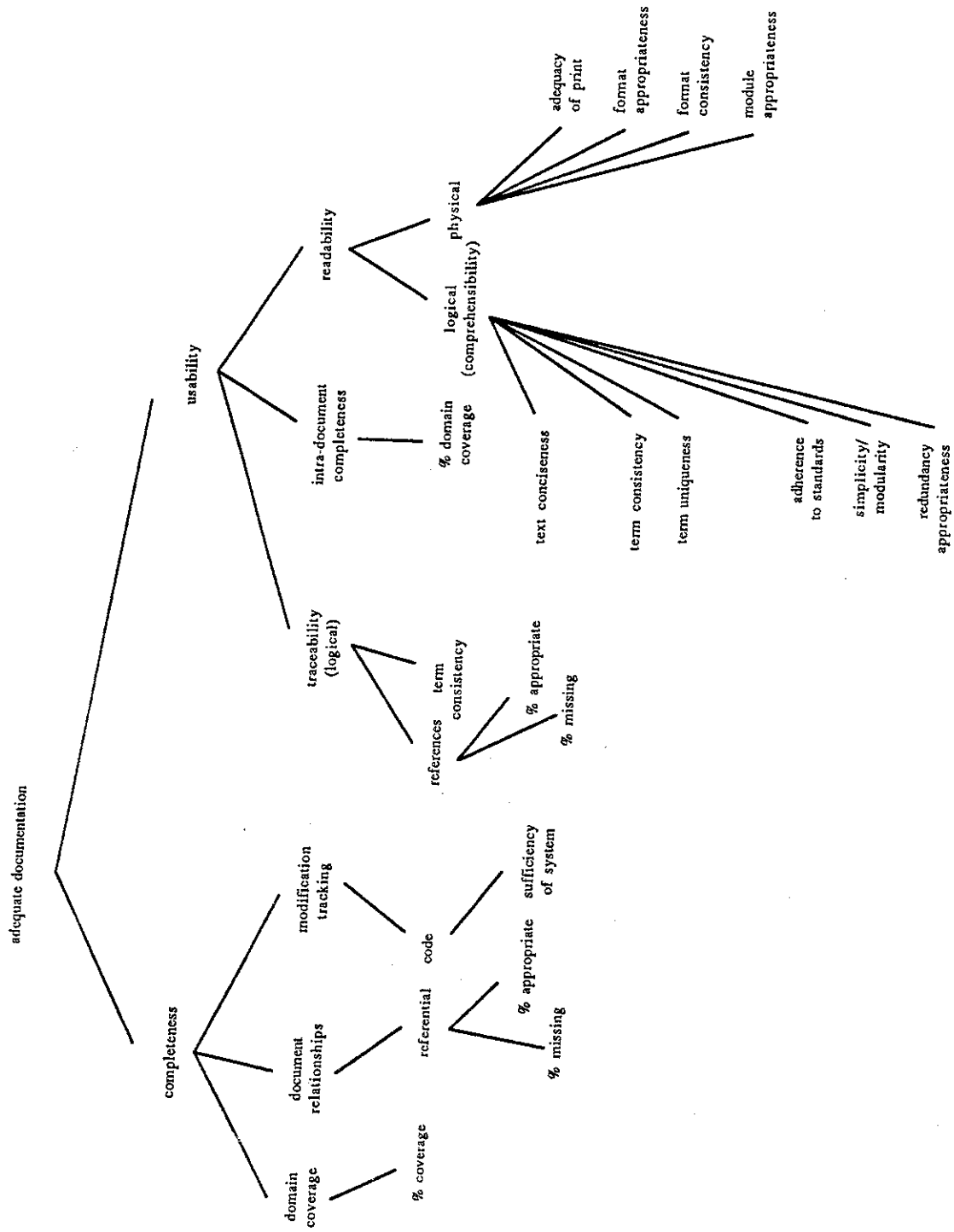


Figure 23. A Taxonomy for Evaluating Documents Synthesized by ADDS

3.4 *Relating the Refined Taxonomy to the ADDS System*

Tasks 2 and 3 of the Statement of Work focus on deriving benefits from the General Evaluation Taxonomy (see Figure 23 on page 80) through modifications and enhancements to ADDS. More specifically, these two tasks combine to focus research directions on

- the identification of DQIs that should be measurable in documents generated from ADDS,
- the identification of DQIs that are and are not currently measurable, and
- methods for establishing a basis for measuring those missing DQIs.

The remainder of this section individually addresses each of these items.

3.4.1 Expected DQIs

Section 2 presents research results culminating in the identification of a well-defined set of DQI applicable to the evaluation general documentation. Initial steps in applying the DQI concept to the evaluation of documents generated by ADDS is

- the identification of crucial characteristics inherent to the domain in which ADDS' synthesized documents are used, i.e. an in-service engineering environment, and
- an exploitation of the implications of those characteristics through successive refinements to the general evaluation framework, leading to an evaluation procedure tailored to the assessment of documentation automatically generated for in-service engineers.

Sections 3.1 through 3.3 outline research directions, assumptions and justifications in achieving the above. One result of those assumptions is the refined evaluation taxonomy presented in Figure 23 on page 80. More specifically, the refined taxonomy implies that a basis should exist within the domain of ADDS for measuring the following DQIs:

Relative to Completeness:

- Domain coverage
- Referential relationships among documents, and
- Tracking of code modifications,

Relative to Usability:

- Intra-document completeness

Relative to Usability based on logical traceability:

- References, and
- Term consistency,

Relative to Usability based on logical readability:

- Text conciseness,
- Term consistency,
- Term uniqueness,
- Adherence to standards,
- Content simplicity and modularity, and
- Redundancy appropriateness, and

Relative to Usability based on physical readability:

- Adequacy of print,
- Format appropriateness,
- Format consistency, and
- Modularity appropriateness.

The authors note that for the presentation clarity the first five (5) elements listed above are actually Quality factors, i.e. document Qualities and related Factor. Quantifiers, the third component of any DQI, are also necessary for actually evaluating the adequacy of documentation. For instance, one indication of document completeness with respect to domain coverage is the extent to which the document (or set of documents) covers the stated standards and/or subject matter. Section 2.4 presents all quantifiers in relation to their corresponding Quality/Factor partners; Figure 23 on page 80 illustrates the complete set Quality/Factor/Quantifier relationships comprising all DQIs applicable to the evaluation documents generable from ADDS.

3.4.2 DQIs Not Found Within the ADDS Framework.

Relative to the assessment process and documents generated by ADDS, each DQIs mentioned above is expected to have an identifiable basis for measurement. That is, one should be able to assess the extent to which each DQI is present (or absent) based on defined metrics that exploit specific document characteristics. The inability to identify such a basis for one or more DQI indicates a deficiency in the process by which documents are created. Note that "process" is emphasized here rather than "product" because, if the basis for assessment is dictated by the synthesis process, then measurements are possible. An assessment of the product might indicate the absence of a DQI, but an assessment (and hence, a measurement) is still possible.

Based on an evaluation of the ADDS system and a representative subset of documents generated from that system [ARTJ88], a comparison of those findings with the refined evaluation taxonomy illustrated in Figure 23 on page 80 leads the authors to conclude that ADDS is deficient with respect to only one DQI: documentation completeness as measured through the sufficiency of the code modification tracking system (Completeness/Code-Modification-Tracking/Sufficiency of System).

3.4.3 A Suggested Extension to ADDS

Given that

- the Statement of Work precludes the consideration of the Expandability and Accessibility
- the document synthesis process is "accuracy preserving", (and hence obviating the need to consider Accuracy in assessing the quality of documentation generated by ADDS),

the findings presented in the previous section suggest an extension to ADDS that can exploit the benefits derived from a code modification tracking system, i.e. a source code control system (SCCS). An SCCS system can provide access to a history of source code changes from which additional, desirable documents can be generated. For example, additional reports might include information reflecting changes made to a particular module, who made them and what was the rationale for such changes. Access to and the generation of documents from a history of source code modifications significantly enhances overall documentation completeness, and hence, the adequacy of a documentation set.

4.0 Summary and Concluding Remarks

Sections 2 and 3 of this report present a detail description of the authors' research findings. The remainder of this report

- summarizes those findings relative to the Statement of Work,
- discusses the findings in relation to earlier work reported in [ARTJ88],
- presents several concerns stemming from assumptions made during this research effort, and
- outlines future research directions.

4.1 A Task Summarization

Motivating the research effort described in this report is the recognition that system and project documentation are crucial for high quality software development and maintenance. Implicit in this recognition is that documentation must be accurate, complete and usable. The Automated Design Description System (ADDS) is a product that addresses a need for the generation of improved and

more complete documentation through an automated process utilizing reverse engineering concepts. The primary focus of this investigative effort is an assessment of ADDS relative to its ability to produce documents that exhibit DQIs, and subsequently, provide a basis for quantitatively measuring the quality of the generated documents, both individually and collectively. The three tasks comprising this assessment effort are described below.

4.1.1 A Summarization of Task 1 Findings

The major emphasis of Task 1 is the investigation of the ability of documentation standards to promote, encourage or enforce adequate documentation. Intuitively, Task 1 suggests a research direction focusing on a well-defined framework for documentation analysis. Such an effort necessitates (a) a review of current literature, (b) the identification of document characteristics amenable to analysis and (c) an investigation and formulation of document quality indicators (DQIs) that can serve as a basis for the quantitative assessment of such characteristics.

In concert with Task 1 and reflecting an extensive literature review, Section 2 identifies several characteristics that "adequate" documentation must possess, e.g. accuracy, completeness and usability. During the course of the research, a categorization of documentation properties emerged: *Qualities*, *Factors*, and *Quantifiers*. Based on recognized relationships among *Qualities*, *Factors* and *Quantifiers*, Section 2 describes a evaluation framework for assessing the extent to which desirable characteristics are present in a document. The framework reflects an undeniable set of linkages among document *Qualities*, *Factors* and *Quantifiers*. In turn, the assessment process exploits these linkages in the evaluation of DQIs, formally defined as a *Quality/Factor/Quantifier* triple. Fundamental to the assessment process is a recognition that

- each *Quantifier* is a non-abstract, measurable document characteristic, and that
- Factors provide the bridge relating *Quantifier* measurements to the evaluation of documentation *Quality*.

In summary, document characteristics crucial to assessing the adequacy of documentation are identified. An assessment procedure is then outlined which recognizes the importance of those identified characteristics through the employment of DQIs.

4.1.2 A Summarization of Task 2 Findings

Task 2 focuses on applying the findings of Task 1 to ADDS. In particular, Task 2 stresses an examination of documents generated by ADDS with a concentration on identifying those DQIs for which a basis for measurement exists. An expected complementary finding is the identification of those DQIs for which no basis for assessment exists.

In addressing Task 2, the major research objective is a refinement of the general evaluation taxonomy to reflect a more precise assessment of documents automatically generated by ADDS. Refinements in the general taxonomy are guided by three following concerns:

- task dictates guiding the framework for evaluation,
- process dictates stemming from automatic document synthesis, and
- refinement dictates based on the identification of a document's user, a document's scope and its scope.

Section 3 of this report provides a detailed description of the refinement process driven by the guiding concerns mentioned above. Findings indicate that documents generated from ADDS should possess characteristics that provide for the assessment of 18 DQIs. Based on a previous

study of ADDS, however, the capability for assessing only 17 DQIs currently exists. The implications of this discrepancy underlie the research directives of Task 3.

4.1.3 A Summarization of Task 3 Findings

Results of Task 2 indicate that documents produced through ADDS lack a basis for assessing one DQI, i.e. (Completeness / Code Modification Tracking / Assessment of System Sufficiency). In recognizing that such a situation could occur, Task 3 of the Statement of Work is structured to emphasize the exploration of requirements needed to provide for the identification, synthesis and measurement of DQIs currently missing from documents generated by ADDS.

As discussed in Section 3.4.3, research findings indicate that ADDS should be extended to exploit the information database associated with a code modification tracking system. Such an extension to the ADDS document synthesis process can provide a basis for the assessment of the absent DQI.

4.2 Relating the Evaluation Taxonomy to Previously Identified Documentation Inadequacies

If the evaluation taxonomy is sufficiently powerful, then one should minimally be able to identify those DQIs that reflect the document deficiencies discussed in [ARTJ88]; that is, those deficiencies already identified in documents generated by ADDS. Figure 24 on page 90 outlines those deficiencies, indicates whether the deficiencies are process or product related, and identifies the DQI(s) that support the recognition of such deficiencies. For example, the presence of "stale" (non-current) data is a process related deficiency that can be detected through those DQIs associated with Accu-

racy. Documents exhibiting confusing (database) terminology are susceptible to deficiency detection through several DQIs, viz. those associated with Accuracy as well as Term Consistency and Term Uniqueness in reflecting Usability through Logical Readability. The authors note, however, that only principal DQIs supporting the detection of the enumerated document deficiencies are identified. Arguments can certainly be given to support the inclusion of other DQIs. Such arguments are consistent with the authors' contention that a reliable assessment of document adequacy is achieved through the evaluation of many complementary DQIs.

4.3 Statement of Concerns

The investigation described in this report is the second of three research efforts. The first effort involved (a) an in-depth study of ADDS relative to the products and the production process, and (b) a sequence of interviews with users who have familiarity with ADDS. This second effort focuses primarily on the evaluation of documentation, and in particular, documentation produced by ADDS. The third proposed effort addresses the composition of an "ideal" reverse engineering approach to document synthesis. Complementary findings of the first two research efforts foster several issues (or concerns) that require discussion. The following subsections outline those concerns relative to an overall evaluation of ADDS.

4.3.1 Assessing the Process (and the Product)

Refining the evaluation taxonomy to support the evaluation of documents generated by ADDS follows the assumption that only the product (i.e. documents synthesized by ADDS) need be assessed. As described in Section 3 and as outlined in Section 4.2, the refined taxonomy does provide

Document Deficiency	Process/Product	Identifying DQI Component
Erroneous Information		
* Non-current information database	Process	(1)
* Overlay problem	Process	(1) (5)
Inadequate Information		
* Insufficient titles and description	Product	(6)
* Need for contextual information	Product	(3)
* Confusing (database) terminology	Process	(1) (7) (8)
* Reflecting design decisions	Process	(2) (4)
* Document accessibility	---------	(9)
Omitted Information		
* Affecting relationship among variables	Process	(2)
* Globally accessed variables	Product	(2)
* Variable name decoding	Product	(2)
* Variable cross-referencing among SYS-PROCS, Procedures, etc.	Product	(2)

DQI Legend:
(1) => Accuracy
(2) => Completeness / Domain Coverage
(3) => Completeness / Referential Document Relationships
(4) => Completeness / Modification Tracking
(5) => Usability / Logical Traceability
(6) => Usability / Inner Document Completeness
(7) => Usability / Logical Readability / Term Consistency
(8) => Usability / Logical Readability / Term Uniqueness
(9) => Usability / Accessibility

Figure 24. DQIs and Deficiencies Identified in ADDS Synthesized Documents