Requirements for a Software Maintenance Methodology

By Richard E. Nance, James D. Arthur, and John A. Ciaramella, Jr.

TR 91-21

FINAL REPORT
(Appendices Not Included)

*Cross-referenced from Technical Report SRC-91-003; Systems Research Center; VPI&SU; Blacksburg, VA 24061; 17 May 1991. Work supported by the Naval Surface Warfare Center under contract number N60921-83-G-A165 B048. This report is not to be reproduced nor cited, in total or in part, without the written permission of the authors. For information on obtaining appendices, please contact the authors at the address listed above.

ABSTRACT

This project ventures into the domain of software maintenance methodologies, an area given relatively little attention in software engineering. Project efforts are divided into three tasks:

(1) definition of maintenance methodology requirements,
(2) development of a model of the AEGIS maintenance process, and
(3) specification of the requirements for an AEGIS maintenance methodology.

The focus of this report is on the third task, performed in the period 1 July 1990 – 20 May 1991.

The research utilizes the Objectives/Principles/Attributes (OPA) framework developed for software quality assessment for time-critical, embedded systems. The overriding maintenance objective is to "realize desired changes in an efficient and effective manner." Four principles are identified that support achievement of this objective. Principles identified in the OPA framework are also applicable. As a consequence, eleven (11) requirements are derived for an AEGIS maintenance methodology.

The AEGIS maintenance process model is used to determine the potential points where any of the requirements can be met. Recommendations are made with respect to restructuring the process, noting the most appropriate place for meeting requirements, and acquisition or development of software utilities to support maintenance. Recommended as an approach for meeting long-term needs is the AEGIS System Evolution Environment, supporting both systems and software maintenance activities.

[Appendices Not Included]

CR Categories and Subject Descriptors: D.2.1 **[Software Engineering]**: Requirements/Specifications; D.2.7 **[Software Engineering]**: Distribution and Maintenance.

General Terms: Documentation, Management.

Additional Key Words and Phrases: Methodologies, reverse engineering, model, maintenance.

AEGIS Maintenance Methodology

1. Introduction

This is the final report for the project entitled, *Development of an AEGIS Maintenance Methodology* (N60921-83-G-A165-B-048). The project is separated into three tasks:

1. Definition of maintenance methodology requirements, investigating the differences in approaches, techniques, and methods that contrast the maintenance and development phases of the software life-cycle.

2. Development of a model of the AEGIS maintenance process, based on earlier and on-going work by NAVSWC and contractors, and through active guidance and participation of an Advisory Panel.

3. Specification of the maintenance methodology based on general requirements for maintaining time-critical embedded systems but also recognizing specific needs of the AEGIS application domain.

The report focuses on the third and final task of the project which covers work performed in the period 1 July 1990 - 20 May 1991.

2. Background

In preparation for Task 3, Task 1 provides a definition of maintenance methodology requirements and Task 2 develops AEGIS maintenance process models using both detailed and aggregated representations. A brief review of prior work is given below to provide a context for understanding the most recent effort.

2.1 Task 1: Definition of Maintenance Methodology Requirements

The foundation for the Task 1 research is from an earlier project exploring software development documentation potentials using current technological advances, such as

hypermedia, mass storage technology, and knowledge representation. The final report for that project, "Documentation Production and Analysis Under Next Generation Technologies," (N60921-83-G-A165 B043-01) proposes the Abstract Refinement Model (ARM) as a means for depicting iterative refinement of development documentation in the specification of a software system. The critical importance of development documentation in the early maintenance activities is demonstrated by the ARM. The ARM also allows one to characterize the objectives of reverse engineering more explicitly than any model to date. Linking maintenance to development more directly than prior models, the ARM underscores the necessity for a software maintenance methodology for time-critical embedded systems.

2.1.1 Literature Search

To bring such a methodology into existence for the AEGIS Combat System, the approach in Task 1 relies on an extensive analysis of software engineering literature, seeking to

(1) compare and contrast the activities of development and maintenance,

(2) recognize explicit treatments of software maintenance for time-critical embedded systems, and

(3) derive requirements for an "ideal methodology" that can form the target for a process model and an AEGIS maintenance methodology (to be developed by the Advisory Panel).

To derive requirements for an "ideal methodology," numerous definitions of maintenance are considered, and a categorization of maintenance forms is presented. Notably absent in the software life-cycle models is the depiction of an explicit linkage

between development and maintenance activities. A detailed explanation of life-cycle model deficiencies can be found in the subtask 1 interim report.

2.1.2 Requirements and Principles: OPA Perspective

In conjuction with complementary research findings presented by other authors, the principles defined by the OPA approach provide significant insights into what one might expect from a maintenance methodology:

(1) The objectives of a particular methodology are, to a large extent, reflective of requirements imposed by a set of system specifications. Within the maintenance framework, for example, a requirement to provide access to development documentation commensurate with the maintenance form underlies an objective of realizing desired changes in an efficient and effective manner. Identifying requirements and recognizing their role within a methodological framework is crucial because one must be able to assess the impact of constraining or sacrificing a particular requirement. The significance of this statement becomes even more evident when one considers that a single requirement often impacts several objectives.

(2) Recognizing the impact of requirements on methodological objectives is not, in and of itself, sufficient to define or guide a maintenance activity. Because maintenance tasks within the time-critical embedded systems domain can have wide-spread impact, a well-defined, systematic approach to performing maintenance activities is essential. Moreover, that approach must recognize, encourage and support the achievement of system-wide objectives through the enunciation of complementary methods and techniques.

(3) Intrinsic to a maintenance methodology, and providing the driving force behind many of the maintenance activities, is a set of principles that must be employed to achieve the objectives emphasized by that methodology.

(4) Finally, the OPA framework provides a basis for arguing the importance of a well-defined, systematic approach to performing maintenance and of the crucial role of principles in supporting the maintenance process. One should recognize, however, the significance of principles in the shaping of the maintenance environment. The effective employment of methodological principles often requires the use of tools with specific capabilities. A study of such principles should yield requirements for tool capabilities, and subsequently, assist in the identification and selection of tools appropriate for the maintenance environment.

The OPA approach provides a framework in which maintenance requirements can be expressed. The elements used are methodology *objectives*, the *principles* by which the objectives can be achieved, and the *attributes* of the resultant system.

The objectives and attributes for a maintenance methodology correspond closely to those for development. One new objective can be stated as "realizing desired changes in an efficient and effective manner," the primary objective of maintenance, which the methodology must promote.

The fundamental maintenance objective - effective and efficient change - leads to the identification of four new principles:

Scope Delineation- The initiation of every task should be the identification of bounding (document) components.

Varied Abstraction - Representations that support multiple levels of abstraction and the transitions among them should be utilized in the maintenance process.

Change Propagation - Recognition of the need to propagate specification changes through multiple levels of abstraction (i.e., throughout the maintenance document set).

Quantification with Abstraction Resolution - Quantification of the product quality should be a constant goal: the potential for quantification is inversely related to the level of abstraction.

2.1.3 Requirements for a Software Maintenance Methodology.

In concert with the principles enunciated in section 2.1.2, requirements for a software maintenance methodology are derived. The requirements are built upon the insight provided by the OPA framework and work toward the fundamental maintenance

objective. Each requirement is stated with a description and reference to the governing principles. Also included are the heuristic rules for applying them to the process model.

1) *Access to development documentation commensurate with maintenance forms.*
Principle: Varied Abstraction.
Purpose: A mechanism for accessing development documentation is the first need, but secondly, this access should be guided by the form of maintenance. The need for accessing specific levels of documentation for particular maintenance forms is described by the Abstraction Refinement Model [NANR89].
Rule for Application: This requirement applies to any activity which utilizes development documentation.

2) *Provide for decisions which maximize product availability (consider system availability).*
Principle: Varied Abstraction
Purpose: This requirement relates to the global objective of the maintenance process being both effective and efficient. The methodology should seek to ensure that quality is not sacrificed but changes are accomplished efficiently.
Rule for Application: This requirement might be interpreted in two ways (neither is incorrect). First, the decisions could minimize time spent in the maintenance process. Second, the decisions could maximize product quality, and in particular, reliability. Both should be combined since neither alone is adequate. However, the application of this requirement is necessary where decisions of either type are made.

3) *Each modification activity should include the following sub-activities:*
(a) Identify source and target (order is source dependent).
(b) Define and effect transformation process.
(c) Record source, process, and target.
(d) Test:
(1) Identify original test specifications
(2) Modify original test specifications for target.
(3) Revise test procedures and apply them.
Principles: Varied Abstraction, Scope Delineation, Change Propagation.
Purpose: The major steps in making a modification are prescribed. Actually making the modification requires the identification of the source of the maintenance actions, the target of the actions (a solution) and the means for realizing that target. Complementing these activities is the recording of the source, target and the process (including design decisions) to ensure compatibility of the documentation with the programs. Finally, testing is required to ensure that the correct target is achieved.
*Rule for Application:*This requirement governs the modifications made during maintenance, and so impacts these activities. It is not necessary that each of the parts of the requirement occur in sequence with no separating activities.

4) *Promote the identification of alternatives, the evaluation of alternatives (risk assessment) and support documentation of both.*
Principle: Varied Abstraction.
Purpose: The maintenance (and development) activities involve the consideration of a number of alternatives. The recording of these alternatives and their evaluation provide a justification of the selected change strategy that may be useful for later maintenance.
Rule for Application: Many alternatives arise in maintenance. The ones governed by this requirement should be those which impact the system at a lower level of decision making. In general, these would be alternatives with more than two choices and would not be of the nature "does the system pass this test?"

5) *Recognize and resolve potential interference among concurrent maintenance activities*
Principle: Scope Delineation.
Purpose: Maintenance activities might require changes which interfere or interact in some way. Recognizing and handling this interference helps ensure that combinations of modifications have a positive result. Otherwise, these modifications might combine to form an undesired change or one that is obviated by a subsequent change.
Rule for Application: This requirement applies in the initial consideration of a modification.

6) *Require and facilitate auditing of the maintenance process (metrics and methodology).*
Principles: Quantification with Abstraction Refinement.
Purpose: An audit of the maintenance process measures the success of the process in terms of product quality. The methodology should support auditing to guarantee the correctness of the process and the quality of the product.
Rule for Application: This requirement impacts all activities which involve preparation for and evaluation of the maintenance process.

7) *Support (enforce) uniformity in maintenance process/activity (procedure, documentation).*
Principles: Change Propagation, Varied Abstraction.
Purpose: Providing for uniformity in the maintenance process means that the procedures used in the maintenance process are the same. The impact of uniformity in the process is uniformity in documentation. Uniformity in documentation increases the maintainability of the system by facilitating understanding of the system.
Rule for Application: This requirement should apply to all activities of the process.

8) *Enable prioritization and coordination of maintenance forms and activities.*
Principle: Change Propagation.
Purpose: Various maintenance forms attach not only a practical order, but also a theoretical order to tasks. The practical order captures the necessity for certain modifications following sequentially. The theoretical order implies that

certain tasks should be performed first for efficiency and effectiveness. The methodology needs to recognize the ordering in the decision-making process.
Rule for Application: This requirement influences planning and scheduling activities, but also the initial consideration of a problem (i.e. CPCR).

9) *Enforce recording of source, process, target, test documentation, decision alternatives, evaluation, and final decision.*
Principles: Scope Delineation, Change Propagation.
Purpose: To provide for the documentation necessary for future maintenance.
Rule for Application: This requirement enforces the recording of information from a variety of sources including modifications, design decisions, and testing.

10) *Enable, promote and enforce the quantification of the process and product quality.*
Principle: Quantification with Abstraction Resolution.
Purpose: Metrics can play a major role in the auditing process if proper support is stressed.
Rule for Application: This requirement impacts all activities.

11) *Facilitate access to documentation that reflects the impact of changes made during maintenance.*
Principles: Scope Delineation, Varied Abstraction, Change Propagation.
Purpose: Documentation access is a critical aspect of the planning process for maintenance. While Requirement 1 establishes the necessity of a means for accessing development documentation according to levels of abstraction, this requirement is intended to deal with the fact that development documentation is gradually replaced by documentation which reflects the activities of maintenance on the system.
Rule for Application: This requirement impacts all activities which utilize documentation of any form. In general, these are planning and analysis activities for modification and testing.

2.2 Task 2: Evaluation of the AEGIS Maintenance Process

The software maintenance principles and generic requirements derived from Task 1 are used in the development of a methodology in the classical top-down definition. In developing a model for the AEGIS maintenance process, the process models developed at the Naval Surface Warfare Center (NAVSWC) by government personnel and SAS Consultants are joined with the software maintenance principles and requirements from Task 1. The derivation of abstract models of *maintenance process intent* assures that particular domain-specific requirements for embedded system software are preserved in a

hard-real-time application. In the attempt to maintain relevancy and realism, factors related to the AEGIS application domain, such as system architecture, programming language, and organizational structure, are included from the beginning.

2.2.1 Understanding the AEGIS Domain

Three models of the AEGIS maintenance process are constructed, each with a different objective in mind. The initial model, also called the detailed model, attempts to represent the process as it currently is performed and understood by all those involved. The high-level process relationship model has the objective of describing the communication between individual processes in the overall AEGIS model. The mid-level or aggregated model is derived from the detailed model as an expression of intent. As such, the aggregated model is intended to represent *what* is being accomplished in the detailed representation but excluding the more precise characterization of *how* the intent is currently achieved.

2.2.1.1 Refinement of the Detailed Process Model

The working (detailed) model developed by NAVSWC and SAS Consultants, with review by Virginia Tech System Research Center, divides the AEGIS maintenance process into nine subordinate processes. During review of each process deficiencies are cited with recommendations for improvements conveyed to NAVSWC. The result is an updated version of each process which reflects a more appropriate flow of the maintenance activities.

2.2.1.2 Development of the Aggregated Process Model

The aggregated model of the AEGIS maintenance process is formed with the goal of identifying the methodological intent of the activities. The model is derived by aggregating detailed activities to describe clearly identifiable functions. The report "Task

2: Models of the AEGIS Maintenance Process" provides the aggregate model for each process.

2.2.2 Realization of Requirements within the AEGIS Domain

The application of the principles and requirements from Task 1 is utilized to configure the "ideal" maintenance model. Specifically, the application is used to identify those points in each process where stated requirements can potentially be met. From the potential sources of improvement, the selection of the most effective or most efficient points of application can be made (Task 3).

Task 2 emphasizes the application of the requirements for an AEGIS maintenance environment to the AEGIS aggregated process model. Each aggregated process is assigned requirements in the areas where the requirement should be applied. For each process, the designation of activities to which the requirement applies suggests the potential for improving the activity.

3. Task 3: Toward a Specification of the AEGIS Maintenance Methodology

Utilizing the aggregated model from Task 2, Task 3 addresses the requirements at the detailed level. Specification of the AEGIS maintenance methodology (Task 3), is separated into two parts. First, the requirements as applied to each process of the aggregated model are mapped onto their detailed counterparts. In applying the requirements to the detailed model, we further refine the aggregated level and assign requirements to specific block(s) in each process. Secondly, key requirements are extracted from a global perspective and then at a level local to each individual process.

3.1 Application of requirements to each process

Closer examination of the aggregated process model, in conjunction with the supporting documents provided by NAVSWC, allows the application of requirements to be refined to a more detailed level. At the aggregated level requirements are associated with several blocks in the corresponding detailed process. The detailed process model enables a more definitive application of requirements to block(s). Detailed diagrams for each process and associated requirements are located in Appendix A.

The application of requirements to blocks within a process is achieved through an in-depth examination of supporting documents. The following information is utilized:

(1) Key performer of task (organization or individual responsible),

(2) Description of process (activity occurring within each block),

(3) Supporting organizations (organization assisting in the activity),

(4) Product(s) created/updated (output of the activity), and

(5) Products used (information or products required to accomplished the task).

An important observation is that several requirements span entire processes, i.e. they and are not limited to any subset of blocks within a process. For example, requirement 9 (Enforce recording of source, process, target test documentation, decision alternatives, evaluation, and final decision) is applicable throughout Process 400 and cannot be limited to a single block within that process.

3.2 Extraction of Key Requirements

On completion of the detailed process model, key requirements are extracted at two levels: 1) global to all processes, and 2) inherent to an individual process. This distinction of key requirements allows for prioritization at both levels and provides a basis for identifying tools and techniques to help meet those requirements.

3.2.1 Global Requirements

Key requirements from the global perspective are identified in one of two ways. First, if only one or two points within the entire maintenance process offer the opportunity to meet a particular requirement, then that requirement is classified as a global key requirement. Requirements 5 and 8 are two that meet such a criterion. Recognizing the limited capability for responding to these requirements is essential. The importance of such requirements is also echoed at the individual process level for emphasis.

The second criterion for determining global requirements entails evaluating a requirement as applicable to a single process. The requirements in this category are 7, 9, and 10. Uniformity, recording, and quantification of all activities must be realized over entire processes. For each global key requirement, recommendations of actions for attaining the requirement are listed below:

(1) *Support uniformity in maintenance process/activity.* (Requirement 7)
SQA must lead the effort in defining uniformity. Configuration Management should assist in the insurance that requirements are met. A review should be conducted by the AEGIS SQA and the Program Office would give the final approval.

With respect to documentation PMS 400 should be advised of the need to negotiate an update of the documentation requirements for AEGIS deliverables to better reflect the progression of documentation requirements stipulated in DOD-STD-2167A, This recommendation is not to force adherence to 2167A, but to recognize the intent therein.

(2) *Enforce recording of source, process, target, test documentation, decision alternatives, evaluation and final decision.* (Requirement 9)
An efficient and effective means of recording all aspects of AEGIS maintenance activities requires an integrated environment (explained in Section 5). Such an environment would then become the foundation of configuration management,

with an adjunct to provide SQA functions. The Functional Development Folder is an indispensable method for recording all relevant maintenance information and should be considered the basis for recording the evolution of change in a product.

(3) *Enable, promote and enforce the quantification of the process and product quality.* (Requirement 10)
Metrification of the process. Any time a change is made it should be done in accordance with a measurement to determine the final effects on the system. For example, what is the effect on quality (increase, decrease, or no change) as a result of a maintenance activity? A justification for the change is also required. Alternatives should also be examined relative to the effects they would impose on the system.

A maintenance program that adheres to the OPA framework would provide the most effective results. In particular, such a program allows the effects to be reflected in terms appealing to and understood by management. In following this framework the metrics necessary to derive software quality measures should be applied throughout the change process.

3.2.2 Local Process Requirements

The detailed model is examined to determine the most crucial requirements in each process. Key requirements are identified in two ways: 1) requirements that are crucial to the success of the individual process and 2) requirements which exist only in limited segments of a process. Identifying key requirements leads to general recommendations for each process. Figure 1 provides a chart listing the processes and the recommendations for meeting key requirements. Appendix B provides a more elaborate explanation of the general recommendations and also includes *Areas of Support* which further refine the recommendations.

4. Guidelines and Recommendations

Review of maintenance activities with AEGIS management and SAS Consultants has led to recommendations for restructuring the process to provide a more efficient and effective maintenance environment. Achieving key requirements can be ameliorated by the adoption of recommended techniques and tools.

Process	Recommendations
Establish Program Library (100)	1) CM must be cognizant when a baseline enters recognition and, with QA designate the initiation of version control 2) All receivables from GESD must be recognized as a deliverable (potential version)
Perform Detailed Design and Code (200)	1) Use PERT to estimate time required to effect changes and set the subsequent schedule 2) Employ metrics to observe effect on software engineering objectives as a result of modifications 3) Identify data extraction points
Element Test (300)	1) Clarification of the relationship between Element QA and System QA needed 2) QA needs to provide continuous feedback as changes and modifications are made 3) System QA needs to be involved in the SI&T decision 4) All changes should be recorded in the FDF
System Integration and Test (400)	1) High-Level Integration Readiness review needed before Block 400 2) Scenario Dependency Graph is used by SI&T so alternatives are explicitly identified and can be rapidly pursued when problems arise in system test 3) Any changes to the system must be documented in the respective FDF
Specification Change Process (500)	1) Evaluating risks and alternatives build a crucial foundation for the following processes 2) Categorize forms of maintenance 3) Record all decisions and alternatives in FDF
Develop QA Ship Delivery Package (600)	1) Documentation review needs to emphasize guidelines (process directed) and information (product directed) 2) QA audit needs to insure uniformity is kept throughout process
Quick Update Process (800)	1) Use trouble reports to evaluate technical/cost risks and aid in defining test specifications 2) Insure patch has no adverse effects on other system elements
Computer Program Change Request (900)	1) Consider concurrent maintenance activities 2) Use FDF to determine if CPCR is a duplicate 3) Assess urgency of CPCR and note if affected by next baseline

Figure 1. Process Recommendations

4.1 Process Restructuring Recommendations

We recommend the following actions be taken:

(1) Process 400, System Integration and Test (SI&T), should provide a path back to Process 300, Element Test. Currently, if a serious deficiency in a particular element is located during SI&T, no provision for effecting a repair by the element is indicated. The SI&T environment is not suitable for identification, analysis and recording of alternatives specific to an element.

(2) A process model representation should be employed that supports the depiction of parallel and/or concurrent activities among the elements. This recommendation is extremely important to prevent conflicting maintenance activities from occurring throughout the processes. Recognition of concurrent activities can also be used to prevent the duplication of identical tasks in separate elements.

(3) Classification of the forms of maintenance (corrective, adaptive, perfective, and preventive) should be done as early as possible to avoid grouping non-compatible maintenance forms together. Combining multiple forms can lead to degradation in the maintenance activities since conflicting requirements are presented with different forms. For example, in corrective maintenance quality may be sacrificed in order to correct the problem. If combined with adaptive maintenance, where quality cannot be sacrificed, the conflicting requirement cause the maintenance activities to produce unsatisfactory results. The risk assessment in terms of technical, schedule and cost also differs according to the form of maintenance being performed.

(4) Use quality metrics to avoid combining source code elements of highly varying quality. The demand for quality assessment of code and documentation is greater for adaptive than for corrective maintenance.

4.2 Recommendations for Techniques and Tools

Meeting several of the requirements enunciated above can be accommodated through adoption of new techniques or tools. Several recommendations are noted in the following sections.

4.2.1 Risk Assessment

Risk assessment should be a vital component of the AEGIS maintenance methodology. Assessment includes three areas of risk: technical, schedule, and cost.

Assessment should be initially performed within either the detailed design and code activities (Process 200) or the quick update activity (Process 800). Technical risk is especially important in the latter. Reassessment should accompany major changes in design or code decisions in other processes. Figure 2 shows a risk assessment model recommended for AEGIS that encompasses three levels of decision making: system, element, and group. The model depicts a decomposition of risk assessment in a tiered fashion, such that at each level estimates become more precise and focused. Synthesis of risk assessment at each level leads to a well informed decision at a critical (design, code, test) review. The risk assessment (reassessment) model would be employed in process 500 (Blocks 542 - 549).

PERT, Program Evaluation Review Technique, is recommended for estimating the time required to effect changes and evolving a subsequent schedule. PERT would aid in both schedule and cost risk assessment. PERT modeling for schedule and cost can be used at the beginning of Process 200: Perform Detailed Design and Code.

4.2.2 Identification of Alternatives

Identification and recording of alternatives within each process is crucial if the ARM [NANR89] is to be used. Design modifications for corrective, and particularly adaptive maintenance can benefit greatly from understanding *what* alternatives have been considered and *why* a specific alternative has been selected. Such information is especially needed when the in-service support agent is not the developer, as is the case with AEGIS.

During System Integration and Test (Process 400), the identification of alternatives can also be performed in another manner. Through the use of a Scenario Dependency Graph (SDG), a test director is able to identify the dependency among test cases and

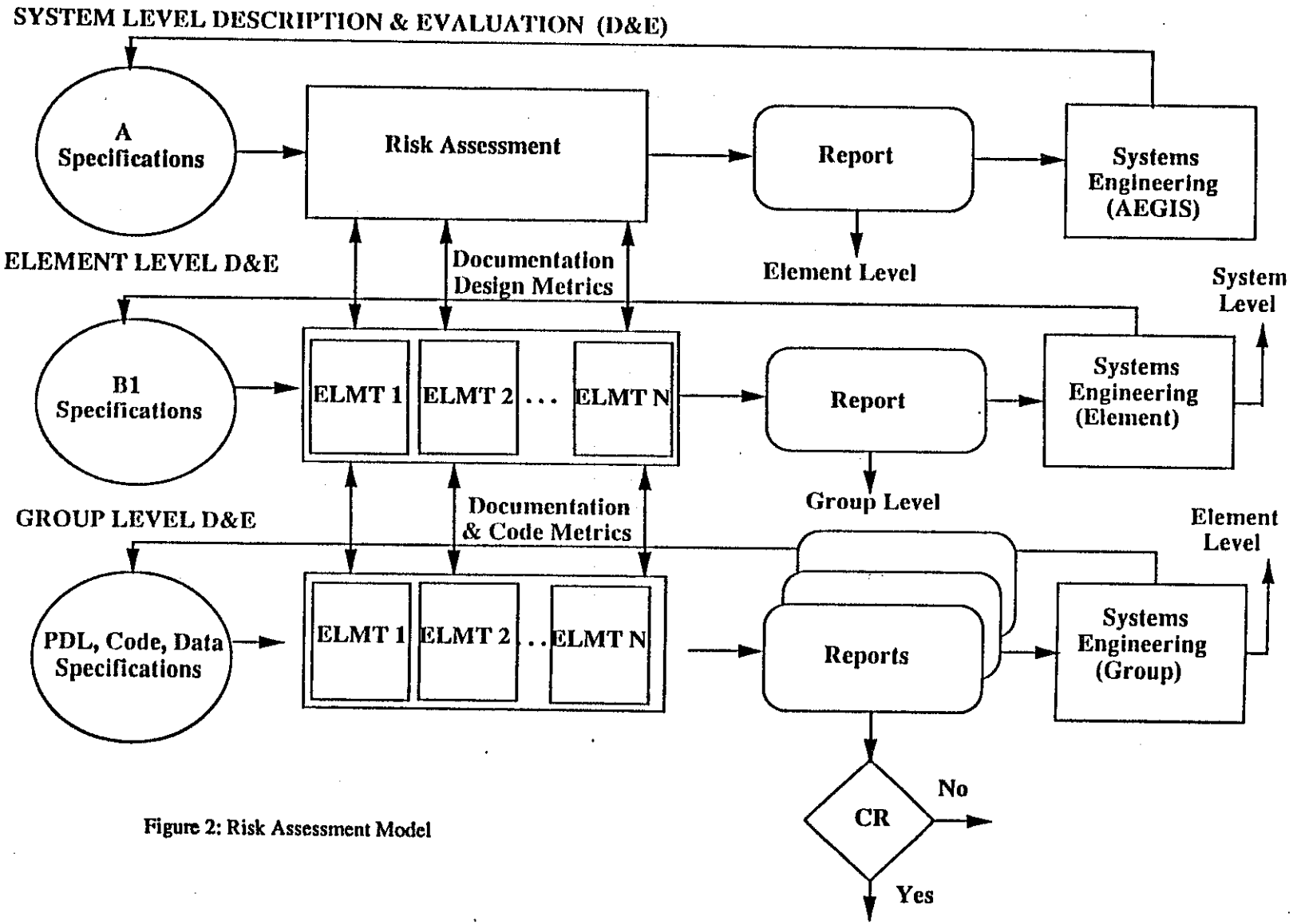

Figure 2: Risk Assessment Model

procedures. Alternate test paths are made explicit for responding to problems during testing, even in the absence of the test director.

A SDG represents relationships among test activities on two levels, as illustrated in Figure 3. The more detailed level shows the required precedence among test cases (following terminology in [ANSI83]) that make up a test procedure; i.e., each test case is represented as a node in the graph and the edges convey the precedence requirements among the nodes (test case *a* must be completed before test case *b* is begun). An estimate of the time to complete each test case is included (although not explicitly shown in Figure 3). Required precedence relationships and the time estimates for completion of test procedures are developed by aggregating the test case representations.

The utility of a SDG is two-fold:

1) The graph provides a readily accessible guide to alternative testing paths when difficulties with an operational software component prohibit continuation of the current test procedure. Note that the SDG does not identify alternatives when test bed problems are encountered.

2) The time estimated for completion of test cases, and subsequent updates of these estimates, provide a measure of test progress and the time required to complete the testing schedule.

4.2.3 Metrics

The application of metrics, both documentation and code, provides a fundamental foundation for assessing the effect of maintenance activities. Metrics also provide a means for managing the maintenance process, however, the selection of metrics must be

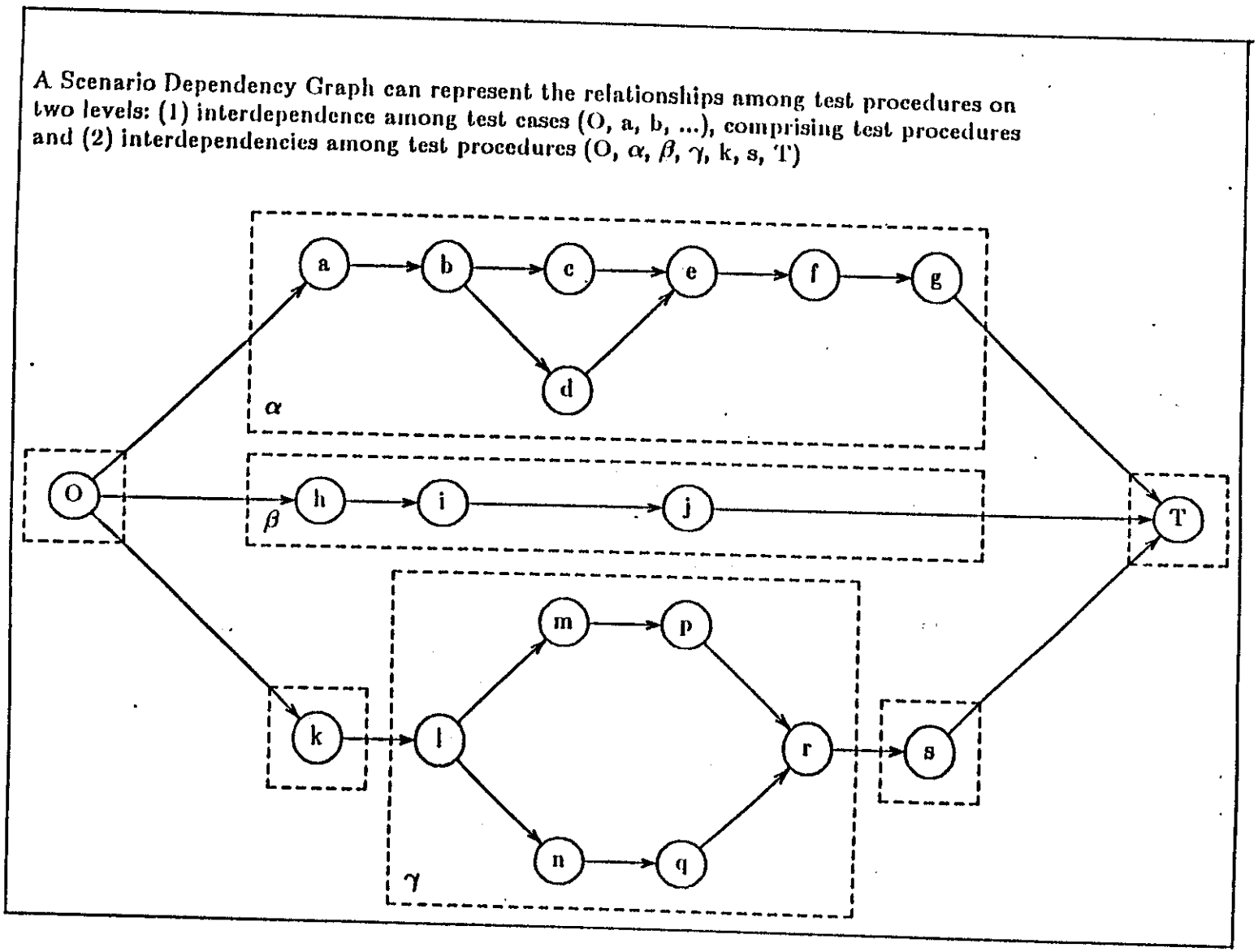

Figure 3: Scenario Dependency Graph Depicting Precedence Realtionships Among Test Cases and Test Procedures

goal oriented. A clear and consensus acceptance of the utility of the metrics employed is essential for success. Multiple data extraction points also need to be recognized throughout the process. For example, in Process 200, code metrics should be applied. However, the location of the data extraction points should directly reflect the measurement goal of each metric applied. If this stipulation is not followed, metrics could be considered as bothersome busy work and uninformative.

4.2.4 Functional Development Folder

The use of a Functional Development Folder (FDF) provides a effective means of recording risk assessment and identification of alternatives. It also enforces the requirement of recording throughout the entire process and creates a foundation on which future maintenance draws information. The FDF is a vital part to the success of the methodology and its importance should not be overlooked.

4.3 An Example: Process 200

This section provides examples as to how the methodology can be applied to a process within the AEGIS maintenance environment. Figure 4, showing the model of Process 200: Perform Detailed Design and Code, is used as an example. A number of key requirements are recognized; a justification for the recommendations and how they should be met is presented. The symbols underneath each block designate the point where a key requirement should be met.

At the initiation of Process 200 a high level design review occurs along with an identification of the impact of changes. At this point PERT (✻) is used to assist the Element Leader in estimating the time required to effect the changes proposed by the design and in assessing how these changes will impact the subsequent schedule. During blocks 200 and 201 it is also important to identify the type of maintenance to be

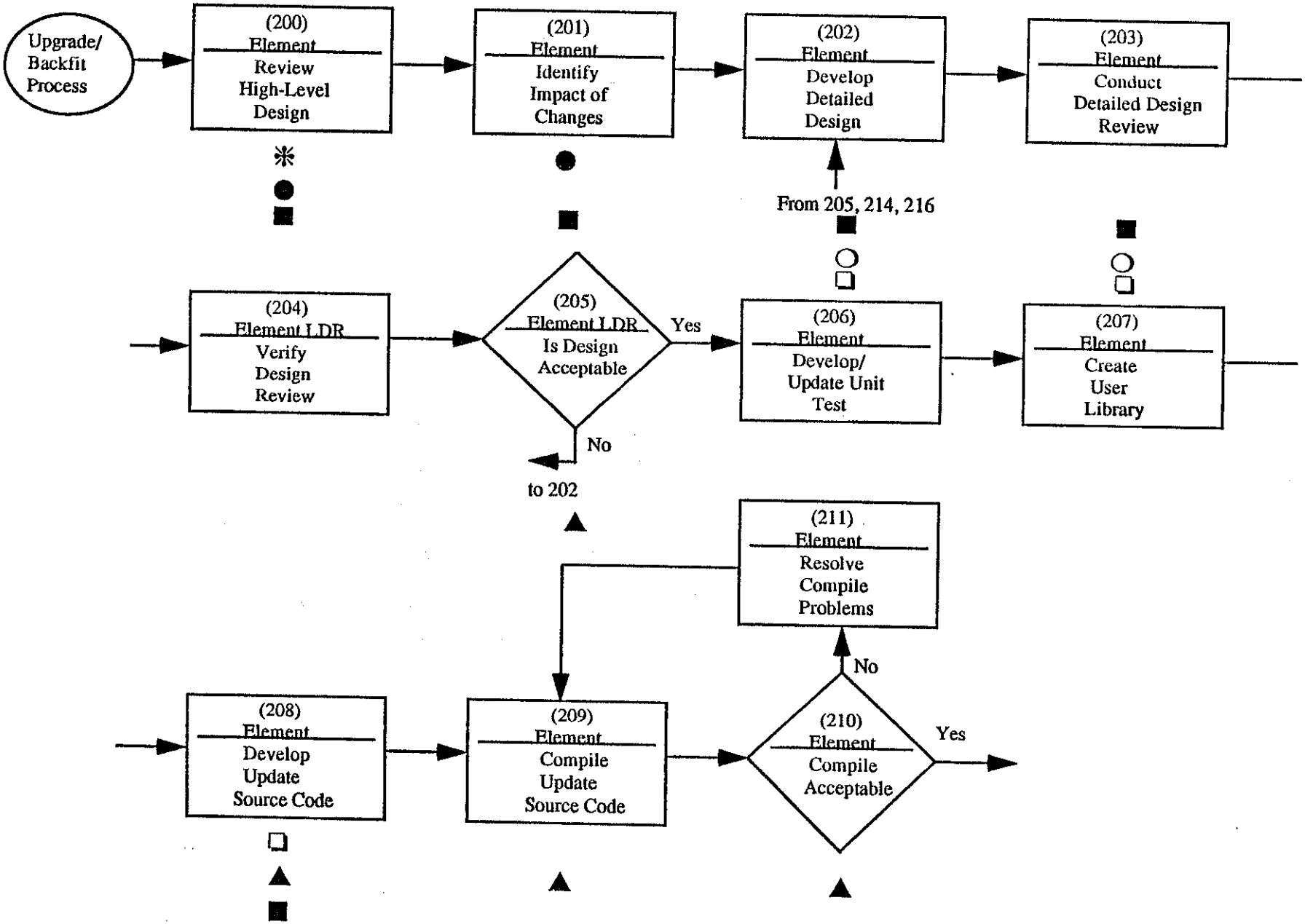

Figure 4a. Perform Detailed Design and Code

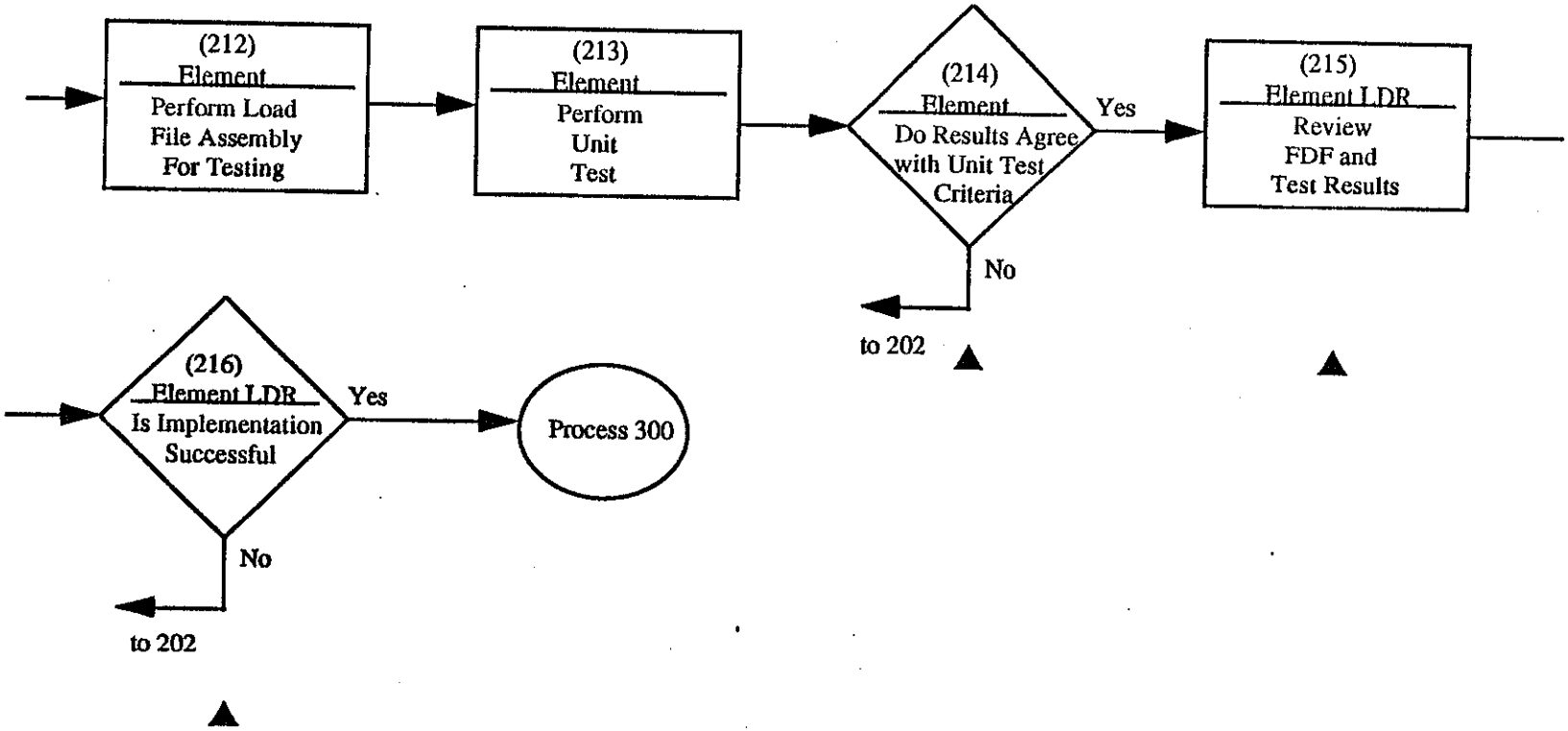

Figure 4b. Perform Detailed Design and Code

performed (●). This identification aids in assessing the time required to implement the proposed changes. Different forms of maintenance demand more in terms of code quality and documentation access. This must be taken into consideration if a realistic schedule is to be constructed.

The detailed design development and review, which occurs in blocks 202 and 203, should be performed in conjunction with risk assessment (○). Are any technical risks imposed by the detailed design, and if so, how do they affect the cost and schedule? Questions such as these must be asked throughout the development and review of the detailed design. Access to documentation must be provided to aid in determining risk factors. This is expressed through requirement 11 in blocks 201 - 204 (■). Documentation must also be provided within block 208, Develop Update Source Code. The documentation should consist of the original documentation and all updates made since the initial release of the current baseline.

Throughout Process 200 numerous points exist where data extraction must occur and where metrics can be used to help guide and manage maintenance activities. Assuming a goal-oriented approach to metrics, the goal of Process 200 for metric application would be to determine if any software engineering objectives are affected by maintenance activities. With this in mind design metrics can be applied at blocks 202 and 203 and code metrics at 208 (❑). The information obtained from the metrics can then be compared with existing values and the difference suggests any change in the software engineering objectives. For each measurement, whether positive, negative, or unchanged, a justification is required. This justification documents and enforces decisions made throughout the maintenance activities and aids in backtracking should future problem be encountered.

Whenever decisions are made that effect the schedule, cost, or technical risk of the system they must be recorded in the Functional Development Folder. Situations such as these occur in blocks 205, 209, 214, and 216. Changes made to the system occur in block 208 and also must be recorded in the Functional Development Folder.

Appendix A contains the detailed diagrams for all nine processes along with the application of requirements to them. By analyzing the key requirements in each process techniques and tools can be applied to aid in meeting the requirements as the above example shows.

5. AEGIS In-Service Software Support: A Planning Perspective

The following assertions concerning the AEGIS Combat Systems Engineering Program at NAVSWC seem incontrovertible:

(1) In-service software support is becoming more complex because of the increasing numbers of: ships, baselines deployed, and diversity of combat system elements.

(2) The complexity stemming from technology pull, i.e. potential sophistication of weapon and system elements, shows no sign of abatement.

(3) Increased quality is desired in the software systems being deployed to meet adaptive and corrective maintenance requirements; i.e., the objective of reliability continues to be emphasized, and maintainability is emerging as an objective demanding high priority.

(4) Increased productivity in the software maintenance process is mandatory as the increasing numbers of ships, baselines, and mixes of elements must be supported with no corresponding increase in staff (and perhaps a reduced number).

Total Quality Management and Increased Productivity must move beyond buzzword status in AEGIS just to keep pace with the increasing responsibilities and complexities imposed by the combat system application domain. This section sketches an ambitious, but realistically achievable, plan to increase quality and productivity to a

level exceeding the pacing advancements in numbers and technology. Such a plan continues the high priority on reliability and raises maintainability to an equally high priority.

5.1 Long-Term Needs

The terminology system evolution in contrast with software maintenance is especially appropriate for AEGIS since the combat is expected to have a life of 30 years or longer and the hardware components are continuously changing (with introduction of a new block or during ship overhaul). The primary role of a methodology is to order and define decisions that must be made, as well as bring to the attention of decision makers the appropriate system concerns. The long-term needs of AEGIS in-service support mandate careful attention to the methodology requirements, the effects of combat system evolution on the embedded software system, and the consequent efforts in training AEGIS personnel. Responding to these long-term needs, the concepts associated with a system evolution environment are outlined.

5.2 Recommended Approach: The ASEE/2000

ASEE/2000 (the AEGIS System Evolution Environment for the year 2000) is advanced as both a programatic and a technical concept. Programatically, ASEE/2000 represents a goal and the outline of a strategy to reach that goal. Technically, ASEE/2000 is an implementation of the emerging AEGIS Maintenance Methodology supported by a set of integrated software utilities that provide combat systems engineering and software engineering support within a framework that embraces training support as well.

The definition of the AEGIS Maintenance Methodology is seen as an evolving process, as rightly it should be. However, the prioritization of objectives and the

recognition of guiding principles are prerequisite to the specifics of methods and techniques prescribed in a methodology. Methods and techniques derived from underlying principles also serve to define the requirements for software utilities that enforce, encourage and enable the effective use of am evolving AEGIS Maintenance Methodology.

5.2.1 The Conceptual Definition

ASEE/2000 is envisioned as a Combat System Evolution Environment following a development process governed by the Objectives/Principles/Attributes (OPA) framework for software quality assessment. Guiding the design of this environment is the Abstraction Refinement Model (ARM), developed by VTSRC to furnish the descriptive linkage between development and maintenance. The ARM has proved very effective in (1) structuring the development process as successive transformations to resolve abstraction, (2) emphasizing the differences in carrying out the different forms of maintenance, e.g. adaptive versus corrective, and (3) explaining the role of and necessity for reverse engineering.

This conceptual definition is manifested in a seamless set of utilities that collectively constitute:

- a systems engineering environment,
- a software engineering environment, and
- the cornerstone of AEGIS training.

5.2.2 The Functional Definition

The systems engineering environment within ASEE/2000 utilizes dual complementary simulation models of the AEGIS Combat System:

(1) a combat system performance model driven by the performance of the embedded software system effectiveness, and

(2) a software system effectiveness model based on changes in system requirements, software design, supporting computing hardware, supported sensors and weapons, or schedule constraints.

This environment enables the effectiveness of the software system to be made visible in the performance of the combat system and, likewise, enabling risk assessment to include software quality objectives such as maintainability and reliability to be factored into trade-off decisions. Such models, coupled with quantitative assessment of software quality, can produce tangible evidence of the importance of sound decisions regarding the development process (specification, design, coding, testing, etc.).

Translation from the traditionally functionally oriented system specification to an object-oriented representation is supported. Configuration management support at the system level is aided by a communications infrastructure that includes hypermedia as well as file transfer and electronic mail capabilities.

The software engineering environment provides formally derived methods based on AEGIS Maintenance Methodology principles that recognize and promote systems/software interrelationships. A primary provision is the software quality assessment utilities based on the OPA framework, which relies on automatic data extraction and analysis for a metrics-driven maintenance process. Software configuration management utilizes automated procedures to a high degree and provides triggers and alerters to prompt and audit human intervention where needed.

The training environment is possibly the most important of the three. ASEE/2000 can provide a clear rationale for viewing AEGIS in-service support as "system evolution"rather than "maintenance." A centralized, uniform, and consistent basis for methodology understanding and learning through utilities that:

(1) explain the role of metrics and the rationale for "metrics-driven software change,"
(2) convey the desired application of TQM initiatives within the software maintenance context,
(3) show the necessity for and importance of decision support through metrification of the product and process, and
(4) provide continuous monitoring and evaluation of training needs through automated feedback on tool usage, assistance requests, and user/student reactions.

The training environment offers a clear prospect for improving software productivity and job satisfaction. New hires and transfers can be rapidly introduced to the AEGIS Maintenance Methodology, acquainted with the prioritization of objectives set by management, and oriented to the taxing demands of embedded, time-critical software systems. On-the-job learning is computer assisted, reducing the training demands on other personnel and permitting individual pacing of the material.

5.3 ASEE/2000: A Summary

The ASEE/2000 proposal is ambitious, but the needs are apparent and the time to plan for meeting those needs is now. Conceptually, the proposal recognizes that methodological links are required between systems and software engineering. Similar principles govern the application of both, and coupling the supporting environments offers clear advantages. Complementary simulation models enable the evaluation of software maintenance decisions in terms of software system effectiveness. Software system effectiveness is made visible in the predicted impact on AEGIS combat performance criteria. Coupled with a software quality assessment based on the OPA framework, tradeoffs in effectiveness versus quality can be made (using software quality

indicators), permitting a form of risk assessment not previously recognized in the software engineering research literature.

6. Concluding Summary and Recommendations

This in-depth investigation of the AEGIS maintenance process has proceeded in three sequential, subordinate tasks:

- Understanding the relationships linking software development and software maintenance, reviewing maintenance concepts for time-critical embedded software systems, identifying principles governing the maintenance activities, and deriving the requirements for an"ideal" model that forms the target for an AEGIS maintenance methodology (Task 1);

- Reviewing and refining a detailed model of the AEGIS maintenance process produced by NAVSWC and SAS Consultants, transforming the detailed model into aggregated models that focus on intent as opposed to procedure, and overlaying the requirements and principles from Task 1 on the models (Task 2); and

- Noting the opportunities for requirements to be met, evaluating the alternatives, suggesting techniques and tools to assist in meeting the requirements, and prioritizing the requirements based on the suggestions of tools and techniques. (Task 3)

The following recommendations are ordered in decreasing priority, with some explanatory comments included:

(1) Adopt a uniform process for maintenance across all elements and throughout the AEGIS program.

Uniformity in activities and procedures (Requirement 7) has the highest potential payoff. Enabling and enforcing uniformity permits a single process model definition at a very detailed level, allows a common environment (set of integrated software utilities) to serve the entire program, and promotes personnel training, mobility and productivity. The ASEE/2000 concept could serve as the basis for a maintenance environment. A secure LAN is also a crucial element.

(2) Employ extensive recording of all pertinent data in performing maintenance activities (Requirement 9).

The Functional Development Folder is a good start toward such a goal. Maintenance personnel need to appreciate that a system with the lifetime of AEGIS mandates that changes be made to prior modifications. Maintenance documentation in time becomes more crucial than development documentation.

(3) Install a metrics program that enables quantification of product *and process* quality, coupled with a mandatory responsibility for audit of the maintenance process (Requirements 10 and 6).

AEGIS SQA needs to focus more on the process and less on the product. Clarification of the relationship between Element and System SQA is needed. A metrics program following the OPA framework would provide the most comprehensive and instructive approach to meeting the AEGIS needs. A metrics-driven maintenance methodology, although still an idea in development, is highly recommended.

(4) Categorize the maintenance forms as early as possible and do not mix different forms, e.g., adaptive and corrective, or combine components widely differing in quality if at all possible.

Mixing maintenance forms unnecessarily complicates the task by introducing confusion regarding the least common abstraction level required and the supporting documentation needed. Combining components of widely differing quality leads to a "leveling" of quality rather than a consistent improvement.

Recommendations regarding individual process structuring and techniques and tools specific to process activities are summarized in Figure 1.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) Systems Research Center SRC-91-003	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Systems Research Center 6b. OFFICE SYMBOL (*If applicable*)	7a. NAME OF MONITORING ORGANIZATION Naval Surface Warfare Center
6c. ADDRESS (*City, State, and ZIP Code*) 320 Femoyer Hall Virginia Tech Blacksburg, Virginia 24061-0251	7b. ADDRESS (*City, State, and ZIP Code*) Dahlgren, Virginia 22448-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Naval Surface Warfare Center 8b. OFFICE SYMBOL (*If applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (*City, State, and ZIP Code*) Dahlgren, Virginia 22448-5000	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (*Include Security Classification*)
Requirements for a Software Maintenance Methodology: Final Report (Appendices Not Included)

12. PERSONAL AUTHOR(S)
Richard E. Nance, James D. Arthur, and John A. Ciaramella, Jr.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*Year, Month, Day*)	15. PAGE COUNT
Final	FROM 7/1/90 TO 5/20/91	1991 May 17	28

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUB-GROUP	Documentation, Management, Methodologies, Reverse Engineering, Model, Maintenance

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

This project ventures into the domain of software maintenance methodologies, an area given relatively little attention in software engineering. Project efforts are divided into three tasks: (1) definition of maintenance methodology requirements, (2) development of a model of the AEGIS maintenance process, and (3) specification of the requirements for an AEGIS maintenance methodology. The focus of this report is on the third task, performed in the period 1 July 1990 – 20 May 1991.

The research utilizes the Objectives/Principles/Attributes (OPA) framework developed for software quality assessment for time-critical, embedded systems. The overriding maintenance objective is to "realize desired changes in an efficient and effective manner." Four principles are identified that support achievement of this objective. Principles identified in the OPA framework are also applicable. As a consequence, eleven (11) requirements are derived for an AEGIS maintenance methodology.

The AEGIS maintenance process model is used to determine the potential points where any of the requirements can be met. Recommendations are made with respect to restructuring the process, noting the most appropriate place for meeting requirements, and acquisition or development of software utilities to support maintenance. Recommended as an approach for meeting long-term needs is the AEGIS System Evolution Environment, supporting both systems and software maintenance activities.

[Appendices Not Included]

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS		
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (*Include Area Code*)	22c. OFFICE SYMBOL

DD Form 1473, JUN 86 *Previous editions are obsolete.*

SECURITY CLASSIFICATION OF THIS PAGE

Unclassified