SAE The Engineering Society For Advancing Mobility Land Sea Air and Space®
INTERNATIONAL
400 Commonwealth Drive, Warrendale, PA 15096-0001

# AEROSPACE INFORMATION REPORT

SAE AIR5121

Issued 1997-01

Submitted for recognition as an American National Standard

## SOFTWARE SUPPORTABILITY - AN OVERVIEW

TABLE OF CONTENTS

1. SCOPE:

This SAE Aerospace Information Report (AIR) provides an overview of the issues relating to the support and supportability of software in computer-based systems. It has general applicability to all sectors of industry and commerce and to all types of equipment that contain software. The software support issues and activities summarized in this report are reasonably easy to comprehend. The reader should not be mislead into believing development of supportable software is easy to achieve. The target audience for the document includes software acquisition organizations, developers, supporters, and end-use customers.

2. REFERENCES:

[COOGAN] Coogan, C., "Re-engineering Supportability: A New View of an Old Problem," Communications in RMS, Volume 2, Number 2, July 1995, pp 56-59.

[DS00-60UK]Defence Standard 00-60, "Integrated Logistic Support," Issue 1, April 1996. "Logistic Support Analysis Application to Software Aspects of Systems," Part 3.

[MIL1388USA]MIL-STD-1388-1A, "Logistic Support Analysis," Notice 4, January 21, 1993. MIL-STD-1388-2B, "DOD Requirements for a Logistic Support Analysis Record," March 28, 1991.

3. TOPICS:

3.1 Software Support in the System Context:

Software supportability is a measure of how well software support activities can be conducted by assigned support personnel using available support environments within allocated schedule and cost constraints. All life cycle activities that are necessary to define, acquire, conduct, and sustain software support should be accomplished within the context of the total system. The system is ultimately what must be supported. Any software support concept must be derived to complement system support during operational use. Support of software during the system's operational use will require a profile of changes to the software: correcting defects, enhancing functional and performance operation, and adapting to improvements in the environment. There will also be a profile of software support activities that are more closely associated with the services of delivery, recovery, help desk assistance, training, and information exchange associated with customer/supplier interfaces.

In [COOGAN], several systems supportability principles were outlined. These principles of supportability do not explicitly include or exclude software, but the following paragraphs provide a context for software supportability principles within this context of system supportability.

3.1.1 Principle 1: Software Supportability is a Systems Performance Parameter: Software support is an essential element of the overall support for any system that has a software content. The support and supportability of software should be viewed as an integral component, in both functional and physical terms, of the equipment or system in which the software operates. Hence, although software has technical characteristics that are distinct from hardware, any assessment of software support requirements should be considered in the light of what is necessary over the full life-cycle for the complete system to meet availability, reliability, maintainability, and performance requirements. The composite system supportability should be a measurable, allocable, and predictable parameter for the overall performance effectiveness of systems or equipment.

3.1.2 Principle 2: Software Supportability is a Life-of-System Concern: Software support covers all activities undertaken to allow continued use of software within a system after it has been fielded. Software should be capable of being supported during operational use so that day-to-day operating needs can be met. Hence, software should be designed for supportability during development so as to anticipate and facilitate the introduction of changes throughout the life of the software. The acquisition and sustainment of software support during operational use are essential to meeting system effectiveness requirements. Thus, software supportability is a concern to be addressed throughout the life of the encompassing system. One purpose of this overview is to highlight the need for all projects to consider software supportability as a life-of-system issue and to identify the key issues that should be addressed by all target audiences.

3.1.3 Principle 3: Management of Software Support Requires a Wide Range of Information: Standards such as [MIL1388USA] and [DS00-60UK] have been used to both direct analysis activities and specify information requirements. With the recent trend to transition such standards from government specifications to industry best practices and guidelines, the need for consistent, system-specific software supportability information will be even more important to achieve necessary system supportability. These standards have rarely been applied to software support and its management. An opportunity now exists for software support to be properly defined within the context of the appropriate system logistic support analysis tasks by adequately defined information for the software support activities. Such information should cover the following support concerns:

a. Support Concept: a strategy or model for how software support of a system is to be accomplished, including the locations, activities, facilities, and resources that will be required.

b. Support Areas: a grouping of major software support task activities by operational field support (e.g., installation, backup, failure reporting, recovery, data loading and unloading, training); modification support (e.g., maintenance tasks, configuration management); and, logistic management support (e.g., problem reporting & corrective action, configuration control of releases, release distribution, operational field help desk).

c. Supportability Characteristics: attributes of software processes (e.g., development, transition, operation, support), products (e.g., documentation, source code, test suites), and environment resources (e.g., facilities, support systems, support personnel) that enhance the capability to support software.

3.1.3 (Continued):

The support concept, support areas, and supportability characteristics provide a comprehensive view of the information necessary to adequately define, acquire, implement, and sustain the support infrastructure for the software components of a system.

3.1.4 Principle 4: Software Supportability Analysis Methods Apply to Public and Private Sector Systems: Existing methods and techniques for supportability analysis can and should be applied to both public (e.g., government) systems and private (e.g., industry) systems. Typical analysis methods include: Failure Mode, Effect, and Criticality Analysis; Fault Tree Analysis; Reliability Centered Maintenance; Level of Repair Analysis; Safety Hazard Analysis; and so forth. The methods and techniques used will vary based on the system and, specifically, the influence of the software components and their desirable support concept, support areas, and supportability characteristics. The application of such methods and techniques may require some interpretation to be most effectively applied to a system that includes both hardware and software components. New methods and techniques to address certain software-specific support issues and hardware-software component trade-offs are needed for both public and private systems.

3.2 Background on the Need for Supportable Software:

In order to ensure a system can meet operational expectations at a time when operational change is standard and timely support task response is critical, the software components of a system must be adequately supported. In many cases, the only responsive method to meet constantly changing threats or enterprise business modifications is to modify the system's software components. Since software is such a pervasive part of systems, there must be provisions to ensure any loss of software operational capability can be restored in a timely manner. And, with cost such an important part of effective business management, software support cost must be optimized. This section reviews some of these issues that focus on the need for supportable software.

3.2.1 Software Functionality is Fundamental to Most Systems: Software is now the predominant means by which functionality is provided in modern business, industrial and military systems. This functionality may be of a nature whereby it is highly visible to a human operator or it may be internal to the system with no visible externally observable effect. However, there are very few, if any, business sectors where the use of software is not integrated at some point into operating processes.

3.2.2 Supportable Software is Critical to System Evolution: The flexibility of software as a design medium is of great potential benefit to all business sectors. The dynamics of a changing operational environment can be rapidly incorporated into computer-based management systems and other support tools to maintain and enhance a business's efficiency and effectiveness. However, despite the capability to more readily change a system through changes to its software components, the embodiment of change to software in a complex system is still a significant task that involves highly expensive resources and skilled personnel. The consequences of failures or inefficiency within this process can be severe to both developer and customer. Hence, software support and supportability are seen as vital issues for cost-efficient systems evolution.

3.2.3 Software Supportability Influences Design: Supportability should always be a distinct concern in the design approach for any software item, which will itself be an integral aspect of the systems development process. However, the importance of supportability will depend on the functional role of the software within the overall system, the system purpose itself, and the likelihood of system evolution over time. There are many products in the commercial and domestic sectors where the role of software is embedded within the functionality of an overall system or equipment and where the product is planned to have a relatively short-term existence before it is superseded. In such cases, software supportability may be secondary to instituting a design and development approach that facilitates minimum time and cost to market for a fully functional and reliable product. For systems where the software is highly interactive with the human user or where it is intended that the system will evolve to meet changing user requirements, then supportability considerations may be of equal or higher weight than time/cost to market and performance. It should be noted that many aspects of supportability reflect quality goals and measures. Hence, the emphasis of software supportability as a design factor benefits the system/equipment development and final product quality.

3.2.4 Software Support is a Major Contributor to Life-Cycle Cost: Software is now accepted to be a major contributor of system life cycle cost. Previous experience has shown that software support can be up to 80% of the total software life cycle cost, although even this proportion may be exceeded for certain types of systems. Furthermore, to compound the problem, reducing the software support cost may not reflect the most optimum software or system cost solution. Generally, software support cost must be considered within the total context of the software and system life cycle cost projections.

Poor software supportability characteristics (process, product, or environment) will tend to increase the resource and engineering costs for change implementation, which will in turn be reflected in the price of support and updates for end-user customers. The cost to design-in the software supportability characteristics and its projected effect on decreasing the resource and engineering costs of software support should be analyzed. In the case of software that has a high user profile, the provision for a continuing design support capability after product release is invariably a commercial necessity. If the developer is to retain and extend a customer base, it is necessary to capitalize on the initial development investment through cost-efficient software support and reuse of software products.

3.3 Software Support Concept:

The requirements capture for a new system should include supportability and support concept analyses and trade-offs for any expected software deliverables. The support concept is principally influenced by the scope and volume of support task activities required in the areas of operational use, product modification, and logistic management control. Supportability characteristics of the process, product, and environment support factors determine the extent to which the planned/required task activities can be achieved. The results of supportability analyses influence how the system (including hardware and software) can be best designed and built to accomplish planned operational use and satisfy support requirements using identified logistic management controls. The effectiveness of the implemented support system is improved through an iterative review and validation process that extends throughout system development and fielding.

3.3 (Continued):

The derivation of a software support concept for a system should be an early project activity, since this will have an impact on system supportability requirements and may influence the design of the high-level software and/or system architecture. A support concept may initially be fluid for a developmental project. As the system engineering process proceeds, derivation of a well-defined support concept provides assurance that:

a. Support requirements are satisfied;
b. Establishment of actual support resources is effectively planned and implemented; and,
c. Life-cycle costs for support are fully understood and reflected in financial plans.

3.3.1 Key Elements of a Software Support Concept: At the simplest implementation level, a software support concept identifies a software engineering capability with the personnel resources and skills, physical facilities, and support systems to undertake on-going development and change implementation. A customer/supplier procedural interface through which queries, change requests and updated products pass must also be defined. The resources committed to the support function represent a significant part of the software life cycle costs in terms of both capital investments and operation expenses. Judging the optimum scale of this investment involves trading off the costs of support against the commercial benefits of improved responsiveness to customer functional and time-to-market requirements. To lower software support cost or make support more effective often requires increased cost to improve the development process and product quality. This trade-off may delay initial time-to-market. The supportability analyses must provide guidance for a support concept that balances the time-to-market, product quality, support responsiveness, and life cycle cost elements.

The software support concept issues become more varied as system complexity increases. In particular, where users are distributed (whether via data networks or in physical isolation), or where their business has a critical dependence on system availability and performance, the software support concept may need to be more elaborate to provide immediate assistance, a rapid fix capability, and periodic software updates to accommodate functional improvements and new features. Some of the major elements of a support concept are outlined below.

a. Support Task Frequency: The number of times a given support task is required over a specified calendar time period to satisfy customer operational or modification needs constitutes the support task frequency. The support task frequency drives the need for support resources (personnel and systems) and hence the support cost. A particularly important part of the support task frequency is based on the number of software product modification requests and the period of time between planned product releases that implement solutions to the requests. A customer may have a specific requirement for product updates to be available at a set frequency, or such updates may be planned because of previous historical information. A balance is required between the benefits of offering customers regular opportunities to make performance and functional improvements, and the higher support costs when the frequency of these opportunities is increased.

3.3.1 (Continued):

b. Support Task Load: The resource (e.g., personnel, support system) time to complete each support task required to satisfy customer operational or modification needs constitutes the support task load. The support task load drives the need for support resources (personnel and systems) and hence the support cost. The support task load along with the support task frequency determines the calendar time cost of the support task.

c. Support Level/Location: Support level/location is concerned with where support for a software item would best be located, either in the field, at an intermediate facility or at a contractor's depot site. The support level/location is driven by the specific support tasks required to satisfy operational and modification needs, the required responsiveness in developing and fielding updated software products, and the benefits of collocation with existing development/support resources. Those support tasks, personnel, support systems, and facility resources that are needed at each support level/location are part of the support concept definition.

d. Support Resources: Support resources include the personnel, support systems, and facilities that are necessary to accomplish the support task activities. There are several key aspects of the support resources that should be included in the supportability analyses.

(1) Contractor Support. Contractor-provided support will normally be the default option, but for software that is expected to be particularly sensitive to operational/business change requirements, the customer may wish to establish an in-house support capability or enter into some form of joint resource arrangement with the supplier.

(2) Customer Participation. A software support concept may require an element of integrated 'intelligent customer' resource to participate in such functions as change analysis/specification /prototyping and so forth. Where such an approach is planned, the trade-off analysis should consider the options for physical integration of customer and supplier facilities and the most cost-effective balance between their respective resources.

(3) Vendor Support. The use of Commercial-Off-The-Shelf (COTS) software product as an integrated part of the operational software system and the associated support systems may have serious support consequences. The extent that COTS software product is used will dictate much of the support concerns. Since COTS vendors are typically in business to provide their software product to more than one customer, it is highly likely that their software product will be updated frequently to accommodate the many improvement needs for all their customers. The frequency of the updates and the nature of the updates may not be of value (in fact, may be a very costly support factor) to a given customer. And, failure to integrate the updates may invalidate the vendor support agreement.

3.3.1 (Continued):

(4) Shared Resource Utilization. A developer will often be in the situation of supporting a fielded product while simultaneously carrying out further development of a related family of products, or of entirely new products using the same development facility. In this situation there may be conflicting demands on resources, and judgments will need to be made on the commercial benefit of additional resource investment.

e. Intellectual Property Rights (IPR): The business failure of a COTS vendor can have very serious consequences for a customer, even if appropriate escrow data rights and service/product warranties have been attained through an Intellectual Property Rights agreement. The extent to which contingency measures are explored in the supportability analysis should be limited to instances where there is considered to be relatively high risk (e.g., software that is immature, has specialized functionality, or has high impact on system operation). In such cases, the analysis should consider to what extent there are suitable alternative products in the marketplace, and whether arrangements could be established for the acquisition of the necessary data rights by an alternative support agency.

If the customer wishes to establish an in-house support capability for a software product, the implications of the software's IPR becomes a major support contractual issue. Other data access rights, such as establishing an escrow capability for vendor-supplied "shrink wrap" software, is also a major concern. In any situation where a customer requires more than the normal usage rights for a software product, or where the use of support agencies other than the original developer will be a consideration, a legal understanding of the right to perform software support actions must be established. Any such constraints need to be exposed and assessed prior to contract award for software development or procurement.

3.3.2 Software Support Tasks and Initiator Events: For any computer-based system there will be a number of different situations that could initiate the need for software support task activities. It is important to examine such support initiators and their consequent support requirements at the same time that equipment design alternatives are being considered. The events or situations that may initiate software support task activities should be grouped according to common operation, modification, and logistic management support impact. A set of top-level software support initiator groups should be defined against which the support requirements of the subject software item may be determined. These initiator groups should be adapted as necessary to the individual nature of specific systems and the impact of the software on system use and mission capability. Some of the software support tasks and possible initiator events are illustrated in Table 1.

TABLE 1

| Support Area | Support Task | Support Task Initiator |
|---|---|---|
| Operational | Installation | Release Distribution |
| | Data Load and/or Unload | Mission Preparation/Completion |
| | Backup | Preventive Maintenance Schedule |
| | Failure Reporting | System Failure |
| | Recovery | System Failure |
| | Training | Personnel, System, Software, Procedures, Update |
| Modification | Corrective Maintenance | Software Failure |
| | Perfective Maintenance | Change in customer functional or performance requirements |
| | Adaptive Maintenance | Change in hardware or COTS software |
| | Configuration Management | Completion of New SW Version |
| Logistic Management | Release Replication | Field Loss of System & Backup |
| | Release Distribution | Release of New SW Version |
| | Installation of COTS | Release Distribution |
| | Help Desk Management | Field Problem Query |

3.3.2 (Continued):

Defining initiator groups, conducting supportability analysis, and identifying an appropriate support concept may be carried out iteratively during the development phase (and even the support phase) of a project. Each iteration should build on the previous analysis and use the results to modify or validate the evolving software support concept. At the earliest project stage, an analysis of support initiators should be undertaken as part of requirements capture. The aim should be to ensure that the software design approach takes account of what post-delivery changes may be anticipated. The support capability must be responsive and efficient in satisfying customer needs, and maximize the return on investment of support resources.

3.4 Software Support Processes:

In broad terms, software support is a natural extension of software development. However, there is a different emphasis with change implementation being the dominant engineering activity. The management role also changes accordingly, to provide product support arrangements through the procedures and logistics of a customer/supplier interface.

3.4.1 Logistic Management Aspects of Software Support: The prime activities in the logistic management infrastructure for software support include: a system for problem reporting and corrective action processing; procedures for configuration control of product baselines; the delivery, installation and check-out of software/data updates; and, the provisioning for user support (e.g., through training and documentation). In addition, these software management support activities must be integrated within the context of similar system-level support activities. The logistic management activities of software support are covered in more detail below.

3.4.1 (Continued):

a. Problem Reporting and Corrective Action: Problems in the form of apparent faults in existing software, requests for new functional features, or modifications required by changes in the operational environment, must receive appropriate management attention. Problems and change requests should be documented in standard formats that provide a clear and complete understanding for system/software engineers of the nature of the problem to be investigated, or the new capability that is required. The compilation of such information may require close interaction between supplier specialists and the customer, and each such instance should be identified and tracked through the configuration control arrangements for the overall system. Help desk support is an extension of the problem reporting and corrective action activity that provides more rapid response to support field queries concerning software operational problems.

b. Configuration Control: Logistic management configuration control includes those procedures by which new software baselines and interim releases are managed. These procedures are distinct from the controls and conventions applied at the software engineering "workface" to manage individual change development and embodiment (see software configuration management engineering process below). Problems and change requests that require design changes to software items are allocated to the software modification process for action through a software configuration control interface. Logistic management configuration control ensures appropriate software releases are installed at each operational site and identifies any operational constraints due to a site's installed software.

c. Delivery, Installation and Check-out: Logistic management also coordinates new software release delivery, installation and check-out to ensure adequate operational availability of the system at each operational site or at intermediate support sites. The output from the software modification and configuration management processes is a fully built and tested software release package, ready for delivery to field sites and users. The logistic management support required for packaging, transportation, installation, and check-out will vary depending on the nature of the system. In some cases it may be entirely feasible for users to undertake installation without any direct supplier support. For situations such as major software updates to networked systems, there may need to be site-specific planning and support activity. Depending on the customer and contractual arrangements, there may need to be an acceptance period and a suite of acceptance tests that are part of the installation and check-out function.

d. User Support: Any new software release may involve training and/or documentation for users, to ensure effective operational use of the updated functionality. Training materials and user manuals should therefore be maintained under configuration control throughout the life of the software to which they relate. When a software release is planned, the implementation of any necessary new or revised training and documentation should be scheduled to ensure that users are fully prepared and supported once installation and check-out are completed.

3.4.2 Engineering Aspects of Software Support: The engineering process within software support involves taking change requests from various sources, analyzing the requested changes for implementation feasibility, developing updated versions of configured software items and associated documentation, testing the revised software, and issuing the versions as production-standard software ready for system installation. Such versions normally implement one or more change requests in a 'block release', with the priority for embodiment of individual changes being determined by customer needs and support resource capabilities. This engineering support activity should take place within a formal environment of configuration management and modification processes for the software and its parent system.

a. Software Configuration Management: Software configuration management identifies, controls, tracks the status and audits implementation of all design changes to a software item. Once the software modification process has completed the embodiment and test of all changes in a new block release, the updated software products are turned over to software configuration management for final configuration builds and formal release as specific product 'versions' by the relevant configuration control authority. The software control authority that interfaces with logistic management configuration control will normally be a software and/or system configuration control board -- a formally constituted body that reviews change requests, authorizes and schedules the embodiment of changes, and releases the implemented changes for use by users. A key aspect of the role of a configuration control board is to determine change priority, i.e. how quickly a requested change needs to be available to users. Typical definitions are: emergency (immediate release required), urgent (interim release required in the near future) and routine (next scheduled release is adequate).

b. Software Modification: The software modification process embodies design changes into a software product baseline, as allocated through configuration management procedures. The engineering activities involved in software modification -- e.g., change analysis, implementation, integrated test, and logistics release -- are broadly common across different system types, although the actual tools and methods used are system dependent and may vary widely. The support tools and methods will normally be compatible with development tools and methods.

c. Support Systems and Facilities: To maintain product consistency, software modification is generally carried out using similar technical facilities to those employed for initial development. In some cases (e.g., military systems), these support systems and physical facilities may be extensive and include special laboratory equipment or test/simulation rigs. The procurement and installation of such facilities for the support function may present a complex planning and management task.