SAE International®

SURFACE VEHICLE/ AEROSPACE RECOMMENDED PRACTICE

SAE® JA1006 MAY2012

Issued 1999-06
Reactivated 2012-05

Superseding JA1006 JAN2004

Software Support Concept

RATIONALE

JA1006 has been reaffirmed to comply with the SAE five-year review policy.

Foreword—Historically, the supportability aspects of software have been given a very low priority in the overall program requirements. This was particularly prevalent during the acquisition phase, where funding and timing constraints were usually the top priorities. The result was inadequate product supportability, inadequate support funding, lack of good field data, and no meaningful analysis and optimization of possible support alternatives.

In order to alleviate these historical concerns, this document presents a top-level structured overview of an overall software support concept and the information associated with it. This document was developed by the Supportability Subcommittee of the Society of Automotive Engineers (SAE) G-11 Reliability, Maintainability, Supportability, and Logistics (RMSL) Software Committee (G-11SW). G-11SW and its different Subcommittees plan to develop several more detailed reports that together will form an integrated task guide for analyzing software supportability concerns, developing a software support concept, and acquiring/conducting software support task activities within the context of the system integrated logistic support process. The SAE G-11 RMSL Division has several other Committees that are developing system level standards and guidelines that will reference the software supportability reports as applicable. The United Kingdom Ministry of Defence efforts such as documented in reference [DEFSTAN0060] and the United States military standards and handbooks related to references such as [MILHDBK347], [MILPRF49506], and [MILHDBK502] serve as source material for this effort.

Development of this document has required dedication by a few participants and extended review by a wider audience of potential users. The professionalism of all these individuals and the support they received from their companies, governments, and other organizations is gratefully acknowledged.

Abstract

This document provides a framework for the establishment of a software support concept related to the support and supportability of both custom-developed and Off-the-Shelf (OTS) software. This document provides information needed to understand the support aspects that should be covered by a software supportability program.

This document has general applicability to all sectors of industry and commerce and to all types of equipment that contain software. The target audience for this document includes software acquisition organizations, logisticians, developers, supporters, and customers. This document is intended to be guidance for business purposes and should be applied when it provides a value-added basis for the business aspects of development, use, and sustainment of support-critical software.

TO PLACE A DOCUMENT ORDER: **Tel: 877-606-7323 (inside USA and Canada)**
Tel: +1 724-776-4970 (outside USA)
Fax: 724-776-0790
Email: CustomerService@sae.org
SAE WEB ADDRESS: http://www.sae.org

SAE values your input. To provide feedback on this Technical Report, please visit http://www.sae.org/technical/standards/JA1006_201205

TABLE OF CONTENTS

1. Scope—This SAE Recommended Practice provides a framework for the establishment of a software support concept related to the support and supportability of both custom-developed and Off-the-Shelf (OTS) software.

This document complements SAE AIR 5121, JA1004, and JA1005 by providing information needed to understand the support aspects that should be covered by a software supportability program. It should be noted that particular information indicated here should not be considered a complete list of all aspects of the support concept. In particular, the information should not be confused with a list of data elements.

This document has general applicability to all sectors of industry and commerce and to all types of equipment that contain software. The target audience for this document includes software acquisition organizations, software logisticians, developers, supporters, and customers. This document is intended to be guidance for business purposes and should be applied when it provides a value-added basis for the business aspects of development, use, and sustainment of support-critical software.

2. References

2.1 Applicable Publications—The following publications form a part of this specification to the extent specified herein. Unless otherwise indicated, the latest version of SAE publications shall apply.

2.1.1 SAE PUBLICATIONS—Available from SAE, 400 Commonwealth Drive, Warrendale, PA 15096-0001.

SAE AIR 5121—Software Supportability - An Overview
SAE JA1004—Software Supportability Program Standard
SAE JA1005—Software Supportability Program Implementation Guide

2.1.2 OTHER PUBLICATIONS

[DEFSTAN0060] Defence Standard 00-60, “Integrated Logistic Support,” (Part 3) “Logistic Support Analysis Application to Software Aspects of Systems,” Issue 2, March 31, 1998.
[IEEE1219] IEEE Standard 1219-1998, “IEEE Standard for Software Maintenance.”
[ISO12207] ISO Standard 12207, “Information Technology - Software Life Cycle Processes,” August 1, 1995
[MILHDBK347] MIL-HDBK-347, “Mission-Critical Computer Resources Software Support,” May 22, 1990.
[MILHDBK1467] MIL-HDBK-1467, “Acquisition of Software Environments and Support Software,” November 19, 2002.
[MILHDBK502] MIL-HDBK-502, “DOD Handbook Acquisition Logistics,” May 30, 1997.
[MILPRF49506] MIL-PRF-49506, “Performance Specification Logistics Management Information,” November 11, 1996.
[PEERCY1] “Assessing Software Supportability Risk: A Minitutorial,” IEEE Computer Society, Proceedings of the Conference on Software Maintenance, 1987, September 21-24, 1987.
[PEERCY2] “A Software Maintainability Evaluation Methodology,” IEEE Transactions on Software Engineering, 7 (1981), 4, pp 343-352.
[SEICMM] Paulk, M., Curtis, B., Chrissis, M., Weber, C., “Capability Maturity Model for Software, Version 1.1,” CMU/SEI-93-TR-024, Software Engineering Institute, February 1993.

3. ***Software Support Concept***—Any attempt to achieve supportable software requires the definition of a software support concept, that is, 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. Software supportability should be addressed as an inherent aspect of the major development processes (e.g., project management, engineering, configuration management, quality engineering, and integrated logistics support).

This section describes a framework of major elements that should be addressed by a software support concept. This framework provides the guidance from which a concept can be derived for specific software applications. Emphasis and inclusion of elements of the framework will depend on many factors, and the intention is that the framework is easily tailorable for different application domains and scalable in its use. Software items that have significant support requirements should be covered by a support concept.

Project management should address software support concerns in reviews and the transition of software from development to support. Design for supportability requires appropriate engineering processes to architect the software so it can be easily modified when changes are required. Configuration management should be a process that easily transitions from development into support. This requires that configuration management processes for software and hardware components are integrated and linked to the logistics activities of product packaging, handling, delivery and installation, and problem reporting. Quality engineering should ensure that planned supportability characteristics are designed into and can be maintained throughout the life of the software product. Development evaluations should provide evidence that adequate characteristics have been designed into the software. Integrated logistics support should ensure that software support concerns are an inherent part of system support.

This document identifies three main elements that taken together, form the overall concept of software support: Support Profile, Support Functions, and Support Classes, as illustrated in Figure 1. The three main elements are briefly summarized in this section and described in more detail in the following sections.

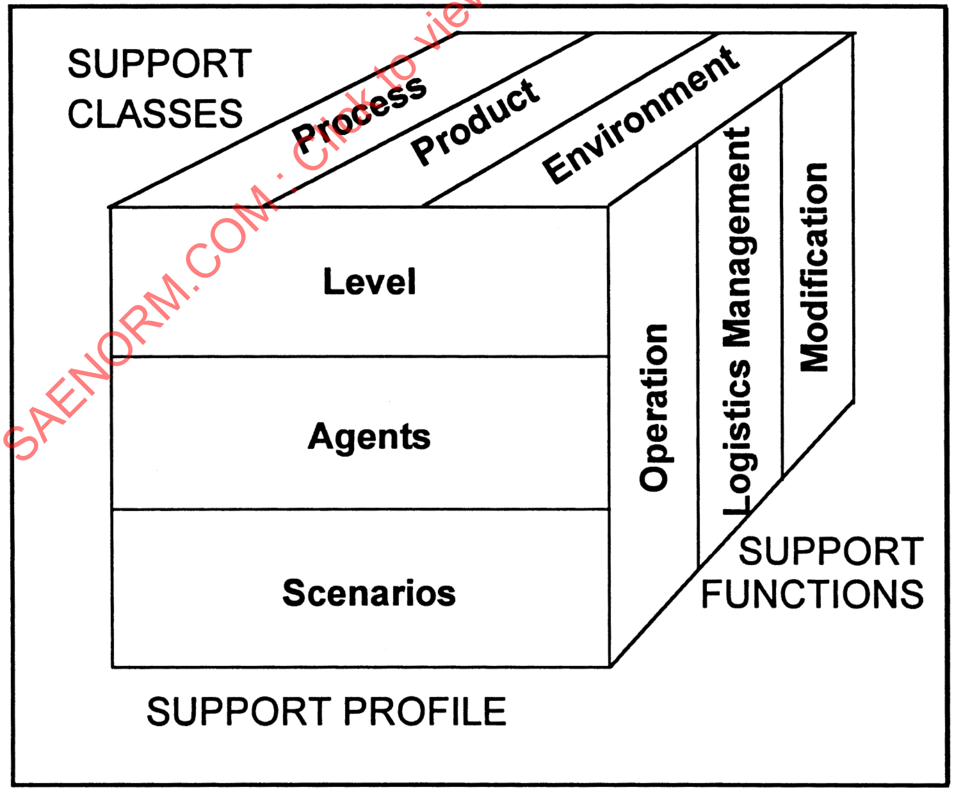

FIGURE 1—SOFTWARE SUPPORT CONCEPT

3.1 **Support Profile**—Covers the overall understanding about how software support should be addressed. The Support Profile consists of three distinct aspects:

a. Support Level—The different (generic) sites where support will be carried out, their capabilities, resources, and overall role in the support process.
b. Support Agent—The generic roles taken by organizations and groups of people that are involved in software support, and the responsibilities and information they require.
c. Support Scenarios—The individual tasks required to carry out all support processes, their interrelationships and sequencing and the people, organizations, resources, and infrastructure required to carry them out.

3.2 **Support Functions—**Refers to the different activities required to be carried out during the software's operational use. Within this approach, software has three distinct, but closely linked, functional support areas:

a. Operational Support—The support activities required by users and operators of the software during application use;
b. Logistics Management Support—The support activities required to establish or restore the operational capability of the software or to provide distribution and control of an upgraded software product; and,
c. Modification Support—The support activities required to modify the software under controlled conditions to correct defects, upgrade performance capability, and adapt to environment changes.

3.3 **Support Classes—**Cover the main quality characteristics of the major factors affecting software supportability. In principle, three factor classes are identified:

a. Processes—The inherent quality characteristics of the support processes that affect the capability to accomplish the required support functions;
b. Product—The inherent quality characteristics that are designed into the software product that affect the capability to accomplish the required support functions; and,
c. Environment—The inherent quality characteristics of personnel resources, support systems, and physical facilities that affect the capability to accomplish the required support functions.

4. Software Support Profile—The software support profile is driven by the location of support services (level), by the various types of personnel or organizations that conduct support activities (agents), and by the different sequences of activities and associated resource requirements that are needed to achieve the required support functions (support scenarios).

4.1 **Software Support Level—**Four categories (Organization/Field, Intermediate, Depot, and Vendor) can be used for the delineation of where software support is performed. Each of these different levels would typically accomplish different support tasks, but there can be various combinations of levels (including elimination or merging) depending on the application. Other terminology may be more appropriate for commercial applications but the descriptions would be similar.

4.1.1 ORGANIZATION/FIELD LEVEL—This support level corresponds to the site where the software is actually used and is oriented mainly towards operation/servicing and basic recovery. Only in extremely rare cases would this level resort to modification, with the exception of relatively trivial programs. The staff required for this level would usually require only elementary computer-and/or software-related skills.

The definition of this level encompasses all organizational aspects of the location, such as a description of the activities to be carried out and the operational procedures to be followed. Resource requirements would also be included, covering such aspects as user manuals, support equipment, and staff numbers and skill levels. If necessary, these requirements would be complemented by the identification of the means required to comply with these prerequisites (e.g., training courses).

4.1.2 INTERMEDIATE LEVEL—This support level corresponds to a site that is oriented mainly towards providing support for operational purposes (e.g., user queries, hot-line), inter-organizational support (e.g., management of computer networks) and limited modification activities (e.g., parameterization of software, mission data preparation). Staffing at this level would require more experienced personnel in computer operation, including extensive training in the software products being supported.

Apart from the site-specific information, it is important to identify applicable software support activities and the relationships of these activities to activities at other support levels. Configuration information is very important as it is necessary to know the build version of the software and associated platforms at operational sites. The limitations or constraints imposed by the operational sites (e.g., maximum turn-round time) might severely limit the quality of the provided service, unless this is properly addressed during software development.

4.1.3 DEPOT LEVEL—The Depot Level corresponds to some central location with many resources, and is usually oriented towards the full-scale modification of software, though it would also coordinate the work of the intermediate sites and provide the working interface to the vendor. Only in exceptional cases (e.g., in case of safety-critical situations) would operational problems be handled directly. The staffing at this level would typically include software engineers and computer scientists with an extensive knowledge about the software and system being supported.

This central site should have available all information about the supported product(s), including the full design documentation. Moreover, it would have all the necessary means (skilled staff, host and target computers, hot benches, etc.) to investigate and fully solve problem reports, or introduce new functionality in the existing software and release it for distribution to operational sites.

4.1.4 VENDOR LEVEL—A Vendor (in many cases, the Original Equipment Manufacturer or Software Developer) is also an alternative location for full-scale modification of software, both for developmental and non-developmental items. The Vendor Level is usually the selected level for outsourcing modification support, either due to its cost, need for very specialized skills, or even because the organization supporting the software does not have an in-house Depot Level. The roles and skills provided at the Vendor Level are similar to those of a Depot, but this level is characterized by being a separate source such as a contractor who will take care of the necessary support details. Usually, contractual arrangements are necessary that not only define payments for Vendor Level support, but also responsibilities and liabilities in the event of failure to meet support performance requirements.

4.2 Software Support Agents—The personnel and specific organizational entities (agents) that facilitate or conduct support functions are basic to the Software Support Concept. Though variants and combinations are always possible, depending on their roles and responsibilities, the support agent is usually either a customer (i.e., support services are provided to this agent) or a Supplier (i.e., support services are conducted by this agent). Customers may be categorized as Buyers of the services and Users that actually receive the services. Supplier agents may be categorized as Supporters or Vendors. Supporters provide special purpose services to the customer that usually are visible and capable of being influenced by the Customer. Vendors provide more standard (e.g., Off-the-Shelf) products and services to the Customer that are not as easily influenced by the Customer. Support agents include both the organizations and the type of people working within those organizations in their individual roles. On occasion, the Customer and Supplier may be the same. There may be many support agents, roles may be partially shared, and their titles may be very application dependent.

4.2.1 CUSTOMER: BUYERS AND USERS—The Buyer is the agent that buys or pays for the development/support of the software. The Buyer wants the software to be delivered on time, within budget, and satisfy the needs of the User, on whose behalf this software is bought. Typical Buyer information includes budget, schedule, progress, quality, potential risks in the development/support and, of course, risk reduction actions. Supervision of the software development/support by the buyer would normally follow typical project management practice. In case of very complex systems it may be desirable for other support agents such as Supporters and Users to assist the Buyer as advisors or experts, in order to ensure that the end product will be satisfactory.

The main concern of the User is to keep the application operationally effective. This usually means application evolution and change, and the software may be the only means or tool to accomplish this during support. Users can have a great variety of backgrounds and skills may vary widely among different organizations and business areas. Users may have very little role in software support to a fairly significant role depending upon the application.

Users initial interest is concerned with installation of the application. Concerns include the particular hardware and software related platforms. The next major concern is about how to use the application software, specifically if it requires special skills or is complex to use. Once in operation, the User wants an (elementary) problem diagnostic and solution capability, along with a quick way of reporting and solving the problems that cannot be resolved by the User.

4.2.2 SUPPLIER: SUPPORTERS AND VENDORS—The Supporter is an agent who is charged to directly support the Users of the software. This may include supplying the software to Users as well as carrying out modifications approved by the Buyer. The main problem for the Supporter is to understand the software in order to be able to quickly respond to User Problems and Change Requests. The Supporter must understand the software's operational use within the general system context, know how the software was designed, and understand the reasons for implementation decisions and the constraints (e.g., safety risk class, timing, target environment) that lead to the decisions. The Supporter role may include standard software modification tasks as well as logistics management tasks such as distribution of the software to the field for installation. Or, there may be separate supporter roles to accomplish these tasks.

The vendor is an agent that corresponds to the Vendor level. The Vendor is typically associated with supplying OTS software products and related services. The OTS products and services may be commercial or may be somewhat more specialized for specific Customers. In most cases, the Vendor writes the software specification with only minimal (if any) assistance from Customers. The specification is derived from the more generic requirements that are established by Customer applications. Apart from the information that is collected and/or produced for internal design or testing purposes, the Vendor may produce management data for status reporting to a Customer. Other information might be required for quality assurance, software certification, or user support. However, and contrary to the Supporter agent, the Vendor typically does not provide the Customer with very much information about the software product and the resources that were required to produce it—as that information is in-house and may be proprietary. Providing this information to the Customer in a structured manner is problematic. Such provisions must be carefully planned through Intellectual Property Rights and Escrow arrangements in case the Customer wishes to assess other support alternatives than the original vendor.

The Supporter and Vendor roles may overlap and/or supplement one another depending upon the application. In some cases, the Supporter may be the same as the Vendor.

4.2.3 SHARED SUPPORT RESOURCES—One special case is when a combination of the Support Agents share support resources. Information for the Support Agents must indicate the heterogeneous support environments and/or skills required for the multiple roles. The necessary ratinalizaton among the different agents may require unique supportability characteristics or a radicaly different support concept. Supportability analysis should be used to evaluate options and achieve optimum decisions on design and support system issues.

When sharing resources, special attention should be given to elements such as conditions of use, priorities, work distribution, cost share and of course access to non-common facilities. These elements and the associated process should be defined early in the program so that they can be incorporated within the necessary contractual arrangements.

4.3 Software Support Scenarios—A support scenario is an integrated representation of resources and processes, showing the steps by which the necessary end results are achieved in response to a demand for support services. Modeling support scenarios is a basic procedure for defining software support requirements and building a complete software support concept for a system. Such analytical activity should be part of the broader software supportability program established for a system. Detailed guidance on a suitable modeling strategy is included in SAE JA1005.

The description of the software support concept for a particular system should include the validated support scenarios arising from the analysis undertaken in the supportability program. This is illustrated in the software support concept template provided at Appendix A.

5. Software Support Functions—The three major software support functions are Operational Support, Logistics Management Support, and Modification Support. Depending on the support organization structure and the type of application, these functions may be conducted by the same or different persons and at the same or different locations.

5.1 Operational Support—Operational Support is related to all the day-to-day support activities for the operational software system. These particular support activities are usually carried out by the User at Organization/Field Level and sometimes at Intermediate Level. The typical activities include Software Loading/Installation, Configuration, Recovery, and Failure Reporting. Training activities for the User may also be conducted as part of the Operational Support activities. Operational Support interfaces with Logistics Management Support for the reporting of problems and the delivery of new software baselines.

5.1.1 SOFTWARE LOADING/INSTALLATION—Software Loading covers software installation and configuration (if required), as well as (mission) data loading/unloading.

At the simplest level, say a stand-alone PC, installation might merely require having the software provided on a compatible medium with a simple installation manual. For large networked systems, the associated system management organization would typically install new or updated software onto centralized server processors, from which copies would be downloaded to individual user machines over a local or external network. For embedded software, the loading process is generally more like a servicing activity, requiring the use of special tools and support equipment to gain access, establish connections, and effect the transfer of the new program code into internal memory devices.

5.1.2 SOFTWARE RECOVERY—Software Recovery covers the basic diagnostic and recovery activities due to system or software shutdowns, that can be handled safely by the User without having to use more expert personnel or implement a software modification.

The recovery process is initiated following the recognition of a failure or fault occurrence. Subsequently, either by purely automatic means or in conjunction with operator intervention, a diagnosis will be undertaken. This might lead to a number of outcomes to immediately rectify the cause of the problem or to sustain full system operation by alternative means, or to revert to a temporary, lower operational capability. In the latter cases a separate support process would be initiated to ensure clearance of the underlying problem. Data about problems and recovery action should in all cases be recorded to enable off-line analysis of system support trends and costs.

Error messages and failure modes should be catalogued and compiled in some kind of user manual, including the diagnostic information/procedures and problem investigation and isolation guidelines. The determination of work-arounds and recovery procedures and necessary preventive tasks (e.g., preparation of periodic backups) for known problems should also be assessed. Where safety issues might be a concern, contingency procedures would be required in order to cater for potentially catastrophic shutdowns.

5.1.3 SOFTWARE PROBLEM REPORTING—In the event of a system failure, a problem report should be generated. The reporting procedure should include the type, severity, and operational circumstances of the [software] failure, and whether any other systems are affected. A [software] problem report should also provide configuration status information concerning the subject software and the platform on which it was running at the time of the incident.

The [software] problem report would be communicated to field support personnel who would investigate the problem further to establish its priority and determine any workarounds and any need for temporary operational limitations whilst the problem was resolved. If a problem could not be fully resolved by field support personnel, it would be communicated to intermediate/depot level supporters for detailed analysis and, if the problem was related to software, for determination of possible software modifications.

5.1.4 SOFTWARE DELIVERY—In the context of operational support, software delivery covers receipt of new software baselines by the User from the distribution and transportation organization controlled by logistics management. Receipt implies a number of actions involving both quality aspects (e.g., checking the condition and completeness of delivered material) and validation aspects such as checking configuration status information and confirming compatibility between the delivered software and the system intended to receive it.

For these purposes, configuration information (e.g., system compatibility, software part number, NATO Stock Number) and licensing information is essential for the correct identification and installation of the delivered item, which might be further complicated by the confidential or classified nature of the software product.

5.1.5 OPERATIONAL CONFIGURATION CONTROL—Configuration Control is an important issue in the field. A specific item of software may only work on certain hardware variants, or with certain combinations of variants. On the other hand, a piece of equipment may have different items of software associated with it, depending on the function it has to carry out at a certain moment. Under these circumstances, it is important to know what is compatible with what, and whether apparently insignificant decisions (e.g., to store the equipment with/without the software installed) might become a major factor in the operational flow, and hence in the desired response times. Users in the field require appropriate management procedures to ensure that these issues are effectively controlled in day-to-day operations.

It is also important to be able to address potential interoperability problems or constraints as soon as possible, and have prepared "fallback" system configurations in case of system or software problems. Operational Configuration Control can support compliance with software licensing requirements.

5.2 Logistics Management Support—Logistics Management Support covers more than the traditional Product Support, adapted to software. This support bridges the gap between Operation Support and Modification Support. Logistics Management Support activities are performed across all levels and carried out by both Users and Supporters. The prime activities covered by Logistics Management 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 (shared with Operational Support); and, the provisioning for user support (e.g., through training and documentation).

5.2.1 PROBLEM REPORTING, CORRECTIVE ACTION, AND SYSTEM EFFECTIVENESS EVALUATION—Logistics Management provides the infrastructure and resources to channel problem reports to the responsible authority, establish work priorities, request the necessary corrective actions, and return software to users, if necessary with interim restrictions or workarounds while a final "fix" is implemented.

Allied to problem reporting, the Logistics Management function may also involve the continuous monitoring of system effectiveness and support costs. The objective would be to identify possible needs to introduce changes to sustain the optimum software performance and supportability throughout the life of the system.

5.2.2 SYSTEM/SOFTWARE CONFIGURATION CONTROL—System/software configuration control can become quite complex when there are a variety of customers and different software variants available for these customers at any given time. The system/software configuration control must be traceable to the configuration of associated hardware, software, and system products. An updated software product may not be compatible with all variants of the same embedded hardware element, or with other software packages designed for the same system.

System/software Configuration control is important for ensuring that problems can be traced back to the proper system configuration, and that individual customers receive the appropriate product. It should also save money for the developer as it will permit the implementation of a single change over multiple configurations, with consequent savings in testing and documentation.

5.2.3 DELIVERY, INSTALLATION, AND CHECKOUT—This activity is closely related with that described in 5.1.4, but in this case, the tasks are carried out by the Supporter, not the User. In particular, after a new software release, the modifying organization will be concerned about the best way of providing its product to the User, which in some cases might be a single client or, in the case of OTS products, millions of clients.

Relevant aspects will include such matters as packaging, transportation media, and distribution. These factors should be considered early on in any development program, and the appropriate measures described within the support concept.

5.2.4 USER SUPPORT—Where users have to interact on an intensive or regular basis with a system's software, they will require support to deal with everyday matters concerning software installation, set-up, and operation (both in normal and fault conditions). The developer and/or supporter will therefore normally provide a channel through which problem reports, queries, help, or suggestions for improvement can be communicated to/from the user.

User Support might be provided through a specific service contract, or in some cases (OTS software particularly) it may be a standing service made available to all purchasers. Whatever the approach, there is a need to strike the optimum balance between cost, user needs, and the broader commercial benefits of establishing a reputation for good quality support.

5.3 Modification Support—Modification Support covers all aspects related to implementing changes to software. This activity is carried out by the Supporters, usually at Depot or Vendor Level, though some less critical activities may sometimes take place at Intermediate Level. Modification Support covers mainly the Software Modification Activity itself (e.g., change request through change implementation), as well as Problem Investigation and collateral tasks such as Configuration Management.

5.3.1 PROBLEM INVESTIGATION AND CHANGE ANALYSIS—The initial concern of problem investigation is usually to determine how the affected system can continue to fulfill its purpose, even with a limited capability. It may first be necessary to verify that there is a genuine technical problem, as opposed to merely user misunderstanding or mishandling. After the problem has been correctly identified and any appropriate interim measures put in place, an analysis will follow of how the system could be changed in such a way that the problem would be permanently resolved.

In all cases, it is necessary to have full technical documentation about the supported systems. For problem investigation, facilities will be needed to allow reproduction of the circumstances that led to the failure mode. In some cases, a simple debugger might suffice, in other cases, expensive equipment and hot benches might be required. Where new requirements arise, change analysis may involve some of the same facilities. However, the focus is more likely to fall on the use of appropriate tools for requirements and design development and for modeling the behavior of the future system.

5.3.2 CHANGE IMPLEMENTATION—Given that prior analysis has defined and verified the specific requirements for a software change, implementation would proceed through the normal development stages of design, coding and testing, and integration. The processes will be similar to those used for initial development, but there are distinctions and additional factors which need to be recognized when producing and managing a support concept.

The basic need of an organization responsible for implementing software changes to a fielded product is access to the full historical design record, and the means to control and account for modification activity. The design function itself is often constrained by structural, sizing, and operational performance characteristics of the existing software. Apart from the impact on the change development process, these factors may also act as a damper on innovation in design, and hence require some consideration in defining human resource requirements for software support.

Coding and testing of modified software components, at least at the lower-level unit stage, would normally be conducted using the same tools and processes as were applied for initial development. Support problems arise when such tools (compilers for example) become obsolescent or need to be superseded for other reasons. This should not be a frequent event over the whole life of a system, but it would have major cost implications and should therefore be considered periodically in support concept reviews.

Integration of software with its target processor, with the parent equipment/system and with other interoperating systems (where applicable) is as important for change development as for initial development. Within a support concept, it will be vital to ensure that facilities to duplicate or simulate the real operating environment for the software, and provide all necessary test inputs and stimuli, will remain available for as long as support is required.

5.3.3 CHANGE RELEASE—Change implementation finishes with the production of the successfully tested and certified executable file(s), which is then combined with other items to form a change ‘release’ package. The associated items may include data files, installation instructions, and user documentation. A support concept should define how such items are produced and replicated, and define the relationship between the change implementation organization and other support agents concerned with change release distribution.

5.3.4 CONFIGURATION CONTROL—Configuration control of software during modification is concerned with managing individual change development, controlling different software versions and their documentation, and maintaining compatibility between specific software products and hardware/software configurations. Information related to the control of the configuration of support tools (e.g., compilers) and equipment will also be required. A support concept should define the relevant standards and procedures which are used.

5.3.5 OTHER MODIFICATION-RELATED TASKS—Other associated tasks might be required in parallel to the actual software modification activities, either to reflect the modification into other products or supporting material, or to improve the efficiency of the supporting organization itself. The evident activity is the collection of metrics (volume and type of change, effort, time, cost) or historical data on software reliability, as well as a compilation of lessons learned that might be of use in future programs. Other activities might cover process improvement, or assessment/recommendations for Reuse.

6. ***Software Support Classes***—The Software Support Classes constitute those quality characteristics of the major factors of the software support infrastructure that influence whether the support function can be accomplished. Three Software Support Classes are discussed in the following paragraphs: process characteristics, product characteristics, and environment characteristics. Such characteristics are discussed in [PEERCY1] in the context of software supportability risk.

Characteristics that are specific to software with security, safety, and/or high-reliability concerns may strongly influence supportability. Specific procedures and tools may be required during the support process to ensure such characteristics are retained when changes are made. It is expected that supportability analysis would identify such critical software support significant items and ensure desirable characteristics are included in the appropriate support classes.

Supportability Analysis [SAE JA1005] is conducted to determine the precise characteristics of each support class that are appropriate for a specific application. The goals of this activity are to deliver a supportable product, ensure that the necessary support infrastructure is in place before the entry into service, and sustain the software support during the software operational life. The support concept captures which support class characteristics have been specified/achieved for a specific application.

6.1 Process Characteristics—A process is a sequence of task activities (with inputs, procedures, outputs, controls, resources) that accomplishes a specific function. The processes applied to the development and support of software will have features and attributes that may influence the capability to accomplish the required support functions. Such process characteristics may be application- or organization-specific, but will generally include the following:

a. Is the process documented?
b. Does the process follow an accepted standard or best practice?
c. Are there measures of time, resources, and effectiveness for the process?
d. Does the process effectively handle specialty engineering requirements such as for safety, security, reliability, supportability?
e. Does the process effectiveness address/satisfy Supporter and User needs?
f. Are the inputs, procedure, outputs, controls, and resources well-defined?

Evaluations of process characteristics can be conducted as part of Software Process Improvement programs using tools as described in [SEICMM].

The characteristics of these processes that are important for supportability include those aspects described previously as well as characteristics that depend upon the specific support tasks that comprise the process.

For example, specific characteristics of the loading/installation of interest might be the expected number of load/unloads required per time period, typical time to perform the operation, and whether there are validation mechanisms to indicate the load/unload is successful. Training characteristics of interest might include the type of delivery (classroom, computer-based), frequency of training by type of support function, certification criteria for the training, and pass/fail criteria for students.

As another example, the characteristics of interest for problem reporting and corrective action may include frequency of backups, multiple backup copies stored at different sites, recovery time, and effort required under non-critical and critical situations, automated mission data logging with trace to post-mission debriefing data, and well-defined steps and assigned responsibilities for automated tracking of problems and change requests from initiation to completion.

6.2 Product Characteristics—Certain supportability characteristics should be designed into the software product artifacts (requirements, design information, source code, test plans, and related development information) and external packaging/handling mechanisms. Various measures of product characteristics can be found in [PEERCY2]. Many of these product characteristics affect the supportability of the software.

6.2.1 SYSTEM INTERFACE—The application domain and type of software may drive system interface support requirements that can only be satisfied if certain software product characteristics exist. Such support requirements may include maximum robustness, years of unsupervised operational use, remote download of software updates, response (quick or slow) time for human operators, response time (emergency or normal) for change request, and frequency of block release updates. Software product characteristics to facilitate achievement of these requirements might affect design of the software load form, replication and installation features, interfaces for loading mission, and software/hardware interface mechanisms to facilitate the logistics management functions.

6.2.2 INHERENT QUALITY—The inherent quality characteristics of the software product artifacts are usually very dependent on how well (or how bad) the software has been designed. Even if the software is well designed, these characteristics can be easily corrupted by bad software support practices.

Important quality characteristics that affect supportability include modularity, simplicity, descriptiveness, consistency, expandability, and instrumentation [PEERCY2]. There are many lower-level indicators and metrics that can be used to define these characteristics. For example, cohesion and coupling are typical indicators of modularity. Use of object-oriented methods may result in software product that is more modular and expandable. Any size or count metric is generally related to simplicity, and fewer is better for supportability. Some simplicity indicators include number of independent paths, number of lines of source code, number of modules, and number of operators and operands. Designing the software to improve these quality characteristics will improve the supportability of the software product.

Certain programming languages include features that can improve the likelihood that the software products implemented in these languages will be more supportable. Language capabilities that improve supportability include strong typing, information hiding, reduced use of global entities, instrumentation of execution trace and variable values, and trapping and processing of exception conditions.

The form and content of software documentation can have a significant influence on the supportability of the associated software product. The more complete, consistent, and descriptive the documentation is, the more likely that software modifications will be understood and correctly implemented without inserting more faults. Furthermore, the use of electronic media for documentation, and integrated tools to automate changes and traceability, will aid the analysis and implementation of software changes and the concurrent updating of the documentation itself.

The internal product characteristics can also be affected by changes to the support environment. For example, a compiler or a design tool updated for new version release may alter the internal product characteristics and require modifications. Import/export facilities of support systems (e.g., to support adaptation and migration) become important in order to avoid corruption of the software product when support systems change. Design features of software components to isolate impact due to changes in the support or operational environment are important supportability characteristics of the software.

6.2.3 OTS PRODUCT SUPPORTABILITY CHARACTERISTICS—The supportability characteristics of OTS software products are the same as product characteristics described in 6.2.1 and 6.2.2, but the supportability focus for the supplier and the customer of the OTS software may be somewhat different. The supplier is concerned with the product supportability characteristics from the perspective of being able to provide cost-efficient upgrades - essentially the inherent quality characteristics. The customer is concerned with the product supportability characteristics from the perspective of capability to easily integrate and validate the OTS software within the operational domain - essentially the external interface characteristics.

6.3 Environment Characteristics—The Environment in which the software support is to be conducted is a major support factor. The support environment has the defining characteristics of Personnel, Support Systems, and Facilities that are required to accomplish the necessary support activities.