Ministry of Defence
Defence Standard 00-60(Part 3)

Issue 3 Publication Date 24 September 2004

Integrated Logistic Support

Part 3 : Guidance for Application of Software Support

AMENDMENTS ISSUED SINCE PUBLICATION

AMD NO	DATE OF ISSUE	TEXT AFFECTED	SIGNATURE & DATE

Revision Note

No substantive change in content.

Details of the changes in all Parts can be obtained by contacting the 00-60 Help Desk on (44) (0)1489-571491 or 0060@bmtrcl.demon.co.uk or via the D Stan Help desk or web site.

Historical Record

May 96 - Initial Issue of Parts 0 to 3, 10, 11 and 20.

Sep 96 - Up-Issue of Part 20 with amendments to Part 0; Issue 1 of Parts 21 to 25.

Mar 98 - Up-Issue of all Parts.

Sept 04 – Up-Issue of all Parts.

Arrangements of Def Stan 00-60

The proposed arrangement of the complete series of Defence Standards comprising Def Stan 00-60 is shown below.

Part 0 -	Application of Integrated Logistic Support (ILS)	-	Defines Requirements for the implementation of ILS
Part 1 -	Logistic Support Analysis (LSA) and Logistic Support Analysis Record (LSAR)	-	Defines the MOD requirements for implementing Logistic Support Analysis (LSA) and the LSA Record (LSAR)

Part 3 -	Guidance for Application of Software Support	-	Defines the MOD LSA for Software requirements
Part 10 -	Electronic Documentation	-	Defines the MOD Electronic Documentation requirements.
Part 20 -	Application of Integrated Supply Support Procedures	-	Defines the MOD requirements for implementing Supply Support within ILS.
Part 21 -	Procedures for Initial Provisioning	-	Defines the MOD requirements for implementing Initial Provisioning
Part 22 -	Procedures for Codification	-	Defines the MOD requirements for implementing Codification

Collation page

INTEGRATED LOGISTIC SUPPORT

PART 3: GUIDANCE TO THE APPLICATION OF LSA TO SOFTWARE ASPECTS OF SYSTEMS

PREFACE

This Defence Standard supersedes
Def Stan 00-60 (Part 3)/Issue 2
dated 31 March 1998

i This part of this Defence Standard provides information and guidance on the application of Part 1 of this Defence Standard to the software aspects of Ministry of Defence (MOD) procurement programmes for defence materiel.

ii This Defence Standard has been agreed by the authorities concerned with its use and is intended to be used whenever relevant in all future designs, contracts, orders etc and whenever practicable by amendment to those already in existence. If any difficulty arises which prevents application of the Defence Standard, the Directorate of Standardization should be informed so that a remedy may be sought.

iii Any enquiries regarding this Defence Standard in relation to an Invitation to Tender (ITT) or a contract in which it is incorporated are to be addressed to the responsible technical or supervising authority named in the ITT or contract.

iv This Defence Standard has been devised for use of the Crown and its contractors in the execution of contracts for the Crown. The Crown hereby excludes all liability (other than liability for death or personal injury) whatsoever and howsoever arising (including, but without limitation, negligence on the part of the Crown, its servants or agents) for any loss or damage however caused where this Defence Standard is used for any other purpose.

CONTENTS PAGE

PART 3: GUIDANCE TO THE APPLICATION OF LSA TO SOFTWARE ASPECTS OF SYSTEMS

Section One. General

0 Introduction

0.1 The UK MOD interpretation of Logistic Support Analysis (LSA) is defined within Defence Standard 00-60. This part of this Defence Standard explains how the requirements should be interpreted for application of LSA to software aspects of systems.

0.2 This part of this Defence Standard is to be read in conjunction with Parts 0 and 1 of this Defence Standard.

0.3 Software support covers all activities undertaken to allow continued use of software within a system; it is required:

(a) To maintain a system's effectiveness as changes occur to the environment within which it operates (including changes to the design of the system that contains the software).

(b) To rectify shortfalls in system effectiveness resulting from changes to user requirements.

(c) To rectify errors made in the software specification and development process.

0.4 Software support is an essential element of the support for all systems that have a software content. The whole-life cost of such support will vary according to the type of system and how the software is procured. However, experience has shown that for many systems, the cost of initial software development has been greatly exceeded by the cost of supporting the software during the system operational life.

0.5 Software support is an intrinsic aspect of the support for any system with software content. The same set of tasks and subtasks, that are defined in Part 1 of this Defence Standard for LSA of the overall equipment, may also be used to cover the analysis of software support.

1 Scope

1.1 This part of this Defence Standard describes the MOD policy and requirements for the application of LSA to software aspects of systems; it comprises 2 sections as follows:

(a) Section One provides an introduction, defines terms and abbreviations, introduces the software life-cycle and lists a number of related documents. Information on the management and planning of LSA for software is also provided, together with a description of a number of factors that have been identified as impacting on software supportability.

(b) Section Two describes the application of LSA to software aspects of systems. It provides tools, interpretations and guidelines to support the application of the LSA tasks and subtasks, described in Part 1 of this Defence Standard, to systems that include software.

1.2 This part of this Defence Standard is intended for use by customers and contractors. It is applicable throughout the equipment life-cycle from concept through to disposal and is applicable to all systems, and parts thereof, that have, or potentially have, a software content.

2 Related Documents

2.1 The documents and publications referred to in this Part of this Defence Standard are shown in Part 0 of this Defence Standard.

2.2 A list of sources of related documents is shown in Part 0 of this Defence Standard.

3 Definitions

3.1 A glossary of terms used throughout this Defence Standard is included at annex A to Part 0 of this Defence Standard.

4 Abbreviations

4.1 A list of abbreviations used throughout this Defence Standard is included at annex A to Part 0 of this Defence Standard.

5 The Software Life-Cycle

5.1 The Software life-cycle is the conceptual equivalent of the equipment life-cycle. As at the equipment level, there is a variety of possible approaches which might be taken to defining the software life-cycle. Whichever approach is taken for a particular procurement strategy, the compatibility and relationships between the equipment and software life-cycles should be clearly established, and be visible in project planning documentation.

6 Management and Planning

6.1 Management

6.1.1 The ILS Manager (ILSM) designated for the overall system should be responsible for ensuring that software support is given due consideration throughout all phases of the equipment life-cycle. LSA activities to cover software aspects of the equipment should be planned, performed, co-ordinated and integrated within the overall equipment LSA programme.

6.1.2 Responsibility for the conduct of ILS and LSA for software aspects of systems should be clearly defined within ILS teams. For programmes with significant software content, ILS teams should include software engineering specialists to conduct the software aspects of LSA.

6.2 Requirements for Planning

6.2.1 Software should be included within the scope of all equipment LSA programmes and plans. For large projects, separate sub-plans may be produced to cover LSA for software aspects. If software specific LSA sub-plans are produced, they should be referenced from, and co-ordinated with, the overall equipment LSA plans.

7 Tailoring

7.1 The approach to tailoring the LSA activity for software will be as described in the tailoring guidance included on the ILS web site.

8 Software Supportability Factors

8.1 There is a range of factors which might affect software supportability, and which will be relevant to the application of LSA. These factors are generally either attributes of the software item itself, or the associated development process, or of the environment within which the software is operated or supported. The factors are not all unique to software, and in some cases will be linked to system-level considerations. Those factors which are of key significance are listed below:

(a) Change Traffic.

(b) Safety Integrity.

(c) Expansion Capability.

(d) Fleet Size and Disposition.

(e) Modularity.

(f) Size.

(g) Security.

(h) Skills.

(i) Standardization.

(j) Technology.

(k) Tools and Methods.

(l) Documentation.

8.2 The following clauses describe each of the factors, explain their relationships to the subject system and the extent of their impact on supportability.

8.3 Change Traffic

8.3.1 Change traffic is a measure of the rate at which software modification is required. It is a complex function of requirements stability, software integrity and system operation. Change traffic will affect the volume of software support activity. Higher change traffic will require more software modification work. Change traffic may only be measured during actual use of the system. Before the software is in use, estimates may be made by comparison with similar applications and projections from requirements change and fault detection rate metrics taken during the software and system testing and trials. Any data available from comparable in-service systems on change traffic and effort will also be of significant value.

8.4 Safety Integrity

8.4.1 The safety integrity required of a software item will be determined by consideration of the safety criticality of the functions that it provides. Safety criticality relates to the likelihood of anomalies in the system causing accidents of varying severity. The overall safety criticality of a system should be established by the application of an appropriate hazard analysis technique. The criticality of particular software items will be consequent upon the partitioning of system functions in the system design. Designs should aim to minimize and isolate software which implements highly critical functions. System requirements should define safety criticality categories and specify appropriate software safety integrity levels. Various constraints and requirements for software development, testing and modification will be associated with each safety integrity level.

8.5 Expansion Capability

8.5.1 Expansion capability is an attribute of system design. It is concerned with the degree to which software may be modified without being limited by constraints on computing resources. Associated physical limitations, such as space, are to be addressed in the context of the parent system. Examples of constraints on computing resources are:

(a) Available memory.

(b) Processor performance.

(c) Mass storage capacity.

(d) Input/Output bandwidth.

8.5.2 Inadequate expansion capability might limit the scope for software modification or significantly impact on modification costs. Even simple changes might involve significant amounts of rework to overcome system limitations.

8.5.3 Limited expansion capability is of particular relevance in the case of embedded, real-time applications. In such cases it is normal to state spare capacity requirements as part of any procurement specification.

8.6 Fleet Size and Disposition

8.6.1 The number of equipments in use (fleet size), and locations at which software support is conducted, will have an impact on software supportability requirements and support costs. Significant sub-groups of users might generate requirements for variations of the software to suit their specific needs. The number and distribution of equipments will influence the magnitude of the software support task and the optimum location of the software support facilities. Moreover, large fleets are more likely to accumulate higher levels of equipment usage, thereby increasing the probability of fault detection and the identification of corrective change requirements.

8.7 Modularity

8.7.1 Modularity is an attribute of the low-level structure of a software design, and relates to the extent to which processes and functions are represented as discrete design elements. The modularity exhibited by a particular design will be a function of the engineering practices applied by the developer, and factors determined by the choice of design method, tools and programming language. However, in general the optimum approach to modularity will be one that balances functional and performance requirements against the need to provide an understandable and supportable design. Poor modularity might result in increased modification costs owing to the need to implement consequential changes in other parts of the software. Requirements for interface control and standardization might be used to influence the modularity of a system design.

8.8 Size

8.8.1 A number of metrics are available to quantify software size. The size of a software item might influence its supportability, both in terms of the level of change traffic expected and the resources required to implement a change. The size of the software within a system is dependent upon the application and the design solution.

8.8.2 Software requirements should state any constraints on the size of run-time software imposed by the system design. Many software support and supportability projections will be based on estimates of software size and complexity. Software development requirements should specify requirements for data collection and analysis to measure software size, and to verify any models or estimates of supportability parameters that depend on software size.

8.9 Security

8.9.1 The security classification of data, executable code and documentation might impose constraints and demands on the software support activities and/or the Project Support Environment (PSE). The main influence on a prime equipment will be to impose special handling requirements. These might limit access to the software and introduce design requirements which give rise to specific software support tasks and equipment.

8.9.2 The security classification of a software item will be dependent upon the application and the equipment design. Wherever possible systems should be designed such that highly classified software is physically segregated from all other software within a system. System security requirements should provide criteria for security classification of software items and should specify modification and handling constraints associated with such classifications.

8.10 Skills

8.10.1 Software modification will require personnel with appropriate software engineering skills. Requirements for particular skills might be associated with the application domain, the technology or the methods used. Skill requirements will be determined by the system design, the software design and the chosen software support policy. Skill requirements will have an impact on personnel and training needs.

8.11 Standardization

8.11.1 Standardization may be applied to the computing environment within which the software executes, and to the technologies and engineering processes used to develop the software and the associated software documentation. Standardization will benefit software supportability by reducing the diversity of tools, skills and facilities required.

8.11.2 The scope for standardization across a system might be constrained by the overall architectural design. Standardization requirements should be included within system and software requirements. Software standardization requirements might be less rigorously applied to software which will only be supported by the original developer utilizing existing facilities, personnel and equipment. Software standardization requirements should avoid constraining the design to software technology which has limited life expectancy or no clear evolutionary path.

8.12 Technology

8.12.1 Technology should be considered in respect of the software engineering methods and tools used in development and implementation together with the hardware and software aspects of both the host and target platforms. Technology issues might include: specification and software design methods and supporting tools; operating systems, programming languages and compilers; software test methods and environments; project specific tools and techniques; processing architectures. Requirements for the use of specific technologies might impose constraints on the system and software design solutions, they will also affect software engineering productivity and integrity.

8.13 Tools and Methods

8.13.1 The selection of tools and methods is dependent on the technologies used to develop and implement the system. The use of particular tools or methods might influence the software productivity and integrity achieved during software modification. The cost of acquiring and supporting tools should be carefully considered, since it might influence the

selection of the software support policy. Depending on the level of standardization achieved, the same tools and facilities might be used to support software items from one or more systems.

8.13.2 Selection of the tools and methods to be used during software development is a design decision and will form part of the design solution. The selected toolset will normally be incorporated in a PSE, which would also provide, depending on the chosen support policy, the basis for the post-delivery support environment for the software.

8.13.3 The tools within a PSE will require support throughout the life of the prime system to which they relate, since the tools themselves will experience change, upgrade and obsolescence. System developers should have a strategy for considering these issues in their initial toolset selection, and for on-going management of overall toolset effectiveness and integrity during the system life-cycle. Aspects which should be considered in respect of each potential tool supplier include the following:

(a) Commercial viability and track record.

(b) Quality of customer service arrangements.

(c) Product upgrade policy, particularly in respect of the maintenance of functional compatibility between succeeding software versions and the continued provision of support for preceding versions.

8.14 Documentation

8.14.1 The term documentation refers to all records, electronic or hardcopy, that relate to the requirements, specification, analysis, design, implementation, testing and operation of a software item. In order to ensure software supportability the documentation must be produced to an agreed standard and it must be available to the organization charged with delivering software support. Any software tools used in the creation of documentation must be included in the support facility and arrangements must be defined for their through-life support. Further guidance on documentation to facilitate software support, and how such documentation is generated, is provided at annex A.

8.14.2 Checks should be made at the end of the software development phase, and prior to formal acceptance, to verify that all documentation needed for software support has been identified and delivered.

Section Two. Application of LSA to Software

9 Introduction

9.1 The LSA strategy for any project involving software will apply LSA tasks concurrently to hardware and software elements. However, at the detailed level of task application, there are some distinct techniques and considerations for software, which are described in this section. The overall LSA process for software is illustrated in Figure 1. Specific amplification of LSA Tasks and subtasks, for application to software, are included in Part 1 of this Defence Standard.

9.2 Software support is affected by the overall equipment design and the software development process. The overall equipment design will determine the functionality and performance required of the software and any constraints within which it is to operate. The software development process will also influence the number of residual faults in the software at the end of development and the ease with which the software can be modified. Contracts for software procurement typically define the software functionality and any required development standards. Therefore, little scope is provided for optimizing designs for supportability. As a result, LSA for software aspects must commence before such contracts are defined.

10 Early Project Phase Activity

10.1 LSA tasks undertaken in respect of software items during the early, pre-design phases of a project offer the greatest potential benefits for supportability and for the achievement of optimum LCC. The realization of such benefits is conditioned, however, by the nature of the system and the procurement strategy (eg developmental versus COTS), and the extent to which design influence is possible.

10.2 In the concept and feasibility stages of a project, preliminary LSA tasks may be undertaken by the customer, or by selected contractors or by a combination of both. The output will be used to compile the supportability aspects of the formal system requirement, and the formulation of provisional software support concepts. The main areas of interest are as follows:

(a) Prime Equipment Technical Requirements. These fall into the categories of functional requirements (eg test and testability, system monitoring, data recording etc) and non-functional requirements (eg growth capacity).

(b) Development Standards. Contractual standards to be applied in respect of software design and development, quality and configuration management.

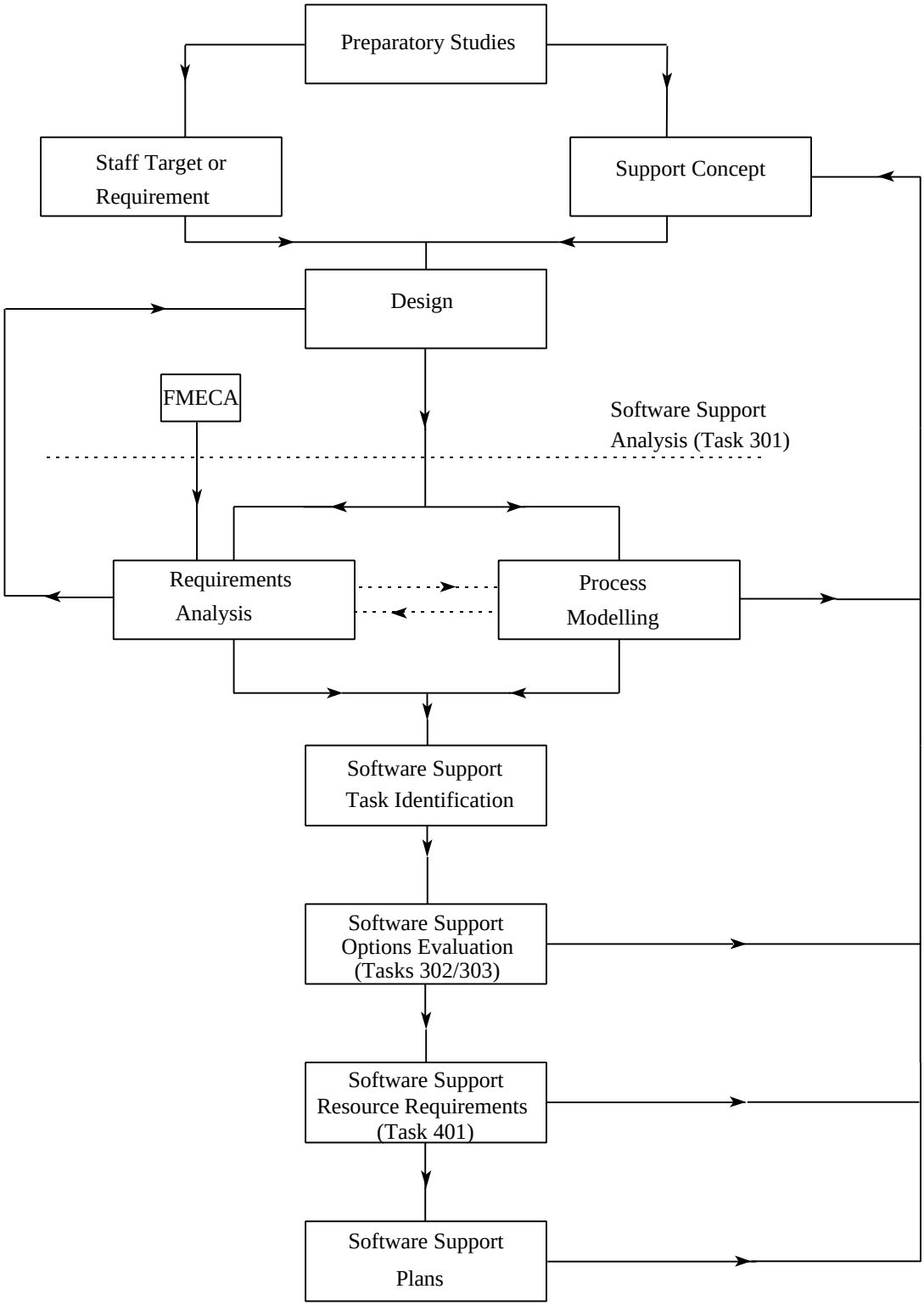

Figure 1 LSA Process for Software

(c) Support System Requirements. The attributes and performance/service levels required for the software support system.

10.3 These areas of a system requirement might be compiled through the LSA task mapping illustrated in Figure 2. (Note that the analysis issues listed are indicative rather than exhaustive, and will vary between projects). These LSA Task mappings for software aspects should be considered in conjunction with the equipment-level guidance given in the tailoring section of the ILS web site.

10.4 The target for the application of LSA tasks in the early project phases will generally be the application software for the envisaged system, and the system architecture (eg processing/data communications/security) over which the functionality will be provided. The primary purpose of any preliminary application of LSA tasks is the clear identification of requirements and constraints, and issues requiring further analysis. This will facilitate linkage into any subsequent LSA task iterations which might be performed. The format of any output reports will vary according to the project phase and the nature of the system, and will be specified in the contract.

10.5 Once software development or procurement has been initiated, the LSA process may proceed as illustrated in Figure 1 (subject to tailoring requirements). LSA tasks during and after software implementation are mainly concerned with the capture or definition of information needed to support the software.

REQUIREMENTS AREA	ANALYSIS ISSUES	RELEVANT LSA TASKS				
		201	202	203	204	205
Prime Equipment Technical Requirements	-Functional Aspects: -- Built-in test, diagnostics and monitoring. -- Recovery, reconfiguration, degradation, mode reversion. -- Data recording/access/upload/download.	*		*	*	*
	-Non-functional Aspects: -- Growth capacity: memory, processing, data communications.			*		*
Development Standards	-Design/development Methods and Tools: -- Requirements capture, systems analysis, design, code, test etc.		*	*	*	
	-Implementation Standards: -- Language, data communications, firmware, security, documentation etc.		*	*	*	
	-Management: Project/Quality/CM standards.		*	*		
Support System Requirements	-Readiness/responsiveness } -Release frequencies } -Intellectual Property Rights } -Use of customer resources } -Support concepts (through Task 302) }	*	*	*		*

Figure 2: Mapping LSA Tasks to the Generation of System Software Requirements

11 Software Support Analysis

11.1 Software Support Analysis (SSA) is a technique to assist in the application of LSA Task 301 to the software aspects of systems. The objectives of SSA are to identify potential requirements for, and constraints affecting, software support, and to identify software requirements to enhance system availability and supportability.

11.2 SSA comprises 2 broad activities: requirements analysis and process modelling. Guidance on the conduct of these activities is provided in this section. The necessary depth of application is variable and will depend upon the project phase, the type of procurement and the role of the equipment. The normal approach to the balance between costs and time for the analysis effort and the level of detail should apply.

11.3 The Applicability of Failure Modes Effects and Criticality Analysis (FMECA) and Reliability Centred Maintenance (RCM) to Software

11.3.1 FMECA is relevant to software, in that the analysis applied to the overall equipment should identify, for any particular failure mode, whether the associated functionality is provided by, or dependent on, software. Further guidance on this topic is given below, within the description of software support requirements analysis.

11.3.2 RCM is not directly applicable to software, since software 'failures' will always be the result of unintended features of the design, rather than wearout or breakage. However, depending on its assessed safety criticality, a software item might attract a safety integrity level (recorded in the LSAR). This will be traceable, through the Software Engineering LSA report, to the software tools and methods which have been selected to provide the required assurances for integrity/reliability.

11.4 Software Support Requirements Analysis

11.4.1 General

11.4.1.1 For a typical system there will be a number of potential software support activities, together with a number of different situations that could initiate that support. It is necessary to systematically examine such support initiators and their consequent support requirements, in order to identify the impact on the overall software support process. This should ensure that the necessary supportability provisions have been made within the system requirements, and in the design solutions intended to satisfy them.

11.4.2 Support Initiators

11.4.2.1 Support initiator is a term for any event or situation that gives rise to the need to perform software support. Support initiators should be grouped according to common operational, technical or user interface implications, (eg threat changes, hardware changes, operating difficulties etc).

11.4.2.2 System functional failure modes due to software design errors represent a primary group of support initiators, and will be determined through FMEA. The extent to which these failure modes should be broken down into sub-groups to facilitate the support task requirements analysis will vary from system to system. However, it is unlikely to be necessary to carry the consideration of failure effects down to the level of particular errors in the detailed design or source code.

11.5 Software Support Process Modelling

11.5.1 Process modelling is a technique to identify the processes, and their interrelationships, involved in software support. It is used to assist in the identification of software support tasks and the associated resource requirements, but is also a basic tool to support trade-off analyses of support alternatives.

11.5.2 Figure 3 provides a top-level generic model of the software support process. Further guidance on development of the elements of this model is given in annex B.

11.6 The Application of SSA

11.6.1 SSA will be carried out on Software Support Significant Items (SSSI) as part of Task 301. The following clauses describe how the techniques outlined above should be applied in practice.

11.6.2 Support Initiator Identification

11.6.2.1 The first approach to the identification of software support initiators should be to define a set of top-level groups, against which the subject software item might be assessed. System failure modes induced by software, as determined by FMEA, will constitute one group of support initiators; other examples are as follows:

(a) Changes in the Human-Computer Interface.

(b) Changes in the threat environment.

(c) Changes to parent system hardware/software.

(d) Changes to inter-operating equipments.

This approach should be adapted as necessary to take account of the individual nature of specific equipments, ie in the light of the role of the equipment and the impact of the software on equipment use and mission capability.

11.6.2.2 Having identified an initial set of support initiator groups, each one should be elaborated to a further level to identify the types of event or circumstance which might trigger a software support activity. For example, under a category of threat environment changes, the support initiator events might comprise: new target characteristics, new target defensive

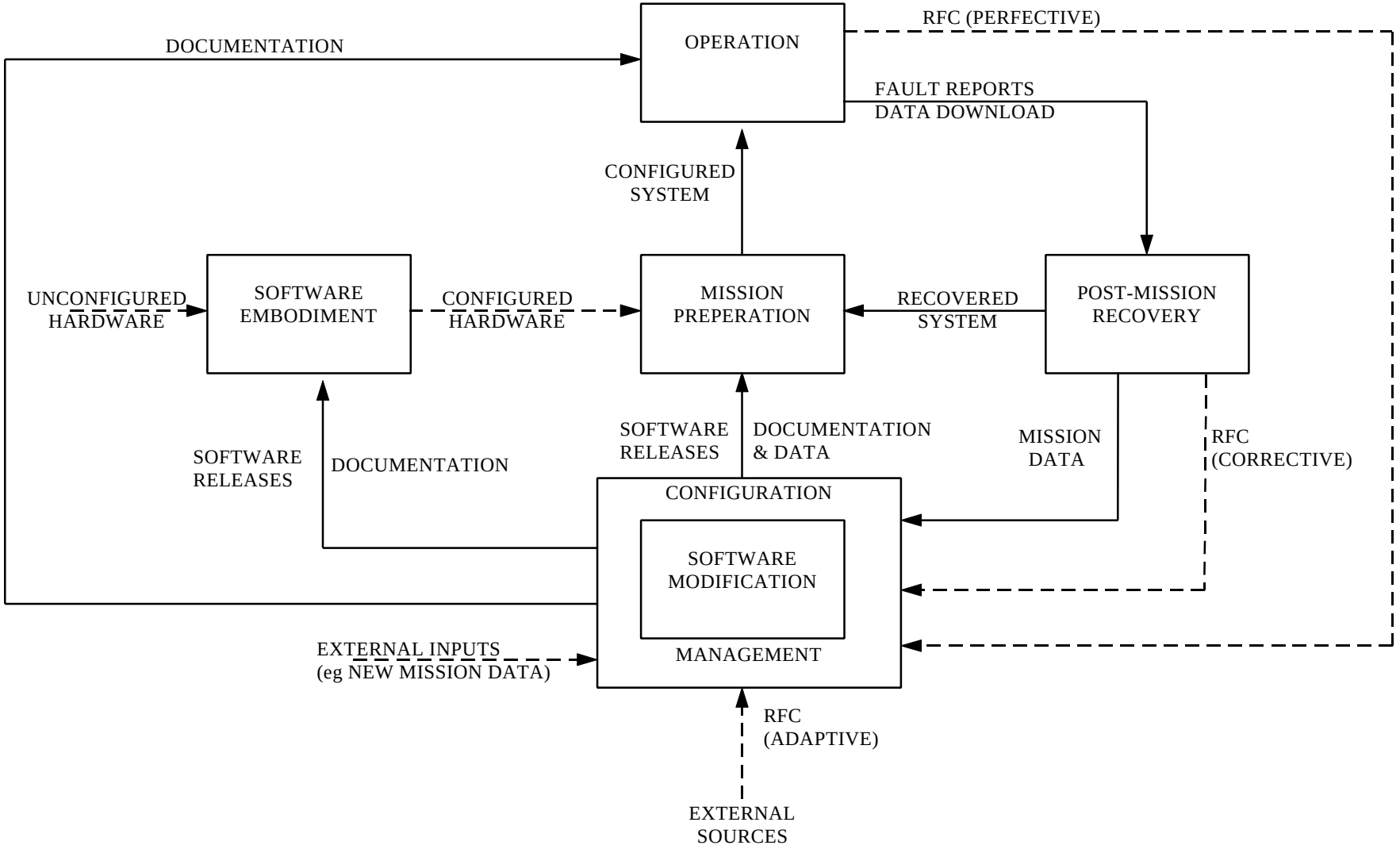

Figure 3: Generic Model of Software Support Process

systems, or a new theatre of operations. This analysis should also serve to validate that the top-level classification of support initiator groups is correct and appropriate.

11.6.3 Process Modelling

11.6.3.1 The analysis of support task requirements will be assisted by the creation and development of software support process models, as described in this section and at annex B. For each deliverable software item, process models should be generated taking account of any support concept constraints and requirements. Such models provide the context for considering the software support activities which flow from each defined support initiator and should be elaborated as the support task analysis progresses. Eventually, a validated model will thus be derived for each SSSI, embracing all the required support processes relevant to that item. The models will also provide the basis for subsequent identification of the hardware, software and human resources to be applied at the various support locations.

11.6.4 The Analysis of Software Support Task Requirements

11.6.4.1 Having identified the set of potential software support initiators and created a provisional support process model, the next stage will be to determine the consequential support tasks for inclusion in the overall task inventory.

11.6.4.2 Failure Mode Considerations. Software design errors might manifest themselves as system failures when a combination of specific operating conditions cause an erroneous code path to be executed. This might result in a fixed or systematic failure mode (eg processing might halt, or a display might 'lock up'), or in an unpredictable transitory effect. In the case of fixed or systematic failure modes, FMEA will identify where these are software-induced, and will drive the provision of appropriate features in the system software design, eg graceful degradation, reversionary modes, reconfiguration etc. However, software errors might also cause transitory system faults which might appear in many different forms. These might not be amenable to pre-design analysis or classification. Each of these cases has specific support task impacts.

11.6.4.3 Software Support Task Identification. Support tasks will arise from the failure mode effects outlined above, and from any other classes of support initiator which have been defined for the subject system. These tasks will fall into the following categories:

(a) System operations and maintenance support tasks (ie those tasks which involve only the handling or transportation (physically or electronically) of software executable code and data).

(b) Software modification tasks (ie the implementation of changes to the design of a configured software item).

The support tasks within each of these categories are described in more detail below. However, in carrying out LSA for software, it is necessary to keep in mind the relationship between the above task categories, and the representation of software items in the functional and physical system breakdowns. (Refer to annex C for guidance on applying LCN to

software.) The basic concept is that support tasks which do not involve modification (ie design change) activity - for example, software up/downloading and transportation - should be analyzed against physical LCN items representing executable code or data. Software or data upload and download tasks should be recorded in the task inventory as actions applied to the appropriate hardware item. Support tasks which are associated with software modification should be analyzed against functional LCN items representing software design information.

11.6.4.4 System Operations and Maintenance Support Tasks. The broad types of task to be considered under this category are as follows:

(a) Operations Support Tasks. Software support tasks associated with system operation are initiated by the need to respond to software-induced failures during the course of a mission. Such tasks mainly fall into 2 categories: action to reload/restart software to attempt to recover system operation, and action to initiate support activity and/or to record information about the system condition/configuration at the time of failure. The latter is particularly important in the fault investigation of transitory effects, where the ability might be needed to reproduce the effect in an appropriate test rig or reference system.

(b) Post-Mission Tasks. Software support tasks carried out immediately after a mission will generally be concerned with data extraction (for operational/engineering analysis) and fault investigation, and non-logistics functions such as the sanitization of classified software/data. In respect of data-related functions, a range of data administration tasks might be required, eg the provision and upkeep of data which is routinely needed by a software program to fulfil its operational role. Fault investigation tasks will relate both to activities on the prime system, and to the deeper-level testing carried out on remote reference systems which replicate or simulate the prime system environment.

(c) Software Embodiment and Mission Preparation Tasks. On being released from the modification process, a software item may either be issued to first-line users/operators for direct loading into the system according to mission/operational requirements, or it may be issued elsewhere in the maintenance organization for embodiment in a system hardware assembly (ie in firmware). In the former case, embodiment is effectively part of mission preparation. In the latter case the software forms part of the overall configuration status of the parent assembly, and is loaded in the prime system as part of a hardware installation task. Where remote embodiment in firmware is the proposed software loading concept, the task analysis must be carried out in conjunction with LORA for the subject hardware item, to ensure that any necessary trade-off analysis of maintenance level for hardware and software support has been addressed.

11.6.4.5 Software Modification Tasks. These tasks will relate to the longer-term process to introduce design changes to a software configured item. Any support initiator may necessitate a design change if it generates a requirement for a change or enhancement to the software functionality. Software modification involves a number of potential subtasks, but the extent to which these will need to be recorded in a task inventory will depend on the software support concept for the subject item, and the form of the modification process model. As a

general guide, software modification subtasks will be broken down only to the level where distinct visibility of associated skills/personnel/resources is required.

11.6.4.6 Use of Task Codes and Verbs. The use of DED 427 task codes might require some interpretation in order to facilitate their application to software. Recommended mappings are given below in respect of the typical software support tasks covered in the foregoing description of SSA.

TASK CODE	INTERPRETATION
Remove/Fit	To cover the use of portable/removable data media by operations/maintenance personnel.
Load/Unload	To cover the up/download of system data and software for mission or test purposes. Also the up/download of programmable memory devices.
Set Up	To cover pre-mission preparation and validation of system software or data by operations/maintenance personnel.
Software Modification (UK-specific code)	The development and implementation of a design change to an in-service software item.

Subtasks should be identified in the normal way using DED 431 verbs. The UK-specific verb 'code software' has been introduced to cover the transformation from low-level design information into source code. It is recommended that the verb 'develop' should be used to cover subtasks concerned with all higher levels of design transformation, and 'implement' should be used to cover the transformation from source code to executable code and the build of an image or file which may be directly loaded into the relevant equipment.

11.6.5 Levels of Software Support

11.6.5.1 For software support tasks not involved with design modification, the normal categories for level of maintenance may be applied, as appropriate to individual customer policy. Hence software loading/embodiment might be recommended (as determined through LSA) to take place in conjunction with equipment maintenance at any level. Associated support facilities should be documented in the appropriate UK LSA 'Facilities' reports as defined in Part 0 of this Defence Standard, with reference as necessary to the equivalent reports for the hardware assembly at the preceding indenture level.

11.6.5.2 In respect of software modification activity a somewhat different approach is required, owing to the distinctive nature of the support tasks, and the fact that for any software deliverable item there would normally be only one support facility, at a single location. This might be at an operational unit, or with a contractor, or at some intermediate facility. The approach taken in this standard is to define the software modification function as 'Specialized Repair Activity' (see DED 277 and 427). Data element 427 is linked to LSA Report 628,

which should therefore be used as the standard means of defining the required support facilities for software modification of a particular software candidate item. An appropriate facility name should be used to distinguish such reports from those for other facilities.

12 Software Support Resource Requirements

12.1 Software support resource requirements will be analyzed and quantified in conjunction with the application of LSA Task 401 to the overall equipment. Software support tasks relating to the operational use of an equipment, and mission preparation and recovery, are essentially equipment-level tasks. A similar approach will apply to software embodiment tasks carried out at the equipment level. However, tasks concerned with software modification require some particular considerations, as outlined below.

12.1.1 Release Frequency. Specialist tools might be used to generate detailed predictions for the expected rate and volume of change to a software program. However, based on information generated by the Use Study, the assessment of comparative systems and the application of SSA, a planned release frequency may be determined. There is provision for this to be recorded in the LSAR. Unless the customer specifies otherwise, the release frequency should be a balance between the resources and costs for a range of release frequency options, and the need for the user to be able to take advantage of progressive improvements in functionality and performance.

12.1.2 Sharing of Resources Between Development and Support Functions. The nature of developmental software procurements often involves the continuation of development work for a considerable period beyond the initial in-service date of the parent equipment. Where concurrent development and support activity require some use of common facilities or resources, their capacity to satisfy the expected overall utilization requirements should be assessed. Alternative resource proposals should be made where necessary.

12.1.3 Critical Resources. Certain elements in a software modification facility represent 'bottlenecks' (ie single, critical resources) in the overall change development and implementation process. The resource requirements analysis should identify such items and estimate the average utilization factor. Remedial action should be proposed where necessary.

12.1.4 Consolidated Software Team Resources. Where it is recommended that support for a software item is undertaken by the customer, the support plan should provide a consolidated support team description covering all required functions and resources, including infrastructure aspects such as QA and CM.

12.2 Use of Software Metrics

12.2.1 A software metric is any attribute of a software product or its development process, about which data may be collected and analyzed. Such activity might fall into a number of non-logistics domains, such as project monitoring and process improvement, but will also be of value to the LSA process in helping to quantify support resource requirements.

12.2.2 Despite these benefits, it is difficult to define a standard set of software metrics which would be appropriate and usable for all types of MOD project. Hence the formal use of software metrics in this Defence Standard is limited to a set of DED, defined in Part 0 of this Defence Standard, which provide program sizing information. Customers might nevertheless specify or prefer the use of particular methods and tools for estimating software support resources and costs. Where no such preference is stated, the contractor should propose an approach to generating and analyzing appropriate detailed supporting data. In either case, the form and content of deliverable reports associated with such methods and tools should be defined in the contract.

13 Software Support Planning

13.1 Software support planning will be integrated with the equipment-level analysis of support requirements and support system alternatives. The activity of SSA represents the application of LSA Task 301 to software, and will consider relevant aspects of the output from FMEA (ie software-induced failure modes). LSA Tasks 302, 303 and 401 will be applied as described in Part 1 of this Defence Standard.

13.2 For software support functions not involved with software modification, the consideration of support system alternatives will be through application of the standard LSA process. However, for the application of Tasks 302/3 to the software modification function, contractors should take into account a number of special considerations.

13.3 Support Policy Options for Software Modification. A wide spectrum of support policy options for software modification is available, ranging from full contractor support at one extreme to full customer support at the other. Contractor support will normally be the default option, but for software which is expected to be particularly sensitive to operational/business change requirements, the software modification process might require an element of integrated 'intelligent customer' resource to participate in such functions as change analysis/specification /prototyping etc. Where such an approach is recommended, the trade-off analysis should consider the options for physical integration of customer and contractor facilities, and the most cost-effective balance between their respective resources. Other issues of significance to support policy are as follows:

13.3.1 Intellectual Property Rights Considerations

13.3.1.1 Constraints on Support Policy. Intellectual property rights (IPR) refer to the legal limitations obtained by the creator of a software design or product on the ability of third parties to use, replicate or modify that software. This is a key support policy issue, in that IPR might place constraints on a customer's ability to undertake software support in-house, or to consider alternative support agencies. Any such constraints need to be exposed and assessed prior to contract award.

13.3.1.2 Vendor Failure. The commercial failure of a COTS vendor, whilst generally a remote possibility, might have serious support consequences for affected systems. However, the extent to which contingency measures should be explored during LSA should be limited to

instances where there is considered to be relatively high risk (eg immature product, specialised functionality etc). In such cases the analysis would consider to what extent there were suitable alternative products in the marketplace, and whether arrangements could be established for the acquisition of the necessary IPR by an alternative support agency.

13.3.2 Location of Support. The location of a software support facility is a key support policy consideration. Whilst contractor support at the original development site will generally be the default option, the following factors should be considered in the analysis of all options under consideration for any particular system:

(a) Responsiveness to change requirements (ie can urgent software updates be implemented and brought into use by the user in timescales which satisfy his operational needs).

(b) Collocation with related system facilities (eg trials units).

(c) Collocation with other system software support facilities.

13.4 Use of LSA Reports. Although most LSA reports are directly applicable to software, a specific report has been devised - LSA-672 'Software Engineering Report' - to consolidate the key logistics aspects relating to a software item and its support requirements. This report is described in Part 0 of this Defence Standard. If it is considered that there would be benefit for a particular project in maintaining visibility of a defined subset of software-related DED, the use of an appropriately structured ad hoc report should be considered.

14 Applicability of LSA Tasks to Software

14.1 In broad terms, most LSA subtasks may be applied to software as they stand. This section has described the application of LSA subtasks in the context of an overall LSA process for software, and Part 1 of this Defence Standard gives the complete subtask descriptions for application to equipments overall. This includes annotations and additional information, where appropriate, to highlight any specific interpretation or activity which should be undertaken for software. Where any specific subtask description has no such modifications, the subtask should be interpreted and applied equally for hardware and software within the overall equipment context. Additional guidance on the application of LSA tasks to software elements of systems is provided in annex D.

14.2 Only 2 LSA subtasks are considered not to be applicable to software. The first - subtask 303.2.10 (Energy Trade-Offs) - is technically incompatible, and the second - subtask 301.2.4.2 (RCM) - is considered inappropriate for reasons which are discussed under the coverage of Software Support Analysis earlier in this section.

Documentation for Software Support

A.1 The generation of documentation to facilitate software support is included within the integrated approach to LSA and documentation which this Defence Standard provides. Specific measures concerning the creation and compilation of Data Modules (DM) for software are covered in Part 10 of this Defence Standard. The guidance in this annex is intended to describe the peculiar aspects of documentation relevant to the software modification task, since this activity, with its emphasis on design change, is significantly different to servicing and maintenance tasks for hardware.

A.2 Software modification can only be carried out effectively if the organization tasked with such activity has access to the necessary supporting information. The information required can be grouped into five areas: design, system, support environment and facilities, development process, and management. A summary of each of these areas is as follows:

A.2.1 Design Information. Software and system design information provides the engineering baselines on which the software modification process operates. Software design information includes specifications and all design representations (ie the design 'record') such as source code, structure charts, data flow diagrams etc. Relevant system design information would cover areas of interaction between the system software and hardware design, eg interface details.

A.2.2 System Information. A support organization will require system information in order to understand how the software operates within the system. System information would typically include a system description, details of system operation, user/operator/maintainer manuals, etc. The investigation of operating problems, and the assessment of the impact on the system of any proposed in-service software modifications (and vice versa), both require access to this type of information.

A.2.3 Support Environment and Facilities Information. Access to this class of information is essential if a support organization is to maintain effective control over the modification process, and the integrity of the end product. PSE information includes details on software tools used, project database, hardware requirements in terms of host and target, test harnesses etc. Facilities information describes the physical environment within which a PSE, and the associated technical resources required for software modification, are accommodated.

A.2.4 Development Process Information. It is necessary to ensure that software modifications are carried out in a manner which is consistent with the original design and development methodology. To this end, information on the software development process and associated software engineering standards is required. This will describe such aspects as codes of practice, programming guidelines, development standards etc.

A.2.5 Management Information. Management information to support software modification is related to the design process and associated management aspects. This may include such items as the following:

(a) Plans (Quality/CM/test/acceptance/safety etc).

(b) Development methodology and procedures.

(c) Personnel aspects (qualifications/experience and training needs for the various roles within the support organization).

A.3 Data Module (DM) Requirements

A.3.1 In LSA, software modification will be treated as a top-level support task, which may be broken down into subtasks (eg design, coding etc) as appropriate to individual projects. For documentation of the software modification task, a DM has been defined which will provide a description of the overall design change process, and identify or reference the relevant information sources as outlined above. Associated DM may be generated for subtasks under software modification, although in many instances the documentation required for support of a software item will mirror that which was generated and used during the initial design and development.

A.3.2 A DM has also been defined which will describe the transformation from a complete, updated functional design, into the actual executable image or data file that will be loaded into a system. This DM will provide the basis for necessary links between documentation of software upload tasks, which LSA will assess in the physical system breakdown against the relevant hardware item, and the documentation for the related design units in the functional breakdown.

A.4 Information Needs for In-service Change Management. During a system's in-service phase, users and operators will generate information concerning software problems and failures, as well as requests to improve the existing functionality. The responsible software support organization will need to define the form and content of such information, in order to establish an efficient user interface and to allow the change management process to be controlled effectively.

A.5 Information Sourcing. As indicated above, some elements of the documentation required for software modification (eg design documentation) will not be provided directly through DM, but will be procured by the customer, if required, through separate contractual arrangements. Separate arrangements might also be necessary to provide documentation to support the in-service change management function, although for software which will be fully supported by the contractor, much of this documentation will probably already be in place. However, for other support options involving customer or third party resources, the relevant Defence authority for system support will be responsible for arranging the provision of whatever documentation is required. This would include such entities as: Software Fault Reports, change control records, Requests for Change (RFC) etc.

Decomposition of Software Support Process Model

B.1 Introduction

In this annex the generic model of the software support process, introduced in clause **11.5.2**, is decomposed to give further guidance and examples of the internal structure of each of the constituent processes.

B.2 Configuration Management

B.2.1 Figure 4 illustrates the Configuration Management process model which covers those activities necessary to maintain configuration control over the software, its PSE and all associated documentation and data. The process includes the management of software modification, data administration and the co-ordination of the release of software items and data.

B.2.2 Inputs. The Configuration Management process receives data from external sources and from the Post-Mission Recovery process. The process also receives perfective RFC from the Operation process and corrective RFC from the Post-Mission Recovery process; adaptive RFC are received from external sources. Perfective RFC are intended to initiate software changes to enable software items to operate in an improved fashion; corrective RFC are intended to initiate the correction of errors in existing software items and adaptive RFC are intended to initiate software changes to enable software items to operate in a changed hardware or software environment.

B.2.3 Configuration Management Sub-Processes

B.2.3.1 Configuration Control. This sub-process covers the registration and evaluation of RFC and management of the authorization of associated work; it also co-ordinates the activities necessary to manage the issue and release of software items.

B.2.3.2 Software Modification. A description of the Software Modification process is given in clause **B.5**.

B.2.3.3 Software Library. This sub-process maintains a reference library of the configured items for use by any approved user. It implements the procedures for entering items into the library and withdrawing them for the purposes of both operational use and modification.

B.2.3.4 Data Administration. This sub-process covers the provision and upkeep of any loadable data required by the software.

B.2.3.5 Issue and Release. This sub-process implements procedures to register new software items and to issue and release them to approved users.

B.2.3.6 Replication and Distribution. This sub-process covers the replication and distribution of configured items to authorized users.

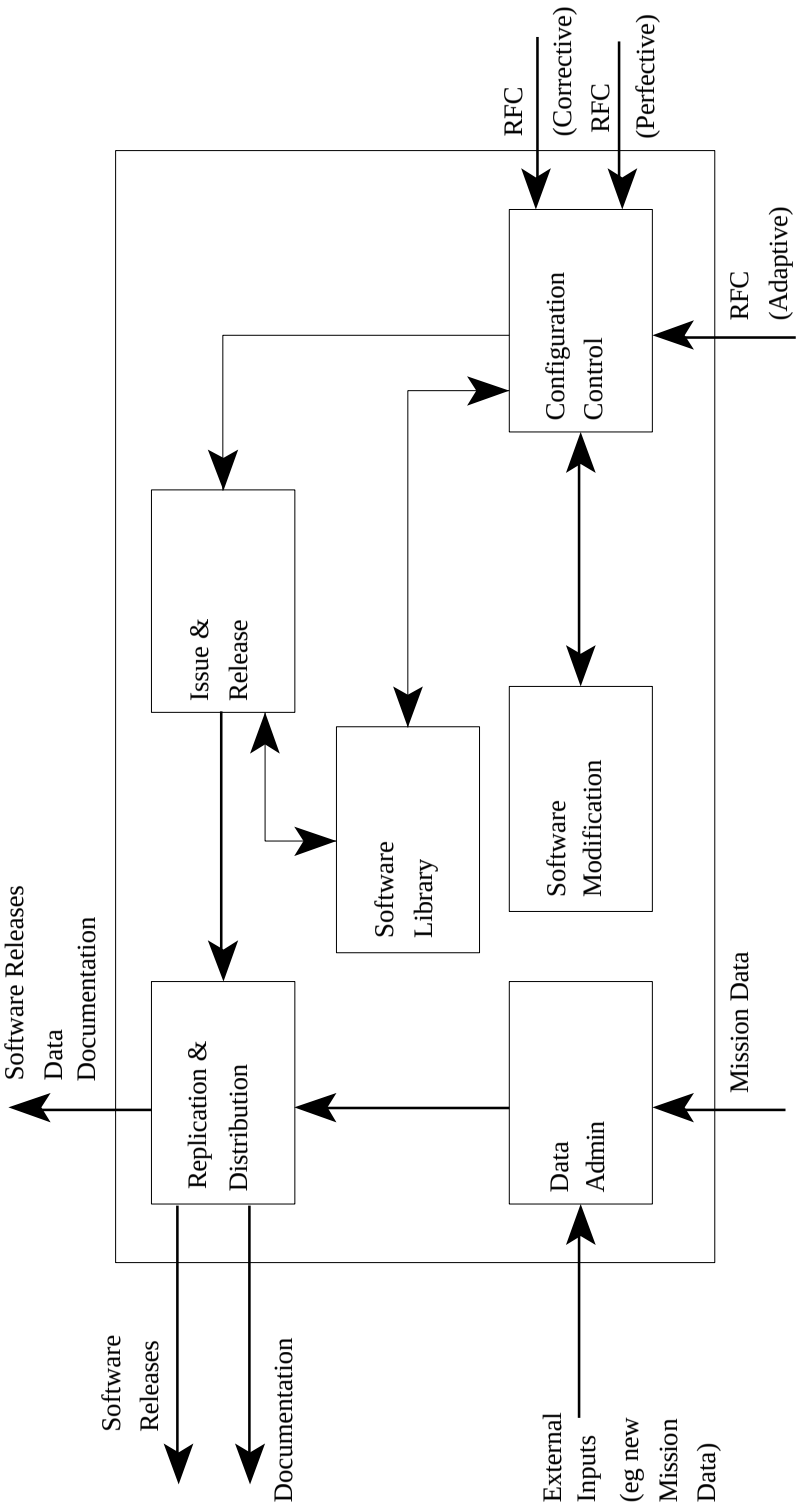

Figure 4 Configuration Management Process Model

B.2.4 Outputs. The Configuration Management process generates new software releases, data and associated documentation.

B.3 Operation

B.3.1 Figure 5 shows the operation process model. This illustrates those sub-processes, involved with the use of software in the system, which generate or are associated with support tasks. Normally this process and all associated sub-processes would be carried out by operational staff and not by members of an organic support organization.

B.3.2 Inputs. The Operation process receives a configured system, ready in all respects for use, from the Mission Preparation process. Documentation pertaining to the operation of the system is received from the Configuration Management process.

B.3.3 Operation Sub-Processes

B.3.3.1 Mission Usage. This sub-process covers the operational use of the software within its normal mission environment and includes simple operational procedures to boot, run or use the software within the system or equipment.

B.3.3.2 Performance Monitoring. This sub-process covers the monitoring and review of the operational use of the software to identify and report perceived faults and requirements for changes to the functions and facilities provided by the software.

B.3.3.3 Fault Recovery. This sub-process covers actions taken by the system operational staff, following a software failure, to recover system operation to its state prior to the failure (eg re-boot, re-load, re-run, re-start).

B.3.4 Outputs. The Operation process generates fault reports which are investigated in the Post-Mission Recovery process. Additionally, the process also generates perfective RFC which are input to Configuration Management process. Mission Data is also passed to the Configuration Management process.

B.4 Mission Preparation

B.4.1 Figure 6 illustrates the Mission Preparation process model; it covers those activities required to load software and data onto equipments to prepare them for operation.

B.4.2 Inputs. The Mission Preparation process receives recovered systems from the Post-Mission Recovery process and Configured Hardware from the Software Embodiment process. It also receives software releases, documentation and data from the Configuration Management process.

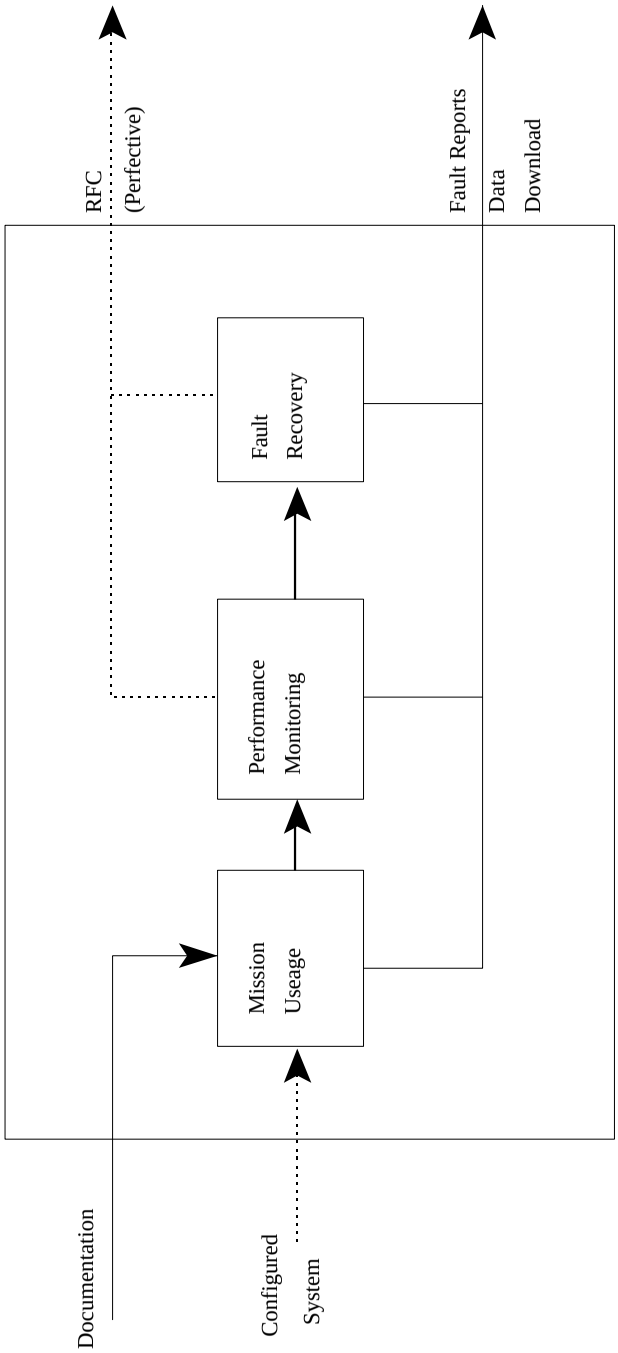

Figure 5 Operation Process Model

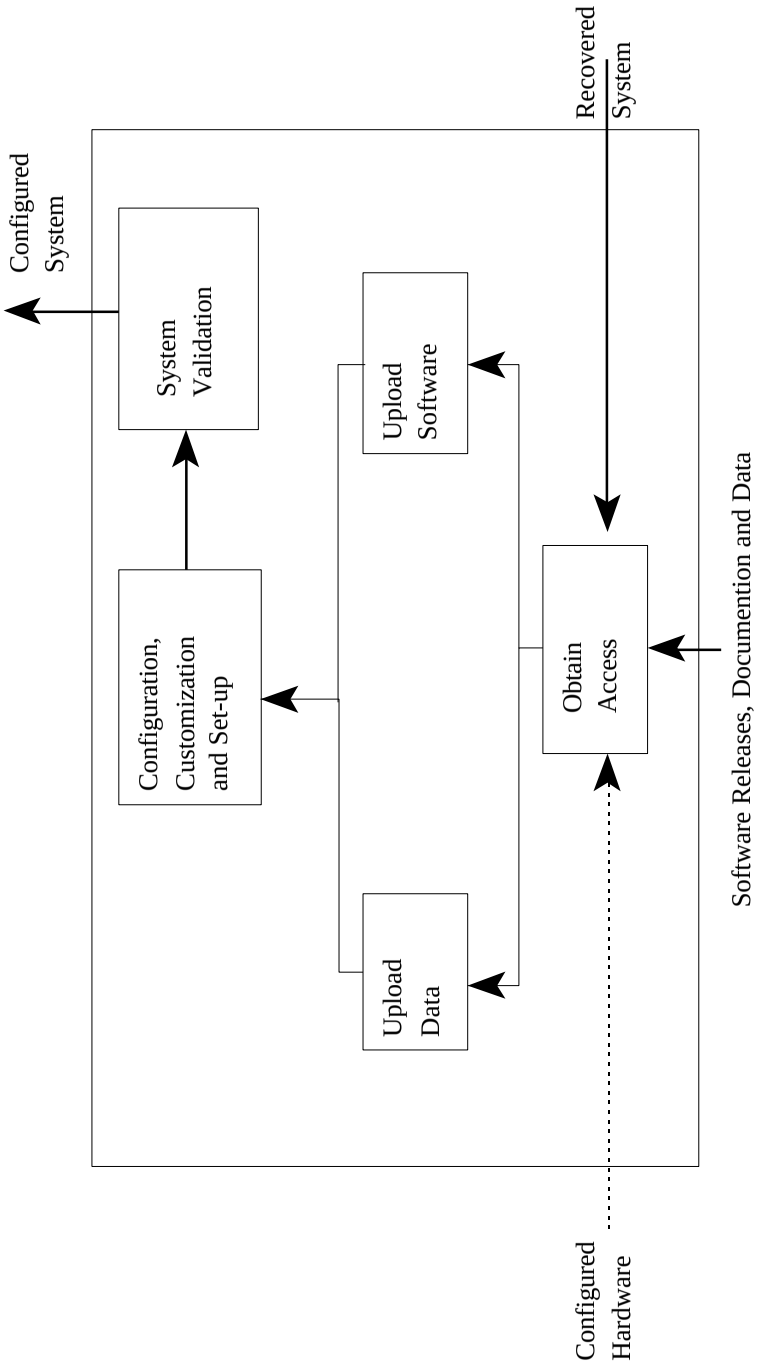

Figure 6 Mission Preparation Process Model

B.4.3 Mission Preparation Sub-Processes

B.4.3.1 Obtain Access. This sub-process covers any activities necessary to obtain access for the purpose of uploading data or software.

B.4.3.2 Upload Software. This sub-process covers the loading of software onto a system.

B.4.3.3 Upload Data. This sub-process covers the loading of data onto a system.

B.4.3.4 Configuration, Customization and Set-up. This sub-process covers any requirements for special tasks to be carried out when a new version of the software is loaded or when the software is to be configured for a new user, role or threat.

B.4.3.5 System Validation. This sub-process includes any activities necessary to validate the configured system before it is released to the Operation process.

B.4.4 Outputs. The Mission Preparation process generates configured systems ready for use within the Operation process.

B.5 Software Modification

B.5.1 Figure 7 illustrates the Software Modification process model which covers all activities necessary to modify software. The sub-processes within Software Modification are all equally applicable to perfective, corrective and adaptive RFC.

B.5.2 Inputs. The Software Modification process takes as its inputs RFC from the Configuration Control sub-process in the Configuration Management process.

B.5.3 Software Modification Sub-Processes

B.5.3.1 Change Analysis. This sub-process carries out the analysis of all RFC to determine their impact on the system or equipment. The sub-process will also define the required changes to relevant software documentation.

B.5.3.2 Test Planning. This sub-process covers the preparation, update and correction of test plans, schedules, and data for use in verification and validation of modified software. The sub-process defines the tests and criteria for unit testing, integration testing, regression testing, system testing and acceptance testing.

B.5.3.3 Design. This sub-process covers all activities required to amend the software design in order to carry out a software modification. This sub-process also defines the changes to software documentation that will be required.

B.5.3.4 Coding. This sub-process covers the amendment and/or production of software code to implement a software modification; it also defines required changes to relevant software documentation.

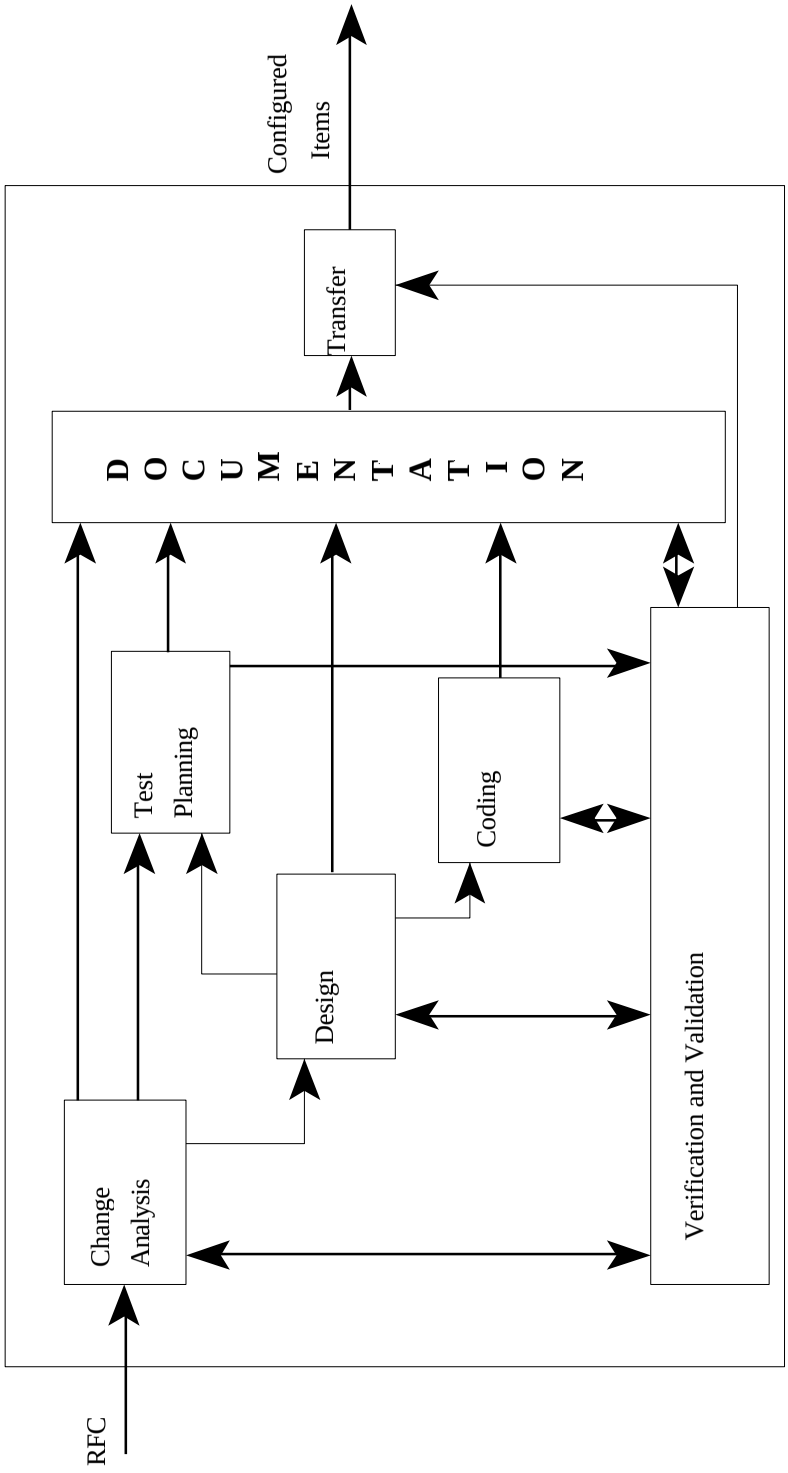

Figure 7 Software Modification Process Model

B.5.3.5 Documentation. This sub-process covers the embodiment of required changes into software documentation as a result of the activities of other Software Modification sub-processes.

B.5.3.6 Verification and Validation. This sub-process evaluates the products of other Software Modification sub-processes to determine their compliance and consistency with both contractual and local standards and higher level products and requirements. Verification and validation consists of software testing, traceability, coverage analysis and confirmation that required changes to software documentation are made. Testing subdivides into unit testing, integration testing, regression testing, system testing and acceptance testing.

B.5.3.7 Transfer. This sub-process covers the transfer of modified software items to the Configuration Management process.

B.5.4 Outputs. The Software Modification process generates modified software items and associated documentation which are handled by the Configuration Control sub-process within the Configuration Management process.

B.6 Software Embodiment

B.6.1 Figure 8 illustrates the Software Embodiment process which covers those activities necessary for loading a software item onto a programmable hardware device.

B.6.2 Inputs. The Software Embodiment process receives software releases from the Configuration Management process and unconfigured hardware items (ie items with programmable memory, the content of which is blank or has been erased in preparation for reprogramming).

B.6.3 Software Embodiment Sub-Processes

B.6.3.1 Disassemble Hardware. This sub-process covers any requirements to disassemble any hardware items in order to load software.

B.6.3.2 Load Programmable Device. This sub-process covers the loading of software onto a programmable hardware device.

B.6.3.3 Reassemble Hardware. This sub-process covers the re-assembly of hardware items after software embodiment. It includes any requirements for installation checks before an item is passed to the Mission Preparation process.

B.6.4 Outputs. The Software Embodiment process produces items of Configured Hardware which are passed to the Mission Preparation process.

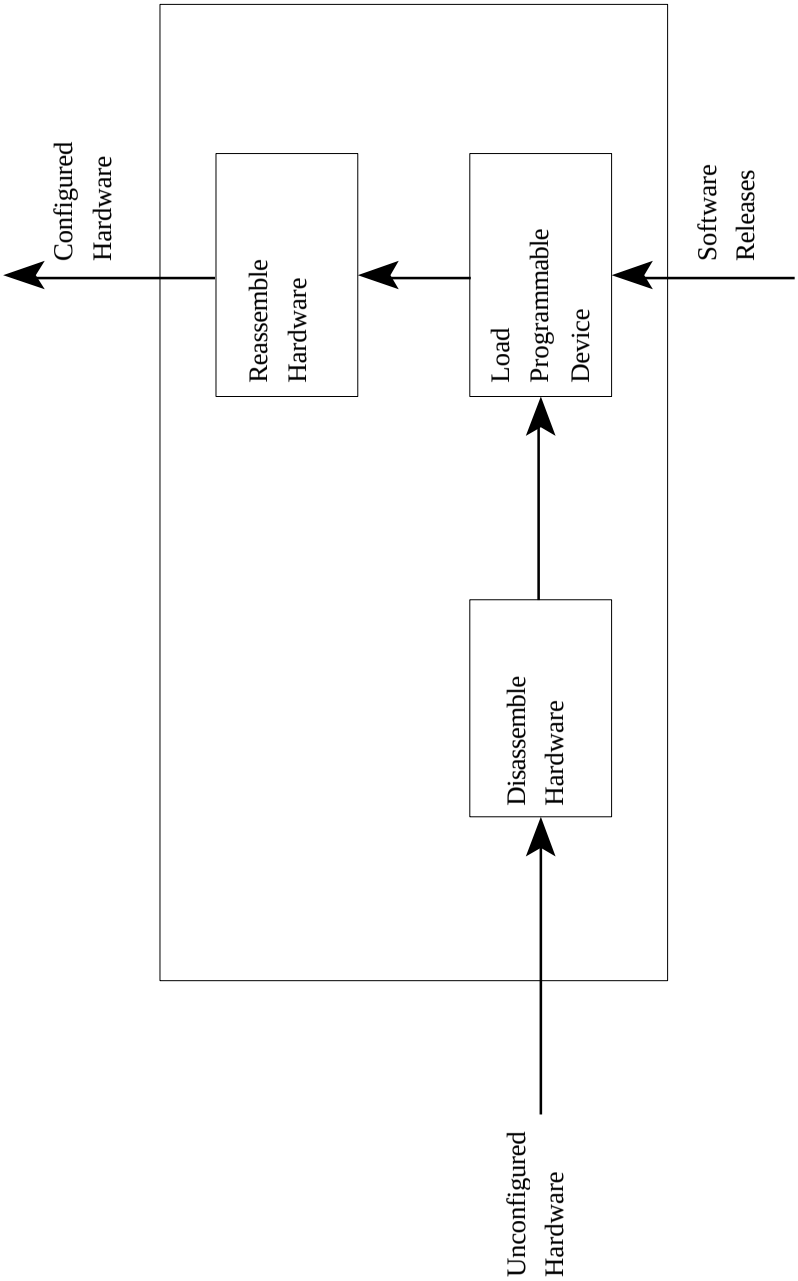

Figure 8 Software Embodiment Process Model

B.7 Post-Mission Recovery

B.7.1 Figure 9 illustrates the Post-Mission Recovery process model which covers those activities necessary to recover a system after operational use; this includes the downloading of data, the sanitization of systems that have processed classified data or run classified software and the initial investigation of fault reports.

B.7.2 Inputs. The Post-Mission Recovery process takes systems that have been subjected to operational usage and fault reports from system users as inputs.

B.7.3 Post-Mission Recovery Sub-Processes

B.7.3.1 Investigate Fault Reports. This sub-process covers the investigation of fault reports, including replication of faults and the generation of corrective RFC.

B.7.3.2 Obtain Access. This sub-process covers any requirement to manipulate hardware components in order to download data.

B.7.3.3 Download Data. This sub-process covers the post-mission downloading of data from a system.

B.7.3.4 Sanitize System. This sub-process covers any requirement to declassify a system after it has held classified data or run classified software.

B.7.3.5 Recover System. This sub-process covers all activities, that have not been addressed elsewhere in the Post-Mission Recovery process, necessary to render a system ready for the application of the Mission Preparation process.

B.7.4 Outputs. The Post-Mission Recovery process generates recovered systems, corrective RFC and mission data.

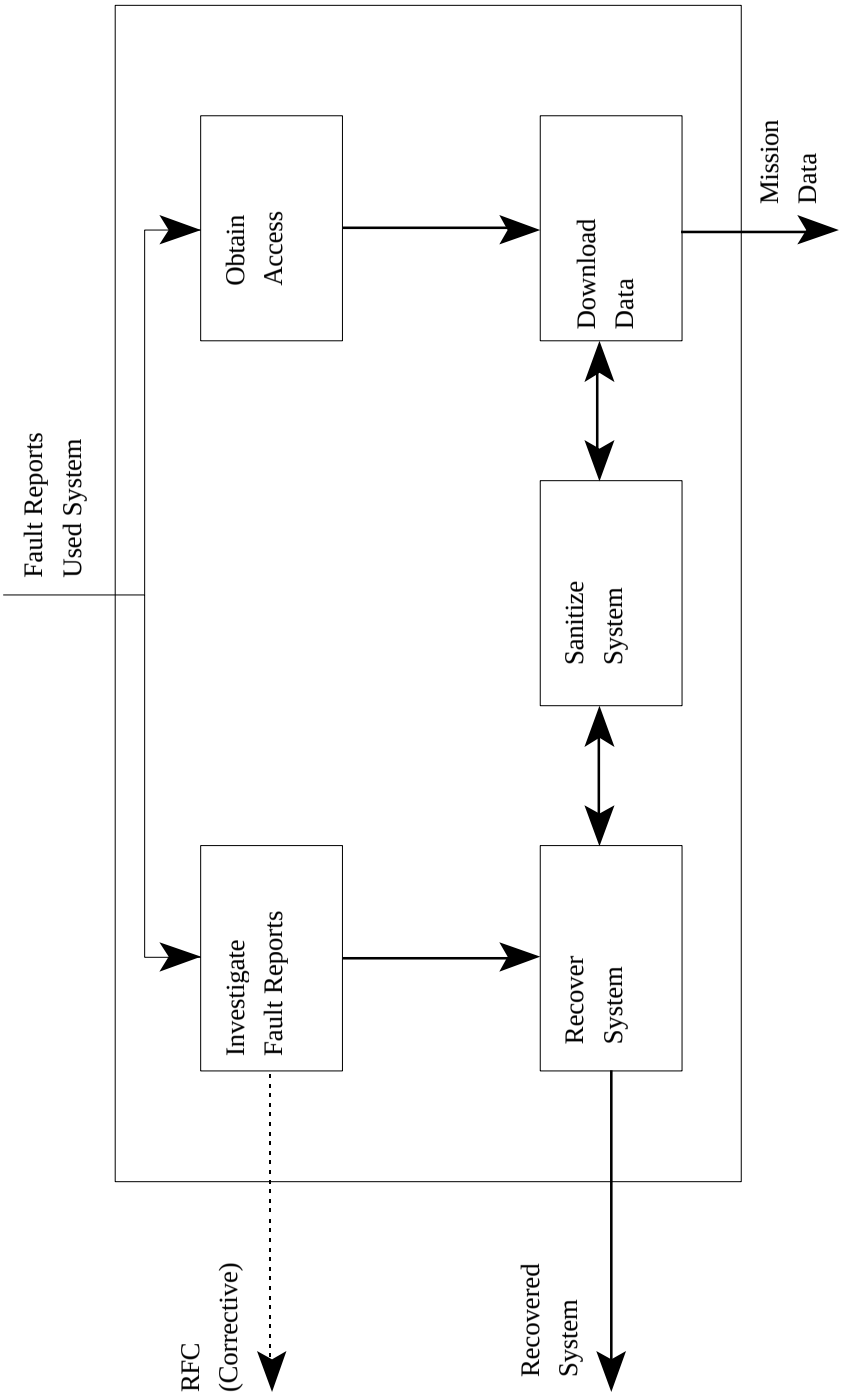

Figure 9 Post-Mission Recovery Process Model

Collation Page

Software in LCN Strategies

C.1 Introduction

Software items should be included within the overall system LSA Control Number (LCN) strategy. The means by which this is to be achieved should be described in the LSA Plan.

This annex describes the principles that underlie the preferred approach to the allocation of LCN to software items. The application of these principles is then illustrated with an example.

C.2 Software LCN Requirements

Due to the nature of software it is not possible to allocate LCN in the same way as for hardware.

Software is not a physical entity, therefore in a physical LCN structure it is not inherently obvious where software items should be located. Nevertheless, a standardized approach is needed to cater for those software support tasks which are concerned purely with the handling of software code and data for the purposes of system operation or maintenance. Software LCN in a physical LCN breakdown should therefore be associated with the LCN of the hardware items on which, or through which, the handling activity is carried out.

This approach does not, however, reveal the functional relationship between a software item and the system within which it operates. Visibility of such functional relationships is necessary to identify software support activities that result in changes to the software design. Furthermore, the functional representation of relationships allows for the aggregation of system attributes (eg rolling-up reliability data).

Therefore, it is necessary to maintain 2 views of each software item that is to be supported. The first view is provided by including executable software, and data, in the physical LCN breakdown. The second is provided by including a design representation of the executable code in the system functional LCN breakdown.

The following example, of an aircraft Defensive Aids System, illustrates the approach outlined above.

C.3 Example

C.3.1 Physical LCN

In a physical LCN structure, 2 distinct types of software may be encountered.

The first is termed 'loadable' software. In the example, Figure 10, 2 items of loadable software - Load 1 and Load 2 - are delivered using a hand held data loading device

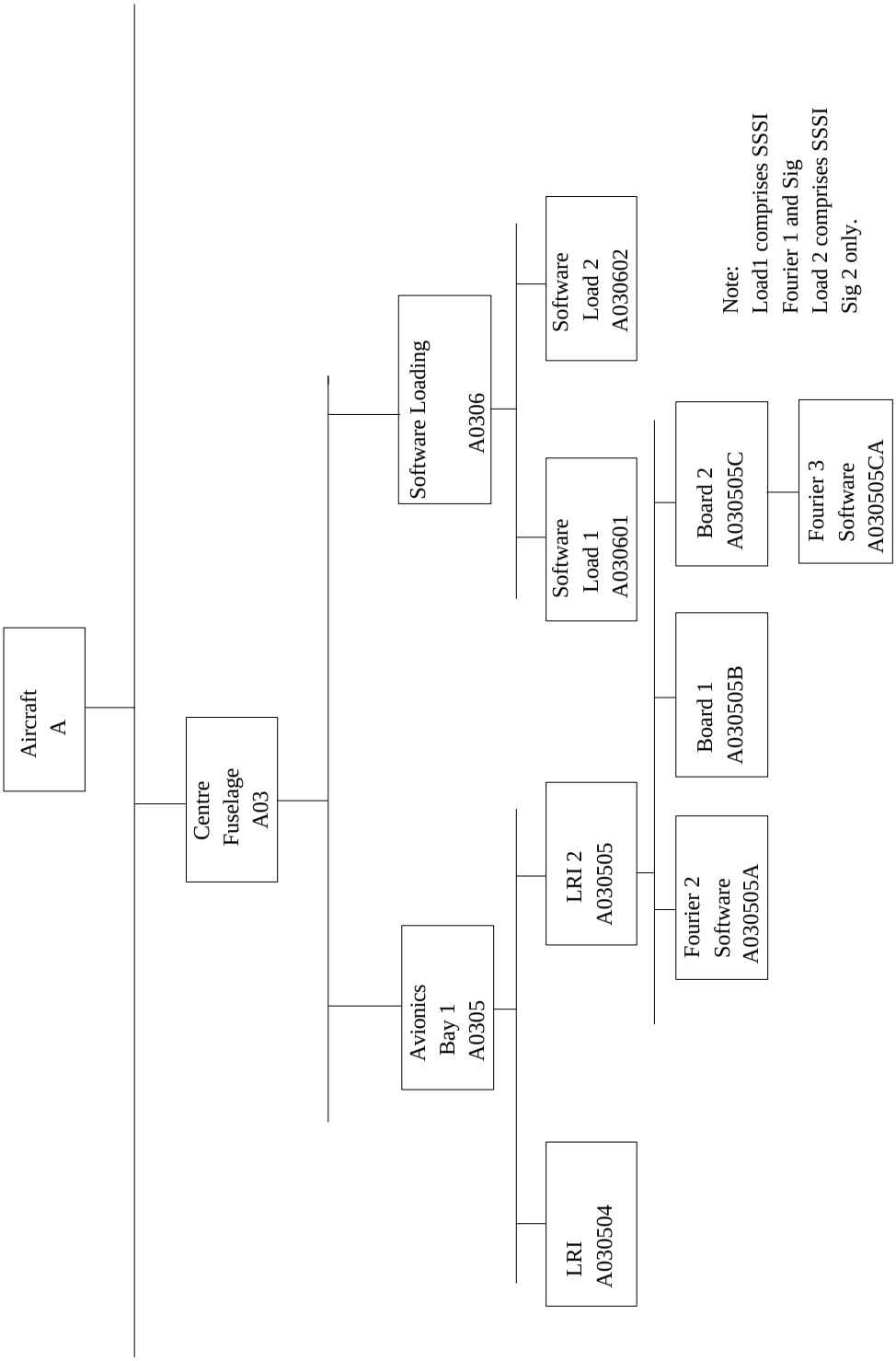

Figure 10 Physical LCN Structure

temporarily connected to a Software Loading Point. In other circumstances these software items might be provided on a tape or disc, in which case they would be shown as children of the disc/tape drive.

The second software type is termed 'resident', that is to say it forms the software element of a firmware device. Software of this type would have been embodied at 2nd, 3rd or 4th line. The example shows 2 resident software items - Fourier 2 and Fourier 3. The first is a child of a Line Replaceable Item (LRI) and the second is a child of a board within an LRI.

In all cases the software item is a child of the hardware maintenance significant Candidate Item where the physical loading activity takes place.

C.3.2 Functional LCN

In a functional LCN structure, SSSI should be shown as children of the sub-system within which they operate. In allocating LCN it is necessary to consider the supportability characteristics of the software items in question; items with particular support needs should be allocated their own LCN. For example, the LCN scheme should distinguish between SSSI that are loadable and those that are resident. It might also be necessary to distinguish between software that has either been provided by, or 'bought in' by, the prime contractor, or to identify SSSI that are safety or security significant. Clearly, during the early stages of a project these supportability issues might not all have been fully explored. The development of a full functional LCN allocation scheme should therefore be seen as an iterative process.

In order to develop an LCN structure in which SSSI are logically grouped at an appropriate level of indenture, and in which the differing support requirements of all SSSI are exposed, items will often be included at intermediate levels of indenture against which only limited amounts of information need be stored. For instance, in the example at Figure 11, little LSAR data will stored in respect of items Fourier Software and Signature Software - LCN A060204 and A060206 respectively; the full range of LSAR information will be recorded against SSSI at the lowest level of indenture.

In all such cases, software is represented by the entities on which the modification activity is carried out; that is to say the software design information as described in annex A.

C.4 Functional/Physical Relationships

It is necessary to maintain both functional and physical views of all SSSI throughout the life of the parent system. It is important to be able establish links between SSSI in the functional domain and the associated executable code in the physical domain. This may be achieved through the use of data table XG.

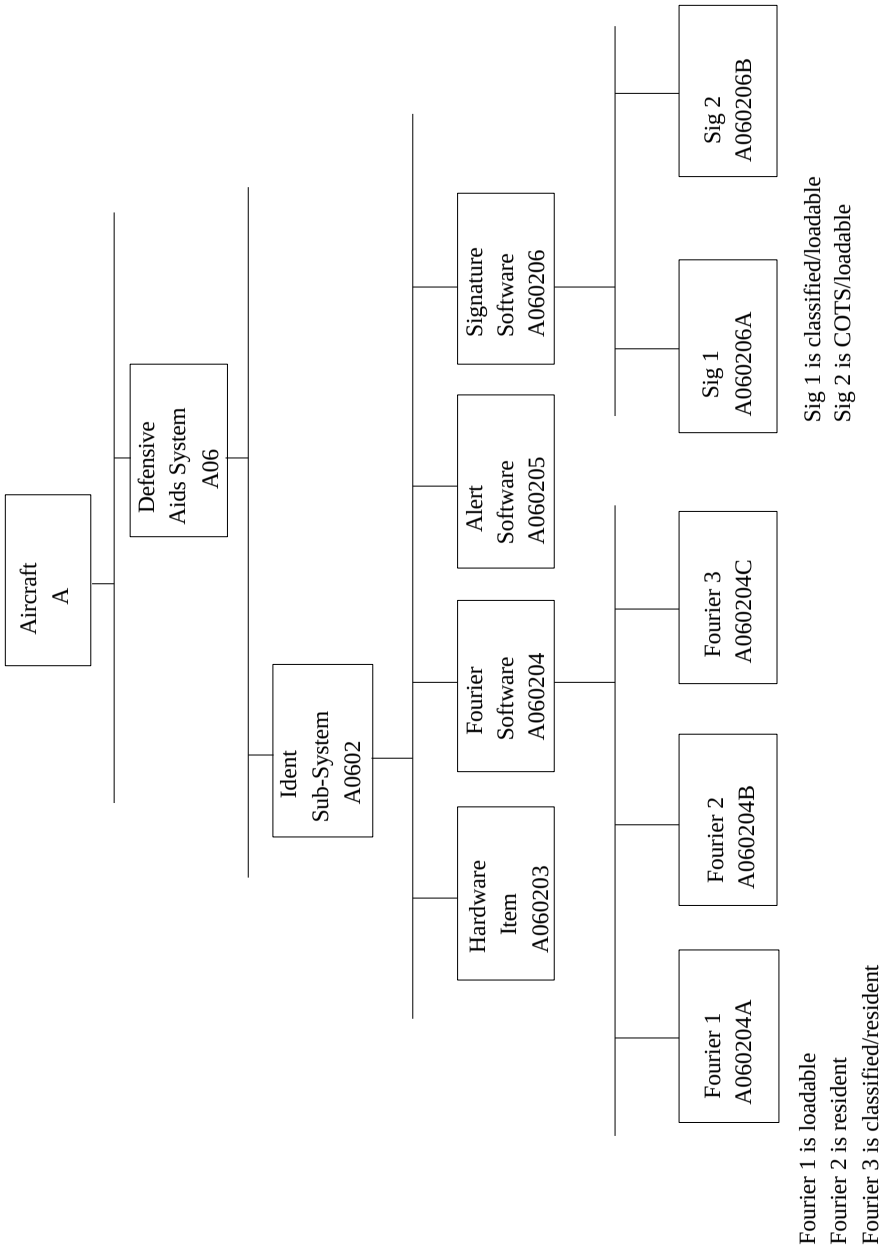

Figure 11 Functional LCN Structure

Guidance on the Application of LSA Tasks to Software

D.1 100 Series Tasks

D.1.1 Task 101 - Development of an Early LSA Strategy

D.1.1.1 The software supportability inherent in a system is largely established by the time that software development is initiated. After this point, any steps that may be taken to improve software integrity or maintainability may be outside the contracted requirements, and changes to the software functionality will be constrained by the system architecture. Key decisions affecting eventual software supportability will often have been taken by tenderers during the preparation of their proposals. LSA must, therefore, seek to influence system architectures and software requirements before software development or procurement contracts are placed. Further information on early project phase activity is provided in clause **10**.

D.1.1.2 The early LSA strategy, established by the Project Office, should include software aspects of systems. The strategy should consider the processes for determining software functionality, specifying software and software engineering requirements, and contracting for software development and procurement. The strategy should select LSA tasks and subtasks and allocate them to appropriate organizations and procurement life cycle phases to ensure their effectiveness. The LSA Strategy should identify requirements for models and estimates to enable quantification of software support resources.

D.1.1.3 Considerable uncertainty may be associated with software supportability issues prior to software development. However, the portion of the life cycle prior to contracting for software development or procurement provides the maximum potential for improving overall supportability. The early LSA strategy should ensure that analysis of software support is not delayed until accurate data becomes available. Early LSA activities for software aspects of systems should be performed promptly using the best data or estimates that are readily available.

D.1.2 Task 102 - LSA Plan

D.1.2.1 Generally, software will be included in the system LSA plan (LSAP); however, there may be instances when a separate software LSAP will be required. All the requirements of the overall system LSAP should apply to any software specific LSAP. The LSAP should define the scheme for the allocation of LSA Control Numbers (LCN) to software items - guidance on software LCN strategies is provided in annex C.

D.1.2.2 Project support environments (PSE) are essential to the modification of software. PSE should be included within the system LCN scheme in a similar manner to any other support equipment essential to the support of the system.

D.1.3 Task 103 - Programme and Design Reviews

D.1.3.1 No further guidance required for software aspects of systems.

D.2 200 Series Tasks

D.2.1 Task 201 - Use Study

D.2.1.1 Software Supportability Factors. Particular factors affecting software supportability are identified in clause **8**. These factors may relate to either, the functional requirements and constraints to be met by the system, or the design and implementation approach selected. The significance of each factor should be considered in respect of each of the logical software functions identified for the system - this process is iterative and should be carried out as new functions are identified and decomposed. The impact of the supportability factors should also be reviewed in respect of the SSSI identified as the architectural design emerges. Any pertinent software supportability requirements and constraints arising from the intended operation and support of the system should be documented.

Software support activities such as carrying out fault recovery actions or setting user variables are subject to all the factors affecting hardware supportability.

Software modification may be needed in response to the detection of faults, changes to the system design or changes to system requirements. Software modification may be initiated by users, system maintainers, support authorities or design authorities; the same agencies may impose constraints on the software support process.

Some software supportability factors will be affected by the application area (eg security classification, safety integrity), whilst others will be affected by the technology employed (eg development environment, implementation language).

D.2.1.2 Quantitative Factors. Quantitative data resulting from consideration of the software supportability factors should be documented. For software support issues, such data should be limited to attributes of direct relevance to the use and support of the system. Examples of such data include:

(a) Allowable software support activity durations (ie time to complete software installation, data entry or transfer, re-boot etc).

(b) Existing resources available for software support.

(c) Response time for rectification of software faults.

(d) Software release frequency.

(e) Data transfer and storage requirements and constraints.

Clause **12** describes the approach to be taken to the quantification of software support resources.

D.2.1.3 Field Visits. Separate field visits might be necessary to obtain information pertinent to the application area and the technology employed. Visits should be made to operational units and support cells of in-service systems using the same or similar technology to that envisaged for the subject system. During these visits data should be captured on support requirements originating from the technology involved. Such data may include: software modification team size; documentation, skill and tool requirements; engineering problems and constraints.

D.2.2 Task 202 - Mission Hardware, Software & Support System Standardization

D.2.2.1 Supportability Constraints. Existing and planned software support personnel, tools and facilities that might be relevant to the subject system should be identified. Constraints on the software development process and the technology employed - stemming from cost, manpower, personnel, readiness and support policy considerations - should be identified for application to the project as a whole.

D.2.2.2 Supportability Characteristics. It is necessary to provide information on software support costs, resource requirements and support activity durations in order to complete an overall assessment of the impact of the system and software standardization efforts. Software support information should be commensurate with the level of hardware and software standardization being pursued; it should also take account the need for the definition of a complete software support process. A software support process model, as described in clause **11**, should be developed and used as a basis for determining software support information. Clause **12** describes the approach to be taken to the quantification of software support resources.

D.2.2.3 Recommended Approaches. Hardware and software standardization initiatives that are of potential benefit to system cost, readiness or supportability should be identified and considered for identification as design or development constraints. Candidates for standardization leading to improved software support include: processors and memory devices; operating systems; interface protocols; software development (& support) environments; compilers; software development and testing methodologies and tools; data formats; coding and naming conventions etc.

D.2.2.4 Risks. All risks resulting from the establishment of constraints through the application of standardization should be identified. Equally, risks that may arise from any decision not to apply standards should be considered. Risks might be associated with dependence on limited software support resources and facilities or on the imposition of constraints on design solutions that conflict with functional or performance requirements.

D.2.3 Task 203 - Comparative Analysis

D.2.3.1 Identify Comparative Systems. A set of existing software systems should be identified to enable informative comparisons to be carried out against a range of possible software system design alternatives. To compare all attributes of interest adequately it will be necessary to identify software systems from comparable application areas, utilizing

comparable technologies, that are of comparable size and that are subject to comparable support policies to those anticipated for the new system. In some sectors, where system replacement cycles are long (20 years or more), it might be difficult to identify systems that will yield useful comparative data, due to advances in technology or changes in policy etc that have occurred in the intervening period. In such cases examples should be drawn from other system domains and where necessary, a range of different software systems should be identified to cover all the relevant features of interest. However, data obtained in this way should be treated with appropriate caution.

D.2.3.2 Baseline Comparative System. A BCS for a software intensive system will generally need to be developed as a composite of aspects of a range of existing systems. Existing systems from comparable application areas should be used to generate data relating to functional requirements, such as: safety integrity, mission criticality, security classification and stability of requirements. Existing systems of comparable size, technology and support policies should be used to collect data associated with the design solution, such as: productivity, servicing and software modification durations, tool requirements, engineering problems and fault densities.

D.2.3.3 Comparative System Characteristics. Data from identified comparative systems may be used in conjunction with software support process models (as described in clause **11**) and the approach to resource quantification (as described in clause **12**) to model the characteristics of the comparative system. Characteristics associated with software supportability should include:

(a) Description, frequency and duration of each software support and operation activity.

(b) Frequency of software releases.

(c) Costs and quantities of resources (tools, personnel, facilities etc) required for operation and support of the software.

D.2.3.4 Qualitative Supportability Problems. Any qualitative supportability problems evidenced by comparative systems should be identified and steps should be taken to avoid their recurrence in the design of the new system. Each element of the software support process model should be considered for all phases of the system life-cycle. Consideration should be given to all storage and transfer media, documentation, tools and facilities required for software support.

D.2.3.5 Supportability, Cost & Readiness Drivers. All software supportability, cost and readiness drivers should be identified and recorded for each comparative system. For example, software data transfer cycle time or software installation and start-up times may be prime system readiness drivers, software modification personnel, tools and facilities may be cost drivers.

D.2.3.6 Unique System Drivers. All software supportability, cost and readiness drivers for the new system that are not reflected within comparative systems should be identified and

recorded. Critical examination of software support process models and the conduct of Software Support Analysis (as described in clause **11**) may be used to assist in the identification of such drivers.

D.2.3.7 Risks & Assumptions. Any risks and assumptions associated with the use of data from comparative systems should be identified and documented. In particular, the validity of data from different technologies or application areas should be carefully considered. Data relating to software fault density and productivity may be critically dependent upon the technology and personnel involved in software development and support. Where there is significant uncertainty over the value of critical parameters, attempts should be made to assess the range of values that might be expected rather than base decisions on point estimates.

D.2.4 Task 204 - Technological Opportunities

D.2.4.1 Recommended Design Objectives. Software supportability may be affected by both hardware technology and software development process technology. Requirements, specific supportability design objectives and recommendations may be established through:

(a) Identifying technology advances in the areas of system and hardware architectures, operating systems, programming languages and application packages, software development methodologies and tools.

(b) Estimating the resultant improvements in overall system supportability, cost, safety and readiness values that might be achieved.

(c) Considering each of the elements of the software support process model to identify potential software engineering techniques or tools that could benefit software support. Examples include: tools for automation of code generation, documentation, configuration, verification and validation; provision of software documentation in electronic format.

D.2.5 Task 205 - Supportability & Supportability Related Design Factors

D.2.5.1 Supportability Characteristics. Quantitative characteristics, relating to a range of alternative design and operational concepts (covering both peacetime and wartime conditions), should be documented for all identified modes of system operation and support.

Operational characteristics should be established in terms of manpower, skills and performance standards for all modes of software operation.

Supportability characteristics should be established in terms of: system readiness requirements, software failure rates, durations for software installation, re-setting or re-booting, feasible software support concepts, software change traffic, software release frequencies, manpower estimates, skill and aptitude requirements, tool and facility costs. Clause **12** describes the approach to be taken to the quantification of software support resources.

D.2.5.2 Sensitivity Analysis. Decisions and estimates associated with software supportability may be based on assumptions about the frequency and type of software change requests that are likely, software fault densities, software modification productivity etc. Since large degrees of uncertainty may be associated with such assumptions, sensitivity analyses should be carried out to confirm the validity of associated supportability decisions and estimates.

D.2.5.3 Identify Proprietary Data. All elements of the software support process model (as described in clause **11**) should be checked in order to identify any design rights, licensing requirements or constraints associated with the use, replication or modification of software. Executable code, source code, documentation and software tools should all be considered. Contingencies should be identified and costed to ensure appropriate support is available for all SSSI.

D.2.5.4 Supportability Objectives & Associated Risks. Establishment of supportability, cost and readiness objectives for the system should take account of all modes of software operation and support.

(a) Operational objectives should be established relating to manpower and time requirements for software operating modes specific to system support or configuration (eg data transfers, menu configuration, etc).

(b) Software support objectives should be established relating to software installation times; re-set, re-run or re-boot times; software failure frequencies; software release frequencies; software modification effort, skill and aptitude constraints; software change traffic; change implementation duration; tools and facility costs; software support policy.

Clause **12** describes the approach to be taken to the quantification of software support resources. Estimates of software modification productivity, software fault density and software change traffic involve significant uncertainties. In particular, significant risks might be associated with any assumptions regarding productivity or software integrity improvements obtained from changes in technology or the application of new tools.

D.2.5.5 Specification Requirements. Supportability constraints and requirements should be established for software elements of the new system for inclusion in system and software specifications, requirements documents and contracts. The following topics should be considered:

(a) Requirements for specific modes of software operation to enable local system changes to be made (eg configurable menus, maintainer adjusted data).

(b) Requirements or constraints on system and software behaviour in the event of a failure (eg to provide system or software tolerance of particular types of faults, to ensure fail safe behaviour, or to provide meaningful but unobtrusive error messages as an aid to fault diagnosis).

(c) Requirements or constraints on system design and software requirements (eg to standardize target hardware and operating systems or to ensure interoperability with other systems/subsystems).

(d) Requirements or constraints on the software development process (eg definition of methods, languages and tools, documentation requirements).

(e) Constraints and requirements on software support (eg specification of the required software support policy, design authority/intellectual property rights constraints).

Conduct of Software Support Analysis (as described in clause **11**) may be used to assist in the identification of specification requirements and constraints.

D.2.5.6 Supportability Goals and Thresholds. Software supportability, cost and readiness objectives, goals and thresholds should be updated as alternative system designs gain definition or as more evidence is obtained about requirements stability, software size, software productivity and software integrity.

D.3 300 Series Tasks

D.3.1 Task 301 - Functional Requirements Identification

D.3.1.1 Functional Requirements. All functions relating to software operation and support should be identified and documented - this should be carried out for each of the alternative systems and support systems under consideration. Functions that should be considered include:

(a) Functions relating to the support of the system; for instance data transfer and installation and configuration of software.

(b) Functions relating to re-starting, re-setting or re-booting the system following a software failure - including any fault reporting and investigation functions.

(c) Functions relating to the software modification process; for instance, requirements analysis, fault diagnosis, design, implementation, verification and validation (including regression testing), documentation, transfer and replication.

The software support process model, as described in clause **11**, may be used to assist in the identification of support functions.

D.3.1.2 Operations and Support Tasks. All likely mission profiles and operational and support scenarios should be analyzed to identify operation and support tasks to be included in the system task inventory. The software support process model, as described in clause **11**, may be used to assist in the identification of software support tasks.

The task inventory should include software support activities for all software within the system and for the software used in the support process. It will be possible to refine the task descriptions as the system design and the approaches to operation and support become better defined.

Software modification tasks - such as analysis, design, coding and test - will be generic, although their actual scope will depend upon the nature of the modification in question.

D.3.1.3 FMECA. The system FMECA input to the LSAR should include software failure modes. Software failures will typically initiate 2 distinct types of support task: the immediate support activities necessary to restore the system to its pre-failure state and the longer term software modification process. SSA, as described in clause **11**, will use the results of the system FMECA to help define the support activities that are necessary. (It should be remembered that SSA will also take into consideration a wide range of other change drivers, such as operational environment changes and hardware environment changes etc.)

Detailed consideration of the failure effects caused by particular software faults is unlikely to be beneficial to the identification of support tasks.

D.3.1.4 RCM. RCM is not applicable to software.

D.3.1.5 Design Alternatives. Identification and analysis of tasks required for the operation and support of software may be used to provide feedback into the system specification or design at various phases of the life cycle as follows:

(a) Requirements for software modification may be alleviated by adding requirements for selectable modes of software operation to enable user control over software functional configuration or program variables.

(b) Changes to system architecture may provide a means of removing or simplifying some software support tasks.

(c) Hardware design may be affected in respect of the provision of suitable access for installation and upgrade of software or to improve facilities for re-setting, re-running, or re-booting systems.

D.3.2 Task 302 - Support System Alternatives

D.3.2.1 Alternative Support Concepts. Software support requirements should be considered in the development of options for overall system support. A range of alternative software support concepts should be considered although the feasible options might in practice be limited depending on the nature of the procurement. A software support process model, as described in clause **11**, should be used to check that all elements of the software support process have been addressed.

D.3.2.2 Alternative Support Plans. Software support requirements that do not involve design modifications should be included within overall system support plans. Software modification plans should be developed and documented to levels of detail commensurate with system development. Software modification plans should cover the following:

(a) Assumptions, targets and requirements regarding software release frequencies and response times.

(b) Details of what software items are to be modified, where modification is to be carried out and who is to conduct the modifications together with a description of the modification process.

(c) Identification of the modification team size and the skills required.

(d) Identification of consumables and parts required for the conduct the software modification, the replication and transfer of software products, and the support of software modification tools and equipment.

(e) Definition of the hardware and software items required for software modification, software and system testing, replication and transfer.

(f) Identification of the documentation to be used and/or maintained as part of the software modification process.

(g) Identification of training requirements (courses, durations, frequencies, etc) for the software modification team. The use and design of the software to be supported should be considered together with the software engineering methods, languages and tools to be used.

(h) Identification of the facilities required by the project support environment and any related constraints.

(i) Definition of the process for replication and transfer of the software including packaging, handling, storage and transport provisions.

D.3.3 Task 303 - Evaluation of Alternatives and Trade-off Analysis

D.3.3.1 Trade-off Criteria. Trade-off decisions resulting from the evaluation of alternative software designs and alternative software support systems should be made in the context of their impact upon the system as a whole.

The design of the software and the software support system may contribute to system supportability, cost, readiness etc. For example, the cost of appropriately skilled manpower, training and tools for software support contributes to overall system support costs.

Any models and assumptions used in the definition of trade-off criteria and in the completion of trade-off analyses should be documented. Sensitivity analyses should be carried out in

respect of variables that have a high degree of uncertainty. Such variables include: software failure rate, change traffic, productivity, software size.

Evaluations and trade-offs should be updated as necessary through the life of the project to reflect latest estimates, revised assumptions and improved models.

D.3.3.2 Support System Trade-Offs. Support system evaluations and trade-offs should give consideration to all software embodiment, pre-mission preparation, post-mission recovery and configuration management requirements. Evaluations and trade-offs should be conducted between alternative software modification policies. Issues such as cost, response times, priorities, inter-operability, safety integrity, personnel and training requirements, tools and facilities should be considered.

D.3.3.3 System Trade-Offs. Trade-offs and evaluations of alternative system designs should give appropriate consideration to their impact on software support. For example, system architectures may influence software support by affecting the number of software items, the means by which they are installed or configured for use. The way in which functions are partitioned between software items may also influence software modification requirements and/or constraints.

D.3.3.4 Readiness Sensitivities. Software support, in particular pre-mission preparation and post-mission recovery, should be considered during system readiness trade-offs. Activities such as data loading, system re-booting and software installation etc should, as far as possible, be assessed for their effect upon system readiness.

D.3.3.5 Manpower and Personnel Trade-Offs. Software support activities should be considered alongside hardware support activities in manpower and personnel trade-off analyses. Skill levels required for software embodiment, pre-mission preparation and post-mission recovery will depend upon the methods of software installation and data loading detailed in the system design. Estimates of software modification effort will depend upon assumptions about change traffic, productivity, size, methods, tools and software integrity etc. Further information is provided under the heading 'Software Supportability Factors' in clause **8**.

D.3.3.6 Training Trade-Offs. Training requirements for all aspects of software support should be considered along with the overall training requirements for the system. Different support options might require relatively more or less investment in such training provision. For example, where the system developer undertakes the support function there might be little or no visible training time and costs, whereas for customer provided support there might be a need for both basic and specialist training, with a lengthy 'on-the-job' element. Training requirements should also take account of likely staff turn-over rates.

D.3.3.7 Repair Level Analyses. Software support activities such as software embodiment, pre-mission preparation, post-mission recovery and data loading should be included in system repair level analyses. However, software modification cannot be treated in the same way as

hardware repair and the approach outlined in clause **11.6.2** should be adopted. Software modification will typically be undertaken at separate facilities from hardware maintenance.

D.3.3.8 Diagnostic Trade-Offs. Software may include features to assist in the diagnosis of faults (eg to provide error messages or records of processor status and memory contents). The provision of diagnostic facilities might affect usability and performance; for instance, software performance could be degraded by the diversion of computing resources to monitor processor status and memory content. Trade-off analyses should be conducted to determine the optimum level and type of software diagnostic support.

D.3.3.9 Survivability Trade-Offs. Certain hardware design features may be provided as a protection against particular modes of system failure, combat damage and human error; for instance, the specification of levels of redundancy. Similarly, software diversity, error traps and fault tolerance may be used to provide protection against the effects of software faults, human errors and certain hardware failures. System designs should ensure that hardware and software behaviours in the event of a fault are compatible and that the intended fault tolerance is realised. The inclusion of fault tolerance in a system, whilst reducing or controlling the effect of given faults, will add to the complexity of the system design, thereby increasing costs and fault rates. Trade-off analyses should be conducted to select the optimum level and type of fault tolerance.

D.3.3.10 Transportability Trade-Offs. Transportability trade-off analyses should review the transfer of software and data during system operation and between all elements of the software support process. Costs, transfer times, security constraints, interfaces, exchange formats, equipment and manpower should all be considered for each potential software transfer option. Transportability trade-offs should consider requirements and constraints necessary to safeguard the integrity and security of the software being transferred. Trade-offs should also take account the disposal or return of software transfer media.

D.4 400 Series Tasks

D.4.1 Task 401 - Task Analysis

D.4.1.1 Task Analysis. A detailed task analysis should be carried out for each software operation and support task within the overall system task inventory. For software support tasks:

(a) Identify and quantify manpower, skills and training, tools, parts, documentation and data necessary for the conduct of each task based upon documented assumptions regarding change traffic, release frequencies and productivity.

(b) Determine any elapsed time requirements for each task.

(c) Determine the maintenance level, or location, at which the task is to be conducted.

(d) Identify any outputs, media or by-products of the software servicing activity that are subject to storage or disposal constraints for safety, environmental, security or other reasons.

(e) Identify the software design authority (and the need for licences, IPR, etc).

The software support process model described in clause **11** may be used to assist in the identification of software support tasks.

D.4.1.2 New/Critical Support Resources. Requirements for new or critical resources necessary for the completion of software support or operation tasks might include:

(a) Personnel to conduct and manage software operations and support for large computer systems and networks (eg allocation of user accounts, performing back-ups).

(b) Tools to enable the embodiment of software in the target hardware (eg PROM blowers).

(c) Software engineering and application area skills necessary to conduct software modification.

(d) Hardware and software tools and data to permit the maintenance of software specifications and designs, the modification and compilation of source code, software verification and validation and software and system testing.

D.4.1.3 Training Requirements & Recommendations. Training for software operation and support should be considered as part of the overall system training requirement. The following points should be borne in mind:

(a) Managers of complex, multi-user computer networks might need specific training in the administration and operation of the system.

(b) Personnel involved in software modification will typically require training in the methods, tools and technology involved, as well as in the application area and the design of the subject software item.

(c) Standard classroom courses might be available for training in languages, software engineering methods and the use of particular tools. Additionally 'On-the-job' training might be needed to cover the application area and the design of the subject software item.

(d) Training requirements for software modification teams should identify general requirements applicable to all team members and specific requirements applicable to particular appointments within the team. Requirements for the generation and maintenance of training material for familiarization and 'on-the-job' training should be identified.

D.4.1.4 Design Improvements. The scope for the introduction of design changes to improve software supportability will be severely limited once the system functionality and architecture have been defined. Prior to this point, assessment of high level task analyses should be

carried out to generate recommendations for alternative system architectures, software requirements and software support policies. Assessment of lower level task analyses should also be carried out in order to generate recommendations for alternative detailed designs and hardware and software implementations.

D.4.1.5 Provisioning Requirements. Software operation and support tasks should be examined to determine requirements for hardware, firmware, transfer media and back-up copies of software. Consideration should be given to the impact of software releases on hardware spares provisioning.

Requirements for software modification might depend upon the scale of the project support environment and the criticality of the time scales for the production of new releases. Consideration should be given to initial provisioning requirements for software development hardware, test rigs, software tools and equipment. Requirements for back-up and/or archive copies of software files and software tools should be included in the provisioning requirements.

D.4.1.6 Validation. At the earliest opportunity support resource requirements should be confirmed and measurements should be made of times for completion of all software operation and support tasks. Where such measurements are dependent upon the system state or operating conditions, averages should be determined over a range of conditions. If measurements are based on non-representative hardware or operating conditions, appropriate allowances should be made and representative measurements carried out as soon as possible. The frequency of some software support tasks will be dependent upon the frequency of software releases and the failure rate exhibited by the software.

Measurements of software failure rates and fault densities obtained during software and system testing might not be representative of those that will arise during system operation. However, such measurements may be used, with caution, in the validation of models and assumptions.

For repeatable software engineering activities, such as compilation and regression testing, the time and resource requirements that arose during development should be recorded. Such information may be used to validate estimates for equivalent elements of the software modification process.

For other software engineering activities, such as analysis, design and coding, the time and resource requirements that arose during development should be recorded. However, such information should only be used with some caution in the validation of estimates for equivalent elements of the software modification process.

The preceding clauses might imply the need for a range of metrics that are not catered for as data elements within the LSAR. Guidance on the approach to be adopted is provided in Clause **12** under the heading 'Use of Software Metrics'.

D.4.2 Task 402 - Early Fielding Analysis

D.4.2.1 New System Impact. New systems with an extensive software content might have a significant impact if existing systems contain little or no software. Lesser impacts will arise if existing systems already contain software. New systems with large scale software modification requirements may generate significant requirements for new facilities, equipment and personnel; alternatively, they may have a significant impact on other systems already making use of such facilities. Potential areas of impact include:

(a) New skill requirements for operators and system support staff.

(b) New types of hardware and software for software transfer, replication and installation.

(c) New procedures and processes for software and firmware configuration management, data management, back-ups and archives.

(d) Functionality, performance and reliability problems arising from the addition of new software to an existing system (possible sources of problem include: mixture of operating systems, network incompatibilities, increases in network or computer system loading).

(e) Software modification requirements for facilities, development platforms, personnel, tools, test rigs, etc.

D.4.2.2 Sources of Manpower & Personnel Skills. Where software support requires significant additional manpower or skills the impact on existing support resources should be analyzed. Software support may generate specific manpower requirements in the following areas:

(a) Software operation, system management and data handling for large/complex computer systems (typically multi-purpose, multi-user).

(b) Software modification for large software items.

Operators of extant computer systems may provide a source of manpower for operation, support and modification of a new or expanded system. Training might be required in new technologies, applications, methods and languages. Existing staff may provide a suitable core for 'on-the-job' training and supervision of additional personnel. Teams used for software development might provide suitable skilled personnel for software modification. Consideration should be given to system needs during transition periods when both old and new systems might need to be supported.

D.4.3 Task 403 - Post Production Support Analysis

D.4.3.1 Post Production Support Plan. Providing for software operation and support throughout the useful life of a system presents particular problems. The likelihood of various

problems arising should be assessed and plans should be formulated to assure the continued support of the system through out its life. The following issues should be considered:

(a) Tools and media used for the transfer of software from the host to the target and for the replication of software might become obsolete. The storage life of magnetic media might preclude the initial provisioning of sufficient items to ensure supply throughout the system's life.

(b) Hardware and software elements of the PSE might become obsolete; it might become necessary to maintain an obsolete host platform, for which parts and support are no longer available, in order to continue to run a particular version of an item of support software. Upgrades to elements of the PSE might be necessary to retain compatibility both within the PSE and with external interfaces and to ensure continued vendor support.

(c) Developments in software engineering might render certain skills obsolescent - for example language or design method skills - and suitable training might cease to be available.

(d) The ability to carry out software modifications depends upon the continued validity of appropriate licences and IPR arrangements. Some of these rights might be vested in third parties.

(e) Verification and validation (including certification) of the system might be required following software modification. This might require continued access to, and support of, system test rigs, equipment and facilities.

Assurance of continued software support is best provided through the anticipation of likely problems and the development of appropriate contingency plans. This topic is discussed further under the 'Tools and Methods' supportability factor in clause **8**.

D.5 500 Series Tasks

D.5.1 Task 501 Supportability Test, Evaluation and Verification

D.5.1.1 Test and Evaluation Strategy. Strategies for the evaluation of system supportability should include coverage of software operation and software support. Direct measurements and observations may be used to verify that all operation and support activities - that do not involve design change - may be completed using the resources that have been allocated. During the design and implementation stage measurements may be conducted on similar systems, under representative conditions.

As software modification activity is broadly similar to software development the same monitoring mechanism might be used both pre- and post-implementation. Such a mechanism is likely to be based on a metrics programme that provides information, inter alia, on the rate at which software changes are requested and on software productivity.

D.5.1.2 System Support Package Component List. The System Support Package component list should include all software operation and support resources, including all elements of the PSE, that will be evaluated/tested during software development, software production, operational tests and system logistic demonstrations. The component lists will include:

(a) The software development platform.

(b) Test equipment, rigs and data.

(c) Facilities, hardware and software tools.

(d) Documentation and source code.

(e) Licences, design rights, etc.

(f) Replication and transfer equipment.

(g) Processes, procedures and controls (quality assurance, configuration management).

(h) Consumables and parts.

(i) Training and skills.

(j) Personnel requirements.

D.5.1.3 Objectives and Criteria. System test and evaluation programme objectives should include verification that all operation and support activities may be carried out successfully - within skill and time constraints - using the PSE and other resources that have been defined. The objectives, and associated criteria, should provide a basis for assuring that critical software support issues have been resolved and that requirements have been met within acceptable confidence levels. Any specific test resources, procedures or schedules necessary to fulfil these objectives should be included in the overall test programme. Programme objectives may include the collection of data to verify assumptions, models or estimates of software engineering productivity and change traffic.

D.5.1.4 Updates and Corrective Actions. Evaluation results should be analyzed and corrective actions determined as required. Shortfalls might arise from:

(a) Inadequate resource provision for operation and support tasks.

(b) Durations of tasks exceeding allowances.

(c) Software engineering productivity not matching expectations.

(d) Frequencies of tasks exceeding allowances.

(e) Software change traffic exceeding allowances.

Corrective actions may include: increases in the resources available; improvements in training; additions to the PSE or changes to the software, the support package or, ultimately, the system design. Although re-design of the system or its software might deliver long term benefits it would almost certainly lead to increased costs and programme slippage.

D.5.1.5 Supportability Assessment (Post Deployment). Data on software failure rates, software fault report rates, software change traffic and software modification productivity etc should be analyzed in order to verify the achievement of supportability goals. Corrective actions that might be required should be investigated and appropriate recommendations raised.

Collation Page

Inside Rear Cover

Defence Standards are Published by and Obtainable from:

Defence Procurement Agency
An Executive Agency of The Ministry of Defence
Directorate of Standardization
Kentigern House
65 Brown Street
GLASGOW G2 8EX

DStan Helpdesk

Tel 0141 224 2531/2
Fax 0141 224 2503
Internet e-mail enquiries@dstan.mod.uk

File Reference

The DStan file reference relating to work on this standard is D/DStan/21/60/3

Contract Requirements

When Defence Standards are incorporated into contracts users are responsible for their correct application and for complying with contractual and statutory requirements. Compliance with a Defence Standard does not in itself confer immunity from legal obligations.

Revision of Defence Standards

Defence Standards are revised as necessary by up issue or amendment. It is important that users of Defence Standards should ascertain that they are in possession of the latest issue or amendment. Information on all Defence Standards is contained in Def Stan 00-00 Standards for Defence Part 3 , Index of Standards for Defence Procurement Section 4 'Index of Defence Standards and Defence Specifications' published annually and supplemented regularly by Standards in Defence News (SID News). Any person who, when making use of a Defence Standard encounters an inaccuracy or ambiguity is requested to notify the Directorate of Standardization (DStan) without delay in order that the matter may be investigated and appropriate action taken.