Using Def Stan 00-60 LSA Reports To Address Software Supportability

MSc Dissertation in Software Engineering

Oxford University

Peter Bruce Hutchison

9th March 2000

ES(AIR)(Wyt)/166501/11/AD STS/AE

ABSTRACT

This dissertation shows, for software, where and how the Logistic Support Analysis (LSA) reports listed in Air Publication 100C-71, the Royal Air Force Integrated Logistic Support Managers' Handbook, can be used to assist in addressing each element of Integrated Logistic Support (ILS) within Defence Standard (Def Stan) 00-60. The manner in which the reports should be used and the applicable content of each report, including any additions or changes will be explained. The dissertation focuses on software items and the resources needed to support them. Where support equipment is identified, and this support equipment, as a piece of hardware, requires ILS to be applied, the means of accomplishing this is not detailed. While this dissertation focuses on avionics software systems, the principle can be applied to any industry sector where the developer and/or customer needs to identify the way in which software should be most effectively supported. For those unfamiliar with ILS, it will give an indication of the issues that need to be considered when examining the options for Software Support; however, if used in an ILS project, this dissertation should be read in conjunction with Def Stan 00-60.

The views expressed in this dissertation are those of the author and do not necessarily represent those of HMG, MOD, the RAF or any government agency.

Contents

List of Tables

List of Figures

INTRODUCTION

The Origins of Integrated Logistic Support (ILS) and Logistic Support Analysis (LSA)

Since the end of the Second World War, technological advances have led to increasingly complex military systems which require large amounts of specialist support equipment, manpower and facilities to operate effectively. However, many military systems were purchased by the UK and US Governments with no consideration of the cost of ownership, the manner in which the equipment would be supported or the spares requirement. Defence contractors developed systems to meet a supposed need and, in general, the support requirements established after the design was fixed. In many cases, the designed equipment operated as required, but the maintenance penalty was not considered a factor.

There are many examples of these situations, such as the case of the US M1A1 Main Battle Tank. The M1A1 was a large, fast tank that, it was discovered, needed a new road and rail transport system to move it and a new concept of operations because few of the available support vehicles could keep pace with it. Another example is the UK's Lightning aircraft with its engines mounted one above the other – if the upper engine failed, the lower engine had to be removed to gain access, resulting in a double maintenance penalty and an increased likelihood that the serviceable lower engine would fail due to its untimely removal.

With rising costs and political oversight of military spending demanding more cost effective procurement programmes, the US Government saw that there was a need to develop defence equipment in such a manner that the operational role of the system, design for supportability and through-life identification of support equipment and spares requirements would be identified as part of the system's procurement. To address such issues, Integrated Logistic Support (ILS) was developed as a system management approach to ensure that, as system support costs were a major part of the lifecycle cost, military equipment was procured with availability, reliability, maintainability and supportability being given the same importance during development as operational performance. Initially developed in the United States in the 1970s by the US Department of Defense (DoD), the standard, MIL STD 1388 [MIL STD 1388, 1973] evolved, and in the UK has been adopted as Def Stan 00-60 [Def Stan 00-60, 1999]. This has itself continued to evolve and, of all Defence Standards, is the most frequently updated with 4 issues, the first being Issue 0, between May 1996 and July 1999. To achieve its aims, ILS considers a number of areas, known as the ILS Elements and those currently detailed in Def Stan 00-60 Part 0 are:

a. Maintenance Planning.
b. Supply Support.
c. Support & Test Equipment.
d. Reliability and Maintainability.
e. Facilities.
f. Manpower & Human Factors.
g. Training and Training Equipment.
h. Technical Documentation.
i. Packaging, Handling, Storage and Transportation.

As this is the order in which they are listed, this dissertation will cover them in the same order.

In the context of Software Engineering, MIL STD 1388 never considered in detail how ILS/LSA would be applied to software. Having reached version 2B, DoD Requirements for an LSA Record, version 2C, which was to have included software aspects, was never produced. One reason was that it was considered too difficult to apply ILS/LSA to software. The first major UK project to attempt to use ILS, the Eurofighter project, has MIL STD 1388 as part of the contract but, due to the problem with assessing Software Support requirements, an in-house methodology was developed. This methodology, termed Support Analysis for Software (SAS) has been hampered by the convoluted relationships between the 4 international customers, 4 international contractors, 4 governments' political aims and the level of technical and performance detail which the process tries to gather. The current practitioners of SAS, including one of its developers, accept that it has failed to achieve its aims because of the difficulty in getting access to data, sorting the useful data from the chaff and the process of approving reviewed data to the satisfaction of all parties. However, having contracted for this process and being too far down the system development process, there is no option but to continue.

While ILS is the over-arching Ministry of Defence (MOD) procurement principle [CDPI/Supp/010, 1998], Logistic Support Analysis (LSA) is the means by which information is gathered and analysed in order to make reasoned decisions on the support of an item. Additionally, by considering the support of a system, design factors affecting that support can be influenced. The LSA process is divided into a number of tasks:

TASK	TASK TITLE	AIM
101	Development of an early LSA Strategy	To develop an LSA programme strategy and to identify the LSA tasks and subtasks which provide the best return on investment.
102	LSA Plan	To develop an LSA Plan (LSAP) which identifies and integrates all LSA tasks, identifies management responsibilities and activities, and outlines the approach toward accomplishing analysis tasks
103	Programme & Design Reviews	To establish a requirement for the contractor to plan and provide for official review and to ensure that the programme is proceeding in accordance with contractual milestones.
201	Use study	To identify and document the pertinent supportability factors related to the intended use of the new equipment.
202	System Standardisation	To define supportability and supportability-related design constraints for the new equipment based on existing and planned logistic support.
203	Comparative Analysis	To select or develop a Baseline Comparison System (BCS) representing characteristics of the new equipment.
204	Technological Opportunities	To identify and evaluate design opportunities for improvement of supportability characteristics and requirements in the new equipment.
205	Supportability & Supportability Related Design Factors	To establish quantitative supportability characteristics and objectives resulting from alternative design and operational concepts.
301	Functional Requirements Identification	To identify the operations and support functions that must be performed for the alternatives under consideration and then to identify the human performance requirements for operations, maintenance and support.
302	Support System Alternatives	To establish viable support system alternatives for the new equipment for evaluation, trade-off analysis and determination of the best system for development.
303	Evaluation of Alternatives & Trade off Analysis	To determine the preferred support system alternative(s) for each equipment.
401	Task Analysis	To analyse required operations and maintenance tasks for the new equipment
402	Early Fielding Analysis	To assess the impact of the introduction of the new equipment on the existing logistics infrastructure.
403	Post-production Support Analysis	To analyse life cycle support requirements prior to closing of production lines to ensure that adequate logistic support resources will be available during the equipment's life.
501	Supportability Test, Evaluation & Verification	To assess the achievement of specified supportability requirements, identify reasons for deviations from projections and identify methods of correcting deficiencies and enhancing system readiness.

Table 1 - The LSA Task Structure.

In order to record the data gathered through development and beyond, a relational database called the LSA Record (LSAR) is used. Information entered here is the result of investigation and

analysis by the contractor and procurer as part of an LSA Task and is reviewed via LSA Reports, either to be used as the input for the next stage or to provide a progressive build-up of information. The LSA reports detail information pertinent to different ILS elements and a review will examine applicability, validity, opportunity for design influence and are used as a basis for making support decisions.

The initial UK MOD ILS standard, Interim Def Stan 00-60, was in the main an anglicised version of MIL STD 1388-2B, which did not consider software. Def Stan 00-60 Part 3, Guide to the Application of LSA to Software Aspects of Systems was introduced only at Def Stan 00-60 Issue One, the second formal issue. While Def Stan 00-60 Parts 0 and 1 include reference to software and Part 3 advocates, correctly, that the principles of ILS and the application of LSA should be applied to software, the guidance given does not provide adequate information on where and how to record data on software in the LSAR. The information gathered is concentrated in LSA report 672, the Software Engineering report and, where, contractually, Def Stan 00-60 has been applied to software, this is the only information that has been gathered.

The use of ILS and the application of LSA tasks needs to be tailored for different projects: there are no rigid rules, ILS/LSA should be tailored to fit the project, the project should not be made to fit ILS/LSA. The tailoring process is required for both contractor and procurer to have a clear idea of what must be done on a project, and with what aim. Different types of procurement require different areas to be addressed, possibly to different depths – a Commercial-Off-The-Shelf vehicle project will have significantly different support considerations to a fully developed aircraft.

To varying degrees, the elements listed previously all apply to the support of software. In some cases there are no differences in the approach, others have superficial differences, mainly in interpretation while others require a completely new thought process to produce the necessary information. These considerations lead to the balance of material being presented on Maintenance Planning and Support and Test Equipment – doing the task and having the equipment to do the task. Where there is no difference, the focus is on the generic aim from a systems perspective and to provide background to those unfamiliar with ILS / LSA.

Applicability to Other Standards

Regardless of whether or not the system to be supported is formally subject to ILS / LSA as a contractual obligation, there may be a need to address the same type of issues. By applying the LSA

task structure and gathering the information described here, the needs of a number of International Standards can be addressed:

a. ISO 9001: 2000 – Quality Management Systems – Requirements. Currently at the Final Draft stage, the ISO 9000: 2000 series has become more process oriented and, in some areas, more specific about the requirements placed on the management of a quality system. Analysing support requirements and using the results of this analysis in the maintenance of software can assist in demonstrating compliance with the following clauses and sub-clauses:

i. 5.1, Management Commitment: ensuring the availability of necessary resources.

ii. 5.4.2, Quality Planning: identifying the resources needed to meet the quality objectives and planning such that change is conducted in a controlled manner.

iii. 5.5.6, Control of Documents: approving documents for adequacy and ensuring that relevant versions are available at points of use.

iv. 6, Resource Management: determining and providing, in a timely manner, the resources needed (personnel, training, facilities and work environment).

v. 7, Product Realization: the sequence of processes and sub-processes required to achieve the product.

b. ISO 12207: 1995 – Information Technology – Software Life Cycle Processes. ISO 12207 models the lifecycle of software in 3 particular areas: Primary Life Cycle Processes, Supporting Life Cycle Processes and Organizational Life Cycle Processes. Areas of applicability include:

i. 5.4, Operation: covering the operation of the software product and operational support to users.

ii. 5.5, Maintenance: the process of modifying code and associated documentation due to a problem or the need for improvement or adaptation.

iii. 6.1, Documentation: the recording of information produced by a lifecycle process and needed by such people as managers, engineers and users.

iv. 7.1, Management: the activities and tasks that may be employed by any party that has to manage its respective processes.

v. 7.2, Infrastructure: establishing and maintaining such things as hardware, software, tools, techniques, standards and facilities.

vi. 7.4, Training: providing and maintaining trained personnel for acquisition, development, operation and maintenance of the product.

c. ISO 14764: 1999 – Software Maintenance. Describing in greater detail the management of the Maintenance Process from ISO 12207, the application of LSA information will provide information required by the following clauses:

i. 6, Implementation Considerations: the factors affecting the manner in which a software product may be maintained, such as the type of maintenance, contractual arrangements, tools for maintenance, early involvement in development by the maintenance organisation, lifecycle stages and documentation.

ii. 7, Software Maintenance Strategy: the development of a maintenance concept and preparation of resources to provide maintenance.

iii. 8, Maintenance Processes: the identification of the activities and tasks necessary to modify an existing software product.

The Society of Automotive Engineers (SAE), dealing in Land, Sea, Air and Space systems, have a Reliability, Supportability and Logistics division, titled G-11. Included in this division is a software committee, titled G-11SW. This committee is actively pursuing the advancement of software supportability as a major logistics area. In doing so, it has published two standards, Software Supportability Program Standard [JA1004, 1998] and Software Support Concept [JA 1006, 1999]. The JA1004 standard provides an overview of Software Support within an overall systems engineering context. JA 1006 provides a framework for the establishment of a support concept and gives information needed to understand the support aspects that should be covered by a software supportability plan for either custom-developed or Commercial Off-The Shelf (COTS) software. A

third standard, Software Supportability Implementation Guide [JA 1005, due publication 2000], addresses the means by which such a program may be implemented. Although not allied to any military standard, they draw heavily on experience from military processes such as ILS.

The information presented in this dissertation complements the JA 1004, JA1005 and JA 1006 standards by providing more detailed information on the gathering, analysis and use of software supportability information. In the JA 1005 standard, Dr David Peercy of Sandia National Laboratories, chairman of the G-11SW committee, has adopted my method for identifying software within a system, detailed in Chapter 1 and will refer to this dissertation in the bibliography.

Aim of this Dissertation

The aim of this dissertation is to provide the MOD and its suppliers with a guidance on the application of ILS / LSA to software in a similar manner to which it is applied for hardware. To do so, it focuses on the issues that must be considered when supporting software and the presentation of the information, gathered as part of the analysis process, in LSA reports. This dissertation is significant in that it covers a topic that has often been considered, by both MOD and its suppliers, as too difficult to tackle and therefore ignored. The content of this dissertation will also be used in a new issue of Def Stan 00-60 Part 3, thereby formalising this work.

THE STRUCTURE OF THE LSAR

Tables, Data Elements, Data Element Definitions and LSA Reports

Central to understanding how information is collated and reviewed in the LSA process is an appreciation of the structure of the LSAR. In the LSAR, information is recorded in a number of tables, each element of a table is called a Data Element (DE) and its purpose detailed in its Data Element Definition (DED). Data elements which originated in MIL STD 1388-2B are numbered up to 518. Data elements, which were introduced in Interim Def Stan 00-60, start at number 601; those introduced in Def Stan 00-60 Issue 1 start at number 801. A number of data elements that were not required in UK usage have been removed, thus leaving gaps in the sequential numbering. Each table holds data on a separate subject, for example the A tables deal with Operational and Maintenance Requirements, the C tables with Task Information and the G tables with Personnel and Skill Considerations. The LSA reports are used to present information on a specific item or range of items from the data contained in a number of tables, for example LSA – 019, the Task Analysis Summary pulls together information from the A, C, E, H and X tables. Reports which originated in MIL STD 1388-2B are numbered up to LSA – 085, Def Stan 00-60 reports are even numbered reports starting at LSA – 602, again there are gaps in the sequence. This structure is illustrated in Figure 1.

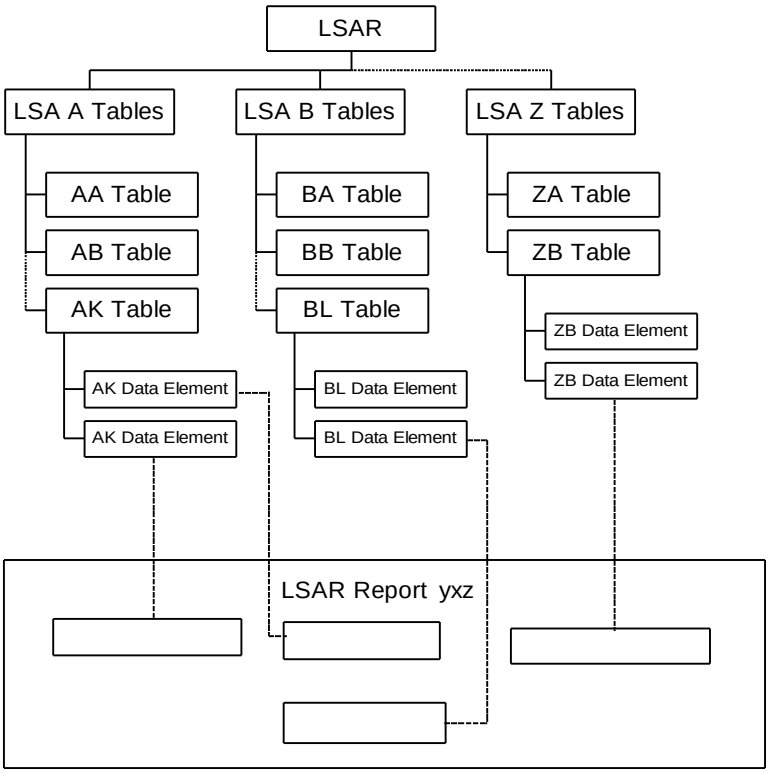


Figure 1 – LSAR Structure.

Physical and Functional System Breakdowns

The method by which items of equipment in the LSAR are indexed is through LSA Control Numbers (LCN), of which there are 2 types, Physical and Functional. While Def Stan 00-60 considers the Physical breakdown to reflect the location of components within a piece of equipment, for example an aircraft may be broken down into front, centre and rear fuselage and so on, in reality the Physical LCN structure reflects how a system is broken down based on the physical make-up of its systems. In this sense, an aircraft may consist of such things as a navigation system, a radar system, a propulsion system and a weapons system. This breakdown continues until all Candidate Items (CI) are identified in a hierarchical manner. A CI is piece of equipment that is being subjected to LSA in order to determine the most appropriate level of support. The difference between theory and practice is due to the fact that, in modern military systems, the structural location of items may change a number of times throughout the design stage. This would lead to continual flux in the LCN structure and make analysis impossible. Although Def Stan 00-60 has not been updated to reflect the explanation of LCNs in this physical sense, the discrepancy between theory and reality is being worked on.

In the Functional LCN structure, the system is broken down to a level where discrete functions can be analysed. As an example, an Attack and Identification function can be decomposed into an active identification function and so on for the receive, process and display functions. In most cases, both a Physical and Functional breakdown should be used in order to give the clearest picture of the whole system. In this case, the Functional and Physical LCN structures will need to be cross-mapped ,since one physical item may provide a number of functions and vice versa. This is achieved using the Functional to Physical LCN Mapping Data Table, XG, as described in Def Stan 00-60 Part 2.

The current Def Stan 00-60 Part 3 method of representing software in a Physical structure is to differentiate between software that is loadable and software resident in a piece of equipment. However, at some stage in production and level of maintenance, all software must be loadable and the LSA process will identify this. In Def Stan 00-60 Part 3 Figure 10, reproduced in Figure 2, software that is loadable is represented at the next indentation level down from the point at which it is loaded. This method ignores the fact that the 2 different software loads are, after the loading action has been completed, resident in a physical location. This location could be the same Line Replaceable Item (LRI) for each load or separate LRIs in separate systems. Since the object of LSA is to establish the resources required to support an item, the software contained in an LRI at the time that it is operating must be identified.

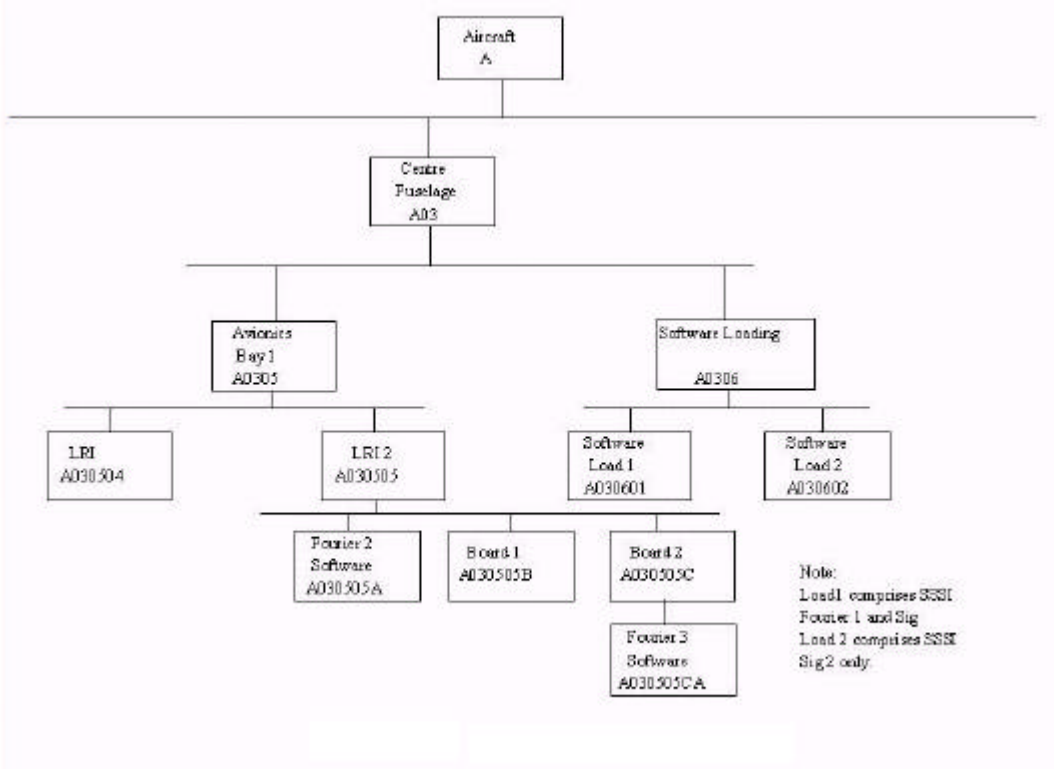


Figure 2 – Def Stan 00-60 Part 3 Physical LCN Structure.

To provide a practical illustration of how the current system shown at Figure 2 fails, consider the situation in which Software Load 1 and Software Load 2 are, during operation, located in LRI 1 (LCN A030504) and in Board 1 of LRI 2 (LCN A030505B) respectively. There is no indication that, if during the development of the system, the examination through LSA of different solutions leads to LRI 1 being replaced with a different piece of equipment or that Board 1, containing a 486 processor, is replaced with a PowerPC processor board, that the software in the system may need to change since Software Load 1 (LCN A030601) and Software Load 2 (LCN A030602) are still considered to be part of the Software Loading Point (LCN A0306). Conversely, changing the Software Load may have an impact on the equipment on which it is hosted and its subsequent support requirements. An analogue for this in terms of a personal computer is loading a piece of software for which the minimum requirements are a PII processor, 64 Mb of RAM, 180 Mb of free hard disk space and a monitor with a resolution of 1280x1024 pixels. In this case, the required support information could not be gleaned from a process that considers only the need to load the software through a CD-ROM. The equipment being considered to fulfil the hardware need must reflect the needs of the software; currently there is no mechanism within the LSAR to record such information.

My proposal for a Physical LCN structure for software, and its link to the Functional LCN structure, is at Figure 3. Here, each piece of software is considered primarily in terms of its parent system, not where it is loaded. In order to cater for modern, distributed avionics systems, Software

should be identified at the highest level of commonality. Thus, within Radar System and detailed at the same level as the Radar LRIs is the Radar Software with the Radar Operational Flight Program (OFP) and the Radar firmware detailed below this. At this stage, no differentiation is made in terms of the level at which software is loaded since this would be recorded in the LSAR in terms of Software Loading level and through the Physical to Functional Cross Mapping.

To accomplish the mapping of physical hardware items to a software program, additional Data Elements are needed in Table XG, the Physical – Functional Cross Mapping Table. The three relationships that need to be identified are the physical point at which a piece of software is loaded, the program or data file that implements a function and the hardware on which the software relies for operation. To show the Load, Implementation and Relies-On interdependencies, they should be created as Boolean fields. Thus, if a function X is implemented in program Y, requiring a specific standard of hardware in terms of processor architecture, memory, mass storage or speed, the physical dependencies of the software, in terms of the hardware in which it resides and operates can now be easily represented in the LSAR.

In Figure 3, the relationships indicate the physical items on which the functional software depends. When analysing a system, it should be noted that there is a 2 way aspect to the relationship in that a change in a physical item may result in a change to the functional software and this effect should be considered. The affected software or hardware could be examined through a query against the relevant LCN on a Yes / No basis and the Software Interdependency narrative, DED 811, used to describe more fully the interdependency and any constraints. Overall, the support resources required for each piece of equipment, down to hardware and software component level, are now more easily identified. Additionally, although a future technology for military applications, reconfigurable systems (manual or automatic) where software can run on a number of processors and use memory throughout a system, dependent on the condition of the system, can also be identified using this method.

Although not all the information recorded in table XG is shown for a particular record, an example of the Cross Mapping relationship and related Software Interdependency information as described above, from the system in Figure 3, is shown in Table 2.

XG Table Data					**BB Table data**
Physical LCN	Functional LCN	Load	Implementation	Relies On	Software Interdependency
1A02B01	AA03A	n	n	**Y**	The Tx software is run in LRI 2 using a 486 processor with 32MB of RAM.
1A02B02	AA03A	**Y**	n	n	The Tx software is loaded into non-volatile memory through an interface card mounted 9 pin serial connector.
1A02D02	AA03A	n	**Y**	n	The Tx software is implemented in Radar Control Software AI24-Tx-2G01.
1A02C	AA03B01	n	n	**Y**	The Rx Display software is run in LRI 3 using a 486 processor with 32MB of RAM, outputting an RGB signal.
1D	AA03B01	**Y**	n	n	The Rx Display software is loaded via the Avionics Databus Load Point.
1A02D01	AA03B01	n	**Y**	n	The Rx Display software is implemented in Radar Display Software AI24-RDS-2G03A
1A02A	AA03B02	n	n	**Y**	The Radar Data Processor software is run in LRI 1 using a 64000 series processor with 128MB of RAM.
1D	AA03B02	**Y**	n	n	The Radar Data Processor software is loaded via the Avionics Databus Load Point.
1A02D01	AA03B02	n	**Y**	n	The Radar Data Processor software is implemented in Radar Display Software AI24-RDP-2G03-0LA.

Table 2 – Cross Mapping and Software Interdependency Information.

The interdependencies described above are illustrated by the links between the LCN structures in Figure 3.

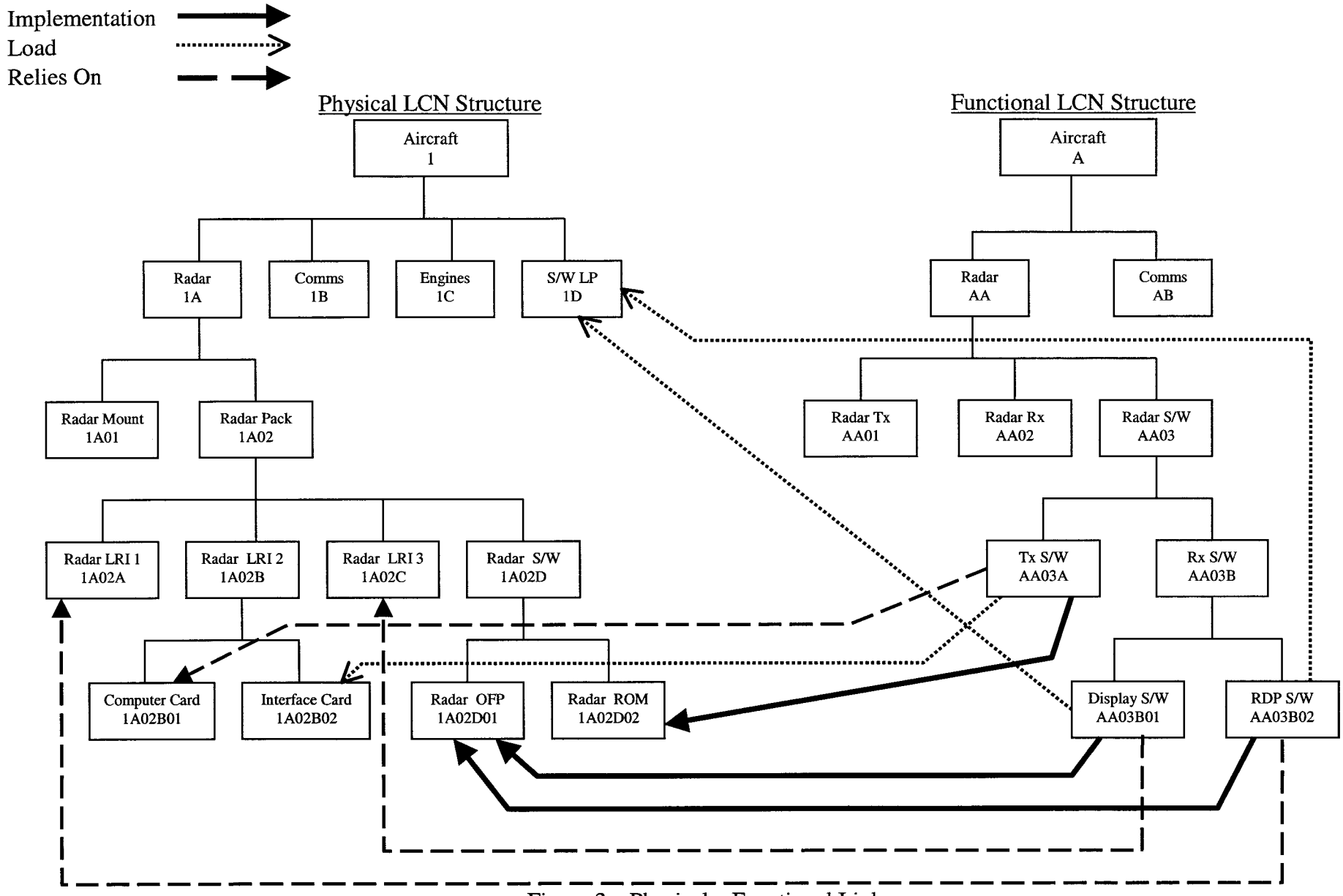


Figure 3 – Physical – Functional Links.

Summary

The organisation of the LSAR is crucial to its success and the identification of software within the LCN structure, a problem with a number of MOD projects, is central to ensuring that the analysis of Software Support can be effectively carried out. The DEDs required to address software supportability are detailed in Annex B. To obtain a complete picture of the system, the location of the software within the system and the interdependencies between hardware and software must be identified. The method proposed here shows the functionality that the software provides; the physical entity encapsulating this functionality; the software loading location and any system interdependencies.

MAINTENANCE PLANNING

As with all systems, Maintenance Planning and the in-Service costs associated with maintenance, take up a significant proportion of ILS / LSA effort and budget. This Chapter is therefore the largest and most important aspect of this dissertation. Maintenance Planning for software differs slightly from hardware maintenance planning in that hardware maintenance is structured around repairing an item after failure or carrying out preventative or scheduled maintenance in order to ensure that an item does not fail in use. For software, maintenance is either corrective, adaptive or perfective [Sommerville, 1995] and can be undertaken when the original item is still in use. Although Software Maintenance activities have distinct initiators that must be identified and processes for their management put in place, each maintenance activity, corrective, adaptive or perfective will have a very similar, if not identical lifecycle. In a structured environment, this lifecycle will be a process of [West, 1993]:

a. Understanding the problem or requirement for change.
b. Identifying and describing the changes to be made.
c. Making the changes.
d. Testing the changes and carrying out regression testing.
e. Implementing and assessing the upgraded system.

Although following a similar process to the original development (requirements, specification & design, code, test and distribution) and most likely using the same environment and tools, Software Maintenance is different from initial development in that the focus is on changing an existing design within a set of criteria, normally using different personnel with less core knowledge of the product. The criteria may be related to the physical environment or functionality of the software. The primary report for Maintenance Planning is LSA – 602, the Candidate Item Maintenance and Upkeep Plan with secondary reports of LSA – 019, Task Analysis Summary, LSA – 604, Failure Modes Effects and Criticality Analysis (FMECA) Summary, LSA – 606, Reliability Centred Maintenance (RCM) Summary, LSA – 608 Preventative Maintenance Summary and LSA – 672, Software Engineering Report.

LSA – 602, Candidate Item Maintenance and Upkeep Plan

The LSA – 602 report can be used to list the maintenance tasks for a software Candidate Item (CI) together with the maintenance options considered and recommended. It can also provide additional information, such as manpower required and a list of the Support Equipment (SE) required

for the maintenance tasks, although these can be provided in more detail under other ILS elements. The report consists of 4 parts, one mandatory and 3 optional, and can be used for review of the LSAR data throughout software development and on into maintenance. Part 3 of this report, Reliability & Maintainability, cannot be used for software maintenance for the reasons described under LSA – 606, but Parts 1 (without reliability data), 2 and 4 can be used. When the LSAR data is mature enough, the report may be used for final approval of the maintenance policy.

Part 1, Identification Particulars, provides a general overview of the CI being reviewed. Although a mandatory part, much of the information presented in Part 1 does not relate to software. The information contained in Part 1 is amplified by more specific information contained in Parts 2 - 4. To identify software dependencies, Part 1 should include a list of LCNs related to the CI under review by searching for a Yes entry in the Load, Implementation, Relies-On fields of Table XG, as described in Chapter 1. This is illustrated in Annex C.

Part 2, Maintenance Overview, contains specific information about the recommended Maintenance Concept and how it has been derived for the given CI. The Maintenance Concept Options should cover first line, on-equipment loading activities, second line bay support activities and third / fourth line Software Support facility activities.

The Maintenance Concept Cost should be derived from the costs predicted by Software Cost Estimation (SCE) tools, used to indicate the manpower and effort required for development and support, and from the cost of facilities, SE, manpower and training costs. As with many other aspects of Software Engineering, the choice of the right SCE tools is difficult and must be done in collaboration with the contractor. The difficulty in this area is that currently available SCE tools focus mainly on initial development time and manpower effort [Giles, 1995] and not on maintenance activities or infrastructure costs. These costs can be identified only by considering the tasks that have to be carried out and summing the whole-life costs, both fixed and recurring, of the manpower, equipment, facilities and training required to enable these tasks to be carried out. Therefore, while SCE tools should not be discounted, the modelling process and any limitations should be understood [Stutzke, 1996]. In addition, any costs imposed by external policies or procedures must be taken into account. These costs may be one time costs during development or they may be recurring through-life costs and could include such things as third party certification of the Quality Management System, independent safety assessment or, for aircraft and airborne systems, satisfaction of Military Aircraft Release, Civil Aviation Authority or Federal Aviation Authority procedures. Early information on a system, gathered under LSA Task 203 – Comparative Analysis, is likely to be based on similar projects from the same contractor or from a knowledge base of projects of a similar nature

and size. As development proceeds, the estimates should improve and during the in-Service phase, comparison of estimated and actual costs will provide a basis for future budgetary planning.

The Maintenance Concept narrative should cover the broad, planned approach to be employed in maintaining the software in-Service. The main options for consideration are:

a. Total Industry Support. Total industry support is most likely to be applicable where the following factors occur, either alone or in combination: a small number of systems are being supported; Annual Change Traffic (ACT) is low; complexity is high; coding language is not a project or MOD 'standard'; cost of maintaining proficient personnel is high, cost of in-Service facilities are high, certification and clearance costs are high and the operational need for change is low.

b. Total Service Support. Total Service support is most likely to be applicable where the following factors occur, either alone or in combination: the operational need for change is high; ACT is high; a large fleet is in service; the system uses a 'standard' language; complexity is low; the cost of maintaining resources (proficient manpower and facilities) is low and the cost of certification and clearance are not significant.

c. Joint Industry / Service Support. Joint industry / service support is most likely where a combination of the factors from each option above apply. Additionally, there may be a need due to intellectual / proprietary rights, Design Authority requirements or workshare agreements to operate a joint venture. The ratio of industry / service personnel will vary based on the factors and their importance.

The Maintenance Plan Rationale will detail the information used in deciding on the Maintenance Concept. Consideration of the factors detailed above, and supporting evidence will be required in order to demonstrate that the concept is the most cost and operationally effective. Analysis of information from similar systems, as detailed in Similar To, Same As, Derived From or Fitted To, should be included when available, this will add credence to any argument.

Part 4, Maintenance Tasks, contains specific information about the maintenance tasks required. The different sections deal with preventative or corrective maintenance but contain the same elements. A separate section, titled 'Software Maintenance' is therefore appropriate. Information listed in Part 4 includes the resources required to carry out individual tasks for each CI

in terms of manpower by Skill Speciality Code (SSC), for both New/Modified and Existing SSC, and SE.

For each task, careful consideration of the guidance given in Def Stan 00-60 for Task Code (Function and Interval) and Task Identification will be required. The Task Code, DED 427, is a data chain of six separate data sub-fields which uniquely identify each operator/maintenance task associated with particular items under analysis. These cover Function, Interval, Operations / Maintenance Level, Service Designator Code, Operability Code and Task Sequence Code.

The Function sub-field is limited to one character, A-Z or 0-9, and so interpretation of the codes is required to provide a suitable mapping of tasks to functional group. Although Software Modification has a code (0), there are other codes which are suitable for specific software tasks. Code 0 should therefore only be used where no other code is applicable. The function codes applicable to software are detailed in Annex B.

To some extent, all software, whether COTS, developmental or subject to in-Service maintenance, will require Operational Support tasks. Examples of these are replication, distribution, configuration management, loading, post-mission downloading, post-mission analysis and failure reporting. For a specific CI, there may be other tasks which should be considered. Task Frequency, DED 430, should be entered as a numeric representation of the annual occurrences of a task. In general, the Task Frequency for Operational Support tasks can be obtained by consideration of some or all of the following:

a. Release Frequency. Release Frequency will drive all Operational Support tasks since the release of a new software version impacts on every aspect of installing, controlling and using software.

b. Host Item Removal Rate. The rate at which the host item is removed from the system may affect loading and configuration management tasks, unless the removal task does not necessitate a software reload.

c. Mission / Role Change Rate. For software that is loaded, downloaded or analysed on a mission by mission basis or when the role of the system is changed, these rates will have an effect on the Operational Support task rate.

d. Associated Maintenance Task. There may be maintenance tasks which are related to Operational Support in that a Maintenance Procedure may call for software to be reloaded.

This may be following equipment declassification, as part of the fault diagnosis or for re-instatement of a system following the loading of a test program.

e. Software Failures. Software Failures will initiate a number of tasks, starting in the Operational Support environment, extending to the Software Maintenance environment and eventually returning to the Operational Support environment for the loading of the corrected software. Initial tasks may include a failure reporting process, Helpdesk tasks, possible re-loading of the current software in order to clear the failure or the loading of a previous version if the failure is attributed to a problem with a new release. Along with these tasks there will be associated configuration management of the failure reports and software loads.

From the assessment of Task Frequency, an entry for the Task Interval sub-field of DED 427 must be made. This interval may fall naturally from the frequency for Operational Support tasks, however, Software Maintenance task frequencies / intervals are likely to be more difficult to define since the interval will depend on the corrective, adaptive and perfective nature of the maintenance plan. For each project, it is most probable that software releases due to corrective maintenance will be bundled together, with or without ongoing adaptive or perfective maintenance. Adaptive and perfective maintenance activities may be planned projects in their own right and, as such, the frequency will be easier to predict or control. Therefore, although Software Maintenance may be anticipated, it is generally difficult to predict the exact nature and frequency of change, Task Interval code G – Unscheduled is the best suited code. This is used for 'unpredictable maintenance requirements that had not been previously planned, but require prompt attention to maintain the system in or restore it to operating condition.'

For the Operations / Maintenance Level sub-field of DED 427, the codes used are as defined in Operations / Maintenance Level, DED 277. Whereas Software Maintenance tasks will take place at either a 3rd Line in-Service facility or 4th Line contractors facility, Operational Support tasks could span all levels of maintenance such as:

a. Those focussed on 1st Line (code O) activities in direct support of operations, for example the loading and unloading of software or Configuration Management of software versions against installations.

b. Activities which may be conducted at 2nd Line (code F) such as replication, distribution, post-mission analysis or data preparation.

c. Activities carried out at the Maintenance Facility (code G for a 3rd Line, in-Service facility or code D for a 4th Line contractor facility), such as a Helpdesk.

There may be a mix of activities at various levels or a compression of activities back towards 3rd or 4th Line. For example, there may be no in-Service activities between 1st Line loading and the contractor's facility.

When identifying Software Maintenance tasks and sub-tasks, a structured method of achieving a balance, between detail so sparse as to make analysis meaningless and task identification down to a level where weight of data make analysis impossible, is required. Identification of Maintenance tasks can be best achieved by taking the declared development model (such as V, Spiral, Waterfall) and, in conjunction with the 5 stages identified by West [West, 1993], designating each stage as a Task. Task Identification, DED 431, will require more specific verbs related to software engineering than those listed in Def Stan 00-60. As an example, Requirements Engineering may be identified as a Task with Feasibility Study, Requirements Analysis, Requirements Definition, Requirements Specification and Requirements Documentation as sub-tasks.

At every stage, Software Engineering tasks will require a range of SE. For Software Maintenance tasks, this will be defined by the Project Support Environment (PSE). The PSE should cover the whole software lifecycle and, where System Testing and Integration Testing tasks are identified requiring the use of host equipment to the same standard as fitted to the operational system, this will need to be reflected in the support strategy for that equipment. Regardless of location, be it development facilities, in-Service Support Team or on-base support facility, Operational Support tasks will also require support equipment. These items may be part of the PSE, a subset of it or entirely different items of equipment. Details of the SE identified in LSA – 602 can be expanded upon in the Support and Test Equipment element.

LSA – 019, Task Analysis Summary

This Summary provides a listing of SE and skill speciality requirements needed to perform maintenance tasks. Only one DED, Technical Manual Function Group Code, DED 438, need not be used, being used for development of maintenance allocation charts, narrative technical manuals and repair parts and special tools lists. The Report may also contain the narrative sequential subtask description for each task, and the description of those subtasks which are referenced. The subtask descriptions will appear in the sequence of the task description requested. The Summary may be requested by maintenance level, Hardness Critical Procedure (HCP), task interval, task function, or

Skill Speciality Code (SSCs)/ Item Catecory Code ICC(s). For ICC, DED 177, a new code is required to catgorise software.

Having detailed the Task Code and Task Identification as used in LSA – 602, an outline of each sub-task, as described previously, can be presented in the Sequential Task Narrative. As stated previously, for Software Maintenance tasks, this should give a level of detail such that the analysis is not overly complicated with the minutiae about a task, but contains meaningful information. The personnel carrying out the sub-tasks can then be identified through the Personnel ID, DED 288, and SSC, DED 387 / DED 257. The personnel details can be further expanded in reports used in the Manpower and Human Factors element.

The current Def Stan 00-60 measurements of task duration are carried out in minutes and elapsed time or summation of task times based on hours. In this manner, the annual man-hour requirement for a component, sub-system or system can be calculated and used in a number of reports. For many systems, even where individual maintenance tasks or sub-tasks take a number of hours, this is sufficient but, in order to accommodate Software Maintenance tasks, the format of Mean Man Minutes, DED 226, will have to change. This is because it is quite feasible for Software Maintenance tasks to take days or weeks to complete and the current format, allowing up to 999.9 minutes, would allow only for a task of 16 hours 39 minutes to be recorded. DED 226 therefore needs to be redefined to use 6 digits instead of the current 4, such that up to 99999.9 minutes (about 41 weeks, 3 days at a 40 hour week) can be entered. Although the result of this is that Software Maintenance tasks will have to be calculated and entered in the LSAR in minutes, the overall recording of time across a system can still be expressed in terms of annual man-hours. A change in the task duration measurement base purely for Software Maintenance tasks would result in significant changes to the structure, format and calculations used in the current standard. The scale of such a change to the LSAR means that it is not a practical solution, whereas a change to the size of DED 226 is easier to implement.

LSA – 604, FMECA Summary

The FMECA Summary, in a hardware sense, is used to identify corrective maintenance tasks and feeds into the Reliability Centred Maintenance (RCM) process, thereby establishing the preventative maintenance plan for a piece of equipment. In this manner, depending on the severity of failure, items can have an operating life or inspection periodicity placed on them in order to try and prevent failure in use. However, if the failure should occur, regardless of the preventative maintenance plan, corrective maintenance will still be required. Within the context of LSA, analysis

of a software failure mode will not guide you to a specific preventative maintenance action but will result solely in corrective Software Maintenance. In well documented, modular software, pinpointing the corrective action to carry out on the software as a result of a functional failure recorded in the LSAR will be made easier.

Data contained in LSA – 604 under Details of Task Per Failure Mode regarding failure rates and identifying task intervals are not applicable. This is due to the nature of software failures: knowledge of a specific fault discovered during development should lead to it being corrected prior to release and failures during operation will be random and, while the exact nature of the failure will not be known, the result will be a known corrective maintenance process. However, the other information contained in the summary is an excellent vehicle for providing an understanding of the general failure modes and Safety Critical characteristics of the system. The identification of characteristics, from item function through to compensating design provisions and compensating operator provisions and, if required, system redesign, provides information for review throughout development and during support of the system. Because the LSAR is intended to be used from project inception to disposal, the data gathered provides a ready source of safety evidence supporting the Safety Case throughout the life of the equipment.

For software, the functional LCN Type should be used in generating the LSA – 604 report since it is on a functional, not physical basis, that software failures will be evident. In carrying out the FMECA, care must be taken in the level to which it is applied. While it is feasible to go down to analysing the failure modes of lines of code, this would be extremely time consuming, costly and, since such critical examination of each line should highlight all coding faults (essentially Static Code Analysis), is only practical to apply to software requiring rigorous examination for certification purposes. Practically, the deepest level at which failures should be considered is at the level at which a discrete function is provided and knowledge of the failure effect required. For example, if a program provides flight information to the pilot on a Head Up Display, an acceptable level of detail would be to consider the provision and display of height information. If the height data displayed was incorrect (higher than actual, lower than actual or not at all) the possible scenarios should be explored. From this, compensating design provisions, compensating operator provisions or system redesign requirements could be identified. At the next higher level of functionality, the analysis of failure modes may cover the provision of radar plotting information, where a number of functions are combined. Throughout, traceability can be achieved by using Specification / Drawing number, DED 687, to record the Software Requirement satisfied at that level of analysis. The level of analysis carried out will need to be agreed on a project by project, equipment by equipment basis taking into account the Safety or Mission Criticality of the system.

LSA – 606, RCM report

As with FMECA, it is not possible to apply RCM methods to software within the context of LSA because you cannot precisely answer 'What maintenance work can be carried out to prevent failure?' It is the concept of software reliability, the difficulty in its measurement and prediction which has hindered software LSA for so long. If software reliability could be easily measured and accurate statistical information gathered on when it would 'break' and in what manner, the associated level of support would be more readily identifiable. The facilities required for Software Maintenance will be essentially the same for any change request but the process, personnel and skills required may differ for each activity.

The range of software reliability models and the difficulty in selecting a model which works for specific data and environments, as outlined in British Standard 5760 Part 8 [BS 5760 Part 8, 1998] or by Musa [Musa, 1999], mean that it is not possible for the MOD to contract to a specific reliability methodology. However, with the increasing interest in software reliability, the Contractor should be able to demonstrate a software reliability programme, commensurate with Def Stan 00-42 [Def Stan 00-42, 1997]. Seeking a software reliability programme without being proscriptive parallels the move within the Royal Air Force Reliability community to taking a more hands-off approach in which the onus is on the contractor to demonstrate that they have a suitable reliability programme. A approach to software reliability in the context of ILS and LSA is dealt with in Chapter 5.

LSA – 608, Preventative Maintenance Summary

Preventative maintenance is an action carried out on an item at a given point in its installed life in order to prevent a failure in normal operation. Within the scope of software engineering, preventative maintenance is essentially the actions taken in initial development to provide software which is fit for purpose - a combination of sound design to prevent likely errors, exhaustive testing to ensure that as many statements, branches and paths through the software are exercised satisfactorily before release and the associated correction of test failures. In reality, once released and in normal operation, there can be no preventative maintenance carried out on software since any action taken will come within the scope of corrective, adaptive or perfective maintenance. As preventative maintenance in terms of LSA – 608 relates to the in-Service prevention of hardware failures, this report need not be used for software maintenance activities.

LSA – 672, Software Engineering Report

The current LSA – 672 report, developed in 1994, was the first attempt in LSA to gather information on a software item. In conjunction with the limited guidance on applying LSA to software in Def Stan 00-60 Part 3, the belief that LSA – 672 would give all the information required on which to base the support of software has hampered all Def Stan 00-60 projects so far. The report contains some data elements which are unusable in a practical environment, contradictory, provide insufficient information or which require further explanation in order to be meaningful to contractor or customer. In order to make the LSA – 672 report more useful, two parts are needed. The new Part One should give high level information on the software CI, regardless of support option. A more detailed Part 2, covering the supportability factors is required where analysis of the CI will be recorded. This part would be selected using the Software Identifier Code, DED 833. As development and analysis progress, Part 2 will become more significant for items that will require in-Service support. The proposed format is detaled at Annex C.

In addition to providing a list of Software Interdependent LCNs based on filtering on a Yes entry in the Load, Implementation, Relies On fields of Table XG, as described in Chapter 1, the changes required to the current LSA – 672 are:

a. Software Version Number. The current DED for Software Version Number, DED 816, is of limited use since it allows only for a sequential incrementing of version number from 0000 to 9999. Since many version numbers are of a similar format to MC-OFP-1.02.3, version number should be changed to allow up to 32 alpha-numeric characters to be used. This should be included in my proposed LSA – 672 Parts 1 and 2.

b. Mission Criticality. While the description for Safety Integrity Level (SIL), DED 814, is derived in accordance with Def Stan 00-56 [Def Stan 00-56, 1996] the Mission Criticality of a piece of software can not be recorded in the LSAR. This is because Annex B to Def Stan 00-60 Part 3 defines DED 814 as representing either SIL or Software Operational Integrity Level (ie Mission Criticality) and, since a CI need not be solely safety or mission critical but will have levels which are a combination of both, one DED is insufficient. In assessing support options, the significance of a piece of software, in terms of Mission Criticality, may be as important a system feature as safety integrity and the need to adapt or correct the Mission Critical software could then be the overriding factor in deciding upon a support option. A DED which fulfils the need to identify Mission Criticality is DED 691, System / Equipment Importance Code. At this time, DED 691 is used in the LSA – 604 report, as is

Safety and Hazard Severity Code. With its current title and definition, System / Equipment Importance Code could cause some confusion with Safety and Hazard Severity Code since it grades the effect of failure on a scale of 1 – 3, 1 being most severe. This use of a different scale shows inconsistency. If the title was changed to Mission Criticality Code with the following definitions, Mission Criticality can be adequately represented in the LSAR:

Mission Criticality	
Definition	**Code**
Failure would cause an unacceptable loss of mission performance or function.	1
Failure would significantly degrade Mission performance or function.	2
Failure would not significantly affect mission performance or function.	3

Table 3 – Mission Criticality Codes.

This DED should be included in the new LSA – 672 Part One.

c. Software Loading Level. The level at which software is loaded onto a piece of equipment is a vital aspect of its operational support. Although a critical cost driver, it is not currently identified as such in Def Stan 00-60 Part 3. In general, the maintenance level at which software is loaded should be as close to first line operations as possible. The rationale behind this is that, if a piece of equipment is unserviceable and requires a software re-load or if an upgrade is issued, then the loading needs to take place with the minimum of disruption to the main equipment as possible. Therefore, loading points should be easily accessible, either on the outside of the equipment or via a databus. If the piece of equipment has to be removed then the further from the main equipment that loading is carried out, the greater the workload, spares requirement and turnround time of the system. The worst case scenario, albeit extreme, would be a piece of equipment which required loading before every mission but had to be returned to the equipment manufacturer to achieve this. In general, this situation would not occur, but there are a large number of systems that do require components to be removed for software loading to be carried out. As a support cost driver, it can be seen that, where a piece of equipment has to removed, the cost of supporting that equipment will increase, particularly if a return to manufacturer is concerned. Although of primary concern for systems that are subject to regular change, the software loading level should be analysed

for all equipment and opportunities for system re-design exploited. This information should be included in the new LSA – 672 Part One.

d. Software Packaging, Handling, Storage and Transportation Summary. In supporting software, there are Packaging, Handling, Storage and Transportation (PHS&T) factors to be taken into account. These factors affect both development and support but are different from those considered in the LSA reports for PHS&T: LSA – 676, LSA – 026, LSA – 085 and LSA – 654. The LSA – 676 and LSA – 026 reports are centred on the packaging of an item, sizes, weights, container details and price information. LSA – 085 details the transportation requirements for large items of equipment and LSA – 654 is for ammunition. The PHS&T aspects of supporting software apply equally to Commercial-Off-the-Shelf (COTS) software as to full development and in-house developed software; the media used, handling details, storage at point of use and archiving at the point of development need to be addressed. For transportation, it is possible that the means of delivery between point of development and point of replication or point of replication and point of installation may need specific consideration, particularly if electronic distribution is required. As PHS&T is a separate ILS element, these issues will be considered in more detail in Chapter 10. However, a new narrative DED is required in my proposed LSA- -672 Part 1.

A significant supportability factor, recorded in the LSAR and presented in the LSA – 672 report under DED 810 is the holder of the Intellectual Property Rights (IPR). However, Def Stan 00-60 Part 3 does not list this as a support significant factor. Where possible, project managers should seek IPR or the right to modify the software as detailed in DEFCON 91 [DEFCON 91, 1992] for systems which are being developed for the MOD. Failure to do so makes it likely that there will be no opportunity for in-Service software maintenance to be carried out. However, it should be noted that this right of modification may be obtainable only at some expense. Where IPR or DEFCON 91 rights can not be obtained in the first instance, it may be possible to gain IPR at a later date when the contractor does not see the software as having the same intellectual value. Such a scenario may suit a project which is adopting a Contractor Logistics Support (CLS) package whereby Software Maintenance is provided under contract for a set period. This period is normally 5, 7 or 10 years and, as at the end of this period the contractor may not be interested in supporting the software for commercial reasons, there may be scope for IPR to change at this point, allowing the MOD to support the system itself or seek third party involvement.

In order to leave only high level information in LSA – 672 Part One, certain items should be moved to Part 2. These are DSLOC, DED 807; Program Size, DED 812; Release Frequency, DED

813; Memory Requirements, DED 815 and Software Engineering Methods and Techniques, DED 809. In addition to the 'standard' header details, the LSA – 672 Part 2 should include development process information, programming language, program metrics and interdependency information.

Development process information should be recorded under DED 809, Software Engineering Methods and Techniques. In order to continue to support a system in-Service, a knowledge of the manner in which it was developed is required. The overarching development methods such as waterfall, V-model, Spiral, Iterative, Prototyping or Incremental should be outlined. Thereafter, for each stage of development, information on the techniques and methods employed should be detailed. This narrative information should be available very early in project development, giving background information on the product and possibly a quick indication of the general resources and level of knowledge required to support the system. In addition, the methods for storing and transporting software need to be identified here to ensure that the method chosen is acceptable in terms of standardisation across the software components in a system and in terms of the availability and future viability of the technology chosen.

The programming language used should be recorded as a separate field from the Software Engineering Methods and Techniques since, across a number of applications, there may be a common language, identifiable only via a separate field. In considering the programming language used, the aim is to identify areas where common development activities and resources apply across a project and, potentially, across a number of projects. Such resources may be as simple as a common piece of test equipment or as complex as a suite of development tools. Additionally, common training and the need to recruit personnel with certain skills can also be influenced by programming language. Where information about programming languages is gathered early enough in a project, there may also be scope for design influence to ensure that applications, originally planned to use different languages can be standardised, using the same languages, environment and tools.

When considering obsolescence across a system, the main issue in software terms, leaving aside processor, memory and storage as hardware issues, is the programming language used. Where a language is used that could be considered non-standard, caution should be exercised in its use due to the constraints on future supportability. A non-standard language is one where there is no large customer base, no commercial support beyond the immediate project and a lack of trained programmers (since there is limited career and financial prospects in programming in an out-of-date language). An example of this is the Spirit 3 language used on Tornado. In this case, the MOD is constrained with supporting the systems using Spirit 3 itself since there is no commercial viability beyond Tornado.

The use of metrics in software projects falls into 2 general categories, although there are some common to both. The categories are those used for management of the procurement project and those that will live with the product throughout its life. Those that live throughout need to be recorded in the LSAR in order to analyse the support requirements and are as follows:

a. Annual Change Traffic. Although ACT can be estimated only through the development period based on changes to requirements, faults detected and by comparison with similar systems, the expected level should be considered significant since the need to change software and the rate of change are major drivers in determining the support option.

b. Release Frequency. Release Frequency is an indirect cost driver in that the proposed Release Frequency will be a reflection of effort as well as expected cost in monetary terms. Where a system has a low ACT and a high Release Frequency is proposed, then this should be questioned, since it will involve considerable effort in system test and integration and on a 'cost per release' basis may have serious budgetary implications. Conversely, where a system is subject to significant ACT, particularly of a mission critical nature, then a higher release frequency would be anticipated, and failure to achieve the required frequency may affect the operational effectiveness of a system.

c. Software Size. While the current Def Stan 00-60 Part 3 recognises the importance of software size as a supportability factor, the only measurement considered is Deliverable Source Lines of Code (DSLOC). While DSLOC is a common measurement, it is not the only metric in use and should not be considered as the only one to use. In addition, when considering the support of Mission Data such as Electronic Warfare Threat Libraries, the concept of lines of code is inappropriate. In this case the measurement base may be the number of records or the size of the database in Megabytes or records. To this end, the measured size of a piece of software should be indicated, along with the appropriate measurement base.

d. Resource Utilisation / Expansion Capability. In determining Resource Utilisation, the aim is to look beyond the systems capabilities when it comes into service, in order to gain a view of the expansion capability of the system. This is significant since, in terms of physical design constraints on projected future maintenance, sufficient spare capacity should be available within a system to avoid a major re-design at the first upgrade due to insufficient resources such as processor capability or available memory. Currently, Def Stan 00-60 Part 3

recognises the need to gather utilisation information on available memory, processor performance, mass storage capacity and Input / Output bandwidth. However, only Program Size and Memory Required (both in Megabytes) are recorded in the current LSAR and similar entries are required that give detail on the other utilisation factors. Program Size, DED 812, should be renamed Program Memory to avoid confusion with Software Size, DED 807, due to DED 812 detailing a hardware memory requirement while DED 807 details the size of the software.

e. Complexity and Maintainability. In supporting a system, the factors which will most directly affect those involved in software maintenance are the language used in programming the system, considered above, the complexity of the source code and the maintainability of the software. A distinction needs to be drawn between complexity and maintainability in that source code that is complex may be maintainable given certain skill or experience requirements and supporting characteristics such as traceability, consistency and testability. However, source code that is not complex may not be maintainable because of negative supporting characteristics. There are a number of methods of measuring complexity, such as McCabe's Cyclomatic Complexity [McCabe, 1976], Halstead Complexity measures [Halstead, 1977] and Welker's Maintainability Index [Welker, 1995] that combines Cyclomatic and Halstead complexities. Although called the Maintainability Index, this measure focuses on the source code and does not consider the wider aspects of maintainability detailed below. While each method will produce different figures, it is the consistent application and interpretation across a project which will provide useful information. The maintainability of the software should not be considered solely on the basis of source code but should include wider elements of software development. In the United States Air Force, an organisation called the Air Force Operational Test & Evaluation Center (AFOTEC) has produced a Software Maintainability Evaluation Guide [AFOTEC, 1996]. This guide has been used on a joint UK/US project and is being considered by the Defence Evaluation & Research Agency (DERA) for anglicising and use on UK projects. Each Computer Software Configuration Item (CSCI) in a system is subjected to an independent assessment by a small team of 5 – 6 people using a standard questionnaire. The hierarchical structure of the evaluation areas is shown in Figure 4.

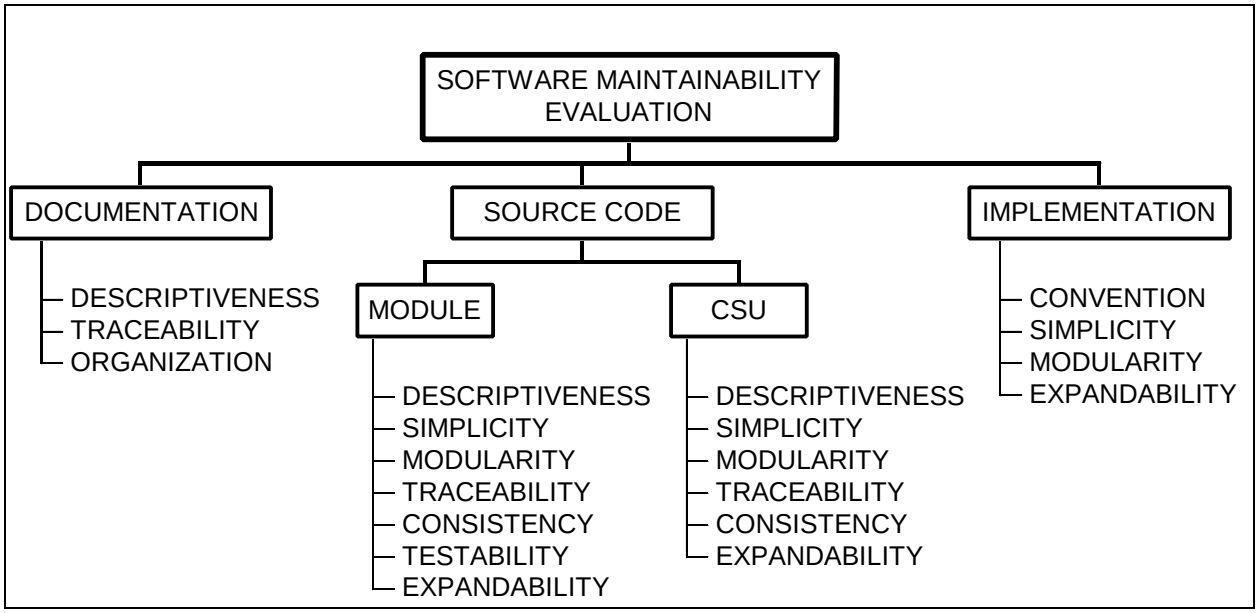


Figure 4 – Software Maintainability Evaluation Hierarchy.

Each element in the categories of Documentation, Source Code and Implementation is assessed on a scale of 1 (bad) to 6 (good), averaged across the evaluators and aggregated to give an overall rating for that category. The rating scale is shown in Figure 5.

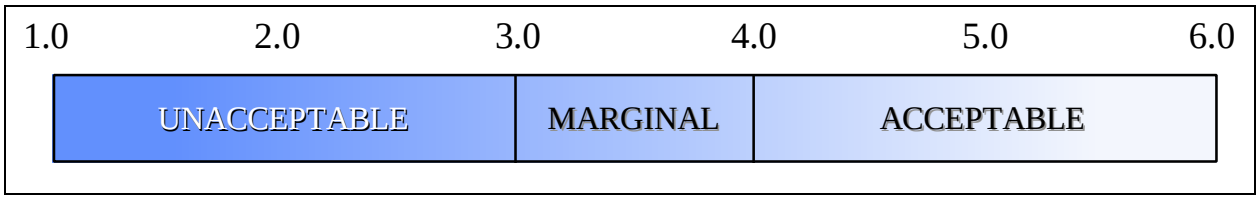


Figure 5 – Software Maintainability Evaluation Ratings.

Where a CSCI scores below 3.0 for any category, the CSCI is considered unmaintainable.

In assessing Complexity and Maintainability, the method and measures to be used should be agreed between the MOD and the contractor. Additionally, information must be gathered at an early enough stage for remedial action to be taken but late enough that the information reflects a realistic assessment of what can be expected in the final product. This may be considered to be when the source code and documentation are in near final form and no major changes are expected that might alter the characteristics of the product. For recording in the LSAR, the results of such measures should be reduced to agreed Complexity and Maintainability factors. This factoring makes data entry and review of the LSAR data a simpler process, if further detail is required for review then the source documents which led

to the gradings should be examined. In recording Complexity and Maintainability, the following systems should be used:

Complexity	
Complexity Factor	**Complexity Evaluation**
A	A simple program.
B	More complex.
C	Complex.
D	Likely to be untestable.

Table 4 – Software Complexity Codes.

Maintainability	
Maintainability Factor	**Maintainability Evaluation**
1	Extremely difficult to maintain, 2 or 3 Maintainability Characteristics graded less than 3.0
2	Difficult to maintain, 1 Maintainability Characteristic graded less than 3.0.
3	May be difficult to maintain, all Maintainability Characteristics graded 3.0 to 3.99.
4	Marginal maintainability, 2 Maintainability Characteristics graded 3.0 to 3.99.
5	Marginal maintainability, 1 Maintainability Characteristic graded 3.0 to 3.99.
6	Good maintainability, all Maintainability Characteristics graded 4.0 to 6.0.

Table 5 – Software Maintainability Codes.

Summary

Maintenance Planning is the most important aspect of software supportability. Everything that follows in this dissertation is directly linked to the maintenance process and the activities to be performed. Key areas include the selection of the most appropriate Maintenance Concept and identification of the maintenance tasks, both in Operational Support and Software Maintenance. The LSA – 672 report, in the format proposed, will enable a comprehensive picture to be gained for each CI, whether COTS or bespoke software. The key support factors of the software in terms of loading

level, language, maintainability, complexity and change traffic, combined with the ability of the system to accommodate future expansion are brought out for review and possible design influence.

SUPPLY SUPPORT

As described in Def Stan 00-60, Supply Support is principally concerned with identifying the range of spares required, the scale of spares required at each maintenance level and in stock, initial provisioning and the re-provisioning requirements of an item. There are no primary Supply Support reports, the secondary reports used are:

LSA – 019, Task Analysis Summary

As described under Maintenance Planning, this report identifies the SE used and the frequency of tasks for a maintenance activity on a piece of equipment. From this, information on the likely consumption rate of spares and the corresponding scale of SE can be assessed. Although software cannot be considered as a spare part since it is not a stock item that can be ordered in specified quantities, the supply of SE, based on task requirements must be considered.

LSA – 648, Provisioning (AECMA 2000M Related Data) Report

This report provides an Initial Provisioning List (IPL) as specified in the AECMA S2000M standard [AECMA, 1998]. The IPL is used to ensure that the correct spares are procured at the outset of a project to meet forecast demands when a system first comes into service. Software is not a spare and cannot be provisioned as a unit, therefore, this report need not be used for Software CIs.

LSA – 650, NATO Codification (AECMA 2000M Related Data) Report

The LSA – 650 report is used in gathering data for the codification process whereby items are given a NATO Stock Number (NSN). The NSN is a means of uniquely identifying equipment and component parts in the military supply system. Although consumable items such as floppy disks will have an NSN, a software CI can not be allocated an NSN since that software CI can not be stored or demanded as an individual item. It is only when a piece of software is transferred to the loading medium, demanded using its NSN, that it becomes useable. If a project were to attempt to codify software, the release of each new version of software would require a new NSN, clogging up the codification process. Configuration Management problems would also occur if a reliance was placed on ordering software by NSN since a number of versions could be released in very quick succession but codification and issuing of NSNs takes considerably longer.

LSA – 652, Illustrated Parts Catalogue (AECMA 2000M Related Data) Report

This report provides data in the form of an Illustrated Parts Catalogue (IPC), although there are no illustrations included. Specifically, it functions as an identifier of spare parts, the physical relationship between parts and the quantities of parts involved and, since a software CI cannot be considered as a spare part, the report need not be used.

Summary

In considering these reports, there is little application of Supply Support directly to software; the areas in which LSA – 648, LSA – 650 and LSA – 652 can be applied are where hardware items are linked with software. Examples of this are the transportation media used to load software, the associated Operational Support equipment and, in the case where a PSE is being purchased, equipment forming the PSE. Items in each of these categories will need to be procured in the correct quantities and in the correct timescale. As hardware items, the normal Supply Support processes and reports can be used for such items with no alteration or explanation required in relation to software LSA and so no further consideration of this ILS element is required.

SUPPORT AND TEST EQUIPMENT

As described under Maintenance Planning, Software Support tasks fall into 2 generic groups, Software Maintenance and Operational Support. In carrying out tasks in these groups, there will be a requirement for Support Equipment (SE), Software Tools and Test Equipment. The primary LSA report for the Support & Test Equipment Element is LSA – 624, Support Equipment Report. This gives an overview of the SE requirements for a given item or system and information on its use. Secondary reports used are LSA – 019, Task Analysis Summary; LSA – 070, Support Equipment Recommendation Data; LSA – 071, Support Equipment Candidate List; LSA – 072, Test, Measurement and Diagnostic Equipment (TMDE) Requirements; LSA – 074, Support Equipment Tool List; LSA – 602, Candidate Item Maintenance and Upkeep Plan; LSA – 626, Support Equipment Data Transfer Report. The contents and use of LSA – 019 and LSA – 602 were described under Maintenance Planning; for Support and Test Equipment, they serve as cross-reference links to the task requiring the SE and the CI on which the SE will be used.

LSA – 624, Support Equipment Report

In the LSAR, software SE, in terms of the hardware used and software engineering tools are both included and will be represented in the LSA – 624. The method of differentiating between hardware SE, tools, test equipment, handling equipment and automatic test equipment is through the use of the Item Category Code (ICC), DED 177. The ICCs currently listed under DED 177 are adequate for use in a software engineering context. Individual identification of SE is then by use of an SE Reference Number, DED 337. In addition to SE used for Software Maintenance, the SE for Operational Support Tasks needs to be identified.

Part 1 of LSA – 624 is mandatory and details the use of each item of SE. Cross referencing to the LCN and task for which the SE is being used is in the lower half of this section and relates to the SE identified for that LCN's LSA – 602 report. Currently, Task Type, DED 433, is either Corrective (C), Preventative (P) for calendar based tasks or Preventative (U) for usage based tasks. A new code specifically for software is required - Software (S) since the existing codes do not correspond to corrective, adaptive or perfective tasks in Software Maintenance or in Operational Support tasks. Part 2 is optional and expands on the characteristics of the SE. Information given includes manufacturer's details, power requirements and installation factors, an analysis of the item being supported and the description and function of the SE.

Overall, the information presented must be reviewed with the aim of reducing the scale of SE to the minimum required level in the same manner as influencing the design to focus on language standardisation. This can be achieved by considering the whole system and identifying areas where duplication occurs or could occur. Duplication is likely where a system contains a range of software products which are being developed and analysed separately but have the same or similar PSEs and software engineering tools. Where duplication is found, each case should be carefully considered since there may be justification for the duplication in terms of segregating safety or security critical development items.

The hardware required for software development and support centres on 4 areas:

a. The host development system, generally a distributed network of computers, possibly of different types.

b. A management system, again generally a distributed network. The management system and development system may be one and the same, merely using different tools.

c. The test environment, consisting of simulators and emulators of the target system and test rigs for the various levels of integration testing. Even where this is of the same type as the development system, the test environment will be separate, ensuring that test failures do not compromise other systems.

d. Operational Support equipment used in the day to day support of software. This will include equipment for loading the software onto the target equipment. Where replication and Configuration Management activities are carried out downstream from the development environment, the equipment required for these activities can be considered to be Operational Support equipment.

Logistically, it may not be possible to procure a hardware environment where all computers are of the same standard or from the same manufacturer. This may raise issues of the interfacing between computers or networks and of the operating systems and software engineering tools that can be used. Additionally, the project under review may be an upgrade project where new equipment has to work alongside an existing support environment, whether this is an upgrade to the full environment or a new test rig.

Where equipment in the test environment is being used for final installation and system integration testing, the equipment should be exactly that of the real environment. This addresses two problems:

a. The test environment must be an accurate reflection of real life in order to avoid any false test results. An example of this is the Ariane 5 scenario where, amongst other failures, the test environment only simulated real components, leading to a false sense of confidence in the system.

b. The equipment used in the test environment must be directly interchangeable with in-use items. On aircraft, it is possible to have two types of equipment, that suitable for flight and that, which although functionally the same and consisting of the same components in the same housing, is not considered to be aircraft-standard and for ground use only. In supplying spare equipment in this situation, there are now two categories of equipment, of separate standards, with different scales of spare provision. These are most likely procured under different contracts, with a resulting increase in costs. While the physical segregation of these items should not present a problem, it does not make economic sense to have two pools of spares which can not be interchanged. More logically, the assessment of the spares requirement should consider each test installation as another operational installation for procurement under one contract.

LSA – 070, Support Equipment Recommendation Data

The LSA – 070 report contains considerable detail on the requirements for an item of SE. This includes a description of the requirements, administrative data, supercedure data, design data and ILS requirements. In tailoring the LSA process for software, this report will only be of value where ILS / LSA is being applied to the support equipment. Where a contractor is providing a PSE for in-Service maintenance, this may be a fixed price option mirroring the development environment with no design influence available to the customer. Action taken in reviewing data as part of the Support and Test Equipment ILS element will be aimed at validating the proposed PSE in terms of content and against the price quoted. The data presented in this report is very detailed as regards the design, delivery and application of ILS to the SE concerned. As a result, and given that the necessary data is available in the other SE reports, this report should be used only for projects where full ILS / LSA is also required to be carried out on the software SE.

LSA – 071, Support Equipment Candidate List

The LSA – 071 report provides a complete list of SE for a selected LCN and the associated tasks. This report is an abbreviated report, providing more concise information on the SE than LSA – 624 and in list format. The only information provided by this report not covered by LSA – 624 is on those items which have been proposed and then disapproved for use. For assessing software supportability, this report is therefore not essential but, if used, the Technical Manual Functional Group Code, DED 438, is not required as it provides no useful software engineering information.

LSA – 072, Test, Measurement and Diagnostic Equipment (TMDE) Requirements

The TMDE that this report deals with is for use on an item of equipment, identified by LCN and it is used to verify the applicability of the test equipment for use. It details the TMDE technical characteristics such as the parameters to be measured, the size, weight power requirements, development and delivery data. The item on which it will be used on is also identified. The testing of software, as a stand alone item, will be accomplished in the test environment of the PSE. Therefore, when the software is installed on an item of equipment, the test equipment requirements are those of the hardware item, not the software, even though a functional test of the equipment may exercise the software.

Following a software release, it can be seen as bad practice to rely on a piece of test equipment further downstream to test for and diagnose software failures. To counter this, when software is initially developed for a system, the design of the hardware TMDE should be done in concert with the design and testing of software to ensure that the equipment TMDE is not testing the software to a greater degree than that done as part of the software system integration testing. However, it is unlikely that the effect of software upgrades can be catered for in the same way and so, given the nature of the discovery of software failures, the design of the equipment TMDE should also allow easy reporting of such failures.

LSA – 074, Support Equipment Tool List

By selection of the appropriate ICC, DED 177, a listing of the tools required to support an item can be identified. For a range of software products, the hardware environment may be the same but different software engineering tools are used. Therefore, by making a distinction between the PSE hardware and the software engineering tools used, the breakdown of all Software Support equipment will be made easier. Depending on the ICC selection made, this report will identify

existing tools in the customer's inventory, either as part of this project or in use on another project, and tools which will have to be developed or purchased.

There is a range of software engineering tools, also known as Computer Aided Software Engineering (CASE) tools available, and in most cases essential, for Software Support. These CASE tools cover such areas as Project Management; Requirements Engineering; Editing, Configuration Management; Method Support; Language Processing; Testing and De-bugging. A comprehensive list and description of tools is given by both Pressman [Pressman, 1997] and Sommerville [Sommerville, 1995]. For the purposes of assessing supportability, computer Operating Systems should be considered as tools since they are distinct entities, separate from the hardware environment. As with CASE tools, issues of version, compatibility and licensing apply to Operating Systems. Given the wide variety of CASE tools available, each facet of development will be handled by different tools, possibly from different vendors. In transitioning between development stages, there will be a need to transfer design data and, to simplify the design process, the tools used should interface with other design tools, sharing common data repositories and presentation formats where possible. An integrated CASE environment will improve the transfer of information throughout development and support both at a technical and personal level. This reduces the effort required in such areas as Configuration Management, Quality Management, documentation production and traceability and assist Project Management functions such as planning, monitoring and control. Three levels of CASE integration are described by Sommerville [Sommerville, 1995]. The difficulty in achieving complete integration is also highlighted; although the need is recognised, the market does not yet have significant demand for fully integrated environments. The levels are:

a. Integrated tools, managing all of their data and functions within the framework of environment services.

b. Semi-detached tools, less tightly bound to the environment, they manage their own data structures but file management is achieved within the environment services, allowing links to be made between files throughout the environment.

c. Foreign tools, hosted on the environment but requiring a data interchange service to communicate with the environment.

Selection and employment of these tools in a support environment will largely depend on those used during development.

Beyond the CASE tools, in the Operational Support environment, there will also be a need for software tools. Such tools may be used in such applications as replication, where there is only limited copies produced by the development environment; for Configuration Management of the software issued throughout the fleet; for data downloading and analysis and virus protection. Software provided securely from source, stored and loaded by service personnel should be free from viruses, but there is still a need to check. In the main the need arises where, as is increasingly happening, standard laptop PCs with adapter cards and running full commercial operating systems like Windows 95 are used as software loading devices or for replication. The threat is more of accidental than malicious infection through the introduction of additional software onto such machines but the subsequent infection of replicated or loadable software should be protected against.

Across a project consisting of a number of software products, there will be scope for the rationalisation of some tools. In particular, for the support of an integrated system across a number of software products, the use of common Project Management and Configuration Management tools will be important. Complete rationalisation may not be possible due to differing design methodologies, languages, programming and testing tools. However, in some areas such as safety or security, diversity may be unavoidable. An early analysis and review of the project and its proposed toolset will avoid costly changes later or acceptance of tools because change is impossible.

The issue of obsolescence, highlighted for the software product as a language issue under Maintenance Planning, can also affect the CASE tools used on a project. In the same manner in which a software language may become obsolete because of a small customer base, no commercial support beyond the immediate project and a lack of trained programmers, tools may be similarly affected. This may be because they are no longer supported by their manufacturer for the same reasons or simply because technology has moved on. Everyday examples of this would be PC users trying to get support for Microsoft Windows 3.0 or WordPerfect 4.0. In developing and supporting major systems, the timescale involved may be as much as 50 years, possibly more and, although projecting the availability and use of CASE tools that far ahead is impractical, there may come a time where the lack of a viable support environment forces the freezing of further development of a system or a complete system upgrade.

LSA – 626, Support Equipment Data Transfer Report

The LSA – 626 report is used in production of the Aircraft Publication Topic 3 C/D, Test and Support Equipment scales and data sheets. Provision of this data will allow a user unit, be it engaged in Operational Support or Maintenance, to hold the necessary SE based on the tasks identified.

Summary

The Support and Test Equipment element, with its identification of the support equipment required, must be carried out in a timely manner. Failure to do so may result in insufficient quantities of SE or the wrong SE being identified and support at the in-service date being compromised. To enable the support requirements to be more easily understood and the equipment procured, SE is divided into the hardware, encompassing the PSE and Operational Support equipment, and the software tools required to be used on the selected PSE. Depending on the scale of the project and the cost of equipment, the hardware required to support software may itself warrant the application of ILS/LSA. This situation is not covered here since the application of ILS/LSA to hardware items is well established.

RELIABILITY AND MAINTAINABILITY

As stated previously, the context of hardware reliability in ILS / LSA differs from that of software reliability. The field of software reliability is a growing area for research and development, with many differing and developing viewpoints [Lyu, 1995]. In the UK, research is being led by the Centre for Software Reliability (CSR) at the University of Newcastle upon Tyne and at City University. In association with the CSR, a number of books have been published including [Fenton and Pfleeger, 1996] [Fenton, 1991] and [Rook, 1990]. A detailed examination of the topic is beyond the scope of this dissertation and only some general points will be considered. Software maintainability, in terms of documentation, source code and implementation was described in Chapter 2.

For both hardware and software, the critical factor in reliability is the failure of a product. In a hardware sense, reliability and failure measurement centres on the material wear out of a product and has little to do with the design process whereas software failure is not a feature of wear-out but of a failure at some point in design. This is reflected in Def Stan 00-56 [Def Stan 00-56, 1996] where, in considering systems, analysis is required of both random and systematic failures for hazard probability targets. To this end, all software failures are considered systematic while random failures result from a variety of degradation mechanisms in the hardware. Therefore, the metrics used to measure hardware reliability are difficult to apply directly to software.

For example, when considering Mean Time Between Failure (MTBF), statistically, by the time the MTBF has been reached for a hardware item with an exponential failure distribution, 62% of those items will have failed. For software, failure is a function of the input domain and this input domain may never be the same for more than one installation of a program. Coupled with this, as software fails and faults are corrected, the time until the next failure can reasonably be expected to increase with time in operation. However, this assumes that correcting faults results in no new faults being introduced. The problem with this is twofold, firstly, while the original fault may be corrected perfectly, it is possible that the correction may uncover other faults which were previously dormant. Secondly, as adaptive and perfective maintenance is carried out, faults may be introduced by these processes. These faults may manifest themselves sometime in the future or activate previously dormant faults. Realistic assertions of MTBF for software are therefore currently unattainable.

Following on from MTBF is Mean Time To Repair (MTTR), the mean time to return an item to a serviceable condition. For a piece of hardware, this can be easily measured since, for a given failure, a finite set of activities which generally take a given time must be completed. For software,

it is almost impossible to measure maintenance tasks in this way since each failure requires analysis, correction and testing. This must be applied to the corrected software and regressed to the system.

It must also be understood that there is a distinction between 'reliable' software and 'safe ' or 'dependable' software. For 'safe' or 'dependable' software, the consideration is not that there may be a failure but that, if there is a failure, it is handled in a safe, predictable manner. Thus, 'safe' software may not necessarily be 'reliable' and 'reliable' software may not necessarily be 'safe'. This concept is explored further by Leveson [Leveson, 1995].

In considering software reliability, the distinction between failures and faults, and how they are treated must be understood. A software failure is a departure of system behaviour from user requirements, these requirements may be explicit or implicit. A software fault is a defect in code, caused by an error. This error may be in requirements, specification or design and is the result of a missing or incorrect action by a person. Software failures are therefore caused by software faults, but software faults may go undetected, since a piece of code may never be executed or a set of input conditions may never be encountered, and therefore never cause a failure. Determining software reliability is therefore an inexact measure of software failures, both discovered and postulated as being undiscovered. Correspondingly, the metrics used are different from hardware reliability models. For this reason, the reliability information recorded in the LSAR and the internal calculations carried out are of little value in reviewing software reliability. However, with the increasing focus on software reliability and the consideration given to the subject in Def Stan 00-42 [Def Stan 00-42, 1997], the issue needs to be addressed. Given the number of software reliability models available and the need to match the use of a model to the development process and operational environment, it is not practical to dictate the model to be used. Hence, although software reliability should be an integral part of a programme and of the ILS strategy, the LSAR is not a suitable repository for this data and the software reliability programme should be reviewed separately.

Software Reliability Planning

From the outset of a project, the reliability of a system will be a main feature of development effort. Software, as a large part of modern integrated systems, will play a significant role in determining the overall reliability of the system. Whether or not one accepts the concept of measuring software reliability and the inclusion of such data in a system reliability model, there are planning and actions which can be undertaken to make a software product less prone to failure. The areas in which both customer and contractor should work together and be proactive are:

a. The development process.

b. The statistical evaluation of reliability.

The Effect of the Development Process

There are a variety of approaches to making the software process contribute to software reliability. At the basic level, this may be certification of the Quality Management System to ISO 9001, required by all MOD contractors. This will demonstrate that there is a set of procedures and that they are being followed. Advancing on this may be use of such models as Carnegie Mellon University's Capability Maturity Model (CMM) [Carnegie Mellon University, 1994], where the maturity of the development process is measured in 5 levels.

Within individual development processes, and aside from adherence to recognised and contracted standards, key areas which have an effect on the end product are:

a. Requirements Capture, Analysis and Specification. Without knowing exactly what it is that is required, it is very difficult to establish whether or not the end product satisfies its requirements or fails to meet the user's expectations. Frequent problems include the customer not knowing what exactly is needed, the contractor misinterpreting what the customer expressed, the customer changing what is wanted on a regular basis and a failure to translate the requirements into a context which designers can take forward. Planned and sustained customer involvement from the earliest stages and techniques which emphasise the correctness of design, such as Formal Methods, can also reduce the risk of producing unreliable software.

b. Verification, Validation and Testing. Verification and Validation of the product is required as it passes through the development process. An internal culture of review where the questions 'Are we building the correct thing?' and 'Are we building the thing correctly?' is essential. Customer involvement in design reviews will also add to the value of the review process, hopefully providing an early indication that avoids costly 'what I said isn't what I meant' mistakes revealed late in the testing phase. To ensure that the product is tested to an appropriate extent and in the right manner, there should be a Testing Strategy defined from the outset. Proceeding in parallel with the development of the product, the testing process should be refined as each development stage is passed and, again where practicable, the customer, the Defence Procurement Agency for UK military equipment, should be involved.

c. Software Tools and Support. The range of software tools used, including Configuration Management, programming, testing and database tools must be appropriate and adequate. Well-designed and coded software that fails in use due to poor testing tools or integration issues is still failed software; the user is not interested in the reason behind the failure, only its effect.

The manner in which the organisation conducts its software development business, regardless of the procedures or standards employed, can have a marked effect on a customer's perception of the reliability or quality of a software product. It is in this vein that maturity models such as CMM consider organisations. There are broadly similar models such as SPICE, Trillium and Bootstrap. In the CMM, there are 5 levels of process maturity:

a. Level 1 – Initial. An ad hoc process with development dependent on individuals. Cost, schedule and quality are unpredictable.

b. Level 2 – Repeatable. There are policies for managing software development. Planning is based on prior experience. The planning and tracking of projects is repeatable.

c. Level 3 – Defined. The software development process is documented and integrated coherently. Processes are stable and repeatable and process activities are understood throughout the organisation.

d. Level 4 – Managed. The processes are measured and operates within limits. The organisation is able to predict trends and produce products with predictable quality.

e. Level 5 – Optimising. The focus of the organisation is on continuous improvement of the software development process. Weaknesses can be identified and there is a proactive approach to preventing faults, improving the process to prevent reoccurrence.

As part of the pre-contract assessment, a Software Capability Evaluation (SCE) against the CMM criteria should be carried out on the potential contractors [CDPI/Tech/ 490, 1997]. This will indicate the process maturity of the contractors and can be used as part of the selection criteria. Ideally, for projects with a long term in development, such as major aircraft or ships, the successful contractor should continue to be monitored through follow-up SCEs. Although at this time there is no mandated maturity CMM Level which must be attained for UK companies, this may not be far

off. In the USA, in the next 18 months the Department of Defense is looking towards placing contracts only with companies that can demonstrate CMM Level 2.

Statistical Evaluation of Software Reliability

The statistical measurement of software reliability is based on the testing of a single instance of an application with a set of test cases. Musa [Musa 1999] focuses on this as a means to achieve faster development and testing where time to market is significant, the aim being to establish the optimum point at which to release the product, given that further testing will produce minimal return on the time and cost invested. There are a number of mathematical reliability models described in Musa [Musa 1999] and in British Standard 5760 [BS 5760 Part 8, 1998] which can be considered for use.

Measuring reliability in testing is subject to certain constraints and assumptions. The basis for measurement is:

a. An expected initial Failure Intensity. Based on historical or predicted data. Failure Intensity is the number of failures per unit of execution time such as processor time or calendar time. In this, care must be taken to ensure that the data used is indicative of the software being developed. A contractor using historical data related to software developed for one application domain, language or development method as a basis for an entirely different set of criteria cannot rely on the validity of this data.

b. An expected Failure Intensity Decay rate, indicating the successful removal of faults uncovered during testing. As time goes and faults are removed, the residual number of faults and consequently the Failure Intensity decreases, provided that no new faults are introduced.

c. A Failure Intensity objective. This objective must be based on the acceptance that the software cannot be perfect and that at some time a latent fault will be activated. Agreement on a reasonable objective between customer and developer is key in this.

d. An estimate of the total failures expected. Again based on historical or predicted data the testing process should show progression toward uncovering these failures. If more failures than this are experienced, the process by which the estimate was derived is flawed.

The assumptions underpinning software reliability models are:

a. Detected faults are corrected immediately. This prevents the fault masking or causing other faults and skewing the test results.

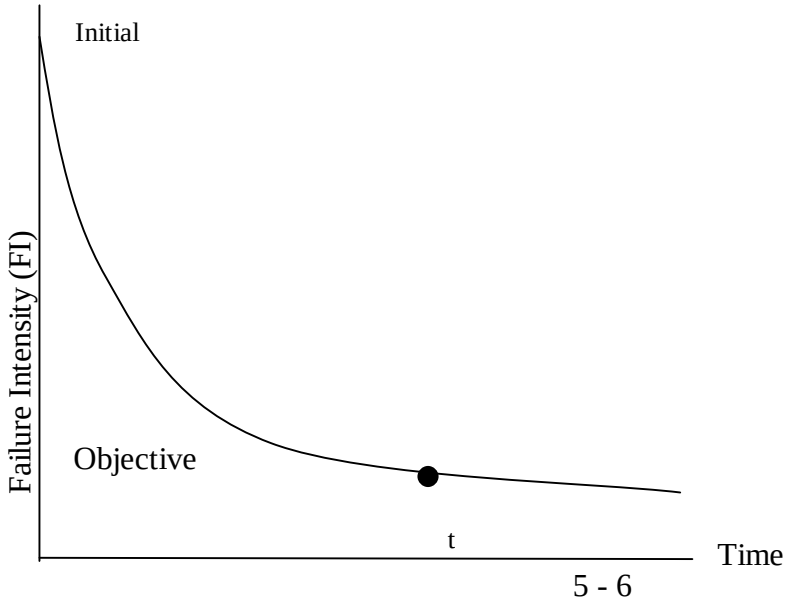
b. Correction of faults introduces no new faults. The validity of this assumption is constrained by the factors described previously on page 5 - 1.

c. The observed failure rate decreases with test time. Linked to the above 2 factors, software which became more prone to failure the more it was tested or used would give serious cause for concern.

d. The failure rate is proportional to the number of faults remaining. Where there is equal probability of executing all portions of code, each remaining fault has the same chance of being detected.

e. Testing is representative of operational use. If the environment in which the software will be deployed is not accurately reflected in the test cases, the data collected will give a false indication of the reliability of the system in the field.

A number of models are described in Musa [Musa, 1999]; particularly recommended is the Logarithmic Poisson model. This model has been favourably studied [Malaiya, Karunanithi and Verma 1992], [Jones, 1991] and [Derriennic and Le Gall, 1995]. The following graph shows that, as testing time increases, faults are uncovered and removed and the Failure Intensity decreases. Beyond a certain point, a reduction in Failure Intensity can be achieved only by a considerable increase in testing. It is around this point that the Failure Intensity Objective should be set. If the failure data collected gives a good fit to the graph, the predictions were accurate and when the objective has been met, the software should be considered for release.



From Assumption d, if all remaining faults are equally likely to appear in operation, at the Failure Intensity Objective, the reliability of the system at this time (t), with a Failure Intensity FI is expressed as $R(t) = e^{-FIt}$. This reliability figure can then be used in system reliability calculations. A limitation of this is that, where software undergoes maintenance post release, the reliability figure for the software will change since, by removing some of the remaining faults or introducing new ones, the residual number of faults changes. It may then be necessary to recalculate the Failure Intensity and reliability figures to ensure that the system reliability is not compromised, taking action as required.

The practical applications of reliability modelling are mainly in systems which operate in a stable environment, predominantly telecommunications systems, [Musa 1999] although there are some US military and space projects in which it has been used [Carnes, 1997], [Keller and Schneidewind 1997]. In the UK defence industry, there has been little practical application of software reliability engineering so far.

Software Reliability Engineering, particularly statistical measurement, is a developing field and one in which the UK defence community is slowly coming to terms with. While process examination can be a very subjective measurement method, the validity, cost-effectiveness and ease of use of statistical methods has yet to be confirmed in UK military applications. The assumptions in software reliability modelling are very dependent on two areas: not introducing faults can be very difficult, and testing representative of operational use can be difficult to achieve. Regarding fault introduction, where corrective, adaptive and perfective maintenance is carried out, fault free maintenance is almost impossible to achieve since completely new elements of code are being introduced to a legacy system, thereby altering the fault density. Although one-off applications or stable environments may lend themselves to testing which is representative of operation, military systems are subject to an ever changing operational environment that may only be superficially covered in testing.

Summary

In considering the reliability strategy to adopt on a project, a combination of both process assurance and statistical evidence should be sought. Customers and contractors need to work together in order to ensure that both approaches can be successful. Each has constraints and limitations and it is only by research and understanding that there can be a successful outcome for both parties.

FACILITIES

Along with identifying the equipment required to support software in-service, the buildings in which such equipment will be housed and any training facilities required will need to be established. The facilities required for Software Support are aligned with the grouping of support tasks: Software Maintenance and Operational Support. Depending on the system and the support concept chosen, the facilities required may be a few rooms containing replication and Configuration Management tools, possibly in an existing building at the location where the system is in operation. At the other end of the spectrum, it may be a purpose built software development facility. This would encompass offices, development areas, test areas, library and archiving facilities where the full range of software lifecycle activities and supporting management functions can take place. In such a situation, the lead-time for justifying, approving, designing and constructing such a facility will be considerable – up to 5 years being a reasonable timescale, longer not being uncommon. Early indications of the need for such a facility may therefore be vital to ensure that the support facility is in place and functional at the start of operations.

The primary report for the Facilities element is LSA – 628, Facilities Summary. Secondary reports are LSA – 012, Facility Requirement; LSA – 019, Task Analysis Summary; LSA – 075, Consolidated Manpower, Personnel and Training Report; LSA – 634, Training Facilities Report and LSA – 636, Facilities Environmental Impact Report. The use of LSA – 019, described under Maintenance Planning, is as a cross-reference to the task and associated SE for each CI being supported. For Software Support, LSA – 075 provides no significant information, cross-referencing to tasks and the levels at which they are carried out is achieved through LSA – 019.

LSA – 628, Facilities Summary

The Facilities Summary report details the facilities required to support a CI and can be selected either by Facility Name, or by LCN range and Operational / Maintenance Level. Software Maintenance tasks are, in general, carried out at one facility, either 3rd Line or at 4th Line as described in Maintenance Planning and so selection by Facility Name would provide a report detailing the CIs supported at that facility and the associated tasks. For Operational Support tasks, a review based on LCN and Operational / Maintenance Level will yield the broader picture of all facilities involved in supporting a particular CI at that level. The narrative fields used in LSA – 628 provide a comprehensive breakdown of factors which need to be considered when assessing the need for any type of facility. Only those for which specific considerations apply to a software

facility are described, the information provided in Def Stan 00-60 Part 0 being adequate for the other narrative fields.

In Facilities Requirements, DED 108, the broader constraining factors are considered, such as planning and development permission, mandatory notifications, Public Relations and environmental criteria, both in terms of the local environment and the working environment inside the facility. Aside from the planning aspects applying to any building, for a Software Maintenance facility, this will cover any requirements for air-conditioning, ventilation and cooling of such areas as computer rooms and test rigs.

The principle factors listed under Facilities Requirements for Operations, DED 109, for a Software Maintenance facility are Communications factors. It is highly likely that a computer network will be required for the PSE and early planning for its installation will be required to ensure that the necessary ducting will be built in and any security issues regarding the layout of the network can be resolved. Integration with, or segregation from, any other networks on-site also needs to be identified. Other communications aspects essential for the running of the facility, such as telephone and Public Address requirements are also covered here.

The Facility Justification, DED 188, describes the major factors which either led to the decision that additional facilities, personnel, training, training material, support equipment are required; or provided the basis for establishing the maintenance concept or making a major programme decision. For Software Support facilities, the justification will be based on the need to provide the level of support anticipated and the need to house the equipment and personnel required. As described earlier, the needs of an Operational Support facility are likely to be small but those of a full development facility may be considerable. The investment in such a facility and associated equipment may consume a significant proportion of the facilities budget and require considerable justification.

The security aspects of a Software Support facility should be detailed under Facility Requirements: Special Considerations, DED 120. Computers, network cabling and telephones will require TEMPEST (electromagnetic emission) clearance, possibly affecting the positioning of equipment. Physical security in terms of control of entry and the storage of classified documentation, including magnetic media, and any classified equipment such as test rigs will also be addressed. This information should be reviewed in parallel with the security requirements of the CIs provided in the LSA – 672 report.

Within a software support facility, whether for Operational Support or Maintenance, there will be a need to store media such as in-use, backup and archive copies of electronic data and documents. For each CI, the storage requirements should be detailed in the new PHS&T element of LSA – 672 and reflected in Facility Requirements: Supply/Store narrative, DED 121.

LSA – 634, Training Facilities Report

If there is a requirement for a training establishment to support a system, LSA – 634 details the training facilities required against each task to be performed. Although the possibility can not be excluded, it is unlikely that a project will seek to establish training facilities covering the whole range of software Operational Support and Maintenance tasks. Within the MOD, there are centralised facilities for Software Engineering training and commercial contracts run to supply more specific training which can not be provided centrally. The narrative Data Elements used in this report are a subset of those used in LSA – 628 and can be interpreted as described above.

LSA – 636, Facilities Environmental Impact Report

This report documents the environmental aspects of introducing facilities required at all maintenance levels. However, as a stand-alone report, it gives no more information than detailed in LSA – 628 since, except one, the Data Elements used are the same. The only Data Element not included in LSA – 628 is Qualitative & Quantitative Maintainability Requirements, DED 315, but from this, the environmental considerations applying to a facility should have been covered in the Facilities Requirements narrative in LSA – 628. There is, therefore, no need to use this report.

Summary

The range of facilities required for Software Support can vary widely from adaptation of a few existing rooms to the construction of a complete complex. Due to the lead time in building such facilities and the associated costs, this element forms a significant part in the overall project plan. The information presented in the Facilities reports will enable a comprehensive breakdown of requirements and provide justification for what may be very expensive buildings.

TRAINING AND TRAINING EQUIPMENT

The Training and Training Equipment element pulls together information from a number of other elements to ensure that training for all tasks is either currently available or that actions needed to make it available are addressed. As would be expected, there is a close correlation between training and manpower, with three LSA reports common to both elements.

Within the MOD, assistance is available throughout the development of training courses, covering Training Needs Analysis, Course Design and delivery of courses or the identification of suitable commercial courses. Centralised training for all Services is available, focussed on programming languages and specific techniques for the maintenance of software systems that are already operational. Where the current courses do not meet the needs of a new project, there may either be the need to arrange for the provision of this training in-Service or, if numbers indicate that this is not viable, arrange contracts for commercial courses. Regardless of application language, in-Service training is most likely to occur for Operational Support tasks where personnel need training on, for example, the equipment used to load the software.

The primary report for Training and Training Equipment is LSA – 014, Training Task List. Secondary reports are LSA – 001, Man-hours by Skill Speciality Code and Level of Maintenance; LSA – 011, Special Training Equipment / Device Summary; LSA – 018, Task Inventory Report; LSA – 019, Task Analysis Summary; LSA – 075, Consolidated Manpower, Personnel and Training Report and LSA – 634, Training Facilities Report.

LSA – 014, Training Task List

Although LSA – 014, Training Task List provides a report by Skill Speciality Code and LCN of every task identified, it is used in this ILS element to identify those tasks for which some form of training is recommended under Training Recommended, DED 463. This is linked to the requirement for training in a New or Modified Skill, DED 257, since this report details all tasks that will require training, whether as part of New or Modified Skill Codes, or as utilisation of existing training courses. Therefore, all personnel selected for employment on the equipment will have their full training requirements identified. By selecting the report on the basis of Training Recommended, the report will show the Rationale For Training Recommendation, DED 462, and the Rationale For Training Location, DED 461. To assist in establishing the scope of the training requirement, the type of training, from Training Recommended; Training Level, DED 702; Task Conditions, DED 428; and Performance Standards, DED 287 are also included.

LSA – 001, Man-hours by Skill Speciality Code and Level of Maintenance

Although a secondary report for Training and Training Equipment, LSA – 001, Man-hours by Skill Speciality Code and Level of Maintenance serves little purpose. The information given about tasks is adequately covered by the other secondary reports. Therefore, unless there is a need to review training against the annual man-hours expended by task, this report need not be used.

LSA – 011, Special Training Equipment / Device Summary

LSA – 011 is produced for LCNs that have a valid entry for mean man-minutes, mean elapsed time and a Training Equipment Required code under DED 358. This indicates that a degree of task analysis has been carried out, rather than mere task identification. It lists all operator or maintenance tasks that require a special training device but does not identify the particular device. According to the Def Stan 00-60 description of this report, it should be used to identify the requirements and provide the justification for the acquisition of training devices. However, this can not be accomplished without identifying the devices and providing the additional information that would be contained in a training device narrative. This information should be defined in a new DED for inclusion in this report.

LSA – 018, Task Inventory Report

In carrying out an assessment of the training needed and in designing the training course, a description of the task in hand is required. LSA – 018, Task Inventory Report provides this information, giving a complete identification and breakdown of tasks. In conjunction with the Task Condition and Performance Standards detailed in LSA – 014, this information can be used to assess the training objectives and subsequent satisfaction of the objectives.

LSA – 019, Task Analysis Summary

LSA – 019, Task Analysis Summary contains essentially the same training information as is provided by LSA – 018 – a narrative description of each task. The additional information given is the mean time that a task takes and the spare parts and support equipment required to complete the task. While LSA – 018 provides an easy to read description of tasks, there is no description of the support equipment and spares that be required to complete the training. Dependent on the task and training required, LSA – 018 and LSA – 019 may therefore be used separately or in combination.

LSA – 075, Consolidated Manpower, Personnel and Training Report

LSA – 075 is the third report shared with Manpower and Human Factors and its use is described in Chapter 7. As stated in that chapter, Part 2 of the report is of most use, being directed at the training requirements for New or Modified Skills and indicating educational, physical and mental requirements for those selected to undertake the training described.

LSA – 634, Training Facilities Report

The LSA – 634, Training Facilities Report has already been described under the Facilities element and, beyond the fact that it concludes the essential training considerations – what training is required, using what equipment, to what standard and where will it be conducted, this report requires no further explanation for this ILS element.

Summary

The provision of training and the associated equipment and facilities parallels the Manpower and Human Factors and Facilities elements. For Software Support, it is important to establish the full training requirements by carrying out a Training Needs Analysis and identifying new or existing training courses that fulfil the needs. Although there is some in-house training available, this may not be adequate. As a result, commercial courses may be required or there may be a justification for the establishment of new courses and the facilities and equipment required to support them.

TECHNICAL DOCUMENTATION

The Technical Documentation element does not have any primary reports. Secondary reports used are LSA – 019, Task Analysis Summary, LSA – 602, the Candidate Item Maintenance and Upkeep Plan and LSA – 674, Electronic Documentation Requirements (AECMA S1000D) Report. AECMA S1000D [AECMA, 1999] is an electronic documentation standard, the use of which is detailed in Def Stan 00-60 Part 10. The significance of Software Support tasks is recognised in this part of the Def Stan. Investigation of the individual elements used to define the AECMA S1000D / Def Stan 00-60 Data Module Code for software systems is beyond the scope of this dissertation and will not be examined further here.

Overall, the technical documentation requirements for Software Support can be considered in two distinct categories:

a. Development documents produced to enable management of the process and used as inputs to subsequent development stages.

b. Procedural documents which define the execution of a task, whether Software Maintenance or Operational Support.

Unlike hardware maintenance, software maintenance relies on the provision of up-to-date design information, right back to the requirements, for success. Without development documentation being passed from developer to maintainer, even if they are within the same organisation, it will be almost impossible to carry out effective Software Maintenance. In modern development environments, the production and interchange of development documentation may be entirely electronic. In this case, the maintaining organisation will need to identify, through the Support and Test Equipment element, the hardware and software resources required to continue using the same documentation methods.

There is no LSAR review process which deals with electronic development documentation and review of this material will require close co-operation with the contractor since the development data may be on a system which spans many projects or customers. In order to review project specific documentation, data may have to be segregated to protect Contract in Confidence information and this should be considered at the outset of the contract.

Regardless of medium, the development documentation required to support a project must be identified as early as possible, using knowledge of the proposed development method or standard. The required documentation should then form part of the Contract Data Requirements List (CDRL), a list of the documents which the contractor is obliged to deliver. While it is one thing to deliver documentation, this documentation must also be reviewed by the customer to ensure that it meets their requirements, provides traceability and reflects the proposed design. Any document review must be timely, there is little point in reviewing the Software Requirements Specification when the system has entered Acceptance Testing.

LSA – 019, Task Analysis Summary and LSA – 602, Candidate Item Maintenance and Upkeep Plan

For hardware tasks, technical authors produce Maintenance Procedures from the Data Module information extracted from the LSAR for use by the support organisations. To do this, detailed information about how the tasks are to be carried out and the equipment to be used is required. This can be extracted from the LSAR from Task and Sequential Sub-task Descriptions and Support, Test Equipment and Tools data. This information is presented in LSA – 019 and LSA – 602. Procedures down to the level of 'Using the long shafted screwdriver, NSN XYZ, remove the 4 screws …..' and so on can then be constructed. In Operational Support tasks, there will be a defined process for loading, unloading, replicating and distributing software. However, as described in Chapter 2, detail down to this level for Software Maintenance tasks is neither practical nor possible to achieve in a meaningful manner. For example, it is not possible to define the precise procedure for a programmer to change lines of code in response to an as yet undetected fault.

LSA – 674, Electronic Documentation Requirements (AECMA S1000D) Report

Although development documentation may be produced electronically, the LSA – 674 report is for the validation of electronic documentation data derived from the LSAR, ie the procedural information about tasks. Having received Maintenance Procedures from the technical authors, LSA – 674 would be used to ensure that the relevant information is being presented. For Software Support, this report would be used to review information presented on Operational Support tasks. The report contains the narrative information about each task and sub-task and the supporting equipment for each task is listed. In order to mirror the ordering of information in a Maintenance Procedure and simplify the review process, the narrative descriptions appear in sequential order.

Summary

Technical documentation is required for both Operational Support and Software Maintenance. Operational Support documentation will be procedural information, suited to the AECMA S1000D electronic documentation format. Software Maintenance documentation may be procedural for some tasks but the scope of such documentation is beyond AECMA S1000D. The requirement for the development organisation to provide design information and then for the maintenance organisation to use and maintain development documents is unique to Software Support. Access to the development documentation is vital and review of such documentation during the development phase is essential to ensure that the documentation provided meets the requirements of the maintenance phase.

PACKAGING, HANDLING, STORAGE AND TRANSPORTATION

As with Supply Support, the application of PHS&T to software does not accord with the normal LSA data gathering and analysis processes. Therefore, the LSA reports for PHS&T, LSA – 676, LSA – 026, LSA – 085 and LSA – 654, need not be used. The software product from a development environment is 'packaged' in some form of transportation medium which may have its own packaging and day-to-day handling requirements. Storage will be also be required for the transportation media and for archive and back-up copies of software. The issue of transporting software has to be considered in the sense of the supply chain delivering software from source to point of installation; with the ability to transmit software electronically, new and innovative techniques are available.

In general, the media used for the loading and storage of software will be a commercially available, consumable item such as CD-ROMs, 3.5" disks, magnetic tape, DAT or flashcard. As such, LSA will not be conducted on these items as in-Service support for consumables is not required. However, if a system used a complex, expensive storage medium which was repairable, then there would be a case for conducting LSA. There are, therefore, no LSA reports which can summarise the PHS&T requirements of software and the information detailed in the following paragraphs should therefore be assessed and recorded in the LSA – 672 report as a new DED.

Packaging

Deliverable code will, at some stage between development and installation, be replicated onto some form of transportation medium. The requirement to load software contained on this medium using a particular item of equipment will be identified as part of the task analysis process detailed in LSA – 019, in addition to being recorded in LSA – 672. The required media can then be procured and specific packaging requirements identified. At the point of replication, where software is transferred to the transportation medium, the end package will require identification of its contents. At this point, the previously blank medium, procured for the project and demanded by the replicating organisation using a NATO Stock Number (NSN), changes state and now exists as, for example, the Tornado Main Computer OFP. The medium can no longer be considered to have an NSN as before, instead it is now merely identified by its contents. As regarding re-ordering and issuing media, this change in state only affects the replication organisation. They will receive a request for a replacement program from the user, they demand the blank replacement media using its NSN and issue it forward, identified by its contents.

When installed on an item of hardware, the procedure for the identification of that hardware item, configured with a specific release of software is detailed in Def Stan 00-22 [Def Stan 00-22, 1991]. This ensures that where there is the possibility of installing or re-configuring a system with different releases or versions of software, the state of the software loaded is visible at the appropriate level of maintenance and recorded in a Master Record Index.

Handling

The handling of software, both in the development environment and from the point of replication onwards, depends on two things: its security classification and its physical characteristics. The development of software may encompass techniques, processes or products which are classified. There will then be a need to provide the required levels of protection, in the sense of access rights, protective marking and data storage and transmission for each element. The security issues surrounding a project may become significant drivers in assessing supportability. An example of this would be a system being developed by a contractor, the end use of which was government approved for supply to a foreign customer but the algorithms used within the product could not be approved for export on national security grounds. All other factors may indicate that the software is maintainable and supportable but if security considerations outweigh other factors, the only option may be for the original developer to support the software. Similarly, downstream factors such as the ease of access to the system or data in normal operation may influence supportability and design features and in-use procedures. For example, from the point of replication, the floppy disk used to install the software may become classified because of its contents, requiring particular handling and storage procedures. Similarly, the target hardware may require de-classification prior to its removal for maintenance or the re-loading of software due to its previous configuration. The security aspects of handling and protecting software are covered in JSP 440 [JSP 440, 1996].

In an environmental sense, there may also be handling considerations for software transportation media. When media is issued for loading onto the host system, there may be warnings and cautions that apply. The media may be fragile, susceptible to electro-static or electro-magnetic influences or have limitations on temperature and humidity at the time of loading.

Storage

Both in development and in use, there will be a requirement to store the media onto which software has been transferred. In the development environment this will entail the safe archiving and storage of master and back-up copies of all files (including documentation) used to develop the end

product. Such storage facilities will be subject to environmental considerations such as those detailed in British Standard 4783 [BS 4783, 1988].

Precautions should also be taken to ensure that there is an accurate mirror image of this archive kept in a safe, separate location, preferably remote from the building in which the master archive is housed. This separate image is required for disaster recovery and implementing it will also require disaster recovery procedures. An outline of these procedures and their effect on Software Support, such as the provision of additional equipment, should be recorded in the LSAR. Archiving material in the correct environmental conditions is both sound practice and ensures that the supportability and maintainability of the software is not compromised.

The in-use storage requirements for software may be somewhat different from that in development. As a piece of media is issued to the user, the conditions under which it will be used and stored are likely to be harsher that the benign storage environment of the master copy; a helicopter squadron operating in Northern Norway is unlikely to have an environmentally controlled storage facility for its software at a dispersed site in a wood. As such, the life of the media will be shorter than its safely stored development counterpart. These conditions should be accepted and catered for. General areas to analyse and address will be storage requirements, such as a safe for classified software, protective packaging for the media and possibly lifing of the media to enable timely replacement.

Transportation

As with any other system component, software must be transported from its point of production to the customer and thence to the end user. The means of transportation may be similar, taking into account British Standard 4783, to a hardware item for delivery through the normal supply channels to its final destination. However, given the dispersed nature of military operations today, the end user may be in an inhospitable environment to which standard supply routes may take a significant time to reach. This may be operationally significant where updated electronic warfare data is required to be loaded before the next mission.

Where the opportunity exists for the digital transmission of software, this should be considered; it may be highly cost effective as well as operationally effective. An example of a scenario where electronic transportation is the only option is in space. Satellites and space exploration programmes need the ability to deliver upgrades to the operational platform without resorting to sending a maintenance mission to carry out the procedure. Although this is not a

widespread practice in the aerospace environment, in assessing technological opportunities under LSA Task 204, there may be benefits in downloading software from a software library directly into an aircraft or other piece of equipment, either by landline or satellite link. Regardless of the transportation method chosen, the issue must be addressed such that, given the release frequency of the software and the dispersion of end users, the logistics chain is in place. This will ensure that the configuration management of the fleet does not become a problem due to remote locations lagging behind more central units.

Summary

The concept of applying PHS&T to software may seem unusual, given the traditional ILS application of this element. However, the fact that it does not conform to the norm does not mean that it can be ignored. The packaging and identification of software, such that the end user can be sure of installing the correct version; handling information, particularly for classified material; the storage of media and the means by which software is delivered from source to point of installation are all issues which are of concern but which differ slightly from the ILS norm. The provision of a PHS&T narrative in LSA – 672 is the best means of recording the information gathered as part of this element.

CONCLUSIONS

The application of ILS/LSA to software is as vital as it is to hardware items in modern military systems. The ILS elements detailed in Def Stan 00-60 can be used, with the interpretation and guidance given here, allowing software to be treated as an integral part of a system. In carrying out a supportability assessment, the LSAR is a suitable tool for recording the information gathered and LSAR reports provide a suitable format for reviewing this information. The most significant ILS element to be addressed is Maintenance Planning; the derivation of a Maintenance Concept and identification of the maintenance processes and activities form the cornerstone of all other work. The element least applicable is Supply Support, being concerned with the initial provisioning of items and subsequent re-provisioning of spare parts. From the work carried out in Maintenance Planning, there will be requirements for equipment to carry out tasks, both Operational Support and Software Maintenance. The identification of equipment required may lead to a requirement to conduct ILS/LSA on that equipment. Other elements where the ILS/LSA work is essentially the same as for hardware are Facilities; Manpower and Human Factors and Training and Training Equipment.

Addressing Software Reliability is the major area in which the ILS/LSA process is different between hardware and software. The reliability information gathered for software cannot be recorded in the LSAR in the same manner as for hardware since some of the data cannot be collected or measured for software and the results of subsequent calculations carried out within the LSAR would make no sense. Software reliability is still a significant factor and should be addressed as a separate programme, running parallel to the ILS and product development processes. The reliability programme should encompass an examination of the development process and a measure of reliability through statistical measurement. Examination of the development process provides information about the ability of the developer to deliver the product required and infers some level of assurance about the product. Such methods could be the use of the ISO 9000 series of standards, employment of [ISO 12207:1995] or the use of process improvement models, for example, CMM. Statistical reliability measurement centres on the prediction of the number of faults within a product and metrics gathered during testing about the rate at which faults manifest themselves as failures. Application of statistical methods should be carried out as a partnership between the developer and customer to ensure that the work carried out is valid, given the operation of the system, and provides realistic data on the software reliability that can be expected when the system enters service. Caution should be exercised since the statistical measurement of reliability during the testing phase will be different from that which can be expected after in-Service maintenance and re-calculation of the reliability figures may be required following maintenance activity.

For Technical Documentation the outcome of the ILS/LSA process is the same for software as for hardware: provision of technical information which will allow Operational Support and Software Maintenance tasks to be carried out but the means of achieving this is slightly different. The technical documentation required for Software Support encompasses design documentation as well as instructions detailed in a technical manual. Operational Support tasks, being more procedural in nature, fit with the standard ILS/LSA process and are suited to the application of [AECMA S1000D, 1999].

The PHS&T element of Software Support, deals with the transfer medium, its marking for use and the security implications of handling both the development information and the transfer medium. As with all products, there is a need for storage but again there is a need to consider the storage of design data and master copies of development files. In the Operational Support environment, storage of the transfer media may also have to be dealt with. Transportation factors, although seeming an obscure consideration for software, should also be analysed. The manner in which the software is to be delivered form the development or maintenance environment to the person tasked with loading it into the host equipment must be understood and suitable arrangements made.

Future Work

In carrying out research into the field of software supportability, there are three main areas that in which I consider future research would be valuable:

a. Software cost estimation for the development and support of software is currently a weak area, compared to the initial purchase costs, spares costs, asset depreciation and manpower costs that can be allocated to hardware. Current software cost estimation tools provide information on the effort required, in time and manpower for software production, either in initial development or during maintenance. As illustrated in this dissertation, the cost of supporting software is much greater than simply the cost of time and manpower. Models should be developed which provide information on the costs associated with each of the ILS elements, being summed to give a total support cost. The cost of supporting software throughout its in-Service life would then be much more visible, providing a better input to project plans and budgets.

b. There is scope for deeper investigation into the supportability factors affecting COTS products. The increase in COTS software in both military and commercial worlds creates some peculiar support issues. In general, these issues can be covered by applying the same principles as for bespoke software supportability, but with some constraints in place from the outset. I believe that in the next few years, when military systems incorporating COTS software are maturing in operational use, there may be support issues which are beyond current thinking and practical experience.

c. The field of software reliability is growing in significance and adequate coverage of the whole subject is beyond the scope of this dissertation. There is a considerable amount of literature on software reliability, both in terms of determining the reliability of a product at the point of release and in determining reliability for safety critical functions for inclusion in system-level analysis. In comparison, the effect of operational experience and software maintenance on the reliability of military systems seems to be less well understood or documented.

ANNEX A

GLOSSARY OF TERMS AND ACRONYMS

AFOTEC	Air Force Operational Test and Evaluation Center.
AOR	Annual Operating Requirement. The estimated or required yearly rate of usage of an item.
BCS	Baseline Comparison System. A current operational system, or a composite of current operational sub-systems, which most closely represents the design, operational, and support characteristics of the new system under development.
CASE	Computer Aided Software Engineering.
CDRL	Contract Data Requirements List. A form used as the main list of data and information which the contractor will be obligated to deliver under the contract.
CI	Candidate Item. An item being considered for support analysis.
CLS	Contractor Logistics Support. An agreed level of support provided by the contractor in place of Service personnel and facilities.
COTS	Commercial-Off-the-Shelf. An unmodified, commercially available product.
CSCI/ CSI	Computer Software Configuration Item / Configured Software Item. A software item which is uniquely identified and controlled within a defined configuration management process.
CSU	Computer Software Unit. An element specified in the design of a CSCI that is separately testable and contains a group of modules that are (usually) functionally related.
DA	Design Authority. The approved firms, establishment or branch responsible for the detailed design of materiel to approved specifications and authorised to sign a certificate of design, or to certify sealed drawings.
DE	Data Element. An individual field in the LSAR tables.
DED	Data Element Definition. The definition of the contents of a Data Element
DERA	Defence Evaluation & Research Agency.
DSLOC	Deliverable Source Lines of Code.
EI	End Item. A final combination of end products, component parts and/or materials which are ready for their intended use. eg ship, tank, aircraft etc.

FMECA	Failure Modes Effects and Criticality Analysis. An analysis to identify potential design weaknesses through systematic, documented consideration of the following: (a) All likely ways in which component or equipment can fail. (b) Causes for each mode. (c) The effects of each failure (which may be different for each mission phase). (d) The criticality for each failure both for safety and for mission success.
ICC	Item Category Code. A code which identifies a type of item and indicates categories into which support and test equipment, spares, repairs parts, etc. may be divided.
IPC	Illustrated Parts Catalogue.
IPL	Initial Provisioning List.
IPR	Intellectual Property Rights.
ILS	Integrated Logistic Support. A disciplined management approach, affecting both customer and industry, aimed at optimizing equipment Life Cycle Costs. It includes elements for influencing equipment design and determining support requirements to achieve supportable and supported equipment.
LCN	LSA Control Number.
LRI	Line Replaceable Item. Any functional item which can be removed from the equiment as part of a single maintenance action.
LSA	Logistics Support Analysis. The selective application of scientific and engineering analyses, during the system engineering and design process to assist in complying with supportability and other ILS objectives.
LSAR	Logistics Support Analysis Record. That portion of LSA documentation consisting of detailed data pertaining to the identification of logistic support resource requirements of an equipment.
Maintainability	The ability of an item under stated conditions of use, to be retained in or restored to a specified condition when maintenance is performed by personnel having specified skill levels under stated conditions and using prescribed procedures and resources.

MOD	Ministry of Defence
MTBF	Mean Time Between Failure.
MTTR	Mean Time To Repair.
NSN	NATO Stock Number.
OFP	Operational Flight Program. A mission dependent program or set of files.
Operational Support	Activities carried out in direct support of the end user.
PHS&T	Packaging, Handling, Storage and Transportation.
PSE	Project Support Environment. The infrastructure of methods and tools used to support the various stages of specification, procurement, development, maintenance and management of a project.
RCM	Reliability Centred Maintenance. A systematic approach for identifying preventative maintenance tasks for an equipment end item in accordance with a specified set of procedures and for establishing intervals between maintenance tasks.
Reliability	The ability of an item to perform a required function under stated conditions for a stated period of time. NOTE: The term reliability is also used as a reliability characteristic denoting a probability of success, or a success ratio.
SCE	Software Capability Evaluation.
SSC	Skill Speciality Code. Describes the maintenance or operator skill required to accomplish the task.
SIL	Safety Integrity Level – see Def Stan 00-56.
Software Maintenance	Corrective, adaptive or perfective maintenance activities carried out through the in-Service life of software.
Software Support	All activities concerned with supporting the operation of software, and with sustaining the ability of software to satisfy the required system performance and functionality during its operational life.

SE / STE	Support Equipment / Support & Test Equipment. All equipment (mobile or fixed) required to support the operation and maintenance of a materiel system. This includes associated multi-use end items, ground handling and maintenance equipment, tools, metrology and calibration equipment, communications resources, test equipment and automatic test equipment, with diagnostic software for both on and off equipment maintenance. It includes the procurement of logistics support for the support and test equipment itself.
Supportability	The degree to which system design characteristics and planned logistic resources, including manpower, meet the system peacetime and wartime availability requirements.
Software	A computer program or data entity, and the associated design or descriptive documentation.
TMDE	Test, Measurement and Diagnostic Equipment.
Tailoring	The process by which the individual requirements (paragraphs, sub-paragraphs, or sentences) of the selected documents are evaluated to determine the extent to which each requirement is most suitable for a specific equipment acquisition and the modification of these requirements, where necessary, to ensure that each achieves an optimal balance between operational needs and costs. Process must conform to provisions of existing regulations governing R&M programmes and take care not to exclude those requirements which are determined as essential to meeting operational needs.

ANNEX B

This Annex lists the DEDs required in an assessment of software supportability and the DED 427 Task Codes applicable to Software Engineering tasks. The list contains the DEDs which are output to LSA reports and those which are required to maintain the structure of the LSAR, such as Key and Mandatory Fields.

DED	Comment
002 ACQUISITION DECISION OFFICE	
007 ADDITIONAL SKILL REQUIREMENT: SKILL REQUIRING A NEW OR REVISED SKILL CODE	
012 ADDITIONAL TRAINING REQUIREMENTS	
015 SUPPORT EQUIPMENT ALLOCATION DATA	
016 ALLOWANCE	
019 ALTERNATE LOGISTIC SUPPORT ANALYSIS CONTROL NUMBER CODE (ALC)	
023 ANNUAL OPERATING REQUIREMENT (AOR)	The AOR of the host equipment.
028 AVAILABLE ANNUAL MAN-HOURS	
037 CALIBRATION INTERVAL	
038 CALIBRATION ITEM INDICATOR	
042 CALIBRATION TIME	
044 SUPPORT EQUIPMENT CHARACTERISTICS	
043 CHANGE AUTHORITY NUMBER	
046 COMMERCIAL AND GOVERNMENT ENTITY (CAGE) CODE	
047 COMMERCIAL AND GOVERNMENT ENTITY CODE ADDRESS	
049 COMPENSATING DESIGN PROVISIONS	
050 COMPENSATING OPERATOR ACTION PROVISIONS	
056 CONTRACTOR FURNISHED EQUIPMENT/GOVERNMENT FURNISHED EQUIPMENT	
071 DATE	
078 SUPPORT EQUIPMENT DESCRIPTION & FUNCTION	
079 DESIGN DATA CATEGORY CODE	
088 DRAWING CLASSIFICATION	
089 DRAWING NUMBER	
090 DUTY	
091 DUTY CODE	
092 DUTY POSITION REQUIRING A NEW OR REVISED SKILL	
094 EDUCATIONAL QUALIFICATIONS	
096 END ITEM ACRONYM CODE (EIAC)	
101 ESTIMATED PRICE	
105 FACILITIES DESIGN CRITERIA	
106 FACILITIES INSTALLATION LEAD TIME	
107 FACILITIES MAINTENANCE REQUIREMENTS	
108 FACILITIES REQUIREMENTS	
109 FACILITIES REQUIREMENTS FOR OPERATIONS	
110 FACILITIES REQUIRED FOR TRAINING	
111 FACILITIES UTILIZATION	
112 FACILITY AREA	
113 FACILITY BASELINE NARRATIVE CODE	
114 FACILITY CAPABILITY	
115 FACILITY CATEGORY CODE	
117 FACILITY LOCATION	
118 FACILITY NAME	
119 FACILITY NARRATIVE CODE	
120 FACILITY REQUIREMENTS: SPECIAL CONSIDERATIONS	

DED	Comment
121 FACILITY REQUIREMENTS: SUPPLY/STORE	
122 FACILITY TASK AREA BREAKDOWN	
123 FACILITY UNIT COST RATIONALE	
124 FAILURE CAUSE	
125 FAILURE/DAMAGE EFFECTS: END EFFECT	
126 FAILURE/DAMAGE EFFECTS: LOCAL	
127 FAILURE/DAMAGE EFFECTS: NEXT HIGHER	
128 FAILURE/DAMAGE MODE	
129 FAILURE DETECTION METHOD	
131 FAILURE MODE & RELIABILITY-CENTERED MAINTENANCE (RCM) NARRATIVE CODE	
134 FAILURE MODE INDICATOR	
135 FAILURE MODE INDICATOR MISSION PHASE CHARACTERISTICS NARRATIVE CODE	
137 FAILURE MODE REMARKS	
147 FUNCTIONAL ANALYSIS	
148 SUPPORT EQUIPMENT GENERIC CODE	
152 HARDNESS CRITICAL PROCESS (HCP)	
153 HARDWARE DEVELOPMENT PRICE	
155 HAZARDOUS MAINTENANCE PROCEDURES CODE	Used wher a Software Operational Support task is hazardous
162 INDENTURE CODE	
168 INPUT POWER SOURCE	
169 INSTALLATION FACTORS OR OTHER FACILITIES	
171 INTEGRATED LOGISTIC SUPPORT REQUIREMENTS CATEGORY CODE	
177 ITEM CATEGORY CODE (ICC)	Requires a new code for software
179 ITEM DESIGNATOR CODE	
180 ITEM FUNCTION	
182 ITEM NAME	
185 JOB	
186 JOB CODE	
188 JUSTIFICATION	
190 SUPPORT EQUIPMENT LIFE CYCLE STATUS	
191 SUPPORT EQUIPMENT LIFE SPAN	
198 LOGISTICS DECISION OFFICE	
199 LOGISTIC SUPPORT ANALYSIS CONTROL NUMBER (LCN)	
200 LOGISTIC SUPPORT ANALYSIS CONTROL NUMBER INDENTURE CODE (LCN-IC)	
201 LOGISTIC SUPPORT ANALYSIS CONTROL NUMBER (LCN) NOMENCLATURE	
203 LOGISTIC SUPPORT ANALYSIS CONTROL NUMBER TYPE (LCN TYPE)	
207 MAINTENANCE CONCEPT	
210 MAINTENANCE PLAN RATIONALE	
217 SUPPORT EQUIPMENT REPAIR MANAGEMENT ORGANIZATION	
224 MEAN ELAPSED TIME	
225 MEAN MAN-HOURS	
226 MEAN MAN-MINUTES	Change of format required for a 6 digit representation of minutes
227 MEAN MINUTE ELAPSED TIME	
237 MEANS OF DETECTION	

DED	Comment
238 MEASUREMENT BASE (MB)	Software Size Measurement Base to be agreed between the Project and Contractor.
244 MINIMUM EQUIPMENT LIST NARRATIVE	
246 MISSION PHASE CODE	
247 MISSION PHASE/OPERATIONAL MODE	
253 NATO STOCK NUMBER	
255 NEW OR MODIFIED FACILITY NARRATIVE CODE	
256 NEW OR MODIFIED SKILL NARRATIVE CODE	
257 NEW OR MODIFIED SKILL SPECIALITY CODE (SSC)	
262 NUMBER OF OPERATING LOCATIONS	
266 ITEM INTEROPERABILITY CODE	
268 SUPPORT EQUIPMENT OPERATING DIMENSIONS	
270 OPERATING WEIGHT	
275 OPERATIONAL REQUIREMENT INDICATOR	
277 OPERATIONS/MAINTENANCE LEVEL	
284 PARAMETERS	
287 PERFORMANCE STANDARDS	
288 SUBTASK PERSON IDENTIFIER	
290 PHYSICAL AND MENTAL REQUIREMENTS	
316 QUANTITY PER ASSEMBLY (QTY/ASSY)	Used to show where identical software is resident in a number of places in an Assembly.
317 QUANTITY PER END ITEM (QTY/EI)	Used to show where identical software is resident in a number of assemblies within an End Item.
319 QUANTITY PER TASK	
320 QUANTITY PER TEST	
327 REASON FOR SUPERSEDURE/DELETION	
330 RECOMMENDED RANK/GRADE	
337 REFERENCE NUMBER	
341 RELIABILITY AVAILABILITY MAINTAINABILITY CHARACTERISTICS NARRATIVE CODE	
347 RELIABILITY/MAINTAINABILITY INDICATOR CODE	
358 REQUIREMENTS FOR	
360 REVISION	
362 SAFETY HAZARD SEVERITY CODE	Conflicts with 00-56 SILs, Def Stan 00-60 Help Desk query raised.
369 SECURITY CLEARANCE	
372 SEQUENTIAL SUBTASK DESCRIPTION	
376 SERVICE DESIGNATOR CODE	
386 SKILL LEVEL CODE	
387 SKILL SPECIALITY CODE (SSC)	
388 SKILL SPECIALITY EVALUATION CODE	
389 SOURCE, MAINTENANCE & RECOVERABILITY CODE (SMR)	
399 SUPPORT EQUIPMENT SPECIFIC AUTHORIZATION	
407 SUBTASK NUMBER	
408 SUPPORT EQUIPMENT SUPERSEDURE TYPE	
412 SUPPORT EQUIPMENT FULL ITEM NAME	
413 SUPPORT EQUIPMENT GROUPING	
414 SUPPORT EQUIPMENT NARRATIVE CODE	
416 SUPPORT EQUIPMENT RECOMMENDATION DATA NUMBER (SERD NUMBER)	

DED	Comment
418 SUPPORT EQUIPMENT REQUIRED	
424 SYSTEM/END ITEM NARRATIVE CODE	
425 SYSTEM REDESIGN/LOGISTICS CONSIDERATION CODE	
426 SYSTEM REDESIGN/LOGISTICS CONSIDERATION RECOMMENDATION, DISPOSITION,	
427 TASK CODE	Codes Listed below
428 TASK CONDITION	
429 TASK CRITICALITY CODE	
430 TASK FREQUENCY	
431 TASK IDENTIFICATION	
433 TASK TYPE	New Code needed - Software Maintenance (S)
437 TECHNICAL MANUAL CODE (TM CODE)	
440 TECHNICAL MANUAL NUMBER	
450 TEXT SEQUENCING CODE (TSC)	
454 TOTAL SYSTEMS SUPPORTED	
461 TRAINING LOCATION RATIONALE	
462 TRAINING RATIONALE	
463 TRAINING RECOMMENDATIONS	
482 TYPE OF CONSTRUCTION	
483 FACILITY TYPE	
490 UNIT OF ISSUE PRICE (UI PRICE)	
491 UNIT OF MEASURE (UM)	
492 UNIT OF MEASURE PRICE	
501 USABLE ON CODE (UOC)	
502 UTILITIES REQUIREMENT	
514 WORK AREA CODE	For on-equipment work such as loading / unloading software
603 AEROSPACE GROUND EQUIPMENT CODE	
604 AFDEETEC/AFDSEC NUMBER	
621 CALIBRATION MARKER (AECMA 2000M TEI CMK)	
629 CURRENCY CODE (AECMA 2000M TEI CUR)	
632 DOMESTIC MANAGEMENT CODE (AECMA 2000M TEI DMC)	
649 MAINTENANCE CONCEPT COSTS	
650 MAINTENANCE CONCEPT OPTIONS	
682 SIMILAR TO, SAME AS, DERIVED FROM AND FITTED TO	
683 SOFTWARE SUPPORT PLAN (SSP) REQUIRED	
685 SPECIAL REQUIREMENTS	
687 SPECIFICATION/DRAWING NUMBER	Use to identify the Software Requirement satisfied.
689 SUPPORT EQUIPMENT CATEGORY	
691 SYSTEM/EQUIPMENT IMPORTANCE CODE	Change title to Mission Criticality Code, definitions as per Table 3, page 2-11
692 TASK/SUBTASK ASSOCIATED NARRATIVE CODE	
693 TECHNICAL PUBLICATION COMMENTS	
694 TECHNICAL PUBLICATION REQUIREMENTS	
695 TECHNICAL PUBLICATION TITLE	
696 THIRD LINE AVAILABILITY	
701 TRADE CODE MANPOWER	The number of personnel, from each trade, required to perform each task

DED	Comment
702 TRAINING LEVEL	
706 UPKEEP/WEAPONS REPAIR POLICY CODE	For software, this is the Maintenance Policy Code
709 SUPPORT EQUIPMENT VALIDATION/COMPATIBILITY CHECK	
711 WORK CENTRE CODE	
801 DISASSEMBLY CODE	
802 DOCUMENT CODE	
803 INFORMATION CODE	
804 INFORMATION CODE VARIANT	
806 ITEM LOCATION CODE	
807 SOFTWARE DELIVERABLE SOURCE LINES OF CODE (DSLOC)	Change Title to Software Size, to be used in conjunction with Measurement Base, DED 238.
808 SOFTWARE DESIGN AUTHORITY	
809 SOFTWARE ENGINEERING METHODS AND TECHNIQUES	
810 SOFTWARE INTELLECTUAL PROPERTY RIGHTS (IPR) HOLDER	
811 SOFTWARE INTERDEPENDENCY NARRATIVE	Provides details about the interdependencies idientified under new DEDs JJJ, KKK and LLL
812 SOFTWARE PROGRAM SIZE	
813 SOFTWARE RELEASE FREQUENCY	
814 SOFTWARE SAFETY INTEGRITY LEVEL	
815 SOFTWARE TOTAL MEMORY REQUIRED	
816 SOFTWARE VERSION NUMBER	Requires change to 32 Alphanumeric Characters
830 MAINTENANCE PROCEDURE IDENTIFIER	
831 MAINTENANCE PROCEDURE TITLE	
832 TASK SEQUENCE NUMBER	
833 SOFTWARE IDENTIFIER CODE	

New DEDs	Comment
AAA Annual Change Traffic	The number of change requests approved and submitted to the maintenance organisation for action.
BBB Language	The programming language or database language used.
CCC Software Loading Level	The Maintenance Level at which the CI is loaded onto the system, codes as per DED 277.
DDD Processor Utilisation	Percentage Utilisation of Processor Capability
EEE Bus Utilisation	Percentage Utilisation of Bus Capability
FFF Memory Utilisation	Percentage Utilisation of Memory
GGG Complexity Factor	Software Complexity Factor, see Table 4, Page 2-17
HHH Maintainability	Software Maintainability Factor, see Table 5, Page 2-17
III Training Device Narrative	A description of the training device proposed and its justification.
JJJ Load	Indicates a loading dependency between a program or file and a hardware item.
KKK Implementation	Indicates a link between a software function and the program or file in which it is implemented.
LLL Relies On	Indicates that a software function and hardware item rely on each other during operation.
MMM Software PHS&T narrative	Software PHS&T factors as described in Chapter 10.

DED 427 Task Codes

For Function, possible codes are:

Debug. To detect and remedy an inadequacy in software.

Debug 2

Evaluate. To determine the importance, size or nature of; to appraise; to give value or appraisal to on the basis of collected data.

Evaluate 8

Inspect. Measure, examine, test, gauge or otherwise compare the item with the applicable requirements.

Inspect A

Load/Unload. To place, insert in or take out a device or piece of equipment; to place or remove components from aircraft or other vehicles; to place or remove weapons; to upload or download software.

Load/Unload 4

Preserve. The action required to treat systems and equipment whether installed or stored, to keep them in a satisfactory condition.

Preserve V

Process. To submit to a series of actions or operations leading to a particular end.

Process 7

Set Up. To prepare or make an item ready for operation; to prepare software prior to mission.

Set up 5

Software Modification. The development and implementation of a design change to an in service software item.

Software Modification 0

Test. Undertake, using the appropriate test equipment, a critical trial or examination of one or more properties or characteristics of the item or system to make certain that it is serviceable and operates correctly.

Test B

Transport. The action required to move systems and equipment from one place to another.

Transport Y

ANNEX C

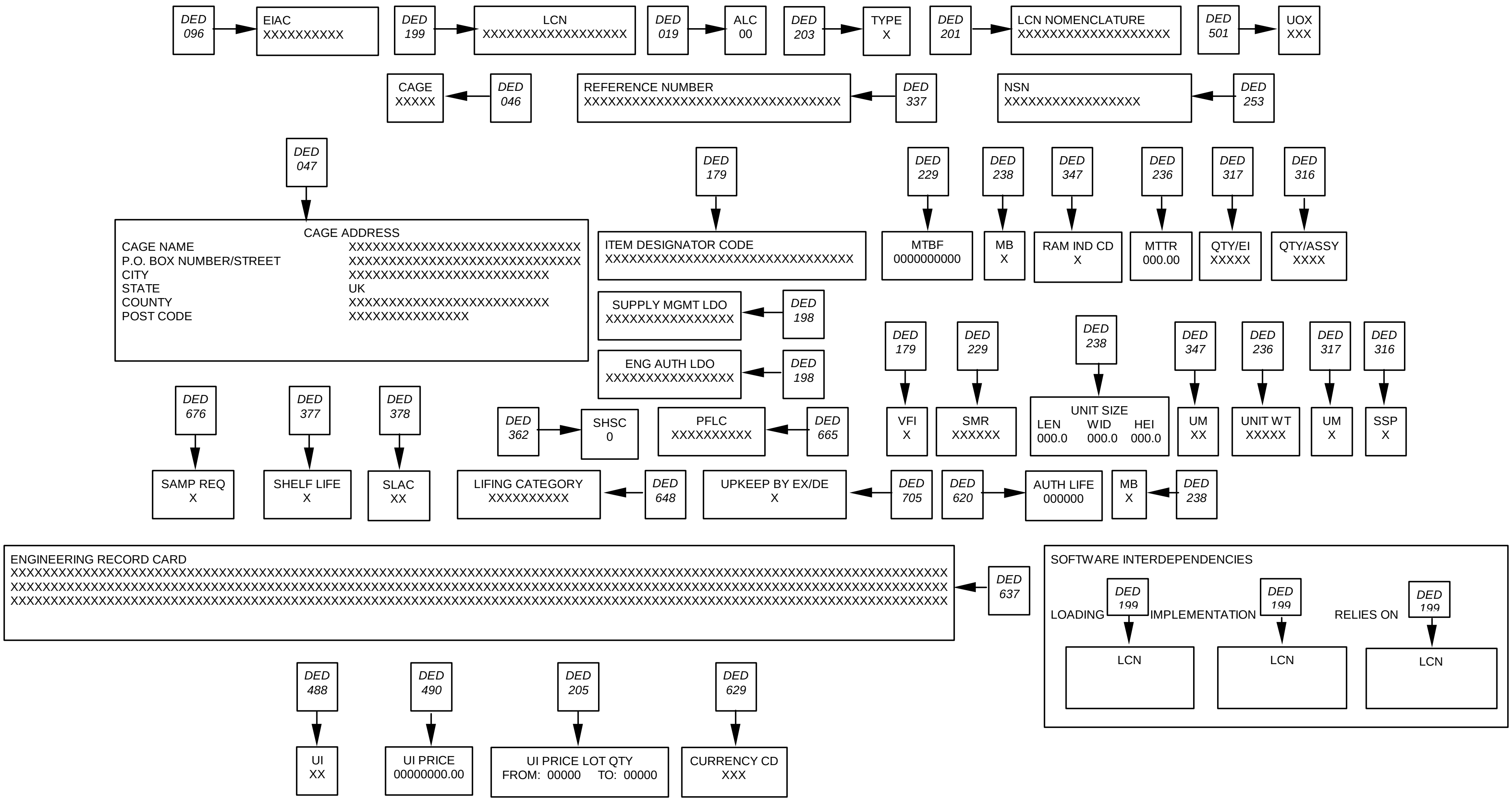
Modified LSA Reports

This Annex shows the proposed changes to the LSA – 602 Part 1 and my proposal for a new LSA – 672 report. The change to LSA – 602 Part 1 shows on page C-2, for any CI, the software interdependencies of that item. These interdependencies may be loading, implementation or relies-on. For LSA – 672, the report is in two parts, Part One, on page C-3, deals with the information required for all software, whether COTS or bespoke; Part 2, on page C-4, provides more Software Engineering detail, primarily for bespoke software. The narrative justification for a piece of training equipment is missing from the current LSA – 011, Special Training Equipment / Device Summary, this has been added on page C-5.

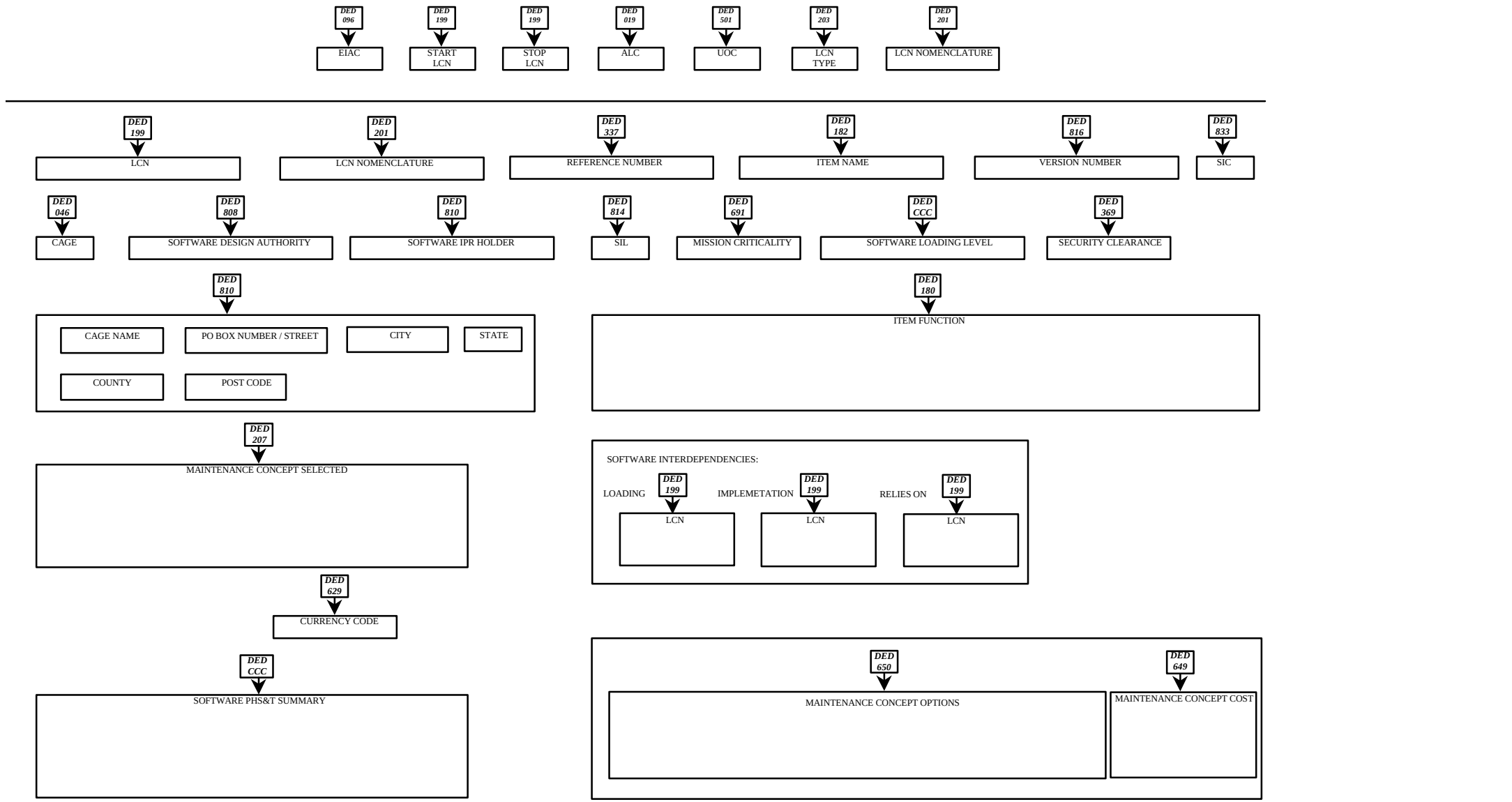
LOGISTIC SUPPORT ANALYSIS RECORD

CANDIDATE ITEM MAINTENANCE AND UPKEEP PLAN

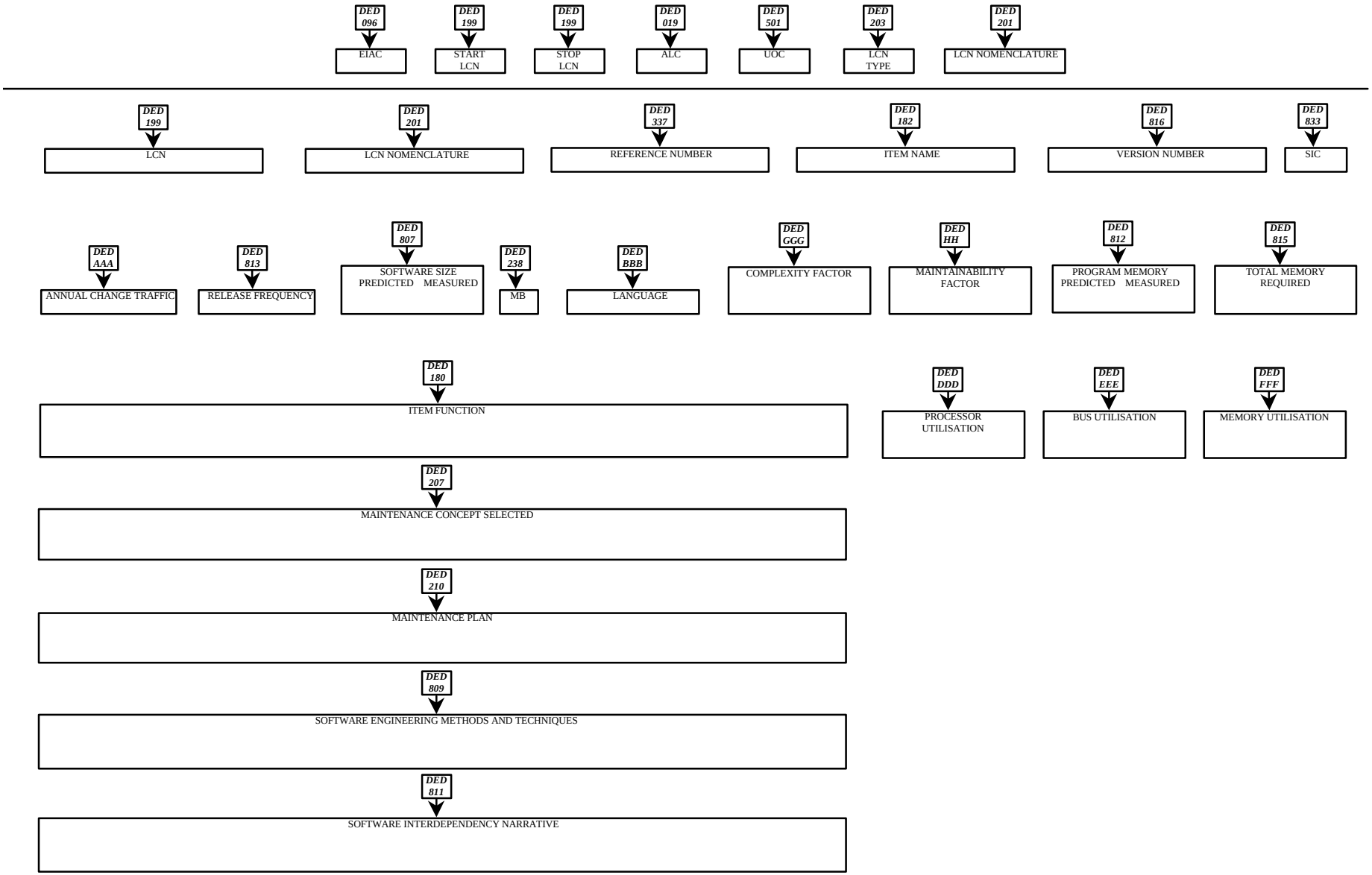
PART 1 - IDENTIFICATION PARTICULARS



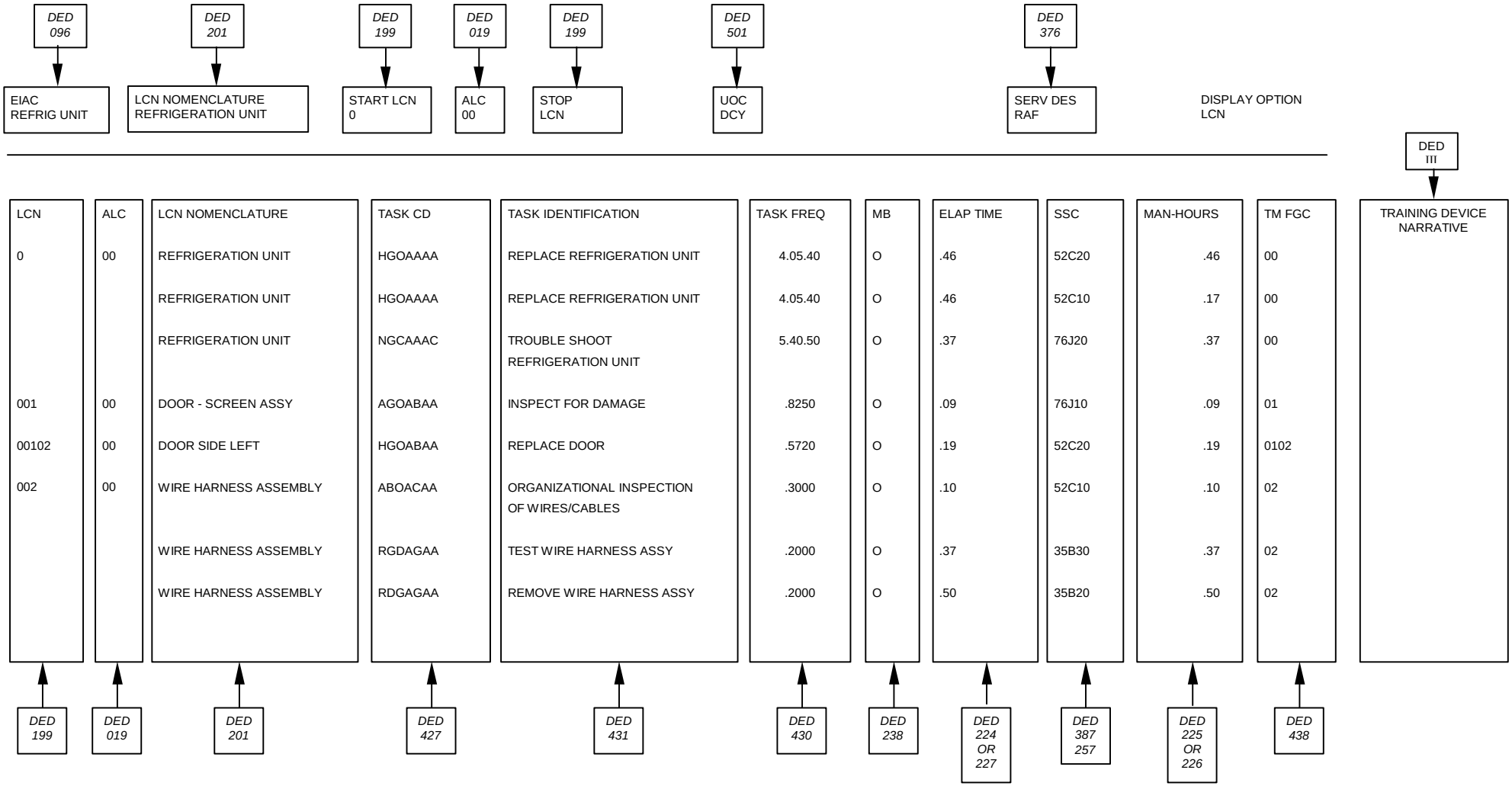
PROPOSED LSA-672 PART 1



PROPOSED LSA-672 PART 2



LOGISTIC SUPPORT ANALYSIS RECORD
SPECIAL TRAINING EQUIPMENT/DEVICE SUMMARY



ANNEX D

REFERENCES

Standards and Military Publications

AECMA S1000D, *International Specification for Technical Data Publications, Utilising a Common Source Data Base,* 1999.

AECMA S2000M, *International Specification for Materiel Management Integrated Data Processing for Military Equipment,* 1998.

AFOTEC, *Pamphlet 99-102, Volume 3, Software Maintainability Evaluation Guide,* HQ Air Force Operational Test and Evaluation Center, September 1996.

British Standard 4783, *Storage, Transportation and Maintenance of Media Used in Data Processing and Information Storage,* 1988.

British Standard 5760 Part 8, *Guide to the Assessment of Reliability of Systems Containing Software,* 1998.

CDPI/Supp/010, *Chief of Defence Procurement Instructions,* 1998.

CDPI/Tech/ 490, *Chief of Defence Procurement Instructions,* 1997.

DEFCON 91, *Intellectual Property Rights in Software,* 1992.

Def Stan 00-22, Issue 2, *The Identification and Marking of Programmable Items,* 1991.

Def Stan 00-42, Issue 1, *Reliability and Maintainability Assurance Guides, Part 2: Software,* 1997.

Def Stan 00-56, Issue 2, *Safety Management Requirements For Defence Systems,* 1996.

Def Stan 00-60, *Integrated Logistic Support,* 1999.

ISO 9001: 2000, *Quality Management Systems – Requirements,* International Standards Organisation, 2000.

ISO 12207: 1995, *Information Technology – Software Life Cycle Processes,* International Standards Organisation, 1995.

ISO 14764: 1999, Software Maintenance, International Standards Organisation, 1999.

JA1004, *Software Supportability Program Standard,* Society of Automotive Engineers, 1998.

JA 1005, *Software Supportability Implementation Guide,* Society of Automotive Engineers, due publication 2000.

JA 1006, *Software Support Concept,* Society of Automotive Engineers, 1999.

JSP 440, *The Defence Manual Of Security,* 1996.

MIL-STD 1388, *1A, Logistic Support Analysis; 2B, DoD Requirements for a Logistic Support Analysis Record,* 1973.

Published Works

Carnegie Mellon University / Software Engineering Institute, *The Capability Maturity Model: Guidelines for Improving the Software Process*, Addison-Wesley, 1994.

Carnes P, *Software Reliability in Weapons Systems*, Proceedings, Eighth International Symposium on Software Reliability Engineering, November1997.

Derriennic H and Le Gall G, *Use of Failure – Intensity Models in the Software Validation Phase for Telecommunications*, IEEE Transactions on Reliability, Vol 44, 1995.

Fenton N, *Software Metrics: A Rigorous Approach*, Chapman & Hall Ltd., 1991.

Fenton N and Pfleeger SL, *Software Metrics: A Rigorous and Practical Approach*, International Thomson Computer Press, 1996.

Giles AE and Dennis B, *Metrics Tools: Software Cost Estimation*, CrossTalk, February 1995.

Halstead M, *Elements of Software Science*, North Holland, 1977.

Jones WD, *Reliability Models for Very Large Software Systems in Industry*, Proceedings, Second International Symposium on Software Reliability Engineering, October 1991.

Keller T and Schneidewind N, *Successful Application of Software Reliability Engineering for the NASA Space Shuttle*, Proceedings, Eighth International Symposium on Software Reliability Engineering, November 1997].

Leveson NG, *Safeware: System Safety and Computers*, Addison-Wesley, 1995.

Lyu M, *Software Reliability: To Use or Not To Use? A Panel Discussion Chaired by Michael Lyu*, CrossTalk, February 1995.

Malaiya YK, Karunanithi N and Verma P, *Predictability of Software-Reliability Models*, IEEE Transactions on Reliability, Vol 41, 1992.

McCabe TJ, *A Software Complexity Measure*, IEEE Transactions on Software Engineering, Vol 2 December 1976.

Musa JD, *Software Reliability Engineering*, McGraw-Hill 1999.

Pressman RS, *Software Engineering, A Practitioner's Approach*, McGraw-Hill, 1997.

Rook P (Ed), *Software Reliability Handbook***,** Elsevier, 1990.

Sommerville I, *Software Engineering*, Addison-Wesley, 1995.

Stutzke RD, *Software Estimating Technology: A Survey*, CrossTalk, May 1996.

Welker K and Oman P, *Software Maintainability Metrics Models in Practice*, CrossTalk, November – December 1995.

West R , *Improving the Maintainability of Software*, HMSO, 1993.