

University of Oxford  
Software Engineering Programme

MSc in Software Engineering  
Dissertation Submission

# **Maturing the Software Logistics Support Analysis Process**

Jamie Francis Brooks  
Kellogg College  
March 2005

## **ABSTRACT**

Within the domain of software engineering, capability sustainment equates to the ability to continually implement and field modifications within required timescales on an affordable basis. However the unique nature of software means that support analysis cannot be performed using traditional hardware oriented Logistic Support Analysis (LSA) techniques. Software LSA guidance provided within Defence Standard 00-60 is inappropriate and incomplete, which has led to software related supportability decisions becoming ill-informed, unjustified and untraceable. As a result of this, the MOD has increased its exposure to contractor exploitation during system support activities.

This project, based on a case study of LSA applied to the Royal Air Force's Future Offensive Air System project, identifies significant inadequacies in the guidance covering the application of LSA to software and suggests more appropriate approaches and techniques for the purpose of maturing the software LSA process, improving capability sustainment, and reducing the through life cost of support.

## **ACKNOWLEDGEMENTS**

Throughout the completion of this dissertation assistance has been gratefully received from family, friends and colleagues. I would especially like to thank:

My project supervisors, Dr Jeremy Gibbons and Dr Andrew Martin, for their guidance.

Mark Bailey, Ian Grose and Lee Cooper, for their challenges, contribution and constructive criticism.

David Gill, for his continued motivation and encouragement.

My devoted wife, for her unconditional sacrifice, tolerance and support.

The author confirms that this dissertation does not contain material previously submitted for another degree or academic award and is the author's own work except where otherwise stated.

**Opinions expressed within this dissertation are solely attributable to the author and do not necessarily represent MOD doctrine or policy.**

## **CONTENTS**

SECTION 1.....	1
INTRODUCTION.....	1
1.1 Motivation.....	1
1.2 Contribution.....	1
1.3 Realisation.....	2
APPLICATION AREA.....	4
1.4 The Nature of Maintenance.....	4
1.5 Software Support Infrastructure.....	5
1.6 Types and Benefit of Software Modification.....	6
SUMMARY.....	10
SECTION 2.....	11
LSA PLANNING AND IMPLEMENTATION.....	11
2.1 The Application of Defence Standards and Publications.....	11
2.2 LSA Scope and Responsibility.....	12
2.3 Approach and Implementation.....	17
SUMMARY.....	21
SECTION 3.....	23
SUPPORTABILITY REQUIREMENTS.....	23
3.1 The Totality of Support.....	23
3.2 Supportability Requirements Engineering.....	29
3.3 Performance Support Requirements.....	30
3.4 Functional Support Requirements.....	31
SUMMARY.....	32
SECTION 4.....	33
SUPPORTABILITY ACHIEVEMENT AND SUSTAINMENT.....	33
4.1 Transposition of Analysis Techniques.....	33
4.2 Risk Based Approach to Software Support.....	35
4.3 Non-Economic LOSA.....	37
4.4 Supportability and Technology.....	39
4.5 Supportability Preservation.....	41
SUMMARY.....	42
SECTION 5.....	45
SUPPORTABILITY VALIDATION AND VERIFICATION.....	45
5.1 Supportability Achievement Through Progressive Assurance.....	45
5.2 Test And Evaluation Strategy.....	46
5.3 Supportability Assessment and Management Techniques.....	47
5.4 Supportability Assessment Objectives And Criteria.....	49
5.5 Supportability Intent Verification.....	51
5.6 Product Assessment.....	52
5.7 Process Assessment.....	53
SUMMARY.....	54
SECTION 6.....	56
DISCUSSION.....	56
6.1 Reflection.....	56
6.2 Critique.....	57
6.3 Future Work.....	59

## **TABLES**

Table 1 – Def Stan 00-60 Defined LSA Task Responsibility .....	15
Table 2 – MOD Related Factors .....	16
Table 3 – Service Supplier Related Factors .....	17
Table 4 – Supportability Performance Requirements .....	31
Table 5 – Support Risk Assignment.....	36
Table 6 – Support Risk Definition .....	36
Table 7 – OFP Qualitative Support Option Selection.....	39
Table 8 – Potential Benefits of Software Technology .....	41
Table 9 – Method Strengths and Weaknesses .....	48
Table 10 – Metric Attributes.....	51

## **FIGURES**

Figure 1 – Dissertation Overview.....	3
Figure 2 – Hardware Maintenance Lifecycle.....	5
Figure 3 – Relative Distribution of Effort Across the Lifecycle .....	6
Figure 4 – Causes of Software Change.....	8
Figure 5 – Maintenance Cost and Benefit.....	9
Figure 6 – The Software Maintenance Life Cycle .....	23
Figure 7 – Def Stan 00-60 Generic Model of Software Support.....	25
Figure 8 – Initial Support System Model.....	25
Figure 9 – Support System Model .....	27

## **APPENDICES**

- A. Bibliography.
- B. Analysis of RAF Change Drivers.
- C. Support Activity Inventory.
- D. Qualitative Support Option Assessment.
- E. Supportability Characteristics.
- F. Technology Demonstrator Programme Summary.
- G. Supportability Related Technologies.
- H. Technology Supportability Benefit.
- I. Capability Maturity Model Common Features.
- J. Transposition of Testing Into the Supportability Domain.

# **SECTION 1**

## **INTRODUCTION**

### **1.1 MOTIVATION**

Within the Avionic Support Group (AvSG), Assistant Directorate Systems, RAF Wyton, work<sup>1</sup> has been carried out to update guidance provided on the application of Logistics Support Analysis (LSA) to software. This work, primarily an evaluation of Defence Standard (Def Stan) 00-60 within the context of the Acquisition Management System (AMS), established new guidance on Concept Phase software LSA and a framework to assist with the planning, implementation and management of LSA tasks.

Accepted within this body of work is the need to establish guidance that will: reach beyond the Concept Phase, assist with the identification of software supportability User Requirements and aid the creation of Software Support Policy Statements. In addition to these stated needs, I believe that the guidance itself needs to be validated as the work is mainly based on a desktop review of LSA related standards and literature.

For the past two years I have been involved in the application of LSA to software for the Future Offensive Air System (FOAS). It is intended that FOAS will replace the defence ground attack capability currently provided by Tornado GR4. However, unlike the GR4, FOAS is not perceived as an aircraft replacement, instead it is due to be a 'System of Systems'. The 'System of Systems' approach is capability focused and as such, FOAS could comprise of a manned aircraft, Un-inhabited Combat Air Vehicle (UCAV), Conventionally Armed Long-Range Cruise Missile (CALCM), and a Non-Penetrating Aircraft (NPA) - which will act as a CALCM and UCAV delivery platform. This work, innovative in its field, has produced a number of reports that have attempted to document software LSA outputs. Unfortunately, as with any immature discipline, the work has fault and many opportunities exist to improve the quality of both the LSA process and its products.

### **1.2 CONTRIBUTION**

This dissertation presents a case study with the aim of maturing the LSA process by:

- Stating the current failings of Def Stan 00-60, both general and software related.
- Defining the nature of software supportability analysis and transposition of LSA into the software domain.
- Assessing the practicalities of LSA scope and depth when analysis is carried out by an organisation external to a Project Team.
- Evaluating existing guidance on LSA for software and suggesting areas for improvement.
- Suggesting new analysis methods that reach beyond the scope of existing LSA guidance.

The dissertation successfully highlights the fundamental issues that govern the need for software support in modern military systems and drive the software Logistic Support Analysis (LSA) process. It identifies some of the common causes of support analysis failure, which has been achieved through an examination of MOD guidance material on the application of LSA to software and a review of the support analysis carried out for the FOAS project.

The totality of software support from an operational perspective has been identified, and awareness of support requirements that are typically implied rather than stated has been achieved. A generic Support System Model has been created and is considered to be a key element of this dissertation, as this model has been utilised throughout further LSA to assist with the production of consistent, complete and correct outputs.

LSA guidance and techniques have been transposed into the software domain to bridge some of the voids and misunderstandings that exist in this area. An informed, justifiable and traceable method, by which critical software items can be identified and their support requirements understood, has been developed.

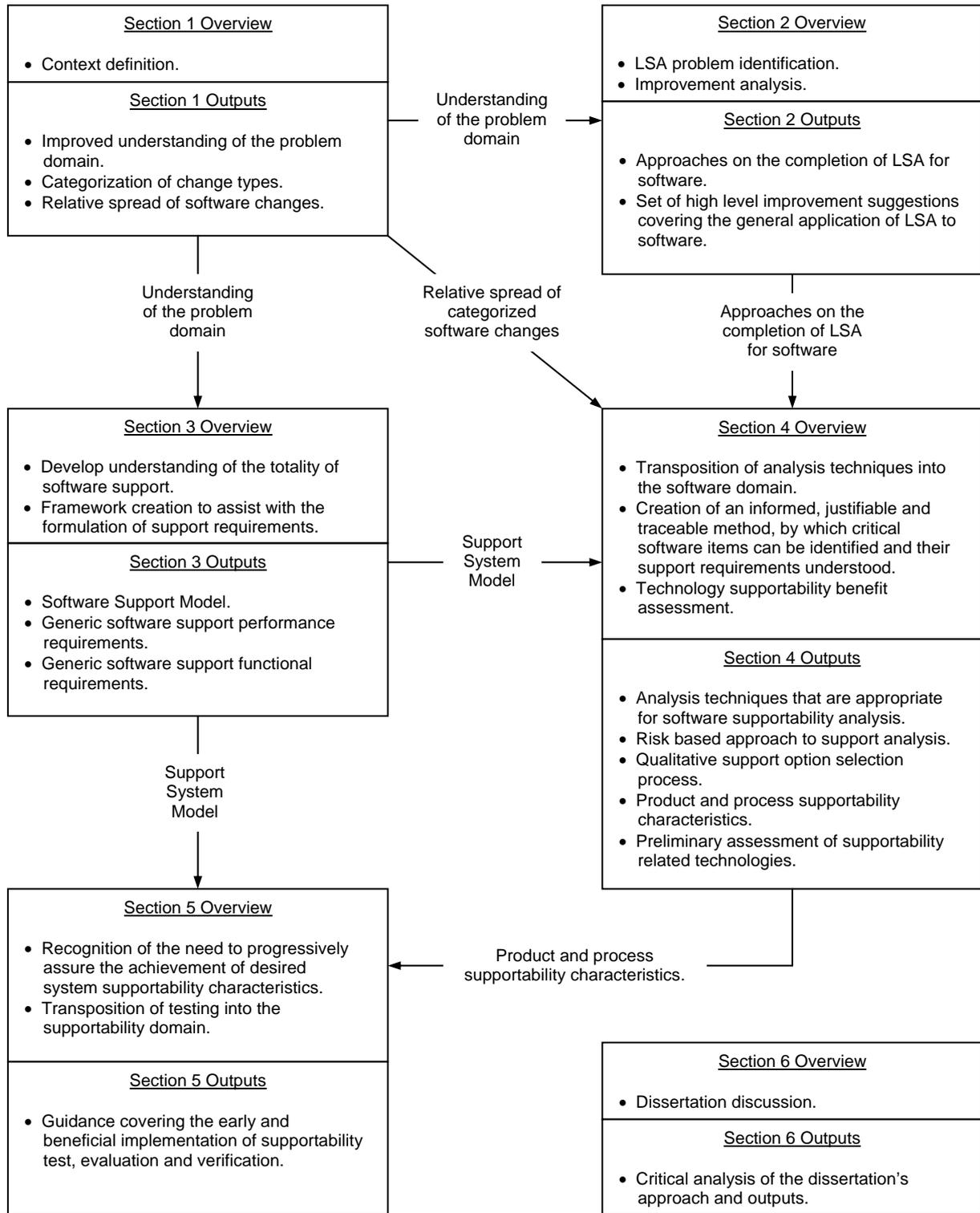
The need to progressively assure the achievement of desired system supportability characteristics has been recognised. Guidance has been formulated on the early and beneficial application of LSA Task 501, which aims to ensure that software enabled capability can be sustained throughout the life of a project.

### 1.3 REALISATION

This dissertation comprises of six sections, covering a broad range of interdependent topics, the relationship between these sections is illustrated in Figure 1.

Much of the dissertation has been directly enabled by principles taught during the course, examples of which include:

- Object Orientation (OOR) – although on the fringe of this subject, support system model refinement was improved by an understanding of the role of modelling and need for consistency.
- Requirements Engineering (REN) – with the definition of supportability performance and functional requirements.
- Managing Risk and Quality in Software (MRQ) – the transposition of Level of Repair Analysis needs to address support criticality in a manner that is informed, justified and traceable.
- Software Testing (STE) – to gain confidence that supportability is being addressed, progressive assurance of software supportability needs to be established. A transposition of software testing into the supportability domain will assist with the achievement of this activity.
- Safety Critical Systems (SCS) – as with safety, supportability is an emergent system property, as such safety management techniques can be utilised within the supportability-engineering domain.



Key:

Arrows represent usage of an output from another section

Figure 1 – Dissertation Overview

## **APPLICATION AREA**

### **1.4 THE NATURE OF MAINTENANCE**

Maintenance of hardware is initiated by random, systematic or predicted failure. These initiators can be defined as follows:

- Random failure is the expected but singularly unpredictable wear out of a physical item.
- Systematic failure is the failure of an item that is attributable to a definable and repeatable set of conditions (typically caused by design errors or system operation outside of specified limits).
- Predicted failure is an estimation technique utilising the fact that failures of a group of items can be estimated to fall within a predicted range of values.

In the field of hardware engineering random failures are dominant and drive maintenance activities. Therefore, historically, system maintenance activities and policies have been based on the management of random hardware failures.

Over the last 20 years a fundamental change has occurred in systems development. System functions are increasingly being enabled by software operating on computer based systems. These software intensive systems have given developers the opportunity to provide highly versatile systems that are easily changed to match new evolving technologies and operational requirements.

The concept that software can be changed with ease to match new technology and operational requirements is now under challenge. The challenge has come from the increased complexity that software intensive systems are exhibiting and limited investment in system supportability engineering. Efforts to manage the efficiency and effectiveness of software support have had mixed success for these same reasons.

Unlike hardware, software only fails systematically; that is to say that it does not wear out like hardware. Failures of a system, attributable to software, are caused by faults that have not been removed prior to delivery, operation of the system outside of its specified limits, or the introduction of faults through subsequent maintenance. The life of software within an operational system is different to that of hardware. Unlike hardware, that at some point in time can have a completely serviceable state, all but the most elementary software contains faults to some extent throughout its life.

For hardware, a difference can be defined between the maintenance and modification of an item. Hardware maintenance will return an item to its original supplied state (see Figure 2) whilst modification will alter the specification or function of an item. Software maintenance does not return a software item back to its original supplied state. Software maintenance requires changes or modifications to the product, even to correct faults. Therefore, in essence, there is little difference between software maintenance and modification processes and activities. As such, software maintenance activities can be referred to as modifications or changes.

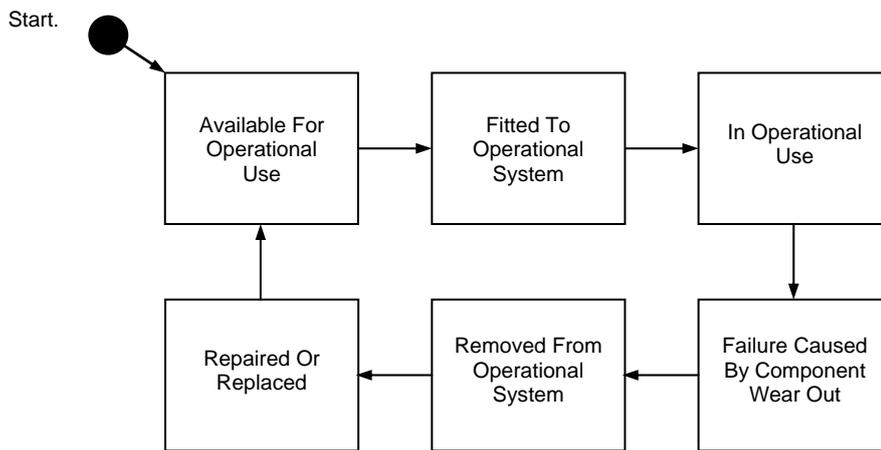


Figure 2 – Hardware Maintenance Lifecycle

## 1.5 SOFTWARE SUPPORT INFRASTRUCTURE

Whilst the engineering resources needed for software development and software modification are similar, differences do exist in the scale of infrastructure requirements between acquisition and support. Acquisition takes place in an industrial context, while support is conducted in an operational context. Industry based software development has an infrastructure tailored to larger scaled projects with longer schedules, work that has greater resource needs than that of smaller projects. In-Service software modification has an infrastructure tailored to smaller scaled projects with shorter schedules, work that often requires greater responsiveness than that of larger projects. At present, this greater responsiveness is typically enabled in the Royal Air Force (RAF) by the utilisation of dedicated Software Support Teams (SST) working closely with the Users.

According to Takang and Grubb<sup>2</sup>, the distribution of effort across the lifecycle differs between software development and software modification, as shown in Figure 3. It should be noted that the traces in Figure 3 are not provided for the purpose of a direct effort comparison between development and modification (i.e. the figure does not indicate that more effort is spent on requirements during modification than development). Instead the figure enables a comparison of relative effort distribution for either development or modification, (i.e. during software modification a larger proportion of effort is consumed during requirements than implementation, where as this is not the case during development).

Software support activities have a different focus than those in development, they include<sup>3</sup>:

- Interacting with users to determine what changes or corrections are needed.
- Reading existing code to understand how it works.
- Changing existing code to make it perform differently.
- Testing the code to make sure it performs both old and new functions correctly – for safety related software modification this also includes re-qualification and certification.

- Delivering the new version with sufficiently revised documentation to support the user and product.

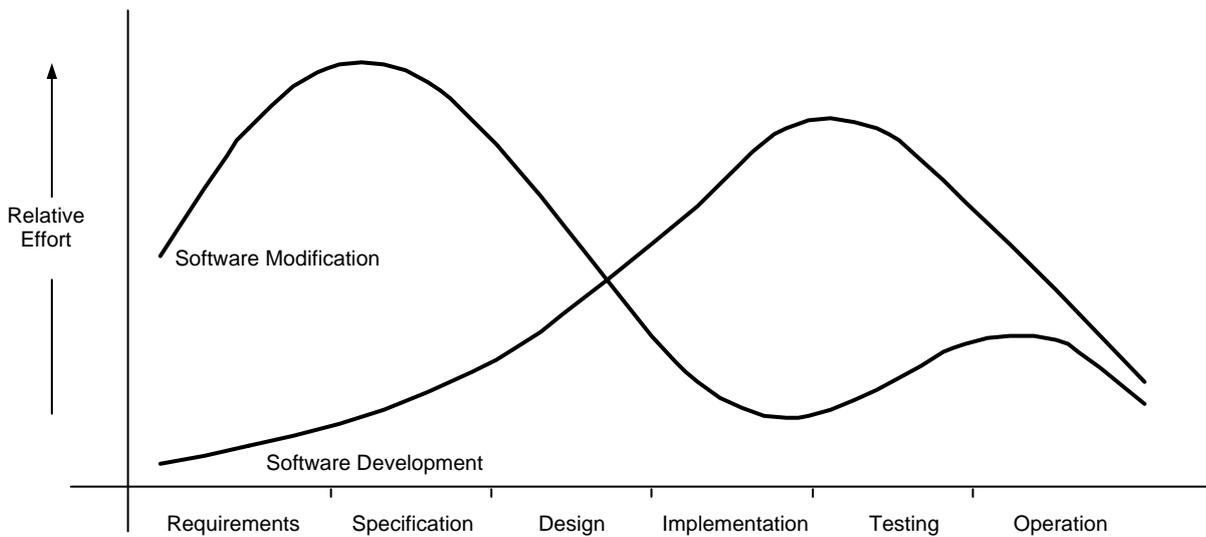


Figure 3 – Relative Distribution of Effort Across the Lifecycle

With reference to Lehman<sup>4</sup>, military systems are embedded in a real-world situation that is constantly changing. As these systems are designed to complete functions within the real world they are therefore subject to a constant flow of possible change triggers. From this observation it can be concluded that the idea that a program can be developed and finished, without further maintenance, is false. Change is inevitable and in fact desirable to protect the software investment, prevent obsolescence and reduce the risk of capability gap. Now that software dominates most new systems as the function provider, the maintenance provision for computer-based systems must recognise the initiators of software modification.

## 1.6 TYPES AND BENEFIT OF SOFTWARE MODIFICATION

According to Musa and Everett<sup>5</sup> the discipline of software reliability engineering exists to maximise customer and user satisfaction by providing and sustaining system availability.

*“Software engineering is about to reach a new stage – the reliability stage – that stresses customers’ operational needs.”*

Musa and Everett suggest that the two activities that specifically enable this customer focus are the establishment of a failure intensity and operational profile, which are summarised as follows:

- Failure intensity is an indication of system reliability determined by measuring the number of failures experienced for a given period of program execution time.
- The operational profile documents the set of functions that a system can perform along with probabilities of their execution. The profile is established

so that derived system reliability will relate directly and accurately to the operational environment.

Like reliability engineering, supportability engineering also aims to maximise customer and user satisfaction by providing and sustaining system availability. As such, benefit should be gained by transposing this work into the supportability domain. To do this we need to identify how this work remains relevant when considering system maintenance rather than failure. It is evident from the summaries above that the main strength of Musa and Everett's work is the way that it focuses on operational use so, with this in mind, we need to ensure that our support analysis considers operational maintenance drivers. As such, rather than thinking about potential sources of failure and how often they might be experienced, we need to understand potential sources of change and how often might they be encountered.

Within the RAF<sup>6</sup>, differing classifications of software modification have been defined, as follows:

#### Corrective.

*"A corrective change modifies a software item to remove a software fault."*

Corrective changes remove faults that are present in delivered software products, causing system operation to deviate from requirements (systematic failure).

#### Adaptive.

*"An adaptive change modifies a software item to enable it to continue to meet its specification in a changed environment."*

Adaptive changes typically preserve system functionality when interfacing system components are changed and no longer operate as originally specified.

#### Perfective.

*"A perfective change modifies a software item to enable it to meet its existing specification in an improved fashion."*

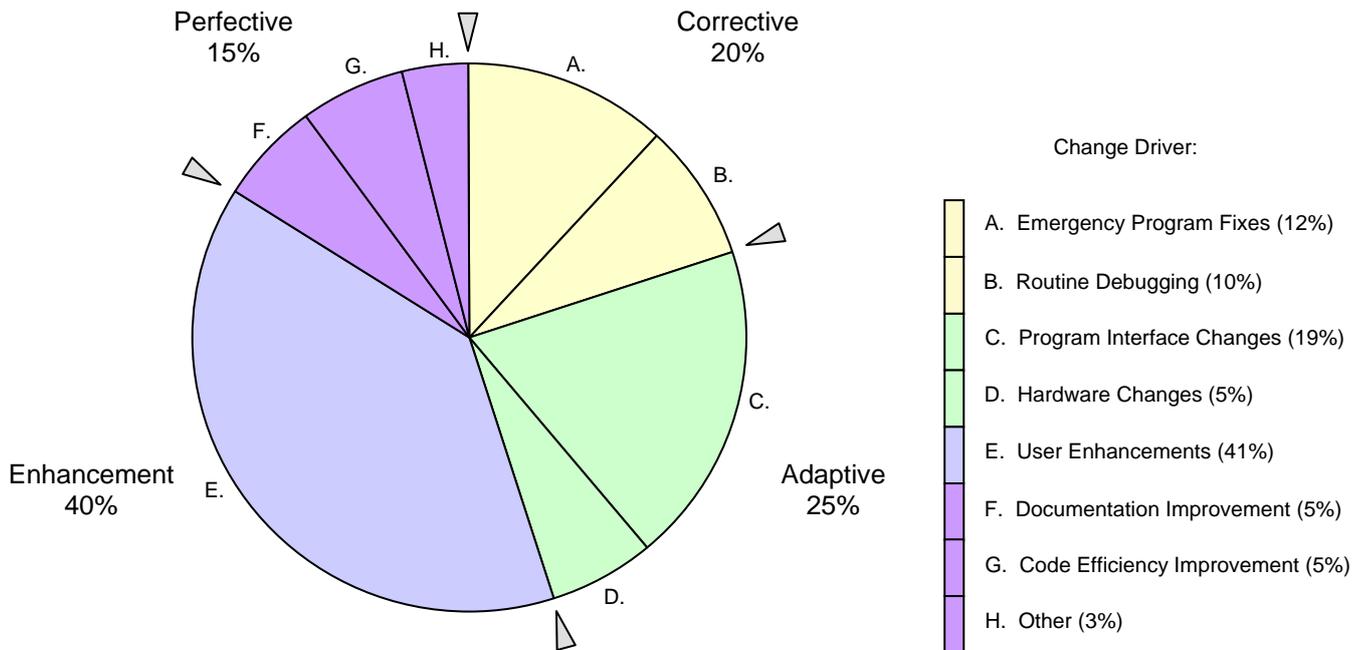
Perfective changes can remove unwanted functionality or re-implement existing requirements to improve inefficient or ineffective items.

#### Enhancements.

*"An enhancement change modifies a software item to add additional functionality to the system."*

An enhancement change is driven by a change to the system requirements. If function change is defined as the addition and removal of functions then enhancements also include modifications that change functionality.

Whist these definitions are useful, they are of no analysis value unless we can gain an understanding of how these sources of change influence our systems. Investigation by Lientz and Swanson<sup>7</sup> into the spread of software change types has existed for some time, as illustrated in Figure 4. As a rule of thumb, functionality improvements and enhancements for users drive 50% of software change (perfective and enhancement changes), while maintaining system functionality accounts for the other 50% (corrective and adaptive changes).



**Figure 4 – Causes of Software Change**

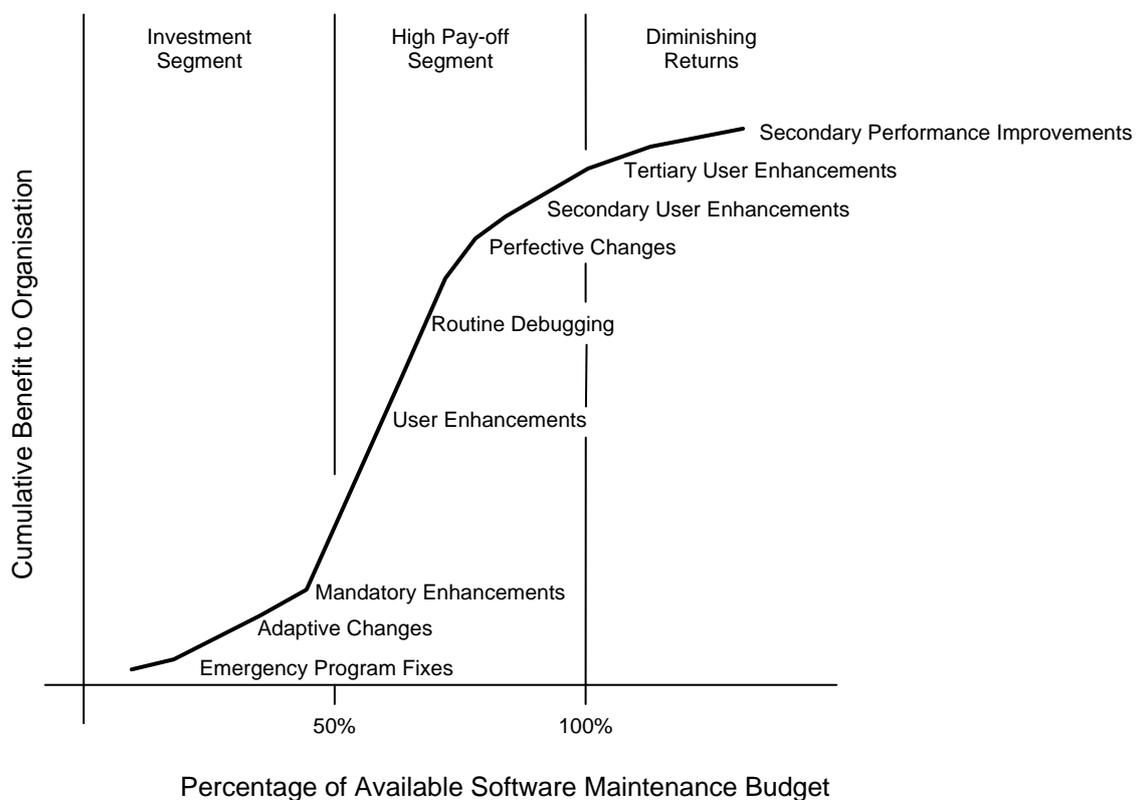
This work is based on analysis of changes experienced by commercial systems in the late 1970's; as such we have to question its suitability for today's use in a military domain. My support for its use is based on the following reasons:

- A second set of analyses produced by Nosek and Palvia<sup>8</sup>, which considered a slightly different categorization of change initiators, supported Lientz and Swanson's initial findings.
- Even if the spread of change types is slightly incorrect, it is important to realise that software maintenance is not just corrective.

To bring this work into a military context and provide extra confidence in its applicability, I have carried out a preliminary analysis of RAF in-service software support teams. In this analysis I reviewed 249 software modifications using Lientz and Swanson's original categorization of change types. This analysis, included at Appendix B, also supports Lientz and Swanson's findings. It has not however been published, the reason for this being that the underlying data set is not considered robust enough to withstand rigorous scrutiny. This situation is not desirable as it results in a poor understanding of why we change our systems, this in turn hampers efforts to predict how our systems might change in the future and what level of support we might require. During presentations of my work

at supportability forums, I have experienced numerous challenges regarding the use of Lientz and Swanson's work, all of which have been countered when the aforementioned rationale is provided. However, this situation could be avoided if data from an in-depth analysis of military change drivers was readily available.

Considering software change from the viewpoint of its ability to enhance or maintain system functionality gives only one perspective on this issue, a second perspective can be taken by considering the degree to which an organisation is free to decide which changes it implements. With this in mind, Boehm<sup>9</sup> has shown that just under 50% of the available software maintenance budget is consumed carrying out mandatory corrective and adaptive changes; change that is forced on the system. The remaining budget is then allocated to changes that offer the organisation most benefit. It is therefore desirable to minimise the cost of mandatory changes, such that a greater proportion of budget can be invested in high benefit change. This is illustrated in Figure 5.



**Figure 5 – Maintenance Cost and Benefit**

With limited resources and a desire to field products\* in a timely manner, a need has been established for effective and efficient software maintenance. These characteristics are in effect maintenance quality goals; as such they should be used to drive all software development and support processes†. The Society of Automotive Engineers (SAE)<sup>10</sup> defines the requirements of software supportability as follows:

\* The term 'product' is used to describe any output created during software development or maintenance.

† The term 'process' is used to describe any set of activities and activity enablers that transform inputs into outputs (or products).

*“General – There are two aspects to meeting a customer’s software supportability requirements. The first is ensuring the delivery of a product with the appropriate design characteristics to facilitate the expected demand for through-life change and enhancement. The second aspect is the provision of a support capability that satisfies the customer’s quality of service needs at an acceptable cost. These are interrelated goals that should be addressed through a coordinated approach to software supportability planning.”*

The achievement of our maintenance goals is therefore vested in the provision and preservation of appropriate software product and process characteristics. Within the MOD the vehicle used to procure supportability characteristics is Def Stan 00-60 (Application of Integrated Logistic Support)<sup>11</sup>, particularly Part 3, which gives guidance on the application of the standard to software. No single Def Stan governs or provides guidance on the preservation of supportability characteristics, instead this function is provided by the various bodies that manage software change.

## **SUMMARY**

This section has endeavoured to highlight the fundamental issues that govern the need for software support in modern military systems and drive the software Logistic Support Analysis (LSA) process.

The unique nature of software means that it cannot be analysed using traditional hardware LSA techniques, nor treated as an adjunct to any system. Software can now be considered the main function provider for many systems and as such has also become the major capability provider.

The belief that system requirements can be defined and frozen prior to a system’s development, or indeed during its operational life, is ill-founded. Threats and counter tactics are constantly evolving; placing ever-increasing demands on our systems. As such, these systems need to be capable of supporting new requirements, a task primarily vested in an ability to modify software. This ability will therefore directly affect an organisation’s potential to sustain capability and protect investment.

Software modification can be grouped into two main areas, those that maintain existing functionality and others that enhance functionality. Enhancements offer the highest benefit to an organisation as they improve capability. The ability to implement enhancements is constrained by the unavoidable limitation created by resource availability and the need to maintain existing functionality. Given the limited availability of resources and a desire to maintain capability, a need has been identified to make the software support process as effective and efficient as possible.

Having introduced the need for software LSA, the next section will discuss the appropriateness and failings of guidance material provided by the MOD and make improvement suggestion where possible.

## **SECTION 2**

### **LSA PLANNING AND IMPLEMENTATION**

#### **2.1 THE APPLICATION OF DEFENCE STANDARDS AND PUBLICATIONS**

Before LSA practitioners can determine support solutions they need to understand the support need. Prior to the advancement of AP100D-10 from version 2 (Management Policy for Operational systems) to version 3 (Support for Mission Software in RAF Systems), software support analysis was carried out to assist with the production of a Software Support Policy Paper (SSPP). This paper captured the support needs based on the characteristics of a supplied software product. Based on this information, a filtering process would be carried out to identify the most appropriate support solution. Further analysis would then be undertaken to identify the infrastructure and personnel requirements of the chosen support option, this information would then be captured in the Software Support Policy Statement (SSPS).

When AP100D-10<sup>12</sup>, version 3 arrived, LSA and the Logistics Support Analysis Record (LSAR) replaced SSPPs. The LSAR brought many advantages to the analysis process by providing a framework and a means of standardising the arduous task of capturing support resource requirements. Unfortunately the LSAR, which was originally established to capture data relating to hardware, struggled to assist with the recording of software related analysis outputs. In effect LSA and the LSAR had arrived but the means of recording support analysis was inadequate, traceability was lost and support decisions became ill-informed, and unjustified.

Historically, support analysis has been carried out by the MOD then, driven by the need to reduce MOD costs and a change in doctrine that focused on front line operations; a paradigm shift occurred that resulted in the majority of LSA being contracted to industry. I now suggest that the MOD should undertake an amount of software LSA itself, or risk erosion the intelligent customer capability and loss of the ability to robustly contract for, or effectively manage, LSA. Should our intelligent customer capability be lost, the MOD will place itself in a position of vulnerability through exploitation, in essence we will no longer understand the software LSA process and the Integrated Logistics Support discipline could fail. With the necessary understanding, contracting for software LSA is a legitimate means of reducing costs and personnel overheads. However, we need to understand what it is we are contracting for and retain the ability to oversee and manage LSA activities.

In accordance with the guidance given in Def Stan 00-60<sup>13</sup> and AP 100C-70 (Integrated Logistic Support In The Royal Air Force)<sup>14</sup>, the assessment of alternative support systems is carried out with reference to data populated in the LSAR. However, by waiting for system design to become suitably mature to facilitate Reliability and Maintainability (R&M) engineering, Failure Modes Effect and Criticality Analysis (FMECA), Level of Repair Analysis (LORA) and subsequent population of the LSAR, the best opportunity to influence system design for support has been lost.

The ability to influence product design and support planning is only possible if analysis is carried out in a timely manner. Unfortunately, the application of Software Support Analysis (SSA), as described in Def Stan 00-60<sup>15</sup>, is reactive, i.e. once a solution has been identified and its design mature. This means that the rigid application of Def Stan 00-60

early in the CADMID<sup>‡</sup> acquisition lifecycle is inappropriate. To accommodate this situation and carry out proactive SSA, the spirit of Def Stan 00-60 needs to be considered at a level that enables supportability design factors to be established at a time when they can be used to influence product and process design.

## 2.2 LSA SCOPE AND RESPONSIBILITY

Work carried out within AvSG<sup>1</sup> suggests that software LSA needs be carried out as early as possible in the procurement lifecycle, specifically during the Concept and Assessment phases; one question that I have become increasingly more focused on is who should carry out this LSA. I believe the answer to this question is dependent upon the LSA tasks to be performed, the depth of analysis required and availability of appropriately skilled personnel, factors that are in turn dependent upon the project's phase in the procurement lifecycle.

LSA comprises of 5 task areas, which are summarised as follows:

- 100 Series LSA Tasks. These planning tasks are performed to enable formal programme planning and review. It should be noted that these tasks are biased towards the management of contracted LSA and are not wholly appropriate for application during the Concept and Assessment phases, when LSA should be carried out by the MoD.
  - Development of an Early LSA Strategy – Task 101. The aim of this task is to provide visibility of the programme strategy such that an efficient and unified approach to support analysis can be achieved.
  - LSA Plan – Task 102. This task covers the creation of the LSA Plan, its acceptance and update.
  - Programme and Design Reviews – Task 103. This task ensures that a review process is established, implemented and maintained. The aim of these reviews is to ensure that the LSA process is effective.
- 200 Series LSA Tasks. These tasks are carried out to gain an understanding of the Users supportability requirements, constraints and goals.
  - Use Study – Task 201. The aim of this task is to identify pertinent supportability factors by considering the nature of software and likely need for support.
  - Mission Hardware, Software and Support System Standardisation – Task 202. The aim of this task is to identify standardisation related constraints and opportunities; this includes an assessment of data standardisation needs.

---

<sup>‡</sup> The lifecycle utilised by the MOD for the acquisition of equipment capability; comprising of the following phases: Concept, Assessment, Demonstration, Manufacture, In-Service and Disposal.

- Comparative Analysis – Task 203. This task provides insight into potential supportability factors through quantitative analysis of existing comparable systems; in effect this task offers an excellent opportunity to identify cost and availability drivers by learning from experience.
- Technological Opportunities – Task 204. The aim of this task is to identify technological related constraints and opportunities. The task endeavours to influence technology selection based on system supportability rather than the provision of functionality.
- Supportability Related Design Factors – Task 205. This task documents supportability factors and the system's sensitivity to their variation. It uses these factors to form the basis of supportability objectives (performance related requirements) that need to be considered during system development.
- 300 Series LSA Tasks. These tasks are carried out to gain an understanding of support system alternatives and enable the selection of the most appropriate support solution based on qualitative and quantitative factors.
  - Functional Requirements Identification – Task 301. This task identifies potential support system functional requirements. These functions are then decomposed into activities, which in turn can be used as the basis for creating a task inventory.
  - Support System Alternatives – Task 302. This task explores and documents potential support options capable of satisfying the support need.
  - Evaluation of Alternatives and Trade-off Analysis – Task 303. This task assesses the support options and generates qualitative and quantitative trade-off criteria that need to be considered during support option selection.
- 400 Series LSA Tasks. This task series is responsible for the identification of comprehensive logistic support requirements. These requirements capture the detailed characteristics of the chosen support option, such that the support need can be fulfilled.
  - Task Analysis – Task 401. The purpose of this task is to identify all support resource, timing, location and IPR requirements.
  - Early Fielding Analysis – Task 402. The purpose of this task is to determine the impact on existing systems and support systems that are attributable to the new equipment. Potential impact areas include; the provision of new skills, new methods for software replication and transfer, new methods for configuration management and back-up, the need to make adaptive modifications, and the need for new software maintenance facilities.

- Post Production Support Analysis – Task 403. This task aims to identify possible through-life support issues including obsolescence management (transfer devices, development environment and software licences), and post modification integrity assurance (including safety and security).
- 500 Series LSA Tasks. This task is carried out to begin the process of progressively assuring that supportability requirements are achieved and deficiencies corrected where necessary.
  - Supportability Test, Evaluation and Verification – Task 501. This task documents supportability assessment considerations and objectives that embrace best practice. These considerations and objectives will then be used to formulate supportability assessment criteria as the project matures.

In general, as a project progresses from the Concept phase through to the end of the Assessment phase, its LSA migrates from that of developing domain understanding, identifying the support need and determining significant support related design factors, to one of maintaining inter-domain congruency, stating the system support requirements and influencing design (product and process) for support. This migration is enabled by the provision of maturing project data, without which LSA cannot fulfil the need to provide an informed, justified and traceable case on which to select and develop a support solution.

In order to carry out software LSA a specialised set of skills is required. Personnel have to understand the procurement process, support analysis techniques and factors specific to the software domain. Within the MOD an Integrated Project Team (IPT) is assigned responsibility for procuring and bringing into service systems that will meet the defence need. As with all government organisations the staffing of these IPTs is limited which, although not discussed here, is a particularly constraining factor during the early procurement phases. In recognition of this situation, organisations such as AvSG exist to support IPTs, this said, whilst AvSG is a support organisation it is not permitted to perform manpower substitution by acting as a pool of freely available personnel.

AvSG contains teams covering many of the system engineering disciplines, Software Supportability being just one of them. The strength of these teams is that they focus on their own domains, creating single points of expertise. In particular the Software Supportability (SS) Team provides a vehicle for ensuring commonality and improvement in the application of software support analysis techniques. From this insight, it is evident that AvSG has to perform a balancing act; it must provide assistance to the IPTs whilst not taking complete ownership and responsibility for the production of support analysis outputs. This balancing act is critical to the continued success of AvSG, as failure in either mode might result in the scrutiny of AvSG tasks and a potential reduction in staffing levels.

Guidance on LSA task responsibility as documented within Def Stan 00-60<sup>16</sup> is given at Table 1. Within this guidance, responsibility is allocated to the Central Customer (the capability owner), IPT, Contractor<sup>§</sup>, or Customer 2 (the capability provider). Unfortunately, I believe that these differing stakeholders serve only to complicate the issue of LSA

---

<sup>§</sup> Where the Contractor is the vendor supplying the equipment being procured.

responsibility and implementation, as whilst Def Stan 00-60 provides recommendation on LSA responsibility, it offers no assistance on the subject of who is best placed to carry it out. The heading ‘Responsible Agency’ used within Table 1, is defined within Def Stan 00-60 as:

*“The term ‘Responsible Agency’ indicates who could be responsible for initiating action under each task.”*

Note that the definition itself is very weak and by no means delegates responsibility to any single organisation, as such the overarching procurement responsibility owned by the IPT must act as the authority under which LSA is managed and carried out.

Task	Description	Responsible Agency
101	Development of an Early LSA Strategy	Central Customer and IPT
102	LSA Plan	IPT
103	Programme and Design Reviews	IPT and / or Contractor
201	Use Study	Central Customer, IPT and / or Contractor
202	Software and Support System Standardisation	IPT and / or Contractor
203	Comparative Analysis	IPT and / or Contractor
204	Technological Opportunities	IPT and / or Contractor
205	Supportability Related Design Factors	IPT and / or Contractor
301	Functional Requirements Identification	Contractor
302	Support System Alternatives	IPT and / or Contractor
303	Evaluation of Alternatives and Trade-off Analysis	IPT and / or Contractor
401	Task Analysis	Contractor
402	Early Fielding Analysis	IPT, Customer 2 and / or Contractor
403	Post Production Support Analysis	IPT, Customer 2 and / or Contractor
501	Supportability Test, Evaluation and Verification	IPT, Customer 2 and / or Contractor

Table 1 – Def Stan 00-60 Defined LSA Task Responsibility

To simplify the issue of who should carry out LSA, I believe that it is best to consider only two basic options, these being the MOD or a Service Supplier \*\*. This belief is based on the premise that there are only two fundamental choices available to the MOD, it either carries out the analysis itself or it pays another organisation to complete it on their behalf.

Of these two options, the latter is often considered to be the most favourable by LSA managers as it enables the MOD to discharge its responsibility to another organisation. My perspective is somewhat different though; I believe that, should the MOD choose to discharge its analysis responsibility to another organisation, it at best increases the risk of support requirements definition failure and at worst becomes exposed to exploitation during software support activities. Therefore, to avoid this situation or at least manage it, the MOD must establish and maintain domain competency, oversee the analysis process and ensure it is fully involved in the requirements identification process.

To assist with the decision of who should conduct LSA, three basic factors need to be assessed and balanced, these being cost, benefit and feasibility. These factors vary in accordance with LSA tasks to be performed, the depth of analysis required and availability of appropriately skilled personnel, which in turn is dependent upon the project’s phase in the procurement lifecycle. Based on experiences gained from numerous projects, the

\*\* Where the Service Supplier is any non-MOD organisation bought in to carryout LSA, either related or unrelated to the vendor supplying the equipment being procured.

relationship between these factors for LSA conducted by the MOD or a Service Supplier is shown in Tables 2 and 3 respectively.

Procurement Lifecycle Phases				
	Concept	Assessment	Demonstration	Manufacture
Strengths	<ul style="list-style-type: none"> <li>• Best placed to understand the problem domain and identify the support need.</li> <li>• Resources available at no additional cost to an IPT.</li> </ul>	<ul style="list-style-type: none"> <li>• Best placed to understand the problem domain and state the support need.</li> <li>• Resources available at no additional cost to an IPT.</li> </ul>	<ul style="list-style-type: none"> <li>• Best placed to understand the problem domain and review LSA outputs from the Contractor.</li> <li>• Resources available at no additional cost to an IPT.</li> </ul>	<ul style="list-style-type: none"> <li>• Best placed to understand the problem domain and review LSA outputs from the Contractor.</li> <li>• Resources available at no additional cost to an IPT.</li> </ul>
Weaknesses	<ul style="list-style-type: none"> <li>• Insufficient resources available to support all procurements.</li> </ul>	<ul style="list-style-type: none"> <li>• Insufficient resources available to support all procurements.</li> <li>• Availability of project data and the completion of quantitative LSA is often hampered when analysis is carried out by non-IPT based personnel.</li> </ul>	<ul style="list-style-type: none"> <li>• Insufficient resources available to support all procurements.</li> <li>• Quantitative LSA is often hampered by poor project data availability when analysis is carried out by non-IPT based personnel.</li> <li>• MOD personnel do not hold necessary technical skills and project experience to complete detailed system analysis.</li> </ul>	<ul style="list-style-type: none"> <li>• Insufficient resources available to support all procurements.</li> <li>• Quantitative LSA is often hampered by poor project data availability when analysis is carried out by non-IPT based personnel..</li> <li>• MOD personnel do not hold necessary technical skills and project experience to complete detailed system analysis.</li> </ul>

**Table 2 – MOD Related Factors**

Procurement Lifecycle Phases				
	Concept	Assessment	Demonstration	Manufacture
Strengths	<ul style="list-style-type: none"> <li>• Improved likelihood of resource availability.</li> </ul>	<ul style="list-style-type: none"> <li>• Improved likelihood of resource availability.</li> <li>• Improved availability of project data if the Contractor carries out LSA.</li> </ul>	<ul style="list-style-type: none"> <li>• Improved likelihood of resource availability.</li> <li>• Improved availability of project data if the Contractor carries out LSA.</li> <li>• Improved availability of technical skills and project experience if the Contractor carries out LSA.</li> </ul>	<ul style="list-style-type: none"> <li>• Improved likelihood of resource availability.</li> <li>• Improved availability of project data if the Contractor carries out LSA.</li> <li>• Improved availability of technical skills and project experience if the Contractor carries out LSA.</li> </ul>
Weaknesses	<ul style="list-style-type: none"> <li>• Additional burden of contract management.</li> <li>• The MOD has to pay for the analysis.</li> </ul>	<ul style="list-style-type: none"> <li>• Additional burden of contract management.</li> <li>• The MOD has to pay for the analysis.</li> </ul>	<ul style="list-style-type: none"> <li>• Additional burden of contract management.</li> <li>• The MOD has to pay for the analysis.</li> </ul>	<ul style="list-style-type: none"> <li>• Additional burden of contract management.</li> <li>• The MOD has to pay for the analysis.</li> </ul>

Procurement Lifecycle Phases				
	Concept	Assessment	Demonstration	Manufacture
	<ul style="list-style-type: none"> <li>• Additional effort is required to gain an understanding of the operational environment and customer support needs.</li> <li>• Increased exposure to exploitation during software support activities when analysis is conducted by the Contractor and insufficiently monitored by the MOD.</li> </ul>	<ul style="list-style-type: none"> <li>• Increased likelihood of requirements definition failure caused by unfamiliarity with operational environment.</li> <li>• Increased exposure to exploitation during software support activities when analysis is conducted by the Contractor and insufficiently monitored by the MOD.</li> </ul>	<ul style="list-style-type: none"> <li>• Quantitative LSA is often hampered by poor project data availability when analysis is carried out by non-Contractor based personnel.</li> <li>• Non-Contractor based personnel will not hold necessary project experience to complete detailed system analysis.</li> </ul>	<ul style="list-style-type: none"> <li>• Quantitative LSA is often hampered by poor project data availability when analysis is carried out by non-Contractor based personnel.</li> <li>• Non-Contractor based personnel will not hold necessary project experience to complete detailed system analysis.</li> </ul>

Table 3 – Service Supplier Related Factors

With these factors in mind, I suggest that where resources allow, there is very little to be gained by having a non-MOD organisation carry out LSA during the Concept phase and early in the Assessment phase. Use of a Service Supplier other than the Contractor would provide some separation of interests between the organisation carrying out analysis and the organisation that will eventually provide support, but this option will suffer from many of the weaknesses common to both the MOD and the Contractor. Based on the grounds that during product and support solution development only the Contractor will have the necessary access to vital product data; I suggest that as a project matures through to the Demonstration and Manufacture phases, it is impracticable to expect any organisation other than the Contractor to carry out LSA.

### 2.3 APPROACH AND IMPLEMENTATION

The goal of FOAS LSA was to produce timely outputs capable of providing a basis on which products and processes might be influence for supportability, with the aim of reducing software whole life support costs. To date this has been managed by focusing on functional analysis, where functional analysis only considers the roles and services that a system is required to provide, and specifically does not concern itself with how these roles or services will be provided. In this way all analyses should remain valid regardless of eventual system implementation. Functional analysis provides further benefit in that it supports analysis evolution through its ability to operate on maturing data. As with all initial or foundation analysis, a number of assumptions have been stated to bridge data and information voids, as such the qualification and maturation of these assumptions has become a vital element of the analysis process.

For the FOAS project, whilst the IPT maintained LSA responsibility, the SS Team within AvSG was tasked with its implementation. To meet schedule requirements and gain the best opportunity to influence design, it was decided that the SS Team would deliver two sets of outputs, qualitative and quantitative. This approach was taken as in the absence of

any other analysis, I had to identify all the areas of concern and gain an understanding of their potential impact on system support. By initially taking a purely qualitative viewpoint, I hoped to gain a broad understanding of the problem domain. This in turn would then be used as the basis to identify areas of high gain on which future qualitative analysis could be focused.

To date only the qualitative work has been carried out, which occurred during the latter stages of the Concept phase and first few weeks of the Assessment phase. This work comprised of an initial iteration of the 200, 300, and 500 series tasks. As expected, it was impossible to carry out an in-depth analysis of each FOAS element at this time, as the project is very early in the procurement lifecycle and no system composition or implementation decisions have been made. Knowing the potential limitations of this LSA, the approach taken was to identify the major areas of supportability concern by covering the broadest range of support analysis topics possible, as such FOAS LSA was carried out as previously described with the exception of:

- 300 Series LSA Tasks. Due to the lack of project specific data these tasks were only carried out qualitatively. Although limited in application this task still added value as it facilitated a first-cut selection of support solutions based on basic viability criteria.
  - Support System Alternatives – Task 302. Although very early in the procurement lifecycle, this task limited potential support options through categorization. The need to limit support options was driven by the reality that analysis cannot be carried out for an infinite number of potential support providers. The use of categorization was appropriate, as it enabled early analysis within minimal constraints.
  - Evaluation of Alternatives and Trade-off Analysis – Task 303. This task qualitatively assesses support options (ranging from unviable to favourable) by considering the relative ability of categorized support providers to fulfil support function needs. This assessment has provided value in that it has enabled the focusing of effort for further quantitative analysis.

Notably missing from FOAS LSA is the 100 and 400 series tasks. These tasks were not carried out as they were considered to be of little value at this time. The rationale for this being as follows:

- 100 Series Tasks. The LSA was carried out solely by a MOD organisation. Whilst a project plan was created to assist with the achievement of milestones; this plan was not established for the purposes of formal programme review or the negotiation staged payments, as such the total application of these tasks would have been inefficient and inappropriate.
- 400 Series Tasks. This collection of tasks focuses on identifying the detailed characteristics of the chosen support option. At this time, when no support option has been chosen, this task cannot be carried to any depth greater than that facilitated under the 300 series tasks. Typically this task series is

undertaken as a project reaches the Demonstration and Manufacture phases of the procurement lifecycle.

This work was presented to the FOAS Integrated Logistics Support Manager (ILSM) and the scrutineers involved in the appraisal of FOAS LSA prior to Initial Gate<sup>††</sup> approval. The outputs were presented in a series of reports, which effectively fulfilled the role previously performed by a SSPP. The work, completed on schedule over a twenty-four month period, was well received by both recipients and placed the FOAS IPT in a good position for its Initial Gate submission. However, as with all emerging disciplines, a number of improvements could be made if the analysis were to be conducted again, the most pertinent of these being as follows:

- Facilitation. Although only qualitative at this time, an amount of project data is still required to enable LSA completion; this is particularly the case for Task 203 – Comparative Analysis. For the FOAS project this data was not readily available as the FOAS IPT was formed to procure new capability and did not have any direct links to IPTs currently supporting similar platforms or equipment. This situation resulted in the SS Team having to approach numerous MOD organisations requesting what was commercially sensitive or even restricted data; requests that often needed a great deal of diplomacy and negotiation to secure data release and use. Unfortunately, due to how the Statement of Work (SOW) had been established between the FOAS IPT and the SS Team, on occasions when the release of sensitive data was delayed, the SS Team held the risk of not achieving agreed project milestones. To avoid this situation from occurring during future projects and improve IPT awareness of the risks associated with the timely completion of LSA, I suggest that the role of LSA facilitator is delegated to a member of the IPT. The facilitator is to take responsibility for the provision of project related data; as such the facilitator is to be of a sufficient grade to enable communication at an appropriate level across organisations.
- Authority and Sponsorship. During the implementation of LSA numerous recommendations, design factors and support requirements are identified. Having been identified, these outputs can only realise some value if they are allowed to influence a project such that supportability is improved. Within an IPT managerial roles in the domains of System Requirements, Support, Safety and Security are rarely performed by the same person, as such prior to the expenditure of any effort on LSA, communication and authority channels need to be established between all necessary stakeholders. To fulfil this function between the IPT and the organisation conducting LSA, I again suggest that a LSA facilitator should be established and in addition to their previously stated role, the facilitator must also hold an authority to influence project development. I further suggest that the facilitator should act as LSA Sponsor, as I believe this will improve the likelihood of LSA outputs being accepted and championed during project development.

---

<sup>††</sup> The point at which the Business Case is submitted to the approving authority, making the case for proposed expenditure on the Assessment phase of the project.

- Method. Prior to the FOAS project, neither the MOD nor a Service Supplier had carried out Software LSA at such an early time in the procurement lifecycle, with the specific goal of influencing software products and processes to reduce Whole Life Costs (WLC). As such, whilst guidance was available within Def Stan 00-60, without the ability to refer to 'best practice' LSA outputs, a degree of uncertainty existed as to the necessary content of some LSA task outputs. As previously discussed, LSA comprises of a number of tasks, however unmentioned until now is the fact that each of these tasks comprises of a number of sub-tasks. These sub-tasks add extra structure to the LSA framework and are intended to assist with implementation. This said, we must now remember that the implementation of Def Stan 00-60, as intended by the standard, is not designed for the early procurement lifecycle phases. With these two major factors in mind, it was decided to carry out 'Top Down' task level LSA by focusing on the overall aim of each task rather than becoming preoccupied with the specific implementation of any one sub-task. This decision was made as it was considered to be the most appropriate means by which LSA could be broadly applied and domain experience developed. For future implementations of LSA, I suggest that this method should not be followed. Now that an initial set of LSA documentation has been produced, and much experience gained, a 'Bottom Up' sub-task approach to LSA can be adopted. In this way each of the LSA component parts can be addressed as opportunity allows, which will assist with the management and timely completion of outputs. Evidence to this fact has been provided by more recent analysis conducted for a mission system upgrade, which was completed quicker and to a greater depth than that of the FOAS project.
- Correctness and Completeness. As a side effect of the original task selection and the completion of LSA in a Top Down manner, some necessary analysis outputs were inadequate or completely overlooked. This required corrective action later in the analysis process, the most notable of these being the late formulation of Supportability Performance Requirements for Initial Gate and inclusion in the User Requirements Document. Now that the initial set of LSA documentation has been produced, it is evident that the Concept phase work, which effectively only addressed the 200 series LSA tasks, was insufficient. Subsequent Assessment phase LSA, consisting of the 300 series and 500 series tasks, could have been produced during the Concept phase; this would have improved analysis completeness and enhanced the quality of outputs. To prevent this situation from occurring in future projects, I suggest that future Concept phase analysis should, without exception, cover 200, 300 and 500 series LSA tasks. With a foundation iteration of this analysis completed by Initial Gate, projects would be in a well informed position prior to embarking upon the Assessment phase, and improved position to select an alternative organisation to conduct future iterations of LSA.
- Reporting. In total five reports were generated to document the LSA outputs. These reports followed RAF writing convention; as such their format was compiled with little communication to or from the task Sponsor. The body of each report covered analysis, conclusions and recommendations; where

necessary annexes were used to document data lists and detailed analysis. This approach proved to be unsatisfactory to the Sponsor as some of the reports were unwieldy and did not readily assist the reader in identifying pertinent supportability issues. This was particularly the case for the first report that covered LSA tasks 201, 202 and 203 due to the high degree of coupling between them. In future I suggest that, whilst still following RAF writing conventions, the body of each report should only cover a summary of the analysis output. The LSA itself should be contained within annexes, and data lists or further detailed analysis should be contained with appendices to the appropriate annex. Great consideration should be made of the potential impact on readability if multiple tasks are to be covered within a single report, and by default LSA tasks should be presented as singular task documents.

## SUMMARY

This section of work has been produced to help identify some of the common causes of support analysis failure. This has been achieved through an examination of guidance material provided by the MOD on the application of LSA to software and a review of the support analysis carried out for the FOAS project.

To assist with the achievement of support effectiveness and efficiency the support need must be understood. However, the MOD does not have a rigorous method to capture and promulgate support analysis, which results in supportability decisions becoming untraceable, ill-informed and unjustified. To further hinder the LSA process, the MOD has allowed its software analysis capability to erode placing the MOD in a position of vulnerability through support exploitation.

Guidance provided within Def Stan 00-60 does not help to reduce the risk of support analysis failure, as outputs created through its application occur when the best opportunity to influence system design for support has been lost. Instead, the spirit of Def Stan 00-60 needs to be considered, such that supportability design factors can be established at a time when they can be used to influence product and process design.

Further to understanding when to carry out LSA, the question of who should carry it out needs to be answered. This issue is dependent upon factors that include: the LSA tasks to be performed, the depth of analysis required and availability of appropriately skilled personnel. Def Stan 00-60 offers little assistance on this matter as it only suggests organisations that could hold responsibility for initiating LSA.

The MOD can either carry out LSA itself or pay another organisation to complete it on their behalf, of which the latter is often considered to be the most favourable by LSA managers. Should the MOD choose to discharge its analysis responsibility to another organisation, the risk of support requirements definition failure and support exploitation increases. However, this situation can be managed if the MOD establishes and maintains domain competency, oversees the analysis process and ensures it is fully involved in the requirements identification process.

There is very little to be gained by having a non-MOD organisation carry out LSA during the Concept phase and early in the Assessment phase. But as a project matures through to the Demonstration and Manufacture phases, it is impracticable to expect any organisation other than the Contractor to carry out LSA.

The goal of FOAS LSA was to produce timely outputs capable of providing a basis on which products and processes might be influenced for supportability. This has been managed by focusing the roles and services that a system is required to provide. In this way all analyses should remain valid regardless of eventual system implementation. Schedule requirements were managed through the division of outputs into qualitative and quantitative phases. Initially a purely qualitative viewpoint was adopted such that a broad understanding of the problem domain could be established.

To date only qualitative Concept phase and Assessment phase work has been carried out, comprising of the 200, 300 and 500 series LSA tasks. Due to project immaturity quantitative analysis of FOAS supportability has not been possible. The work, completed on schedule over a twenty-four month period, was well received by the FOAS Integrated Logistics Support Manager (ILSM) and project scrutinisers. However, if the analysis were to be conducted again improvements could be made in areas such as: task establishment and data provision, LSA influence and value realisation, method and approach, and the timing, quality and appropriateness of outputs.

Unfortunately, understanding the failings of existing guidance is not enough to overcome all the problems associated with software LSA. This is because in addition to the aforementioned failings, software LSA guidance does not cover the complete set of factors that need to be addressed in order to ensure its outputs meet the customer's need. To resolve this situation the following sections will aim to identify the totality of support and present methods that should assist the analysis process.

## SECTION 3

### SUPPORTABILITY REQUIREMENTS

#### 3.1 THE TOTALITY OF SUPPORT

In order to derive support requirements we first had to gain an understanding of the functions that constitute an adequate support system; making visible those activities that are typically implied rather than stated, such as data support and integrity assurance. This visibility will aid support acquisition and, where query and change drivers are common to multiple system elements, provide an indication of areas that should be highlighted as potentially critical items in terms of operational availability and capability sustainment. According to Boehm<sup>17</sup>, the software maintenance process is a continuous closed-loop cycle, as shown in Figure 6.

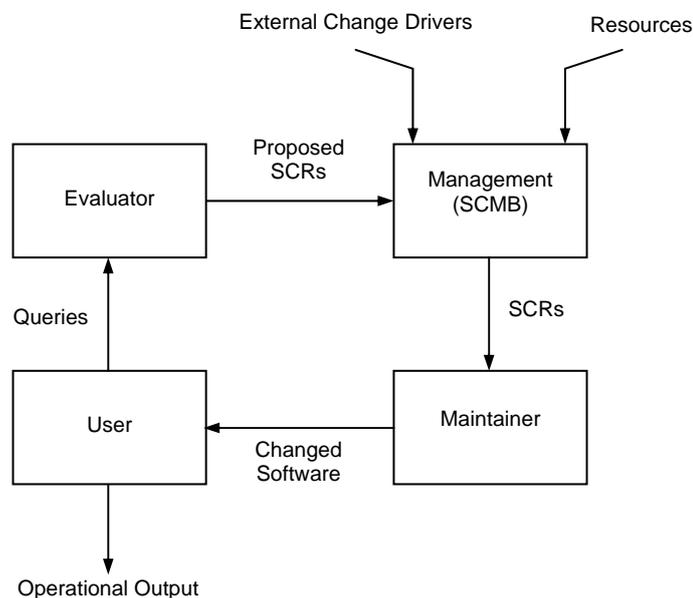


Figure 6 – The Software Maintenance Life Cycle

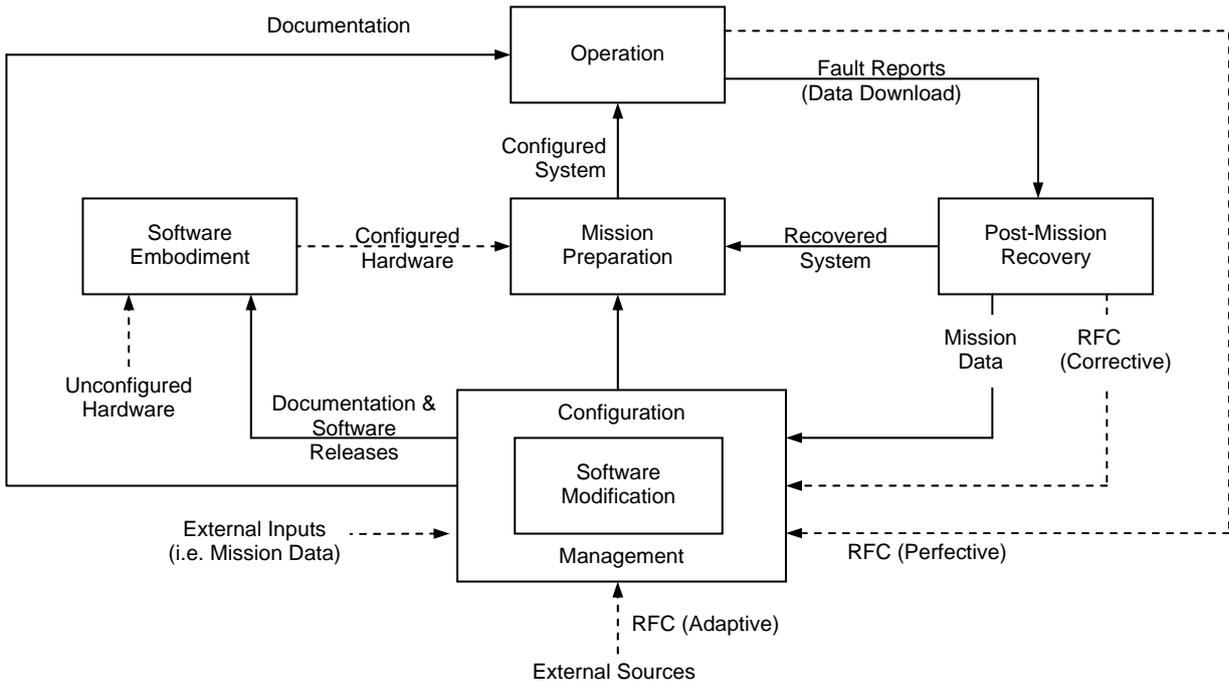
Whilst Boehm's model provides a foundation for defining a support environment, it requires a little explanation to improve our understanding of the roles identified within it. Within a military support context these roles are typically as follows:

- User. The User represents all the personnel that will utilise the software's output to perform some task. Users will provide feedback on the effectiveness and efficiency of the performed tasks, raising Software Queries if the system performance or integrity is questioned.
- Evaluator. The Evaluator examines and filters Software Queries (SQ). The filtering of queries is carried out to identify the nature and cause of the originating event and justify query acceptance or rejection. An accepted SQ will generate an associated proposed Software Change Request (SCR). The evaluator typically performs the following activities:

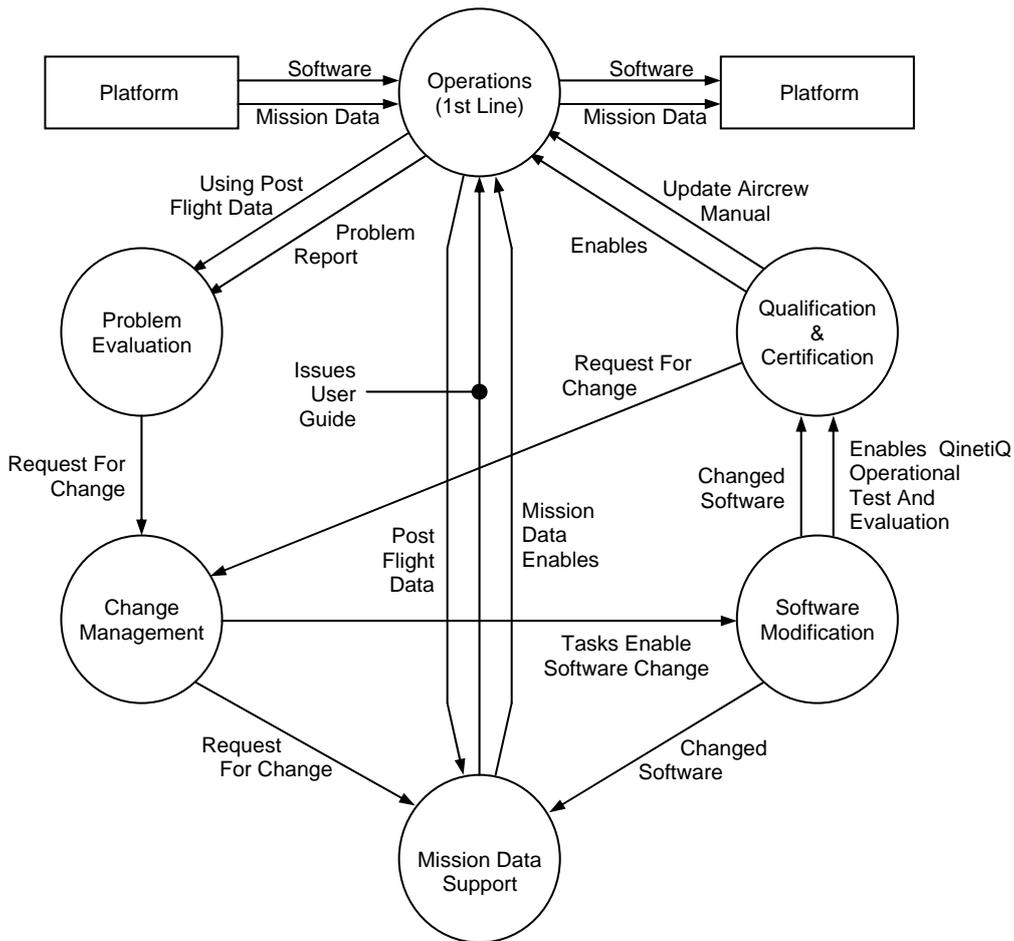
- Reproduction of the problem.
- Collection of information about each SCR.
- Setting up of mechanisms to categorize results.
- Management. The Management represents the organisation that manages operational readiness through the control of software changes. Decisions on the implementation of SCRs are based on the outputs of cost-benefit evaluation and risk analysis. Evaluation is carried out for the financial cost of implementing the change against the need to maintain operation readiness. For systems in RAF Service the Software Configuration Management Board (SCMB) carries out this function<sup>18</sup>. Management typically perform the following activities:
  - Cost-benefit evaluation and risk analysis.
  - SCR authorisation.
  - Generation of SCR priority list.
- Maintainer. The Maintainer represents the organisation that will implement the SCR. The Maintainer is authorised by the Management to modify the software and satisfy the SCR. The Management funds the Maintainers activities (and possibly their resources). Maintainers typically perform the following activities:
  - Design of changes and tests.
  - Building of a new release, (comprising of; editing source code, archiving and quality assurance).
  - Testing.

Within Def Stan 00-60<sup>15</sup> a generic model of software support is provided, as shown in Figure 7. This model, and the accompanying sub-models, are fundamentally different to Boehm's, in that they have a function-based perspective.

On inspection, although viewed from differing perspectives, these models exhibit a great deal of commonality. However differences do exist, specifically in the areas of query evaluation and software operation. As such, the sole use of either model would result in an incomplete support model and potential derivation of incomplete support requirements. Recognising this, work<sup>19</sup> was undertaken within AvSG to create a new support model that would combine Boehm's and Def Stan 00-60's models and include any other functions of support significance; the initial output of this work is given in Figure 8.



**Figure 7 – Def Stan 00-60 Generic Model of Software Support**



**Figure 8 – Initial Support System Model**

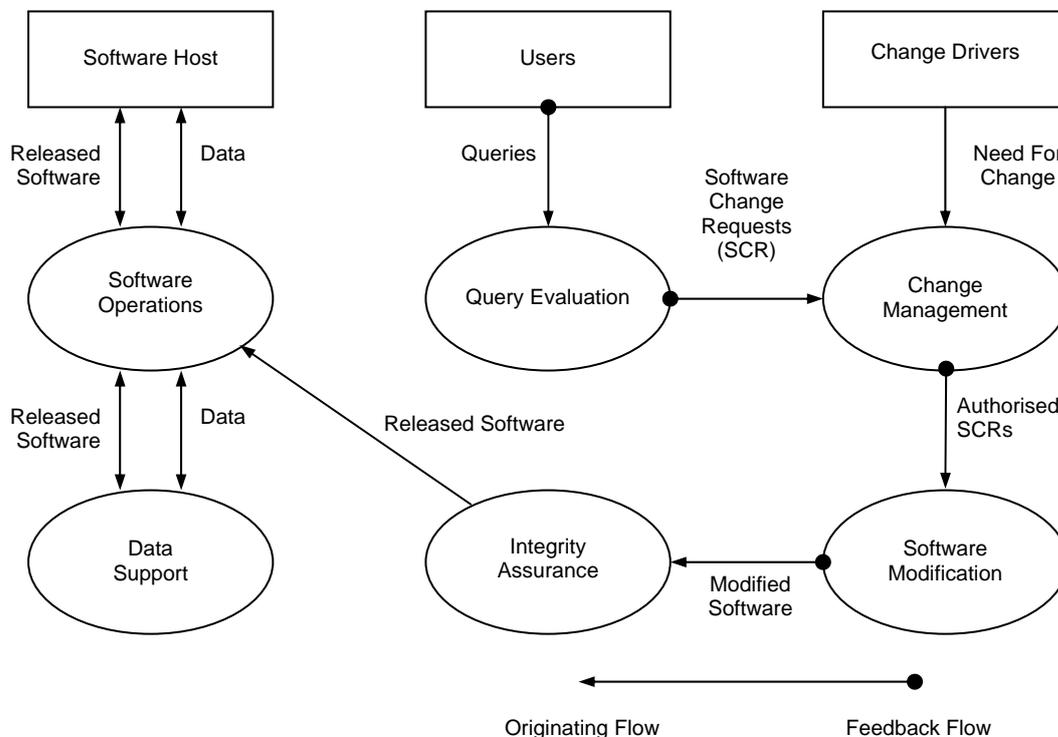
Unfortunately my view of this model is not complementary; I consider it to be ill-formed and a potential source of harm to any report's credibility. However, without an alternative it was used reluctantly within the 200 series LSA reports. My opinion is based on a number of factors, which include but are not limited to the following:

- The model lacks consistency – flows are allowed to represent both entities and actions.
- The model does not represent the real world – problem reports can only originate from the Operations function.
- The model is constrained – specific organisations are implied as function providers.
- The model is not appropriate for universal application – system qualification and certification might not be a significant factor in all situations.
- The model is prone to misinterpretation – the model had no supporting documentation.

To rectify this situation, I refined the model prior to its reuse during the 300 and 500 series task reports, as illustrated in Figure 9. With specific consideration of the aforementioned failings, the refinement process was based on a defined set of model rules and considerations, as such these rules and considerations have become a vital element of the model's documentation set.

- Model Rules. The rules are based on the rationale that a model without consistency is not interpretable and that a model without fidelity is meaningless. As such, these characteristics were used to define rules that govern how the model can be drawn and state its context and scope. These rules are 'unbreakable' and any model that fails to accommodate even one rule is unacceptable. The model rules are as follows:
  - All ovals are Functions; their names are 'verb phrases' and describe the tasks they perform.
  - All rectangles are Actors; their names are 'noun phrases' and describe things that interact with the support system.
  - All lines are Flows; their names are 'noun phrases' and describe items that transit around the support system.
  - The model must be able to accommodate all instances of modification types (Corrective, Adaptive, Perfective and Enhancement both internally and externally driven).
  - The model must be able to accommodate all support profiles (Peace Time, Crisis, Tension and War).

- We must be able to discharge any assumptions made about support within a valid model.
- Model Considerations. The considerations have been established to improve model value and give additional insight on its application. The considerations help to keep the model generic and hence reusable, irrespective of the software item being analysed or the alternative support solutions available. The considerations are not unbreakable; in fact as a project matures it would be appropriate to annotate models with support provider information to improve understanding.
  - The model shows support functions, which only describe the tasks to be performed. Specifically, the model does not dictate organisational boundaries or physical locations.
  - The model should be equally valid for the system as an entity in itself and for each of its component parts. This may include, but is not limited to, mission software, maintenance software and simulation software. However, it is accepted that not all functions will attract the same level of support criticality.
  - Both the Flows and Software Operations function require analysis to understand how software and data products move throughout the support system.



**Figure 9 – Support System Model**

The final item of documentation is the model description. The model description is as important as the model illustration itself, as the description details many of the subtleties through which the Support System Model obeys its 'rules' and accommodates the 'considerations'. During FOAS analysis the Support System Model was presented without a description, which promoted misunderstanding and incorrect interpretation. As such, I suggest that model illustrations and their descriptions are treated as inseparable elements of any Support System Model. A generic model description, complementary to Figure 9, is as follows:

- Users. The term Users refers to all the personnel that interact with the system, specifically this includes the operators and support personnel. Inevitably, as these Users interact with the system software they will have questions about its operation, discover problems and generate ideas for adaptations, improvements and new functions (enhancements). These queries, which capture all internally generated change needs, are formalised by creating a Query Report. The Query Report captures all relevant information and is forwarded (along with any relevant data) to the Query Evaluation function.
- External Change Drivers. The term External Change Drivers refers to a source of change needs that originate from outside normal system operation. Examples of these change drivers include the need to sustain capability in response to data format changes and the need to preserve functionality in response to changes in interfacing software components or underlying hardware.
- Software Host. The term Software Host refers to the physical equipment in which the software and data resides, such that through its operation some function of the system is enabled.
- Software Operations. The Software Operations function comprises of: Software Operations Support – actions necessary to load, re-load, replicate, copy, label, store, distribute, recall or carry out any handling activity on software or firmware, Data Preparation And Recovery – the transfer of data to and from the Software Host for mission, maintenance, analysis or sanitization purposes.
- Query Evaluation. The Query Evaluation function evaluates and filters Query Reports to identify the cause of any query, categorizes the nature of any problems, remove duplicates and justify query acceptance or rejection. Some queries will generate Software Change Requests (SCR), which might relate to either the system documentation or code itself. For all SCRs the Evaluators must assess the operational benefits, costs and risks of each change in support of the Change Management function.
- Change Management. The Change Management function manages operational capability and readiness through the control and prioritisation of software changes. Change Management deals with user-initiated SCRs (via evaluated Query Reports) as well as externally driven change needs. The

function reconciles demands for change with the business goals, constraints and available resources.

- Data Support. Within the Data Support function, Data refers to information, both mission and engineering related, loaded to or from the Software Host. This Data enables system functionality or performance analysis. Specifically, the term Data does not relate to internally created variables, which have no use outside of system operation. The Data Support function captures all the activities necessary to create, preserve, modify and analyse data, such that it becomes operationally useful.
- Software Modification. The Software Modification function is responsible for the implementation of Authorised SCRs. In addition to this, it has to assess its own capacity for tasking and communicate this capacity to the Change Management function during the assessment and prioritisation of SCRs. The output of Software Modification is a new software load, which after its release (and integrity assurance where necessary), is ready for use by the Software Host (as the system software) and Data Support function (in support of its own activities).
- Integrity Assurance. The Integrity Assurance function is responsible for verifying that software products are acceptable for release, i.e. they remain acceptably safe, secure, reliable or supportable for use after the implementation of Authorised SCRs. It is important to realise that this function only represents the formulation of evidence into a statement of assured integrity for a desired quality characteristic. Specifically, the function does not represent all the activities that build towards product integrity; these activities exist throughout the maintenance model. This function can be bypassed where software products do not attract a specific need for integrity assurance

### 3.2 SUPPORTABILITY REQUIREMENTS ENGINEERING

Within the acquisition lifecycle support requirements should be derived via a two-stage process, an overview of this process is as follows:

- The User's support need is captured within the User Requirement Document (URD). These needs describe the level of support considered necessary to satisfy projected operational usage of the new system.
- The system characteristics, necessary to meet the User's needs, are defined within the System Requirements Document (SRD). These characteristics are used to form detailed requirements, which are then apportioned via the system and support system architecture and design, such that the desired supportability characteristics will emerge as the project develops.

Within the procurement process, measures are in place to ensure that the SRD is traceable from the URD. To control requirements creep and assist contract management, it is very difficult to impose SRD level supportability requirements without a parent User

requirement, as such the formulation of an appropriate and complete URD is a vital element of the procurement process.

### 3.3 PERFORMANCE SUPPORT REQUIREMENTS

Large systems such as FOAS have the potential to generate a vast amount of software queries and change drivers, but a system in a constant state of flux is unacceptable, as operators never gain the opportunity to become instinctive users. This means that the support solution must be capable of both enabling and controlling change, balancing the need to sustain capability whilst maintaining operational effectiveness. Within AvSG generic supportability performance requirements have been developed based on the outputs of FOAS and other<sup>20</sup> LSA. These performance requirements, which were developed after an assessment of the Support System Model, potential sources of change and a review of changes made by current RAF Software Support Teams, are documented in Table 4. It is intended that elements of these requirements written in italics must be refined and ratified, such that they accurately represent the required support capability. Unfortunately, whilst the refinement of these requirements will be assisted by the 'quantifying' phase of LSA, I believe that LSA alone is not an adequate means of achieving this goal due to the following factors:

- Quantitative LSA is dependent upon system data that is not always immediately available. Waiting for this data to become available could add a considerable and unacceptable delay to the acceptance of supportability related performance requirements.
- LSA by its nature is limited as a means of refining and ratifying support requirements, this is because it focuses on data and analysis rather than discussion and agreement.

In light of the limitations of LSA an alternative approach is necessary, one such method being initial requirement identification followed by stakeholder negotiation and ratification (potentially achieved at appropriate working levels of the Capability Working Group<sup>††</sup>). Once agreed, these requirements should then be managed and matured as with all other system requirements.

Within the Support System Model rules an inference<sup>§§</sup> is made for scalable support in recognition of change driver criticality. For the RAF saleability is accomplished through the application of an Urgent Operational Requirement (UOR) process. The UOR process is typically initiated during times of Crisis, Tension or War when the implementation of mission critical modifications is often required in extremely short timescales. In overview, the UOR process shortcuts effort intensive modification activities, accepting that modification rework will be necessary during peacetime to meet quality requirements. The support performance requirements listed in Table 4 recognise the need for scaleable support by quoting response times for both routine (peace time) and mission critical changes (operationally essential – i.e. Crisis, Tension or War).

---

<sup>††</sup> The CWG is an adaptive, flexible and social process aimed at providing the advice from which the empowered Director of Equipment Capability (the capability owner) makes decisions.

<sup>§§</sup> "The model must be able to accommodate all support profiles (Peace Time, Crisis, Tension and War)."

Suggested URD Section	Unique Identifier	Proposed Requirement
General	1.1	The Authority requires that the <b>capability</b> shall be sustained and enhanced throughout its Service life.
	1.2	The Authority requires that support of the <b>capability</b> addresses technical, tactical and User (air/ground operator and maintainer) issues in an integrated manner.
Query Evaluation	1.3	The Authority requires that queries about the <b>capability</b> are answered, on average, within <b>5 working days</b> (queries may be answered by raising a Change Request (CR)).
Change	1.4	The Authority requires that authority-approved CRs for mission-critical functions, wherever found in the <b>system</b> , shall be satisfied and released for flight within, on average, <b>4 weeks</b> (for UOR and safety-critical <sup>***</sup> CRs) and <b>18 months</b> (for all other CRs) of the CR being raised. This change capability includes:
	1.4.1	<ul style="list-style-type: none"> <li>CRs to sustain the <b>capability</b> in the presence of a change in environment software interfaces or hardware.</li> </ul>
	1.4.2	<ul style="list-style-type: none"> <li>CRs to correct <b>system</b> problems and deficiencies.</li> </ul>
	1.4.3	<ul style="list-style-type: none"> <li>CRs to improve <b>system</b> performance, usability or support.</li> </ul>
	1.4.4	<ul style="list-style-type: none"> <li>CRs to enhance <b>system</b> capability.</li> </ul>
Survivability	1.5	The Authority requires that a UOR for a mission data update <sup>†††</sup> , resulting from a new or changing threat, be available for operational use within <b>48 hours</b> , maximum.
	1.6	The Authority requires that mission data sets be routinely updated for each theatre of operation every <b>6 months</b> .
Interoperability	1.7	The Authority requires that the <b>system</b> shall accept and use updates, to mission and engineering data sets, within the timescales specified ( <b>TBD</b> ) by the data provider for UOR and routine changes.
Operations Support	1.8	The Authority requires that changes to data and software be incorporated into the <b>system</b> under normal support arrangements and without <b>adversely</b> impacting operations or routine training.

Table 4 – Supportability Performance Requirements

### 3.4 FUNCTIONAL SUPPORT REQUIREMENTS

Once refined and ratified, the performance requirements given in Table 4 will capture the level of service expected from the support system process. However, the derivation of these performance characteristics is not all that can be achieved during early LSA; support function decomposition is also possible. Functional requirement decomposition breaks down each of the functions identified within the support model and lists typical activities that might be performed within each function. The purpose of this decomposition is to provide greater visibility of the MOD's support expectations, facilitate future LSA, and assist initial supportability verification work through the creation of a task inventory against which reviews can be completed. It is not however, the intention of functional decomposition to impose a support solution or constrain support provider options. A support activity inventory distilled from the Support System Model is given in Appendix C, this provides a basic list of tasks that should be considered during further LSA; the list is not a finalised product and will require maturation as a project develops.

<sup>\*\*\*</sup> Safety-Critical CRs are those that result with the aircraft being grounded.

<sup>†††</sup> It is assumed that mission data changes will not affect system certification.

## SUMMARY

This section is focused on identifying the totality of software support from an operational perspective and specifically endeavours to identify support requirements that are typically implied rather than stated. The creation of an acceptable support model is considered to be the key to this work, as this model will be utilised throughout further LSA to assist with the production of consistent, complete and correct outputs.

In order to derive support requirements an understanding of the functions that constitute an adequate support system is necessary, this understanding has been achieved through analysis of both academic and military support models. Due to their differing perspectives, these models have been combined and refined to represent the totality of military software support. The derived Support System Model is supported by a document set covering the model rules, considerations and description. These documents are an integral element of the model as they help to gain confidence in its pedigree and discourage misunderstanding and incorrect interpretation.

Within the acquisition lifecycle support requirements are developed via a two-stage process. At first the User's need (URD) is determined, then system characteristics (SRD) necessary to meet the User's needs are defined. Due to the relationship between the URD and SRD the formulation of an appropriate and complete URD is a vital element of the procurement process.

It is unacceptable to allow operation systems to exist in a constant state of change as this prevents operators from becoming instinctive users. As such, support systems must not only enable change but control it as well. Performance support requirements need to be established that capture the level of service expected from the support system. Unfortunately, LSA alone cannot act as the authority by which performance support requirements are placed upon a system. This is because LSA focuses on data and analysis rather than discussion and agreement.

In addition to the documentation of performance support requirements, functional requirement decomposition is also possible at this time. The decomposition of Support System Model functions will enable the identification of typical support activities, which will provide greater visibility of the MOD's support expectations, facilitate future LSA and assist initial supportability verification work.

In addition to understanding the support need, LSA practitioners have to be capable of analysing support from a software perspective. This perspective change will require the transposition of traditionally hardware oriented guidance and techniques into the software domain. The following section will discuss the most significant hardware analysis guidance and techniques, and suggest ways in which they can be interpreted for software.

## **SECTION 4**

### **SUPPORTABILITY ACHIEVEMENT AND SUSTAINMENT**

#### **4.1 TRANSPOSITION OF ANALYSIS TECHNIQUES**

In the world of hardware engineering, component failure drives maintenance activities. Failure has two properties that need to be understood in order to derive an appropriate support solution, these being probability and severity. Standard analysis techniques used to explore these properties include Reliability and Maintainability (R&M) engineering and Failure Modes Effects and Criticality Analysis (FMECA). In addition to these techniques each item of hardware then undergoes Level of Repair Analysis (LORA) in order to ascertain the most cost effective support solution or maintenance policy. To transpose these disciplines into the software engineering domain the nature of each technique needs to be defined and an assessment of its suitability made. The definition<sup>21</sup> and assessment of each discipline is as follows:

- **R&M.**

*“Reliability – The ability of an item to perform a required function under stated conditions for a stated period of time.”*

*“Maintainability – The ability of an item under stated conditions of use, to be retained or restored to a specified condition when maintenance is performed by personnel having specified skill levels, using prescribed procedures and resources.”*

- **FMECA.**

*“An analysis to identify potential design weaknesses through systematic, documented consideration of the following:*

- *All likely ways in which a component or equipment can fail.*
- *Causes for each mode.*
- *The effects of each failure (which may be different for each mission phase).*
- *The criticality for each failure both for safety and for mission success.”*

- **LORA.**

*“A systematic procedure to determine the cost of alternative maintenance options, taking into account such variables as spares support, ground equipment and manpower costs.”*

When considering R&M for hardware, item failure and its relationship with reliability is the main support activity initiator. As discussed, software does not wear out, although it does fail, often resulting in the need for corrective maintenance. It has already been shown that

the need for corrective changes only accounts for approximately 20% of all software modification, as such it is both the need for corrective change and the other change drivers (Adaptive, Perfective and Enhancement changes) that need to be considered as support activity initiators.

FMECA can be carried out for software with the aim of influencing product design to improve reliability. However, because all software changes require product modification, a greater need exists to influence design to improve changeability. Whilst the use of FMECA might theoretically be possible, the use of this technique requires careful consideration for the following reasons:

- *“For systems that exhibit any degree of complexity, identifying all possible component failure modes - both singularly and in combination – becomes simply impossible.”<sup>22</sup>*
- *“Waiting for detailed design completion is too late, but starting fault analysis without a design can be intimidating. The Systems Engineer should understand how failure affects system behaviour before the design is completed. Appropriate mitigation can then be designed into the system via the requirements.”<sup>23</sup>*
- *“Detailed consideration of the failure effects caused by particular software faults is unlikely to be beneficial to the identification of support tasks.”<sup>24</sup>*

Recognising the limitations of FMECA, it is clear that analysis techniques, more appropriate and beneficial to the early procurement phases, are required. Such techniques include:

- Functional Failure Analysis (FFA) and Risk Analysis. FFA and Risk Analysis can be used to explore the causes and impact of support process failure. This will give an indication of support function criticality and facilitate the focusing of effort in these high value areas during latter LSA.
- Supportability Characteristic Identification. The early identification of supportability related product and process characteristics could provide a direct feed into the development process such that system design and support concepts can be influenced for improved changeability.

For software, the standard application of LORA is only applicable where equipment is removed or disassembled in order to carry out loading activities. This is because in this instance the need to load software can be treated as just another trigger that initiates a hardware maintenance activity. The use of LORA on software maintenance itself is not suitable for the following reasons:

- Software is not repaired, that is, it is not returned to its original supplied state. As such, software LORA needs to consider all of the drivers of software modification and the effort and resources consumed by them.

- Unless a diverse<sup>+++</sup> software backup has been developed, which is a highly improbable occurrence; there is no such thing as spare software – although it might be possible to regress to an older version of software of lesser capability. Hence the ability to utilise spares, as an offset against product reliability, is an invalid premise.

Therefore, software LORA, which would be better named Level of Service Analysis (LOSA), needs to consider the ability to modify and field software in time to sustain operational availability at an acceptable cost.

#### 4.2 RISK BASED APPROACH TO SOFTWARE SUPPORT

The spectrum of support that can be applied to system is wide ranging; all elements of software can be fully supported for each function within the Support System Model (see Figure 9), or in contrast, no support might be established. However, both of these options are too costly, one in a financial sense, the other in terms of capability sustainment. In short, the level of support must be balanced to meet the support need. In the absence of an existing method, I propose that the balancing of support should comprise of a simple two-stage process, the identification of Software Significant Support Items (SSSI), and a determination of SSSI support criticality. As the provision of software support can directly affect future capability, the selection process must be informed, justified and traceable, therefore I also recommend that proven techniques should be utilised to assist with this task.

Taking risk to be a function of probability and impact<sup>25</sup>, support risk can be assigned for each functional element of system software within each of the Support System Model functions. These risk assignments can then be used as a basis for SSSI selection. This risk-based approach is not dissimilar to the techniques used in Def Stan 00-56<sup>26</sup> for the management of system safety. To illustrate this point, an example risk definition table is given in Table 5. Risk characteristics for the considered support function are given under the Probability and Impact headings; where probability relates to the frequency of support function use and impact relates to the resultant effect on capability if the function fails. The significance of the support function is then derived by consideration of the risk criticality, as defined in Table 6.

---

<sup>+++</sup> A diverse software product is one that has been developed following an intentionally different design such that the probability of design related failure is reduced.

		Probability				
		Frequent	Probable	Occasional	Remote	Improbable
Impact	Severe	Level 4				
	Significant		Level 3			
	Marginal		Level 2			
	Negligible			Level 1		

**Table 5 – Support Risk Assignment**

Risk Criticality	Interpretation
4	Intolerable support risk.
3	Undesirable support risk, tolerable only if risk reduction is impracticable or if the costs are grossly disproportionate to the improvement gained.
2	Tolerable support risk if the cost of risk reduction would exceed the improvement gained.
1	Negligible support risk.

**Table 6 – Support Risk Definition**

For example purposes, use of Table 5 on a frequently modified item of mission critical Human Computer Interface software in the Software Modification function during peacetime, might be as follows:

- Inputs
  - Operational support profile = Peacetime (Routine Change).
  - Probability of support function use = Probable.
  - Potential capability impact if support function fails = Marginal.
  
- Outputs
  - Support risk = Unacceptable.
  - Support criticality = Level 3.

The results of this example show that not providing a software modification support function for this element of software would be unacceptable, and specifically that the support system provided must be commensurate with its associated criticality. However, without defining the impact and probability factors, the method can only serve as a guide as to how the identification of SSSIs could be carried out. Definition of these factors will require an appropriate system functional breakdown. For each functional element, an assessment of capability criticality needs to be made and rates of support function usage estimated.

### 4.3 NON-ECONOMIC LOSA

Once the support significant software items have been determined and their relative support criticality assessed, alternative support options can be suggested for the purpose of exploring the most effective and efficient support solution. In theory there are an infinite number of potential support option permutations, a fact that does not assist with the assessment alternatives. To limit the scope of analyses and assist in the production of meaningful outputs early in the procurement lifecycle, the following support options were considered during FOAS analysis:

- Industry Original Equipment Manufacturer (OEM). Industry (OEM) solely manages and provides the support function.
- Industry (3<sup>rd</sup> Party). The support function is solely and independently managed and provided by Industry other than the OEM.
- Service. The support function is solely managed and provided by a MOD organisation.
- Diverse. The support function is provided by a combination of Industry (OEM), Industry (3<sup>rd</sup> Party) or Service, each working independently to fulfil allocated tasks.
- Partnered. The support function is provided by a fully cooperative and integrated combination of Industry (OEM), Industry (3<sup>rd</sup> Party) or Service.
- No Support. No support function is established.

These categorized options were utilised as they accommodate all existing and proposed support scenarios and enable early analysis within minimal constraints. Of these six options, Industry (3<sup>rd</sup> Party), Partnered and No Support are in addition to the traditional options. These extra options were adopted as they enable beneficial analysis, as follows:

- Industry (3<sup>rd</sup> Party). Analysis of an Industry (3<sup>rd</sup> Party) option can simulate the need to establish software support if an Industry (OEM) based support solution fails. Usage of Industry (OEM) support is commonplace within the MOD. Failure of this support mode can arise when the MOD makes itself vulnerable to exploitation by having no alternative support option. In this situation the MOD must pay the support provider, irrespective of cost, if they wish to maintain capability.
- Partnered. Present MOD support doctrine is focused on migrating from Organic Support<sup>§§§</sup> to Contracting for Availability<sup>\*\*\*\*</sup>. As such, analysis of a partnered support option is necessary to explore the relative merits and trade-offs associated with this shift in support ethos.

---

<sup>§§§</sup> In which a MOD organisation is the sole provider of support.

<sup>\*\*\*\*</sup> In which support is provided by a MOD - Industry hybrid, or by Industry alone.

- No Support. Consideration of the No Support option will provide an indication of the potential affect of support function failure and allow each of the support options to be assessed with respect to a common baseline. This option is very appropriate for system components, such as embedded actuator software, that are well isolated from potential sources of change. The No Support option has further relevance for Commercial Off The Shelf (COTS) products, where the establishment of a bespoke support system might not be feasible.

During the Concept phase for FOAS a full effectiveness and efficiency assessment of these options could not be carried out due to a lack of project data. Notwithstanding this, a high-level qualitative assessment, based on a hypothetical Operation Flight Program (OFP)<sup>++++</sup>, was carried out to illustrate how the assessment technique can determine the feasibility of each support option. This qualitative assessment constitutes a non-economic LOSA by sifting viable from unviable support options. This makes the analysis process more efficient by enabling effort intensive cost related LOSA to be prioritised for the most favourable options.

The assessment technique, as applied in Appendix D, has been refined from work<sup>27</sup> carried out within AvSG to identify an appropriate support solution for the Harrier GR9. The refinement was necessary as the existing work approached this task from the perspective of reducing the cost of an existing product and support system, and considered only four supportability factors, these being:

- Design Authority. “[If the authority to make changes] *was not vested in either [the Original Equipment Manufacturer] or MOD then it was decided that the process of obtaining it would prove impracticable or financially prohibitive.*”
- Intellectual Property Rights (IPR). “*If the IPR of the SSSI was vested in an organisation other than [the Original Equipment Manufacturer] or MOD, then it was felt that the cost of obtaining it (if at all possible) for the purposes of software modification could be potentially prohibitive.*”
- Frequency of Change. “*If the software frequency of change was historically (and expected to be) greater than 5 years [between changes], then it was decided that LSA would not identify and improve support alternatives other than that already in existence.*”
- Safety Criticality. “*If the SSSI contained [safety critical software] it was agreed that this would fall outside the current software modification capabilities of [the Original Equipment Manufacturer] and MOD.*”

This approach is inappropriate for the creation of new support systems as it does not consider the full extent of supportability factors and focuses on support constraints rather than the capability driven need to implement software modifications. For new systems greater value can be achieved if products and processes are designed for supportability such that they satisfy the operational requirement.

---

<sup>++++</sup> A military aircraft's real time control application.

In overview, the refined technique assesses the relative advantages and disadvantages of an organisation’s ability to perform necessary support functions for each SSSI, and is based on an understanding of product and process characteristics that influence supportability. These characteristics, identified through consideration of the Support System Model, Capability Maturity Model<sup>28</sup>, systems engineering principles and FOAS LSA, are documented within Appendix E. Unfortunately, the assessment of support options is a subjective task, as a single organisation will not completely satisfy the support requirement for all functions of the Support System Model. To reduce the variance of this subjectivity and maintain an informed, justified and traceable approach, a defined set of assessment and trade-off criteria must be established.

The assessment output, given at Table 7, is reflective of current support capabilities, does not consider SSSI product characteristics and has no reference to support costs; which makes the analysis limited in its use and somewhat unattractive to IPTs. This limitation must not however be allowed to derail the analysis process, as I believe the real value of this work will be gained when quantitative LOSA is completed. Although not yet undertaken for the FOAS project, support cost estimation has been carried out for other projects within AvSG and, although in its infancy, this discipline has indicated its usefulness as a means for balancing support option selection for both appropriateness and affordability.

Provider Tasks	Support Option					
	Industry OEM	Industry 3 <sup>rd</sup> Party	Service	Diverse	Partnered	No Support
Software Operations	Unviable	Unviable	Favourable	Viable	Viable	Unviable
Query Evaluation	Viable	Viable	Favourable	Viable	Viable	Unviable
Change Management	Unviable	Unviable	Favourable	Viable	Favourable	Unviable
Software Modification	Viable	Unviable	Unviable	Viable	Viable	Unviable
Integrity Assurance	Viable	Viable	Unviable	Viable	Viable	Unviable
Data Support	Not Assessed	Not Assessed	Not Assessed	Not Assessed	Not Assessed	Unviable

Table 7 – OFP Qualitative Support Option Selection

To counter any challenges made on the worth of qualitative LOSA, I have presented an argument based on the premise that it is an invaluable stage of the support option selection process. Further to this, I also suggest that it provides an opportunity to discover situations where product or process characteristics could to be improved in order to reduce the risk of support function failure; and that the optimum time to change these characteristics is early in the life cycle, when the ability to influence product design and support planning is greatest.

#### 4.4 SUPPORTABILITY AND TECHNOLOGY

The use of new or emerging technologies is managed within the MOD through the application of a Technology Demonstrator Programme (TDP), as summarised at Appendix F. TDPs are used to de-risk projects through progressive assurance by

ensuring that new technologies are of sufficient maturity prior to their utilisation on any military equipment. The TDP is supported by a method for demonstrating technology maturity called the Technology Readiness Level (TRL). Throughout the TDP process significant consideration is given to system support in recognition of the potential impact this can have on whole life costs. For each TDP tender, Industry will respond with both the technology analysis and a means of demonstrating TRL achievement; TDPs are finally assessed through the use of a Tender Assessment Matrix, where a significant weighting is applied to supportability.

For the FOAS project, whilst the TDP and associated TRL assessment processes provide a robust approach to the exploration and risk management of new technologies, I am concerned that the specific requirements of software supportability will not be addressed. I base this concern on the fact that it is expected that software support will be addressed as part of general support analysis, and yet software supportability has not been explicitly stated as a primary enabler of capability sustainment. Further to this, in the software domain, product maturity is more closely related to obsolescence than stability, as such the need to de-risk projects should be satisfied by a more appropriate method; which will be discussed under the heading of Supportability Validation And Verification.

During the completion of LSA Task 204 – Technological Opportunities, a study of the Joint Strike Fighter<sup>\*\*\*\*</sup> (JSF) was carried out to gain insight into software related technologies already being utilised by new projects, these technologies include<sup>29</sup>:

- Auto Coding. Reducing the amount of manual coding.
- Commercial Off The Shelf (COTS) Software. Reducing the amount of development.
- Reuse. Reducing the amount of development.
- Multiuse. Using JSF developed code in several areas, both on and off aircraft, with the aim of reducing development and the total cost of ownership.

Whilst it is commonly accepted that these technologies have the potential to reduce the initial development effort and cost, I have not been able to identify any analysis showing consideration of these technologies with respect to supportability benefit. On further investigation, many technologies, such as those documented in Appendix G, claim to have the potential to improve software supportability through either product or process improvements. Yet again no measure or index of these potential improvements could be found. This lack of evidence, either academic or empirical, makes justification of technology utilisation problematic, as this should consider both the cost of implementation and potential impact on whole life savings.

In an effort to at least partially solve this situation, I have ranked the potential benefits of the technologies listed in Appendix G. This work, shown in Appendix H, was produced by assessing each technology for its ability to limit the occurrence or impact of system change drivers (see Figure 4). The relative benefit of software technologies shown in Table 8

---

<sup>\*\*\*\*</sup> American developed aircraft being utilised by the MOD as a replacement for the Harrier in 2012.

lacks credibility as it is based on just one assessment method, as such it can only be used to identify issues for further consideration. However, the results have already caused discussion, as they directly challenge MOD policy on the aggressive use of COTS products, which according to my analysis will only improve supportability by providing some isolation from adaptive changes. Further studies on the role of technology within supportability could add great value to this work by providing alternative assessment methods and outputs; which could then be used to verify results and improve credibility. I suggest that this could be taken forward by consideration of how technologies affect the distribution of effort, and relative cost of problem correction, within the maintenance lifecycle.

Technology Opportunity	Benefit Ranking (Sum change driver influence).
Redefinition and refinement of the support infrastructure and processes.	97%
Utilisation of reuse and multiuse.	97%
Application of standards for development and support.	92%
Process Improvement.	92%
Organisationally defined and standardised CASE tools.	92%
Organisationally defined and standardised software management tools.	92%
Organisationally defined and standardised Software Engineering Environment (SEE).	92%
Exploitation of automatic code generation.	87%
Application of Object Orientation (OO).	65%
Utilisation of Open Standards.	65%
Standardised approach to mission data formatting and data loading.	51%
Utilisation of Integrated Modular Avionics (IMA).	24%
Utilisation of Commercial Off The Shelf (COTS) software.	24%

Table 8 – Potential Benefits of Software Technology

#### 4.5 SUPPORTABILITY PRESERVATION

The FOAS Support Strategy<sup>30</sup> identifies the need for capability provision and sustainment with optimised WLC. In software engineering terms, sustainment equates to the fulfilment of one basic function, the ability to continually implement and field modifications within required timescales on an affordable basis. When software is modified the originating change driver should be satisfied or no value will have been added. However, in addition to this, functionality not related to the modification must be maintained and original system quality characteristics preserved, examples of these characteristics are as follows:

- Safety Certification. For safety related systems a significant amount of effort will have been invested in obtaining the necessary system certification. In large and complex systems the relationships and dependencies between system elements can play a significant role in the mitigation of safety related functions. The implication of any software modification must be predictable if system safety integrity is to be maintainable.
- Security Accreditation. The data classification requirements for military systems range from Unclassified to Top Secret; as a result of this the systems themselves attract an appropriate security classification. The implementation of any software modification must not breach system security integrity else platform capability might be compromised in theatre.
- Reliability. Reliability from a software perspective is different to hardware in that it cannot be measured as the Mean Time Between Failure (MTBF). This is because when a software error is corrected it is not restored back to its original supplied state; it is in fact modified and a new product is created. The essence of reliability does however hold, in that it captures the characteristic of how often failure is experienced by the user. Software modification should not degrade the user perceived reliability of the system by implementing the desired change incorrectly or introducing errors.

Supportability. To produce maintainable software that can be modified in a timely manner and at an affordable cost, significant effort must be invested during its development. If software modification is carried out without consideration of this non-functional characteristic, the ability to continually implement modifications will be eroded. Once the ability to support software is lost the software investment is no longer protected and all of its stakeholders become exposed to the risk of a capability gap.

In light of these factors it is evident that it is not enough to merely make changes, we need to be able to do so in a supportable manner. Hence, I suggest that supportability must focus on three factors:

- Product characteristics.
- Process characteristics.
- Characteristics preservation.

## SUMMARY

Within this section LSA guidance and techniques have been transposed into the software domain. The purpose of this work is to bridge some of the voids and misunderstandings that exist in this area.

In the world of hardware engineering disciplines such as Reliability and Maintainability (R&M) engineering, Failure Modes Effects and Criticality Analysis (FMECA) and Level of Repair Analysis (LORA) are used to derive appropriate support solutions. Unfortunately

these disciplines are not directly applicable to software and require transposition prior to their use.

When considering R&M for software, it is the need for corrective, adaptive, perfective and enhancement changes that need to be considered as support activity initiators. The application of FMECA to software is very effort intensive and provides few valued outputs for supportability engineering, as such analysis techniques such as Functional Failure Analysis and supportability characteristic identification are required. Unlike hardware, software is not repaired, that is, it is not returned to its original supplied state. As such, software LORA needs to consider all of the drivers of software modification and the effort and resources consumed by them. Software LORA, which would be better named Level of Service Analysis (LOSA), needs to consider the ability to modify and field software in time to sustain operational availability at an acceptable cost.

To achieve efficiency, the level of support established must be balanced to meet the support need. However, at present, an informed, justified and traceable means to achieve this balance does not exist. To resolve this situation a process is required that identifies Software Significant Support Items (SSSI) and determines their operational criticality. Processes similar to this are already used by many risk-based management disciplines and, common to all of them, the appropriate definition of impact and probability factors is required.

In theory there are an infinite number of potential support option permutations, a fact that does not assist with the assessment alternatives. To overcome this situation support options have been categorized into six basic types. These categorized options accommodate all existing and proposed support scenarios and enable early analysis within minimal constraints.

Although limited in its use as a method for determining the most appropriate support solution, non-economic or qualitative LOSA is a fundamental stage of the support option selection process and provides an opportunity to discover situations where product or process characteristics can be improved in order to reduce support related risk.

The use of new or emerging technologies is de-risked within the MOD through the application of a Technology Demonstrator Programme (TDP). Whilst the TDP provides a robust approach to the exploration and risk management of new technologies, it does not explicitly recognise the role of software support as a primary enabler of capability sustainment.

Whilst it is commonly accepted that many technologies have the potential to reduce the initial software development effort and cost, there is little or no analysis showing consideration of these technologies with respect to supportability benefit. This lack of analysis, either academic or empirical, makes justification of technology utilisation problematic, as this should consider both the cost of implementation and potential impact on whole life cost. In an effort to partially solve this situation, numerous technologies have been ranked according to its ability to limit the occurrence or impact of system change drivers. Unfortunately this analysis lacks credibility as it is based on just one assessment method.

In software engineering terms, capability sustainment equates to the ability to continually implement and field modifications within required timescales on an affordable basis. When software is modified the originating change driver should be satisfied or no value will have been added. However, in addition to this, functionality not related to the modification must be maintained and original system quality characteristics preserved. Hence, supportability must focus on three factors: product characteristics, process characteristics and characteristics preservation.

LSA will only be of benefit to a project if its outputs are allowed to influence system design for supportability. Assuming adequate channels have been established to facilitate design influence, the remaining responsibility of the LSA practitioner is to progressively assure the achievement of desired system supportability characteristics. The following section will discuss how progressive assurance can be initiated at the very outset of a project.

## **SECTION 5**

### **SUPPORTABILITY VALIDATION AND VERIFICATION**

#### **5.1 SUPPORTABILITY ACHIEVEMENT THROUGH PROGRESSIVE ASSURANCE**

Having previously argued that the ability to influence product design and support planning is only possible if analysis is carried out in a timely manner. I now suggest that this idea should be extended to include the early resolution of supportability product and process deficiencies. In terms of corrective action costs, the most effective time to correct faults within a project, both product and process related, is during the requirements definition phase. In fact Boehm<sup>9</sup> has stated that a fault discovered during the requirements definition phase is approximately one thousand times cheaper to correct than if corrected during the in-service phase.

Hopefully it is now evident that the development of a supportable system is dependent upon many interrelated characteristics. Supportability is an emergent property and cannot be manufactured as an adjunct to any system; as such it can only be achieved by designing for supportability. With consideration of these factors supportability verification and validation, just like any other form of testing, needs to be progressively assured from the very outset of the development process.

Now that the need to test and indeed test early has been established, the first transposition of testing into the supportability domain is required. This transposition will make visible some of the limitations relating to this subject, which will in turn assist with their management; the limitations are as follows:

- The effectiveness of early testing will be influenced by the ability to predict the eventual operational environment and use profile, for instance it might not be possible to accurately predict operational software change triggers. Early supportability testing, although beneficial, will therefore be limited in its application and usefulness. As the system matures any test case assumptions and their associated analysis should be revisited to ensure they remain valid.
- Due to the almost infinite permutations of support factors, testing can never be absolute. To balance the need to demonstrate support solution acceptability with the limitations created by time and resource constraints, the use of testing assumptions and hypotheses will be required. As the system matures these assumptions and hypotheses should be revisited to ensure they remain valid.
- Testing is carried out to find yet undiscovered faults and relies upon test cases that have a high probability of finding these faults. As such, the test cases themselves should be reviewed by differing supportability stakeholders to ensure completeness of coverage and correctness.
- The true output of testing is an indication of the existence of faults, this is best summarised by the following Pressman<sup>31</sup> statement:

*“Testing cannot show the absence of defects, it can only show that errors are present”<sup>§§§§</sup>.*

Residual risk, originating from undiscovered faults either in the analysis, assumptions or implementation, will always be present and as such will require management throughout the life of the system.

- It is very easy to overlook or discount testing until products and processes become established. This can often result in the late or rushed formulation of test objectives, or worse, it might become evident that some critical requirements are in fact not testable. To avoid this situation, test objectives or at least a testing approach should be developed as soon as a requirement is identified and accepted.

## 5.2 TEST AND EVALUATION STRATEGY

Complementary to the concept of progressive assurance, an initial iteration of LSA Task 501 – Supportability Test, Evaluation and Verification was carried out for the FOAS project. According to the MOD’s own guidance<sup>32</sup>, this task is carried out to:

*“Validate the support solution adopted, once the system/equipment enters service (500 series tasks).”*

Unfortunately, this statement leads many people to believe that LSA Task 501 can only be undertaken once the system or equipment being procured enters into operational use. This however is not the case and, as already discussed, leaving the implementation of this task until that time can have a detrimental effect. To combat the numerous challenges made on the appropriateness of this task early in the procurement lifecycle, the first activity undertaken was to define the scope and objectives of work to be completed at this time, as follows:

- Scope. This task covers the documentation of supportability assessment strategy considerations and objectives. Without a defined product or support solution, considerations and objectives will be established that embrace best practice and principles which will remain valid irrespective of system implementation. These considerations and objectives will then be used to formulate supportability assessment criteria as the project matures.
- Objectives. The principle objective of this task is to raise awareness of issues relating to software supportability assessment. A secondary objective of this task is to provide early visibility of the MOD’s supportability assessment intentions, such that product design and support planning can be influenced to assist supportability demonstration.

In line with Def Stan 00-60<sup>24</sup>, strategy considerations and objectives include coverage of both software operation and software support. To give the analysis consistency and

---

<sup>§§§§</sup> Although not referenced by Pressman, this quote is similar to Dijkstra’s: “*Program testing can be used to show the presence of bugs, but never to show their absence!*” (Ref: “Notes on Structured Programming”, E. Dijkstra, T.H. – Report 70-WSK-30, 2<sup>nd</sup> Ed, p. EWD249-7, 1970).

enhance traceability, the implementation of Task 501 was undertaken with the continued application of the Support System Model (see Figure 9).

### 5.3 SUPPORTABILITY ASSESSMENT AND MANAGEMENT TECHNIQUES

Guidance within Def Stan 00-60 on the formulation of a test and evaluation strategy of Task 501 is as follows:

- *“Strategies for the evaluation of system supportability should include coverage of software operation and software support. Direct measurements and observations may be used to verify that all operation and support activities (that do not involve design change) may be completed using the resources that have been allocated. During the design and implementation stage measurements may be conducted on similar systems, under representative conditions.”*
- *“As software modification activity is broadly similar to software development the same monitoring mechanism might be used both pre- and post-implementation. Such a mechanism is most likely to be based on a metrics programme that provides information, among other things, on the rate at which software changes are requested and on software productivity.”*

Having worked with the FOAS IPTs on the formulation of a Software Strategy, I find this guidance is inappropriate and inadequate; inappropriate in that it encourages supportability assessment deferment, and inadequate in that it offers no information on how early confidence can be established that supportability is being adequately considered and managed. To resolve this situation, and provide a foundation on which an informed strategy can be developed, my first action was to develop this guidance and provide useful information on the application of Task 501 to software supportability.

Throughout development, confidence in the achievement of supportability characteristics can be established and maintained through the use of techniques such as: conformance review, conformance demonstration, development monitoring and maturity assessment; as follows:

- Customer Product Conformance Reviews. Periodical or event driven technical reviews carried out to assess the degree to which products and processes exhibit supportability characteristics. Typically these reviews are carried out by the Customer or their nominated representative and are held at the developer’s premises.
- Supplier Product Conformance Demonstration. Periodical or event driven presentation of evidence used to demonstrate the degree to which products and processes exhibit supportability characteristics. Typically the Supplier demonstrates supportability characteristics for an item or area selected by the Customer.
- Customer Development Representation. The placement of a Customer’s representative in the development environment throughout product and support process development. The representative can either be, invited to,

or have the right to, access all areas of development and their associated technical or review meetings. Confidence is gained through the continual monitoring of implementation decisions and milestone achievement.

- Product and Process Development Maturity. The selection of products, support processes or suppliers based on their maturity may improve confidence for the following reasons:
  - Reused software products and processes may already exhibit supportability characteristics without need for further work or demonstration.
  - Organisations may have already proved themselves to be reliable providers of supportable products and support processes.

Each of these techniques exhibit differing strengths and weaknesses, a factor that needs to be considered during their inclusion into any supportability assessment strategy. An overview of these strengths and weaknesses is given in Table 9. From this, it is evident that no single technique is sufficient, and a hybrid of techniques will be required in order to achieve an acceptable level of confidence at optimal cost.

Technique	Strength	Weakness
Customer Product Conformance Reviews	Can provide a very high degree of confidence.	Very effort consuming process.
		Requires very specialised skills that might not be freely available to the Customer.
Supplier Product Conformance Demonstration	The Customer does not own the burden of evidence formulation and presentation.	The Supplier is afforded an opportunity to hide problem areas.
Customer Development Representation	Provides day-by-day feedback of supportability achievement rather than a potentially artificial snapshot.	The Customer needs to permanently commit resources.
		Problem issues can be discussed under the cover of unrelated projects.
Product and Process Maturity	Minimal cost to the Customer.	Heavy reliance upon historical performance. This might not be appropriate if the project or application under development utilises new or evolving principles or technologies.
	Maturity assessment can provide good confidence in an organisation's ability to consistently deliver products and services.	

Table 9 – Method Strengths and Weaknesses

As previously mentioned, supportability is an emergent property dependent upon many interrelated factors. The management of supportability is therefore not a simple task. Standards such as SAE-JA-1004<sup>33</sup> suggest the use of a software supportability case as a means by which suppliers demonstrate that customer supportability requirements have been satisfied. Such a case presents a readable overview of evidence, including references to more detailed evidence as appropriate to justify supportability claims. This idea is very similar to the safety case used in the world of safety engineering to assist with the establishment and through life management of system safety integrity. Therefore, I suggest that tools more traditionally related to the safety domain such as Claim-Argument-Evidence and Goal Structured Notation (GSN) are equally applicable to supportability engineering, and should be considered as a means of presenting and managing a supportability case.

## 5.4 SUPPORTABILITY ASSESSMENT OBJECTIVES AND CRITERIA

Assuming that the fulfilment of supportability objectives will build towards the attainment of supportable products and processes, objectives need to be established that are meaningful to those allocated responsibility for their achievement. To assist with their achievement, each objective should be assigned a set of criteria by which objective satisfaction can be determined. For the FOAS project, even though the support requirements have not yet been accepted, work was still undertaken in this area. This work endeavoured to standardise and improve the quality of objectives through the definition of 'SMART' characteristics<sup>34</sup> that each objective should exhibit:

- Specific.  
*"The objective should provide a precise description of expected outcome."*
- Measurable.  
*"It should be possible to determine when the objective has been achieved. Some means of making this judgment should be established."*
- Assigned.  
*"Responsibility for objective achievement should be assigned and agreed."*
- Realistic.  
*"Expected outcomes should be realistic for the assigned resources."*
- Time-Related.  
*"Objectives should be allocated a completion date and time."*

During investigation into the most suitable definition of 'SMART' characteristics numerous variations were encountered. The definition given above was selected on grounds that it best represents sentiment contained within the Capability Maturity Model<sup>28</sup> Common Features and Level 2 Key Process Areas. Of these SMART characteristics, it is expected that measurement will cause the greatest problem, especially during procurement, when many of the things we are trying to value are not tangible. Taking reference from Fenton and Pfleeger<sup>35</sup>, I believe that the MOD needs to raise awareness and understanding of the factors associated with measurement, and provide guidance on the ways in which products and processes can be measured or assessed for their appropriateness.

*"The gap between how we do measure and how we could measure is still larger than it should be. A key reason for this gap between potential and practice has been the lack of a coordinated, comprehensive framework for understanding and using measurement."*

In summary of their work, two distinct forms of measurement can be taken in two differing ways, these being:

- Direct Measurement.

*“Direct measurement of an attribute of an entity involves no other attribute or entity.”*

For example, the recording of team size, where team size is the count of personnel assigned to a given task, is an example of a direct measurement.

- Indirect Measurement.

*“...where we take [Direct Measurements] and combine [calculate] them into a quantified item that reflects some attribute whose value we are trying to understand.”*

For example, the recording of expended effort, where effort is the product of utilised personnel and hours expended on a task, is an example of an indirect measurement.

- Objective<sup>\*\*\*\*</sup> Measurement.

*“When measuring attributes, we strive to keep our measurements objective. By doing so we make sure that different people produce the same measures, regardless of whether they are measuring product, process or resource.”*

For example, a measurement of product size, where size is determined through a count of the source lines of code, is an example of an objective measurement.

- Subjective<sup>++++</sup> Management.

*“[Subjective measures] depend on the environment in which they are made. The measures can vary with the person measuring, and they reflect the judgement of the measurer. What one judge considers bad, another may consider good, and it may be difficult to reach consensus on attributes such as process, product or resource quality.”*

For example, a measurement of product usability, where usability is determined through a number of normalised individual assessments, is an example of a subjective measurement.

Based on experience gained through Software Capability Evaluations<sup>36</sup> and procurement reviews, it is very evident that the collection of metrics within the MOD can be inefficient and ineffective. Often measurements are taken without further analysis or intent for future use; instead remaining dormant within project documentation. Unfortunately, even if the will was present, these metrics offer no value to the MOD, as they often capture incomplete or irrelevant data sets and do not facilitate the creation of meaningful or valued

---

<sup>\*\*\*\*</sup> Objective – “not influenced by personal feelings or opinions in considering or representing facts”

<sup>++++</sup> Subjective – “based or influenced by personal feelings, tastes, or opinions”

“The New Oxford Dictionary of English”, Clarendon Press, Oxford, 1998.

outputs. On further investigation, I have found one root cause that is common to many projects, this being the lack of a well defined approach to metrics collection as part of a balanced and managed metrics programme; a problem itself resulting from a lack of organisational commitment and well defined business goals. From this work it has become clear to the author that metric collection needs to be embraced by the MOD as a vehicle for understanding and improvement. Appropriate business goals need to be established and necessary resources committed, as the enabler for all other improvement activities. Guidance also needs to be established on metric program implementation; covering the differing types and forms of measurement, and embracing best practice such as the attributes listed in Table 10<sup>37</sup>.

Metric Attribute	Attribute Description
Appropriate	Metrics must support the analysis need.
Clear	The intended use of all metrics must be understood and well defined.
Faithful	Metrics must reflect the real situation; they must not be marred by the fact that metrics are being taken.
Accurate	Metrics must truly represent the item being measured. The use of automated tools should be considered to reduce errors.
Consistent	Metrics must be repeatable irrespective of the time of measurement or personnel making the measurement. The use of automated tools should be considered to improve consistency.
Controlled	The collection burden must be managed; metrics must not be collected at random.
Sustained	The commitment to metric collection must persist for the whole period of analysis significance (for some metrics this might be the life of the system). Metric collection commitment, ability and motivation must be established and sustained.
Traceable	To facilitate metric program management and improvement, traceability from analysis need to metric collection must be established and sustained.

Table 10 – Metric Attributes

## 5.5 SUPPORTABILITY INTENT VERIFICATION

The validation of system supportability characteristics can be achieved through a supportability demonstration, which is effectively a validation of the User's performance supportability requirements. For the FOAS project, these requirements have not yet been ratified, and until this is carried out their abstraction into objectives is not possible. When these requirements have been accepted, objectives will need to be established at the earliest opportunity. These objectives should address the need to validate that support activities can be completed using allocated resources within defined periods of time.

The complete validation of system supportability can only be carried out very late in the development lifecycle and will in fact evaluate both product and process characteristics. This situation is undesirable as it delays the discovery of any problems until the ability to rectify them becomes impracticable or cost prohibitive. To address this limitation, timely verification of software products and processes needs to be carried out.

Early in a project's lifecycle the availability of measurable outputs is often limited or non-existent, hence there is no direct way of gaining confidence that supportability is being addressed. Where this is the case, process evaluation can be used to make an assessment of supportability intent. I suggest examples of evidence that might be sought, or items that can be measured to gain early supportability confidence are as follows:

- Explicit action has been taken to ensure the mutual understanding of supportability requirements prior to any contractual commitment.
- Estimates for schedules, resources and cost have been produced. Independent reviews of estimates have been carried out for comprehensiveness and realism. Checks are made for actual usage against original estimates.
- Effort has been expended, funds have been expended, resources have been assigned and training requirements have been identified and satisfied.
- Plans have been produced; progress and milestone achievement is tracked.
- A corrective action system is established; supportability related items have been raised and close within the corrective action system.
- Regular reviews of supportability risks, assumptions and analyses are carried out.
- A means of incentivising supportability improvement is established.
- A supportability preservation strategy has been established and implemented.

These support intent assessment criteria, are again based on the Common Features of the Capability Maturity Model<sup>28</sup>. For reference purposes definitions of the Common Features, named as follows: Commitment to Perform, Ability to Perform, Activities Performed, Measurement and Analysis, and Verifying Implementation, are given at Appendix I.

## 5.6 PRODUCT ASSESSMENT

The timely demonstration of product supportability characteristics is a technically challenging discipline. This is because the goal of developing a supportable system is at a significantly higher level of abstraction than the individual items of evidence that will contribute towards its achievement. Even with detailed information on the software products involved, the specification of supportability assessment objectives and criteria requires a good understanding of the project and specialist technical competencies.

As previously discussed, one approach capable of providing the necessary framework to manage the gap between system goals and supporting evidence is the use of a supportability case, perhaps utilising Claim-Argument-Evidence or Goal Structured Notation (GSN). Early development of a supportability argument will not only identify the individual items of evidence required to justify any supportability claims, but also their roles and relationships within that system. Once the necessary body of evidence has been identified, objectives can be placed on these items, such that their timely achievement can be managed. The characteristics identified in Appendix E could provide the initial basis for supportability argument formulation, adding extra value to the LSA already carried out for the FOAS project. Once products are in development, it is possible to carry out reviews to determine the existence of supportability characteristics. The AFOTEC - Software Maintainability Evaluation Guide<sup>38</sup> provides an example of this type of review.

Although on the fringe of software supportability, the system hardware's ability to accommodate expected software growth also needs to be managed. Demonstration that this factor has been addressed can be assisted by the availability of system growth analyses and evidence of system design influence. However the formulation of hardware objectives in this area is beyond the scope of this dissertation.

## 5.7 PROCESS ASSESSMENT

The verification of software support processes is a relatively immature discipline, although some work has been carried out to verify operational support characteristics through the demonstration of software loading activities. Generally however, verification does not cover the full breadth of support functions as identified in the Support System Model. Recognising this failing, I believed that a transposition of mature software testing techniques into the supportability domain was necessary to assist with the informed establishment of objectives and criteria that are as robust and complete as possible; a copy of this work has been included at Appendix J.

Without detailed information on FOAS's intended support system it is only possible to specify areas for which objectives and criteria should be established. Based on the work carried out in Appendix J, objectives and criteria should be established that:

- Define the entry and exit criteria that will control how products will move through the support system.
- Promote support function testing for: individual support functions, parts of the support system and the support system as a whole (where inputs and outputs are unavailable they should be simulated to facilitate early assessment).
- Verify the establishment and effectiveness of supportability functions and sub functions. Utilise the CMM Common Features to enable an early assessment of supportability intent.
- Test how products will be passed from one support function to another.
- Explore how the system will degrade in the event of extreme circumstances, by testing for tolerance of:
  - Rates of demand in excess of defined maximums.
  - Demands are made beyond the scope of each function.
  - Demands are made beyond the capability of each function.

## **SUMMARY**

This section has recognised the need to progressively assure the achievement of desired system supportability characteristics. LSA Task 501, which runs in parallel to those previously discussed, is a vital element of the LSA process, as it is the foundation on which the MOD ensures that software enabled capability can be sustained throughout the life of a project.

In terms of corrective action costs, the most effective time to correct faults within a project, both product and process related, is during the requirements definition phase. Supportability is an emergent property and can only be achieved by designing for supportability. As such, supportability verification and validation, just like any other form of testing, needs to be progressively assured from the very outset of the development process.

To make visible some of the limitations relating to this subject and assist with their management, transposition of testing into the supportability domain is required. This transposition will consider ideas such as: the effectiveness of early testing, the completeness of testing and role of hypotheses, residual risk originating from undiscovered faults, and the need to conduct testing in a timely and robust manner.

Unfortunately, the MOD's own guidance leads many people to believe that LSA Task 501 can only be undertaken once the system or equipment being procured enters into operational use. This however is not the case and leaving the implementation of this task until that time can have a detrimental effect.

Throughout development, confidence in the achievement of supportability characteristics can be established and maintained through the use of techniques such as: conformance review, conformance demonstration, development monitoring and maturity assessment. Each of these techniques exhibit differing strengths and weaknesses which need to be considered prior to their inclusion into any supportability assessment strategy.

The management of supportability is not a simple task. Standards such as SAE-JA-1004 suggest the use of a software supportability case as a means by which suppliers demonstrate that customer supportability requirements have been satisfied. This idea is very similar to the safety case used in the world of safety engineering to assist with the establishment and through-life management of system safety integrity. Further to this, tools more traditionally related to the safety domain such as Claim-Argument-Evidence and Goal Structured Notation (GSN) could prove equally applicable to supportability engineering, and should be considered as a means of presenting and managing a supportability case.

Investigation has shown that supportability objectives need to be established that are meaningful to those allocated responsibility for their achievement. Achievement of this goal can be assisted through the use of 'SMART' characteristics; of which, it is expected that measurement will cause the greatest problem. To assist with the resolution of this situation, the MOD needs to improve understanding of the factors associated with measurement and provide guidance on the way it expects products and processes to be measured and assessed for appropriateness.

The complete validation of system supportability can only be carried out very late in the development lifecycle. To address this limitation, timely verification of software products and processes needs to be carried out. Reviewing development products and process for evidence of supportability intent can assist with the completion of this task. Suitable items of evidence can be identified through a second transposition of testing into the supportability domain.

Because the goal of developing a supportable system is at a significantly higher level of abstraction than the individual items of evidence that will contribute towards its achievement, the timely demonstration of product supportability characteristics is a technically challenging discipline. Once development begins, it is possible to carry out reviews to determine the existence of product supportability characteristics. The AFOTEC - Software Maintainability Evaluation Guide provides an example of this type of review.

Taking reference from the Support System Model and the testing domain, process objectives and criteria should be established that: define entry and exit criteria, promote Top Down and Bottom Up testing, assess supportability intent, cover the passage of products around the support system, and explore how the system will react in the event of extreme circumstances.

## **SECTION 6**

### **DISCUSSION**

#### **6.1 REFLECTION**

As stated in the project proposal, this dissertation presents a case study with the aim of maturing the LSA process by:

- Stating the current failings of Def Stan 00-60, both general and software related.
- Defining the nature of software supportability analysis and transposition of LSA into the software domain.
- Assessing the practicalities of LSA scope and depth when analysis is carried out by an organisation external to a Project Team.
- Evaluating existing guidance on LSA for software and suggesting areas for improvement.
- Suggesting new analysis methods that reach beyond the scope of existing LSA guidance.

Through the persecution of these aims, which I believe are fully satisfied by the dissertation, a large range of topics have been explored for the purpose of initiating process maturation. Within these topics supportability discussions have been developed resulting in the identification of some very significant software supportability issues. Specific examples of these issues worthy of summarisation include:

- The ability to influence product design and support planning is only possible if LSA is carried out in a timely manner. Unfortunately, the application of Software Support Analysis (SSA), as described in Def Stan 00-60, is reactive. This means that the rigid application of Def Stan 00-60 early in the CADMID acquisition lifecycle is inappropriate.
- Timely LSA can be carried out by focusing on functional analysis, where functional analysis only considers the roles and services that a system is required to provide, and specifically does not concern itself with how these roles or services will be provided. In this way all analyses should remain valid regardless of eventual system implementation.
- By allowing its software LSA capability to erode, the MOD has placed itself in a position of vulnerability through support exploitation. Should the MOD no longer understand the software LSA process, the Integrated Logistics Support discipline will become exposed to the risk of failure.

- To facilitate understanding of the support need, projects must recognise the potential sources of change and how often might they be encountered. Support planning must accommodate the fact that software modification is driven by corrective, adaptive, perfective and enhancement changes.
- Within the software domain, product maturity is more closely related to obsolescence than stability, as such the need to de-risk projects should be satisfied by a method that progressively assures the achievement of supportability characteristics.

The dissertation has been influenced by many referenced sources, however one source of influence, the Capability Maturity Model (CMM)<sup>28</sup>, has been of particular use. I believe that the reason behind the CMM's usefulness is that like LSA it is process based. As such, initiatives in the CMM aimed at improving the software modification processes transpose with relative ease into the supportability domain. In particular, they offer approaches to assist with the notably unguided task of supportability test, evaluation and verification (LSA Task 501).

Within the dissertation refinement of the Support System Model (see Figure 9) has been a key enabler for other LSA activities. The resultant model and its accompanying documentation set has proven itself to be a major factor in establishing completeness and correctness of analysis, which has significantly assisted in the production of outputs which are informed, justified and traceable.

## 6.2 CRITIQUE

Throughout the completion of LSA for the FOAS project numerous challenges have been received on the usefulness of early and essentially generic analysis outputs, as such these challenges are equally valid to this dissertation. On each occasion these challenges have been successfully countered by referral to two basic premises, these being:

- The LSA presented here does not aim to present finalised solutions; instead it is the first pass through of an iterative process. As a first pass, it aims to capture the fullest range of factors relating to software support such that over time these factors can be explored for their significance and acted upon accordingly as a project matures. The value of this work comes from its completeness and not its depth, as omission at this stage could result in significant voids in the analysis process, potentially resulting in the development on an unacceptable or unaffordable support solution.
- Each of the factors presented within the dissertation is supported by academic reference. This approach is not by accident and has been taken in an attempt to establish dissertation pedigree in excess of discipline maturity. Whilst it is accepted that the transposition and utilisation of academic outputs is not infallible, the ideas presented are based on sound and established engineering principles that will always remain valid.

The dissertation structure is complementary to the LSA task framework presented in Def Stan 00-60. This approach results in a document containing many separate but interdependent sections. However, this approach is deemed necessary as it aids read across to the Def Stan and improves its usefulness to the LSA practitioner.

On reflection, I believe there is one area of the dissertation that, although academically sound, will struggle to prove its worth during application. This area relates to the identification of Software Support Significant Items (SSSI). My concern is based on the fact that the proposed method is reliant upon the definition of support impact and probability factors, where definition of these factors is reliant upon an appropriate breakdown of system functions and an associated assessment of capability criticality and estimated rates of support function usage. Historically, the MOD has not invested in the collection of data that will enable this method and unless there is a radical change in culture, project teams will struggle in the completion of this task. However, if the MOD truly wants to develop support systems that satisfy the support need at an acceptable cost, this change in culture must occur.

Having met the stated aims of the dissertation proposal, I have reviewed the Motivation statement made in Section 1, part of which reads:

*“In addition to these stated needs, I believe that [the software LSA guidance work<sup>1</sup> produced by AvSG] needs to be validated as [it] is mainly based on a desktop review of LSA related standards and literature.”*

In general, I am satisfied that the validation exercise has been carried out within the dissertation as a whole but accept that this is not particularly evident to the casual reader as this activity has not been completed in an explicit manner. The reason for this approach is that LSA comprises of a generic process, which is then tailored by project specific data to fit the characteristics of the system being procured. As the focus of this dissertation is the maturation of the LSA process, it is this theme that has to take precedence. However, project data and support performance issues are not ignored by the dissertation even though their relevance to specific LSA tasks is not stated.

During the completion of FOAS LSA, I attended a meeting to determine the support analysis that will be undertaken to identify the support requirements for a new mission management system. This system will lie at the heart of ground-based mission planning activities both pre and during operational sorties. During this meeting, the concepts considered in this dissertation were presented to a 3<sup>rd</sup> party Contractor brought in to conduct LSA on behalf of the Supplier developing the equipment being procured. This meeting was very successful, with both the 3<sup>rd</sup> party Contractor and Supplier expressing a great deal of interest in my work. As a direct result of this meeting support analysis intentions were refocused to meet the Customer’s need and prevent the expenditure of inappropriate LSA effort.

### 6.3 FUTURE WORK

Whilst this dissertation has endeavoured to mature the software LSA process, the matter discussed is very much the beginning of this venture and will require application, review and update before a robust and proven output is established.

Throughout the dissertation suggestions for future work have been made relating to the improvement of LSA. However, additional suggestions have been made that will generally develop the discipline of supportability engineering, in summary these include:

- The relative benefit of software technologies shown in Table 8 lacks credibility as it is based on just one assessment method. Further studies on the role of technology within supportability could add great value to this work by providing alternative assessment methods and outputs; which could then be used to verify results and improve credibility. I suggest that this could be taken forward by consideration of how technologies affect the distribution of effort, and relative cost of problem correction, within the maintenance lifecycle.
- Supportability is an emergent property dependent upon many interrelated factors. Unfortunately the demonstration of product supportability characteristics is a technically challenging discipline as the goal of developing a supportable system is at a significantly higher level of abstraction than the individual items of evidence that will contribute towards its achievement. Based on the similarities between supportability engineering and safety engineering, I suggest that tools more traditionally related to the safety engineering such as Claim-Argument-Evidence and Goal Structured Notation (GSN) are equally applicable to the supportability domain and should be considered as a means of presenting and managing system supportability.
- Supportability focused metric collection needs to be embraced by the MOD as a vehicle for understanding and improvement. A metric program needs to be established that captures sufficient data to enable the prediction of support requirements base on the analysis of comparable systems.

This dissertation has been developed in a military context but its generic supportability focus makes it equally applicable to any system requiring through life support. Accepting that the implementation of LSA is an effort consuming process, systems considering its application to software should carry out an initial cost benefit analysis to gain an understanding of the potential value it will add. However, systems that are expected to be in service for a long period of time could benefit from some form of supportability analysis and should review the contents of this dissertation.

*Collation Page*  
*(for two sided printing)*

## **APPENDIX A**

### **BIBLIOGRAPHY**

- <sup>1</sup> "Logistics Support Analysis Guidance For Software", M. Bailey, 2003.
- <sup>2</sup> "Software Maintenance: Concepts and Practice", A. Takang, P. Grubb, International Thompson Computer Press, 1996.
- <sup>3</sup> "Guidelines for Successful Acquisition and Management of Software-Intensive Systems", [www.stsc.hill.af.mil](http://www.stsc.hill.af.mil).
- <sup>4</sup> "Program Evolution", M. Lehman, Academic Press, London, 1985.
- <sup>5</sup> "Software-Reliability Engineering: Technology for the 1990s", J. Musa and W. Everett, from IEEE Software, Vol 7, No 4, pp. 36-43, Nov 1999.
- <sup>6</sup> "Support for Mission Software in RAF Systems", AP100D-10, Issue 3, Lft 101, 1998.
- <sup>7</sup> "Software Maintenance Management", B. Lientz, E. Swanson, Addison-Wesley, Reading MA, 1980.
- <sup>8</sup> "Software Maintenance Management : Changes in the Last Decade", J. Nosek, P. Palvia, Journal of Software Maintenance, Vol. 2, Part 3, pp. 157-174, 1990.
- <sup>9</sup> "Software Engineering Economics", B. Boehm, Prentice-Hall, Englewood Cliffs NJ, 1981.
- <sup>10</sup> "Meeting Customer Requirements", Software Supportability Program Standard, SAE-JA-1004, 1998.
- <sup>11</sup> "Guidance for Application - Software Support", Def Stan 00-60, Part 3, Issue 2, 1998.
- <sup>12</sup> "Support for Mission Software in RAF Systems", AP100D-10, Issue 3, 1998.
- <sup>13</sup> "Application of Integrated Logistic Support", Def Stan 00-60, Part 0, Section 3, Para 7.1.5, 2000.
- <sup>14</sup> "Integrated Logistic Support In The Royal Air Force", AP 100C-70, 2<sup>nd</sup> Ed, Chap 1, Para 10, 1996.
- <sup>15</sup> "Guidance for Application - Software Support", Def Stan 00-60, Part 3, Issue 2, Clause 11, 1998.
- <sup>16</sup> "Guide to the Application of LSA and LSAR", Def Stan 00-60, Part 2, Issue 4, Para 7.5, 2000.
- <sup>17</sup> "The Economics of Software Maintenance", B. Boehm, in R. Arnold, editor, Proceedings, Workshop on Software Maintenance, Silver Spring MD, IEEE Computer Society Press, pp. 9-37, 1983.
- <sup>18</sup> "Support for Mission Software in RAF Systems", AP100D-10, Issue 3, Lft 110, 1998.
- <sup>19</sup> "The Common Software Support Framework", L Cooper, T Laverick, ES(AIR)(WYT)/SS/(2421/02/0/01)/AD SYS, 2004.
- <sup>20</sup> "Astor Software and Data Support Requirements", M Bailey, ES(Air)(WYT)/SS/(2260/01)/AD SYS, 2003.

- 21 "Integrated Logistic Support In The Royal Air Force", AP 100C-70, 2<sup>nd</sup> Ed, Glossary of Terms, 1996.
- 22 "Fault Tree Handbook", Technical Report NUREG-0492, W. Vesely, F. Goldberg, N. Roberts, and D. Haasl, U.S. Nuclear Commission, Washington, D.C., 1981.
- 23 "Fault Analysis for Systems Engineers", INCOSE 2003, [www.incose.org](http://www.incose.org).
- 24 "Guidance for Application - Software Support", Def-Stan 00-60, Part 3, Issue 2, Annex D, 1998.
- 25 "Managing Software Quality And Business Risk", M. Ould, Wiley, Chichester, 1999.
- 26 "Safety Management Requirements for Defence Systems", Def Stan 00-56, Part 2, Issue 2, 1996.
- 27 "Logistics Support Analysis for Harrier GR9 / GR9A / TMK12 Software", D. Gill, ES(Air)(WYT)/SS/(2237/01)/AD SYS, 2004.
- 28 "The Capability Maturity Model: Guidelines for Improving the Software Process" Software Engineering Institute, Carnegie Mellon University, Harlow: Addison-Wesley, 1994.
- 29 "The JSF System Development and Demonstration Phase Software Presentation", 2002, [www.jsf.mil](http://www.jsf.mil).
- 30 "Future Offensive Air System Use Study", Issue 1, 2002.
- 31 "Software Engineering – A Practitioners Approach", R. Pressman, 3<sup>rd</sup> Ed, London: McGraw-Hill, 1992.
- 32 "MOD ILS Information", [www.ams.dii.r.mil.uk/content/docs/ils/ils\\_web/lsaf.htm](http://www.ams.dii.r.mil.uk/content/docs/ils/ils_web/lsaf.htm).
- 33 "Software Supportability Program Standard", Surface Vehicle / Aerospace Standard, The Engineering Society For Advancing Mobility Land Sea Air and Space, SAE-JA-1004, 1994.
- 34 "Managing Systems Engineering", The Open University, T837.
- 35 "Software Metrics, a Rigorous & Practical Approach", N. Fenton, S. Pfleeger, London: PWS Publishing Company, 2<sup>nd</sup> Ed, 1997.
- 36 "Software Capability Evaluation, Version 3.0, Method Description", Technical Report CMU/SEI-96-TR-002, P. Byrnes, M. Phillips, 1996, [www.sei.cmu.edu](http://www.sei.cmu.edu).
- 37 "Software Metrics for Product Assessment", R. Banche, G. Bazzana, London: McGraw-Hill, 1994.
- 38 "Software Maintainability Evaluation Guide", Kirtland AFB, NM: HQ Air Force Operational Test and Evaluation Center (AFOTEC), 1989.
- 39 "Guidelines for Successful Acquisition and Management of Software-Intensive Systems", [www.stsc.hill.af.mil](http://www.stsc.hill.af.mil).
- 40 "The National Computing Centre, Open Source – The UK Opportunity", Issue 2, 2002, [www.ncc.co.uk](http://www.ncc.co.uk).

## APPENDIX B

### ANALYSIS OF RAF CHANGE DRIVERS

An initial analysis of software changes experienced by RAF Software Support Teams (SST) up-holds the observations made in Section 1.6. The data used for this analysis was obtained from the Eurofighter 2000 (EF2000) International Air Forces Field Team (IAFFT)<sup>B\*</sup>. Unfortunately, the data set does not reflect the latest work carried out by in-Service SSTs and is limited in sample size to the implementation of 249 changes, as such this analysis can only be used to support opinion rather than state fact.

The data collected by the EF2000 IAFFT was gathered for the purpose of identifying EF2000 mission support factors, work that is complementary to the intentions of this dissertation. The types of modification used in the EF2000 report closely matched those contained within Section 1.6 with the exception of Enhancements, which had been named New Requirement.

To accommodate the fact that change types were not consistently categorized across the three SSTs supplying change information a normalising process was applied. It is accepted that the normalisation process could be used to influence analysis outputs, but for the purpose of this dissertation the benefit of improved comparability is considered to outweigh this disadvantage. The normalisation process was carried through the application of two rules to the modification descriptions, as follows:

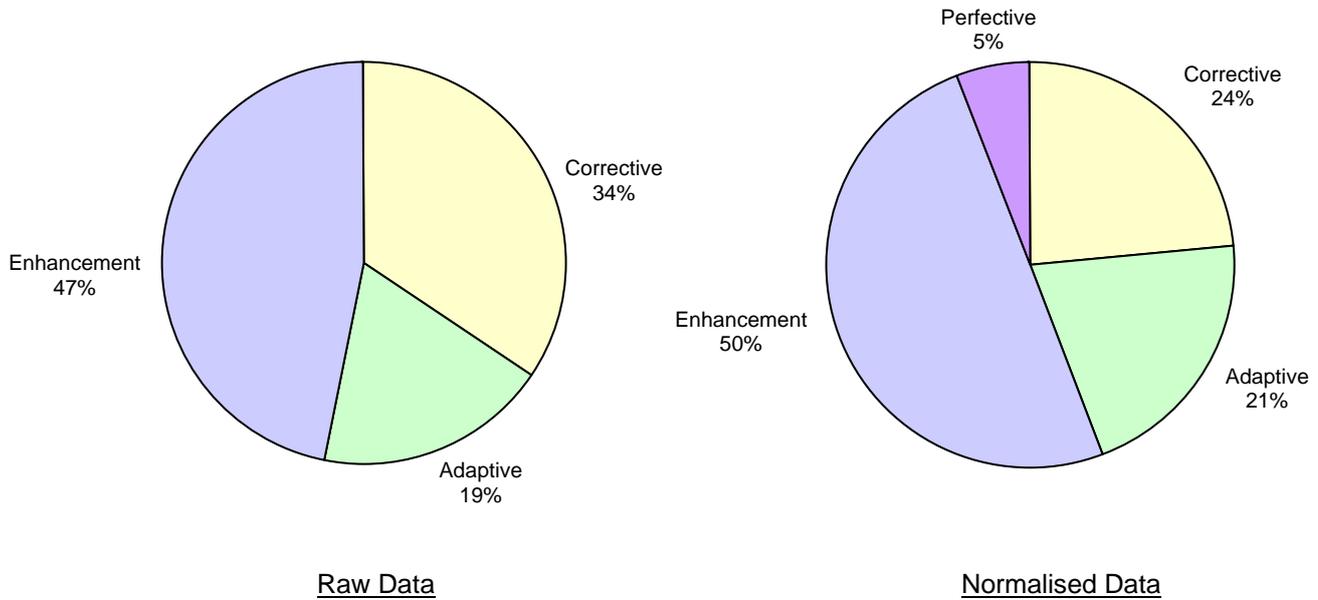
- Modification Re-categorization and Decomposition. The re-categorization and decomposition of modifications to best capture the amount and type of task completed. This rule allowed for the analysis of single data entries that were in fact multiple entries and corrected instances where the work descriptions did not match modification categorization.
- Urgent Operational Requirement (UOR) Rework Filtering. The filtering of modifications that were specifically attributable to the correction and perfection of functions initially developed under the UOR process. This rule was required to ensure that the rework tasks generated by UOR development did not mask statistics relating to the standard development process.

### HARRIER SOFTWARE MAINTENANCE UNIT (HSMU)

HSMU data consisted of information from 32 modifications. Normalisation was carried out that re-categorized and decomposed the data to best capture the amount and type of tasks completed. The raw and normalised spread of modification types is illustrated in Figure B1.

---

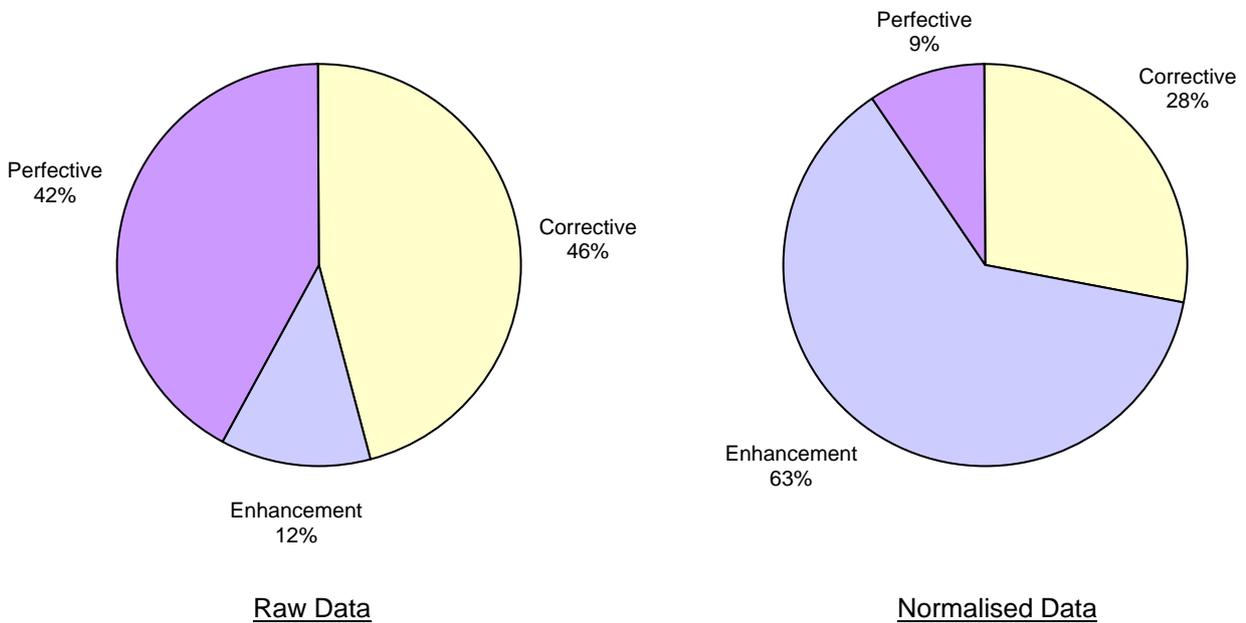
<sup>B\*</sup> MOD file reference: LC/166836/9/LSS2/WTN dated 26 Jun 96.



**Figure B1 – HSMU Change Data**

**TORNADO IN-SERVICE MAINTENANCE TEAM (TISMT)**

TISMT data consisted of information from 169 modifications. Normalisation was carried out that re-categorized, decomposed and filtered the data to best capture the amount and type of tasks completed. The raw and normalised spread of modification types is illustrated in Figure B2. The normalisation process had a notable effect on the data, which was caused by the large proportion of data (137 entries) that captured the correction and perfection of functions initially developed under the UOR process.



**Figure B2 – TISMT Change Data**

## TORNADO AIR DEFENCE VARIANT SOFTWARE MAINTENANCE TEAM (ADVSMT)

ADVSMT data consisted of information from 48 modifications. Normalisation was carried out that re-categorized the data to best capture the type of tasks completed. The raw and normalised spread of modification types is illustrated in Figure B3.

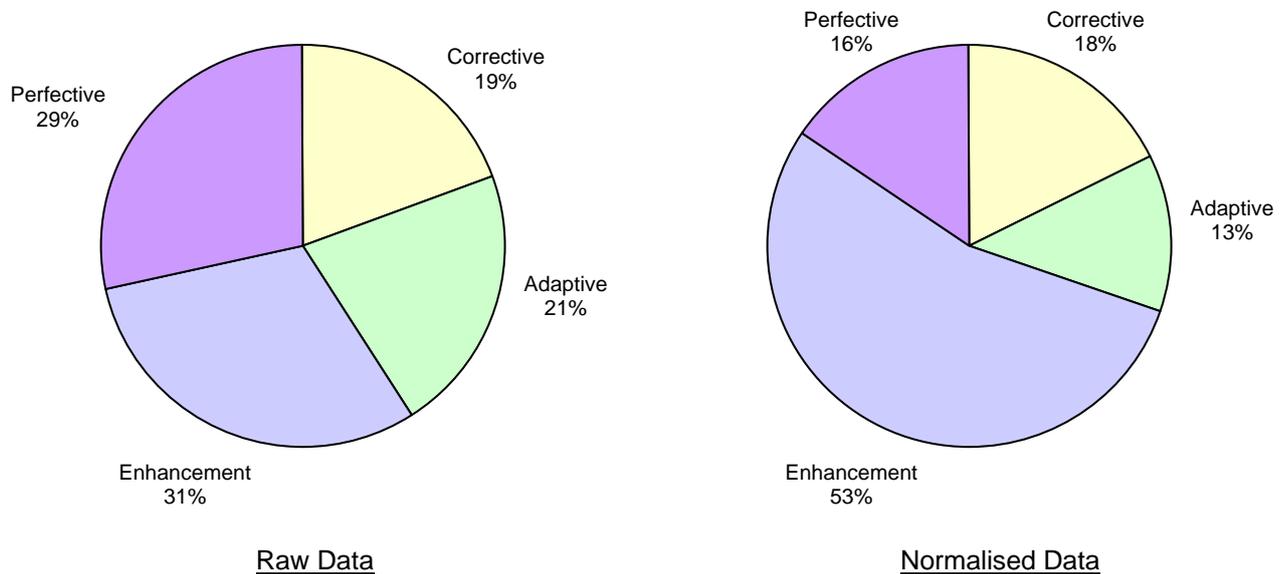


Figure B3 – ADVSMT Change Data

### DATA ANALYSIS LIMITATIONS

The data has a limited analysis capability and is noticeably constrained by the unavailability of associated information such as collection period and software change request rejection rates. However, the data does support the suggestion that software modifications driven by user enhancements are more common than those driven by the need to correct faults.

*Collation Page*  
*(for two sided printing)*

## **APPENDIX C**

### **SUPPORT ACTIVITY INVENTORY**

Now that the Software Support Model has been defined (see Figure 9), support activity decomposition is possible. The purpose of this decomposition is to provide greater visibility of the MOD's support expectations, facilitate future LSA, and assist initial supportability verification work through the creation of a task inventory against which reviews can be completed. It is not however, the intention of functional decomposition to impose a support solution or constrain support provider options. A support activity inventory distilled from the Support System Model is given in Tables C1 – C8.

Software Operations Related Processes	
Processes	Support Activities
Software Installation.	<ul style="list-style-type: none"> <li>• Distribution and recall.</li> <li>• Loading and unloading.</li> <li>• Load verification.</li> </ul>
System Support and Recovery.	<ul style="list-style-type: none"> <li>• System failure support (e.g. re-boot, re-load, re-run, re-start).</li> <li>• Product monitoring and review (to identify and report perceived faults and potential needs for corrective change).</li> <li>• Query and problem reporting.</li> </ul>

**Table C1 – Software Operations Related Processes**

Query Evaluation Related Processes	
Processes	Support Activities
Query and Problem Answering.	<ul style="list-style-type: none"> <li>• Operational User support.</li> </ul>
Query and Problem Analysis.	<ul style="list-style-type: none"> <li>• Queries and problem analysis, including:               <ul style="list-style-type: none"> <li>- Root cause analysis.</li> <li>- Operational impact and benefit analysis.</li> <li>- Risk and feasibility analysis.</li> <li>- Security and Safety impact assessment.</li> <li>- Reliability impact assessment.</li> <li>- Supportability impact assessment.</li> <li>- Cost(s) estimation of solution(s).</li> <li>- Software Change Request (SCR) generation (as necessary).</li> </ul> </li> </ul>

**Table C2 – Query Evaluation Related Processes**

Change Management Related Processes	
Processes	Support Activities
Change Authorization.	<ul style="list-style-type: none"> <li>• Software change trade-off assessment, including consideration of:               <ul style="list-style-type: none"> <li>- Major hardware upgrades.</li> <li>- Hardware obsolescence issues.</li> <li>- System goals, constraints and strategies.</li> <li>- Available resources.</li> <li>- System interoperability.</li> </ul> </li> <li>• SCR authorisation or rejection.</li> </ul>

**Table C3 – Change Management Related Processes**

Software Modification Related Processes	
Processes	Support Activities
Requirements Capture and Management.	<ul style="list-style-type: none"> <li>• All processes required to capture traceable modification requirements, (functional and non-functional) including: <ul style="list-style-type: none"> <li>- Preparation of associated documentation.</li> <li>- Preparation of requirement tests and criteria for testing.</li> </ul> </li> </ul>
Requirements Specification and Management.	<ul style="list-style-type: none"> <li>• All processes required to specify traceable modification requirements, including: <ul style="list-style-type: none"> <li>- Preparation of associated documentation.</li> <li>- Preparation of specification tests and criteria for testing.</li> </ul> </li> </ul>
Design Modification.	<ul style="list-style-type: none"> <li>• All processes required to amend the software design in order to implement a software modification, including: <ul style="list-style-type: none"> <li>- Preparation of associated documentation.</li> <li>- Preparation of design tests and criteria for testing.</li> </ul> </li> </ul>
Product Modification.	<ul style="list-style-type: none"> <li>• All processes required to implement a product modification, including: <ul style="list-style-type: none"> <li>- Preparation of associated documentation.</li> <li>- Preparation of product tests and criteria for testing.</li> </ul> </li> </ul>
Integration and Testing.	<ul style="list-style-type: none"> <li>• All processes required to integrate, verify and validate the modified products, including: <ul style="list-style-type: none"> <li>- Unit integration.</li> <li>- Module integration.</li> <li>- Unit testing.</li> <li>- Integration testing.</li> <li>- Regression testing.</li> <li>- System testing.</li> <li>- Acceptance testing.</li> </ul> </li> </ul>

Table C4 – Software Modification Related Processes

Integrity Assurance Related Processes	
Processes	Support Activities
Product Release.	<ul style="list-style-type: none"> <li>• Product release readiness assessment (fit for purpose).</li> <li>• Product migration.</li> </ul>

Table C5 – Integrity Assurance Related Processes

Data Support Related Processes	
Processes	Support Activities
Data Availability.	<ul style="list-style-type: none"> <li>• Formulation and provision of mission related data loads.</li> <li>• Formulation and provision of engineering related data loads.</li> </ul>

Table C6 – Data Support Related Processes

Customer and Supplier Related Processes	
Processes	Support Activities
Contracting.	<ul style="list-style-type: none"> <li>• Tender proposal and assessment.</li> <li>• Contract monitoring, review and update.</li> </ul>
Communication and Interfacing.	<ul style="list-style-type: none"> <li>• User training.</li> <li>• Product acceptance.</li> </ul>

Table C7 – Customer and Supplier Related Processes

Omnipresent Lifecycle Processes	
Processes	Support Activities
Product and Service Acquisition.	<ul style="list-style-type: none"> <li>• Data utilisation monitoring.</li> <li>• Sub-team, and sub-contractor utilisation management.</li> </ul>
Configuration Management.	<ul style="list-style-type: none"> <li>• Control and management of versions and variants.</li> <li>• Identification and safekeeping.</li> <li>• Replication and replication verification.</li> <li>• Disaster recovery.</li> <li>• Disposal.</li> </ul>
Personnel Management.	<ul style="list-style-type: none"> <li>• Team management, including: <ul style="list-style-type: none"> <li>- Training.</li> <li>- Personnel review and appraisal.</li> <li>- Administration.</li> </ul> </li> </ul>
Project Management.	<ul style="list-style-type: none"> <li>• Modification capability management, including: <ul style="list-style-type: none"> <li>- Skills provision and sustainment.</li> <li>- Facilities provision and sustainment.</li> <li>- Tools and rigs provision and sustainment.</li> </ul> </li> <li>• Project tracking and oversight.</li> <li>• Project communication and feedback.</li> <li>• Project estimation and planning.</li> <li>• Responsibility negotiation, allocation and acceptance.</li> <li>• Project debriefing (lessons learnt).</li> </ul>
Quality Management.	<ul style="list-style-type: none"> <li>• Quality policy management.</li> <li>• Quality planning.</li> <li>• Quality achievement, including: <ul style="list-style-type: none"> <li>- Standard suitability assessment (both international and local).</li> <li>- Method suitability assessment.</li> </ul> </li> <li>• Quality control, including: <ul style="list-style-type: none"> <li>- Ensuring compliance and consistency to standards.</li> <li>- Ensuring compliance and consistency to higher-level products and requirements.</li> </ul> </li> <li>• Quality preservation, including: <ul style="list-style-type: none"> <li>- Ensuring product traceability and consistency (products match their descriptions).</li> <li>- Functional and non-functional characteristic maintenance (see query evaluation).</li> </ul> </li> <li>• Audit.</li> <li>• Review, including: <ul style="list-style-type: none"> <li>- Project reviews.</li> <li>- Technical reviews.</li> </ul> </li> </ul>
Documentation Management.	<ul style="list-style-type: none"> <li>• Standards review and amendment (international standards for appropriateness, local standards for appropriateness and correctness).</li> <li>• Records covering production and maintenance, including: <ul style="list-style-type: none"> <li>- Audits.</li> <li>- Reviews.</li> <li>- Work products.</li> <li>- Task analyses and Terms of Reference (ToR).</li> <li>- Training records.</li> </ul> </li> </ul>
Process Improvement.	<ul style="list-style-type: none"> <li>• Process and product measurement.</li> <li>• Measurement analysis.</li> <li>• Improvement initiatives.</li> </ul>

Table C8 – Omnipresent Lifecycle Processes

*Collation Page*  
*(for two sided printing)*

## APPENDIX D

### QUALITATIVE SUPPORT OPTION ASSESSMENT

The ability to support system software will be influenced by the supportability characteristics of the system's products and processes. Given a set of Software Support Significant Items (SSSIs), and an understanding of the characteristics that influence supportability (see Appendix E), alternative support systems can be assessed. Support provider selection should be based on an organisation's ability to provide both effectiveness and efficiency. By considering the relative strengths and weaknesses of potential support options an early qualitative assessment can be made.

For example purposes an OFP qualitative support assessment, reflective of current support capabilities, is given in Tables D1 – D6. For definitions of the potential support options refer to Section 4.3. Within Tables D1 – D6, relative advantages and disadvantages have been attributed to functional areas within the Support System Model (see Figure 9) and are identified by the following abbreviations:

- All functions (All).
- Software Modification (SM).
- Software Operations (SO).
- Change Management (CM).
- Query Evaluation (QE).
- Integrity Assurance (IA).
- Data Support (DS).
- Not Attributable (NA).

Industry Original Equipment Manufacturer (OEM)			
Relative Advantages		Relative Disadvantages	
Experienced in managing both large and small scaled projects.	SM	Product support is dependent upon economic viability.	All
Experienced in managing subcontractors.	SM	Placing all support responsibility with one provider increases the risk of customer exploitation.	All
Well positioned to exploit technology to overcome obsolescence and improve supportability.	SM	Equipment users and requirements analysts exist in diversely different organisations.	SM
Experienced in implementing modifications across the full extent of complexity, size and scope.	SM	Sustainment of the Software Development Environment (SDE) is only possible if economically viable.	SM
Often the IPR holder of bespoke system software.	SM	Sustainment of required skills is only possible if economically viable.	SM
Transition of concepts, documentation and working practices from development to support should be easy.	SM	Urgent Operational Requirements (UOR) responsiveness is hampered by the need for contracting.	SM

Industry Original Equipment Manufacturer (OEM)			
Relative Advantages		Relative Disadvantages	
Placing support responsibility with industry removes the Service's burden to establish and sustain the SDE.	SM	During periods of surge capacity support flexibility will be dependent upon the ability to reallocate resources, this will only be possible if economically viable.	SM
Placing support responsibility with industry removes the Service's burden to establish and sustain the required competencies.	SM	It may not be appropriate or possible for industry to carry out some 1 <sup>st</sup> line functions, especially when equipment is deployed.	SO
Having a single provider of support will assist with the integration of software changes.	SM	Meeting the customer's operational needs will not always complement the business needs.	CM
Experienced in quality and configuration management at a systems level.	SM IA	Equipment users cannot discuss their problems and queries with Service personnel that appreciate the operational environment.	QE
Competent in safety, security and systems engineering disciplines.	SM IA	Interdependency between the OEM and Integrity Assurance organisation will be harder to define and establish.	IA
Experienced in both the system as a whole and individual functional areas.	SM IA		
Change management is assisted as there is only one organisation implementing changes.	CM		

**Table D1 – Industry OEM Support Option Advantages and Disadvantages**

Industry 3 <sup>rd</sup> Party			
Relative Advantages		Relative Disadvantages	
Experienced in managing both large and small scaled projects.	SM	Product support is dependent upon economic viability.	All
Experienced in managing subcontractors.	SM	Placing all support responsibility with one provider increases the risk of customer exploitation.	All
Experienced in implementing modifications across the full extent of complexity, size and scope.	SM	Transition of concepts, documentation and working practices from development to support may be problematic.	SM
Well positioned to exploit technology to overcome obsolescence and improve supportability.	SM	Authority or the IPR itself will need to be acquired in order to implement modifications.	SM
Placing support responsibility with industry removes the Service's burden to establish and sustain the SDE.	SM	Equipment users and requirements analysts exist in diversely different organisations.	SM
Placing support responsibility with industry removes the Service's burden to establish and sustain the required competencies.	SM	Sustainment of the Software Development Environment (SDE) is only possible if economically viable.	SM
Having a single provider of support will assist with the integration of software changes.	SM	Sustainment of required skills is only possible if economically viable.	SM
Competent in safety, security and systems engineering disciplines.	SM IA	Urgent Operational Requirements (UOR) responsiveness is hampered by the need for contracting.	SM
Experienced in quality and configuration management at a systems level.	SM IA	During periods of surge capacity support flexibility will be dependent upon the ability to reallocate resources, this will only be possible if economically viable.	SM
Interdependency between the OEM and Integrity Assurance organisation will be easier to define and establish.	IA	It will be hard for the support organisation to gain experience in both the system as a whole and individual functional areas.	SM IA

Industry 3 <sup>rd</sup> Party			
Relative Advantages		Relative Disadvantages	
Change management is assisted as there is only one organisation implementing changes.	CM	It may not be appropriate or possible for industry to carry out some 1 <sup>st</sup> line functions, especially when equipment is deployed.	SO
		Meeting the customer's operational needs will not always complement the business needs.	CM
		Equipment users cannot discuss their problems and queries with Service personnel that appreciate the operational environment.	QE

Table D2 – Industry 3<sup>rd</sup> Party Support Option Advantages and Disadvantages

Service			
Relative Advantages		Relative Disadvantages	
No risk of customer exploitation.	All	Transition of concepts, documentation and working practices from development to support may be problematic.	SM
Maintains the Service's 'Intelligent Customer' role.	All	Authority or the IPR itself will need to be acquired in order to implement modifications.	SM
Resources can be reallocated as required to satisfy UORs.	SM	Major capital investment will be required by the MOD to attain necessary skills and infrastructure.	SM
Work can be completed without the need for contracting.	SM	The cost of skills provision and sustainment is owned by the MOD.	SM
Equipment users and requirements analysts exist in the same organisation.	SM	The cost of SDE provision and sustainment is owned by the MOD.	SM
Having a single provider of support will assist with the integration of software changes.	SM	No experience in the management of large scaled projects.	SM
The tasking organisation has relative freedom to negotiate the level of testing in order to meet operational need.	SM	Inexperienced in the management of subcontractors.	SM
Change management is assisted as there is only one organisation implementing changes.	CM	Badly positioned to exploit technology to overcome obsolescence and improve supportability.	SM
Meeting the customer's operational needs is the support organisations core business.	CM	Inexperienced in implementing modifications across the full extent of complexity, size and scope.	SM
Equipment users can discuss their problems and queries with Service personnel that appreciate the operational environment.	QE	The release of software will be dependent upon the MOD's ability to attain product certification.	SM
Well experienced in carrying out Front Line software tasks.	SO	Inexperienced in the disciplines of safety, security and systems engineering.	SM IA
Independency between the OEM and Integrity Assurance organisation will be easier to define and establish.	IA	Inexperienced in quality and configuration management at a systems level.	SM IA
		It will be hard for the support organisation to gain experience in both the system as a whole and individual functional areas.	SM IA
		Against current MOD policy (Provider to Decider).	NA

Table D3 – Service Support Option Advantages and Disadvantages

Diverse			
Relative Advantages		Relative Disadvantages	
Reduced risk of customer exploitation through partial sustainment of an organic support capability.	All	Some exposure to business related factors would still exist.	All
Helps to maintain the Service's 'Intelligent Customer' role.	All	Communication is made more complex as there are multiple support organisations.	All
Industry is experienced in managing both large and small-scaled projects.	SM	Problems in the transition of concepts, documentation and working practices from development to support might still exist.	SM
Industry is experienced in managing subcontractors.	SM	The integration of modifications into the software baseline will become more complex.	SM
Industry is well positioned to exploit technology to overcome obsolescence and improve supportability.	SM	The potential for integration induced function regression ('Saw Tooth Effect') is created.	SM
Industry is experienced in implementing modifications across the full extent of complexity, size and scope.	SM	Inefficiency is introduced through OEM rework of modifications to gain certification.	SM
IPR is easier to secure as the OEM is involved.	SM	Interdependency between the OEM and Integrity Assurance organisation will be harder to define and establish.	IA
Service resources can be reallocated as required to satisfy UORs.	SM	It may not be appropriate or possible for industry to carry out some 1 <sup>st</sup> line functions, especially when equipment is deployed.	SO
Service allocated tasks can be completed without the need for contracting.	SM	Change management is made more complex as there are multiple support organisations.	CM
Equipment users and requirements analysts exist in the same organisation.	SM		
Placing support responsibility with industry reduces the Service's burden to establish and sustain the SDE.	SM		
Placing support responsibility with industry reduces the Service's burden to establish and sustain the required competencies.	SM		
Industry is experienced in quality and configuration management at a systems level.	SM IA		
Industry is competent in safety, security and systems engineering disciplines.	SM IA		
Industry is experienced in both the system as a whole and individual functional areas.	SM IA		
Meeting the customer's operational needs is the Service's core business.	CM		
Service personnel are well experienced in carrying out Front Line software tasks.	SO		
Equipment users can discuss their problems and queries with Service personnel that appreciate the operational environment.	QE		

Table D4 – Diverse Support Option Advantages and Disadvantages

Partnered			
Relative Advantages		Relative Disadvantages	
Reduced risk of customer exploitation through partial sustainment of an organic support capability.	All	Some exposure to business related factors would still exist.	All
Helps to maintain the Service's 'Intelligent Customer' role.	All	The support option is new and immature; as such it will attract an increased level of risk.	All
Industry is experienced in managing both large and small-scaled projects.	SM	Specific management issues will exist relating to balancing the needs of industrial and Service personnel (i.e. career management, cultural expectations).	SM
Industry is experienced in managing subcontractors.	SM	Independency between the OEM and Integrity Assurance organisation will be harder to define and establish.	IA
Industry is well positioned to exploit technology to overcome obsolescence and improve supportability.	SM	It may not be appropriate or possible for industry to carry out some 1 <sup>st</sup> line functions, especially when equipment is deployed.	SO
Industry is experienced in implementing modifications across the full extent of complexity, size and scope.	SM		
IPR is easier to secure as the OEM is involved.	SM		
Dedicated Service resources are available to satisfy UORs.	SM		
Tasks can be completed by Service personnel without the need for contracting.	SM		
Equipment users and requirements analysts exist in the same organisation.	SM		
Transition of concepts, documentation and working practices from development to support should be easy.	SM		
Placing support responsibility with industry reduces the Service's burden to establish and sustain the SDE.	SM		
Placing support responsibility with industry reduces the Service's burden to establish and sustain the required competencies.	SM		
Having a partnered support solution will assist with the integration of software changes.	SM		
Certification and release related rework is minimised through close integration of OEM and other support providers.	SM		
Industry is experienced in quality and configuration management at a systems level.	SM IA		
Industry is competent in safety, security and systems engineering disciplines.	SM IA		
Industry is experienced in both the system as a whole and individual functional areas.	SM IA		
Service personnel are well experienced in carrying out Front Line software tasks.	SO		
Meeting the customer's operational needs is the Service's core business.	CM		
Having a partnered support solution will assist with the management of change.	CM		
Equipment users can discuss their problems and queries with personnel that appreciate the operational environment.	QE		

Table D5 – Partnered Support Option Advantages and Disadvantages

No Support			
Relative Advantages		Relative Disadvantages	
No additional costs once development is finished.	All	Only applicable to the most insignificant software. Exposes the customer to the risk of capability loss if at some time modification is required but not possible. Without support the investment place in developing or procuring software is unprotected from obsolescence.	All

Table D6 – No Support Advantages and Disadvantages

With the assessment now complete, the selection of the most appropriate support option is possible. This selection process is not clear-cut, as there will always be trade-offs that cannot be balanced. At this qualitative stage of work, the selection process is based on the relative strengths and weaknesses of each support organisation, the needs of each function within the Support System Model and the requirements of Customers 1 and 2 (see Table 4)

The qualitative assessment of support options by nature will be a subjective task. To reduce the variance of this subjectivity, assessments need to be made against a set of defined criteria. These criteria must have an operation focus whilst balancing constraints relating to support feasibility, as such the following criteria are suggested.

- Favourable. Selection of this option:
  - Will fully satisfy all of the requirements of Customers 1 and 2.
  - Will fully satisfy the support task requirements.
  - Has support organisation advantages that beneficially outweigh their disadvantages.
  - Can accommodate the implications of SSSI product characteristics.
- Viable. Selection of this option:
  - Will fully satisfy all of the requirements of Customers 1 and 2.
  - Will fully satisfy the support task requirements.
  - Has acceptable support organisation advantages and disadvantages.
  - Can accommodate the implications of SSSI product characteristics.
- Unviable. Selection of this option either:
  - Will not satisfy one or more of the requirements of Customers 1 and 2.
  - Will not satisfy the support task requirements.

- Cannot accommodate the implications of SSSI product characteristics.
- Not Assessed (NA). Insufficient data or analysis is available at this time to make an informed judgement.

An example qualitative assessment has been made for support of an Operational Flight Program. The assessment, given at Table D7, is reflective of current support capabilities, does not consider SSSI product characteristics and has no reference to support costs.

Support Tasks	Support Option					
	Industry OEM	Industry 3 <sup>rd</sup> Party	Service	Diverse	Partnered	No Support
Software Operations	Unviable <sup>D*</sup>	Unviable <sup>D*</sup>	Favourable	Viable	Viable	Unviable <sup>D‡</sup>
Query Evaluation	Viable	Viable	Favourable	Viable	Viable	Unviable <sup>D‡</sup>
Change Management	Unviable <sup>D†</sup>	Unviable <sup>D†</sup>	Favourable	Viable	Favourable	Unviable <sup>D‡</sup>
Software Modification <sup>D§</sup>	Viable	Unviable <sup>D*</sup>	Unviable <sup>D*</sup>	Viable	Viable	Unviable <sup>D‡</sup>
Integrity Assurance <sup>D**</sup>	Viable	Viable	Unviable <sup>D*</sup>	Viable	Viable	Unviable <sup>D‡</sup>
Data Support	Not Assessed <sup>D‡‡</sup>	Not Assessed <sup>D‡‡</sup>	Not Assessed <sup>D‡‡</sup>	Not Assessed <sup>D‡‡</sup>	Not Assessed <sup>D‡‡</sup>	Unviable <sup>D‡</sup>

Table D7 – OFP Qualitative Support Option Selection

---

<sup>D\*</sup> Unviable on the grounds that the support provider cannot fully provide the necessary support function.  
<sup>D‡</sup> Unviable on the grounds that, based on empirical evidence, the support function will be required.  
<sup>D†</sup> Unviable on the grounds that Customers 1 & 2 need to be involved in the function.  
<sup>D§</sup> A favourable option has not been identified as the real benefits of a Partnered approach are as yet untested due to option immaturity.  
<sup>D\*\*</sup> A favourable option has not been identified as the required level of independency from the OEM is at present unknown.  
<sup>D‡‡</sup> Insufficient data analysis carried out.

*Collation Page*  
*(for two sided printing)*

## APPENDIX E

### SUPPORTABILITY CHARACTERISTICS

General, though not exhaustive, product and process characteristics that influence supportability are documented in Tables E1 and E2. These characteristics have been identified through consideration of the Support System Model (see Figure 9), Capability Maturity Model<sup>28</sup>, systems engineering principles and previous LSA reports.

Product Characteristics	
Characteristic	Description
Architecture	System architecture should be influenced by the need for ease of change.
Requirements	The requirements should be: understandable, complete, unambiguous, consistent, organised, compliant to standards and traceable, such that the impact of change can be assessed and managed.
Design	The design should be: <ul style="list-style-type: none"> <li>• Understandable - traceable from the requirements.</li> <li>• Logical - following the structure of the system.</li> <li>• Robust - degrade gently in the presence of faults.</li> <li>• Modifiable - isolating likely areas of change to reduce the impact of change.</li> </ul>
Complexity	The design should balance the needs of low coupling and high cohesion.
Code / Language	The implementation should represent the design. The choice of language should balance the following needs: <ul style="list-style-type: none"> <li>• Technical application.</li> <li>• Clarity of source code.</li> <li>• Maintainability, reliability and portability.</li> </ul>
Test	Adequate test cases should be established, implemented and maintained to demonstrate that product build quality and requirement satisfaction is acceptable.
Safety Risk Classification	Safety Related Software (SRS) will require specific attention regards its modification and subsequent certification and release. Where possible SRS should be isolated from expected areas of change.
Security Risk Classification	Software identified as security related will require specific attention regards its modification and subsequent accreditation. Where possible security related software should be isolated from expected areas of change.
Conformance to Standards	The appropriate application of standards can improve product quality; the use of Open Standards is particularly desirable as they promote interoperability via the standardisation of interfaces and protocols.
Documentation	In addition to the requirements characteristics (understandable, complete, unambiguous, consistent, organised, compliant to standards and traceable); documentation should be appropriate (for both current and future use), in that it should be sufficient to convey the necessary information, be up to date, and in a format most appropriate to the task.
Obsolescence Status	Obsolescence is tightly coupled to the environment and equipment in which the product is developed and operates; as such software product obsolescence is best explored by considering software process obsolescence. The only way that the software product can become obsolete is through gradual, maintenance induced, degradation. When a need for change occurs and modification is no longer possible, product obsolescence has occurred.
Appropriate use of New and Evolving Technologies	Technology utilisation must balance the need to meet both functional and non-functional characteristics. Evidence should be established demonstrating the supportability benefit of utilised software technologies.

Table E1 – Product Characteristics

Process Characteristics	
Characteristic	Description
Requirements Management	Change is inevitable, this must be appreciated and managed such that capability can be sustained and enhanced, and the development investment protected.
Project Management	The timely achievement of milestones will directly affect both the equipment and the capability users. As such modification needs to be both planned and managed in order to generate meaningful milestones and estimates.
Subcontractor Management	Where support activities are allocated to other organisations or subcontractors, the prime support organisation should demonstrate that they have adequate measures in place to ensure products are delivered on time, to specification and within budget.
Configuration Management	Once product quality has been achieved it needs to be preserved. Identification, Change and Configuration Management are activities that assist quality preservation.
Quality Management	As well as the initial achievement of product quality, the support organisation needs to demonstrate how quality will be sustained throughout the life of the system.
Change Management	Change must be balanced against the capability need, available resources, given constraints and User abilities.
Obsolescence Management	The rapid speed of technology change within information systems demands a robust approach to obsolescence management. Technology should be exploited to offer the best balance between product performance and lifetime supportability.
Software Development Environment (SDE) Provision and Sustainment	<p>The creation of software products is heavily dependent upon the SDE. As such the SDE itself needs to be managed so that the ability to implement modifications is not lost. Specific issues relating to the SDE include:</p> <ul style="list-style-type: none"> <li>• Rigs must be of adequate fidelity (utilise or accurately emulate aircraft standard spares), enabling testing appropriate to the need.</li> <li>• Software development tools need to be licensed and obsolescence managed for the life of dependent software products.</li> <li>• Facilities should be provided that enhance productivity by balancing the needs of individuals and the support organisation.</li> </ul>
Skills Provision and Sustainment	<ul style="list-style-type: none"> <li>• Software Change: <ul style="list-style-type: none"> <li>- The ability to assess, manage, modify and release software is an intellectual process, as such it is highly dependent upon the skill of individuals that perform these functions. These skills must be managed for the expected period of software maintenance.</li> </ul> </li> <li>• Software Fielding: <ul style="list-style-type: none"> <li>- Tasks carried out during software operations are of a procedural nature. Some overhead will be required to train individuals to perform these tasks, but this should not be significant. To accommodate the Service rotation of personnel the skills required to perform these tasks should be kept to a minimum.</li> </ul> </li> </ul>
Competency	Specific skills issues exist in the domains of safety, security and systems engineering, in that they require increased competency levels. Specific attention should be given to the provision and sustainment of these skills.
Scalability	Individual modifications will vary in their complexity and size and scope. The support solution must be capable of accommodating the full extent of all modifications.
User Support	As well as the need to modify products, Users require a support service capable of answering any queries or problems that arise in a timely manner.
Intellectual Property Rights (IPR)	Having the ability to make change is irrelevant if it is not accompanied with the legal right to do so. IPR must be acquired and sustained for areas likely to experience the need for modification. Once a product is modified, IPR also needs to be considered for the modification itself.
Organisational Support Focus	The development and support organisations need to have a whole life approach; their products must be created and modified with supportability in mind. The ethos of support must be congruent with the business goals and capability needs. When necessary, the 'can do' attitude must prevail in order to meet operational need.

Process Characteristics	
Characteristic	Description
Organisational Dependency	Dependencies exist for both the Customer and Supplier; the Customer can become totally dependent on the Supplier if no other supply option is available or the Supplier can become totally dependent upon the Customer if they have only one purchaser of their services. From this it is evident that there needs to be a balanced Customer – Supplier relationship. The Customer needs to provide enough business to ensure the Supplier remains healthy, whereas the Supplier needs to reliably deliver an agreed level of service. To enable the relationship balance, the Customer must be able to determine and validate their support requirements.
Organisational Structure and Communication	The support organisation needs to manage the needs of the system as well as the individual functions, as such, the organisation needs to be both Project and Functionally oriented. Good communication must be established across functional areas and between project layers.
Change Integration	Software products should continually improve and mature. The integration of changes should be managed such that function regression (sometimes referred to as the ‘Saw Tooth Effect’) does not occur.
Change Release	The process of releasing changes to service should not impact on operational capability. Release processes should be established that are both commensurate with the operational need and reduce the need for rework to a minimum.
Holistic Approach	Software needs to be recognised as part of the system, as such the development and support organisations need to have a holistic ‘systems’ approach that balances the needs of hardware and software support.
Support Maturity	It is highly unlikely that support arrangements will be optimal from the outset; as such a means of incentivising, maturing and improving support should be established.
Responsiveness	Individual changes will have an associated priority level. The support organisation needs to be able to service high priority changes in a timely manner, reallocating resources as necessary to provide the required level of support flexibility.
Operational Location	Where the Software Host resides at distributed locations, the activity of software distribution and recall needs to be assessed for its impact on availability. Appropriate measures need to be implemented to ensure product distribution and recall does not impact on capability availability.
Human Factors	In addition to the provision of adequate training, individual and team needs also need to be satisfied. Therefore, the support organisation needs to provide a working environment and regime that accommodates rest, recuperation and cultural expectations (this is particularly relevant where mixed civilian and Service teams are established).

Table E2 – Process Characteristics

*Collation Page*  
*(for two sided printing)*

## APPENDIX F

### TECHNOLOGY DEMONSTRATOR PROGRAMME SUMMARY

The TDP sequence of events is illustrated in Figure F1.

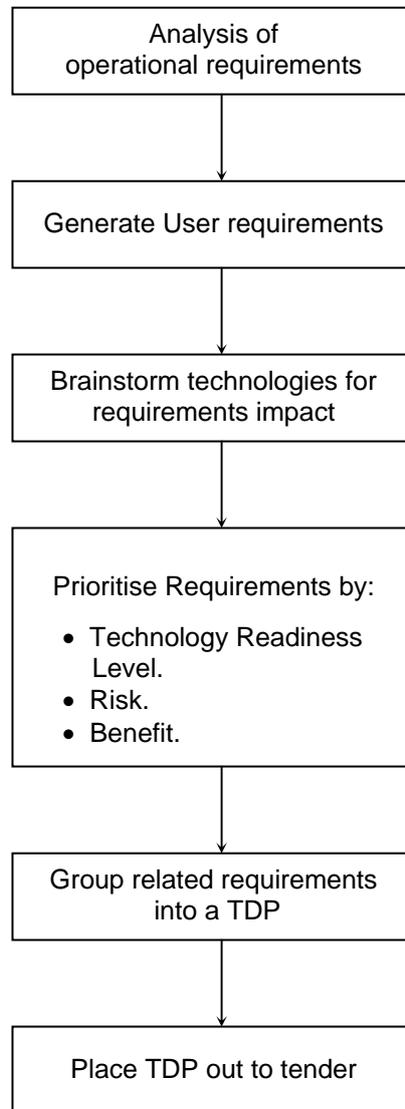


Figure F1 – TDP Sequence of Events

*Collation Page*  
*(for two sided printing)*

## **APPENDIX G**

### **SUPPORTABILITY RELATED TECHNOLOGIES**

Potential software technological opportunities are identified within Table G1.

Technological Opportunity	Support Context	Potential Supportability Benefit
Redefinition and refinement of the support infrastructure and processes.	An opportunity exists to reduce the cycle time between requirement identification and the release of software modifications	Reduction in the time taken to release modifications through improved working practices.
Appropriate adoption and application of standards for development and support.	Inappropriate standards are often mandated within projects such that no value is added to the quality of outputs.	Appropriately standardised software products will improve understandability and supportability.
Adoption of a Process Improvement initiative.	Process Improvement endeavours to ensure that the product is not only built correctly but appropriately to the business needs. Process Improvement is continual.	Better quality of final product, reduced risk of process failure and continual improvement. There is anecdotal evidence that mature organisations also benefit from reduced manpower turnover.
Organisationally defined and standardised utilisation of Computer Aided Software Engineering (CASE) tools.	CASE tool utilisation is the only realistic approach for many engineering activities across product development and support. These tools however must support the organisation's supportability needs.	Organisationally defined and standardised tools will assist with the management of personnel skills and training. Improved flexibility and responsiveness can be achieved once a defined set of skills has been established.
Organisationally defined and standardised software management tools.	The complexity of modern avionics makes the achievement and preservation of non-functional characteristic almost impossible without the assistance specialist management tools covering such disciplines as: <ul style="list-style-type: none"> <li>• Requirements management.</li> <li>• Configuration management.</li> <li>• Safety management.</li> <li>• Supportability management.</li> </ul>	Organisationally defined and standardised tools will assist with the management of personnel skills and training. Improved supportability can be achieved through the use of tools that assist in the management of system non-functional characteristics. Software release times can be reduced through the improved management of integrity assurance.
Organisationally defined and standardised Software Engineering Environment (SEE) such as: <ul style="list-style-type: none"> <li>• Languages.</li> <li>• Methodologies.</li> <li>• Rigs and simulators.</li> </ul>	Standardisation of the SEE will be complementary to the utilisation of CASE tools. The SEE however must support the organisation's supportability needs.	Organisationally defined and standardised SEE will assist with the management of personnel skills and training. Improved support flexibility and responsiveness can be achieved once a defined set of skills and engineering infrastructure has been established.

Technological Opportunity	Support Context	Potential Supportability Benefit
Adoption of a standardised approach to data formatting and loading.	Software Operations Support (SOS) comprises of two different activities: Software and data loading, and pre-mission data formulation. Both areas offer opportunities for supportability improvement if data loading operations and mission data formats are standardised across platforms.	Whilst the effort required for mission data formulation is at present unknown, it is evident that this process consumes a significant amount of effort. This effort appears to be much larger than that expended on data and software loading activities. Standardised mission data formats and loading mechanisms can improve supportability and interoperability by reducing rework and loading effort.
Application of Object Orientation (OO).	Some platform functions will be susceptible to modifications originating from the frequent need to improve Human Computer Interfaces (HCI). In addition to this, understandability is related to the modularisation of software functions <sup>39</sup> .	OO techniques have the potential to improve supportability through the separation of computational and operational data from interface calls. OO encourages the modularisation, and hence understandability, of software products.
Exploitation of automatic code generation.	Most of the needs for supportable software are generated by the assumption that software needs to be developed and supported to enable capability sustainment. A challenge to this assumption comes from the potential ability to automatically generate software (where the level of automation defines the residual dependency on support activities).	The ability to automatically produce integrity assured software would revolutionise capability sustainment. Software coding will no longer exist; it will just be generated as required. Where automation is on a smaller scale the potential benefits of this technology will be reduced. <b>Note:</b> Software implementation is only one activity in the development lifecycle. Effort intensive disciplines such as requirement definition and specification will still be required (see Figure 3).
Utilisation of reuse, multiuse and Commercial Off The Shelf (COTS) software.	The Joint Strike Fighter project advocates the maximisation of reuse, multiuse and COTS software <sup>29</sup> . The utilisation of reuse, multiuse and COTS is based on the assumption that it is more economic to exploit existing products rather than develop new ones.	<i>“Where the reuse, multiuse or COTS product meets, without modification, the system development and support requirements it will almost always cost less to acquire than the development of equivalent software”<sup>31</sup>.</i>

Technological Opportunity	Support Context	Potential Supportability Benefit
<p>Utilisation of Open Standards.  <i>“The term open has spread beyond the world of software development and is diluting the original meaning of the term. It now commonly refers to the general application of a distributed collaborative development method and/or free participation in production and access to product in any sphere of activity”.</i><sup>40</sup></p>	<p>The utilisation of open standards will assist with the modularisation of systems by providing a consistent and readily accessible definition of interface standards. In conjunction with a complementary architecture, once the interfaces between modules are standardised software modifications should be easier to implement through improved abstraction, encapsulation and information hiding.</p>	<p>Reduced occurrence and impact of software modifications driven by adaptive program interface changes. Improved supportability through module commonality and reuse potential.</p>
<p>Utilisation of Integrated Modular Avionics (IMA).</p>	<p>The utilisation of IMA provides architectural separation between hardware and the operating system, and between the operating system and the application layer. This will assist with the management of hardware driven modifications.</p>	<p>Reduced impact of software modifications driven by hardware changes. Improved supportability by designing for scalability and accommodation of changes in hardware.</p>

Table G1 – Technological Opportunities

*Collation Page*  
*(for two sided printing)*

## APPENDIX H

### TECHNOLOGY SUPPORTABILITY BENEFIT

It is possible to associate potential benefits to the technological opportunities proposed in Appendix G. This has been carried out within Table H1 by assessing each technology for its ability to limit the occurrence or impact of system change drivers (see Figure 4). Undesirably, this work lacks credibility as it is based on just one assessment method and does not adequately quantify how product supportability characteristics might have been improved, as such it can only be used to identify issues for further consideration.

Technology Opportunity (Refer to Appendix G for descriptions)	Potential Improvement (Sum change driver influence).	Potential to Reduce Incidence or Impact of Change Driver						
		Corrective (22%)		Adaptive (24%)		Enhancement (41%)	Perfective (10%) Not including 'Other 3%'	
		Emergency Program Fixes (12%)	Routine Debugging (10%)	Program Interface Changes (19%)	Hardware Changes (5%)	User Enhancements (41%)	Documentation Improvements (5%)	Code Efficiency Improvements (5%)
Redefinition and refinement of the support infrastructure and / or processes.	97%	Yes. Redefinition and refinement of the support infrastructure and processes will not reduce the incidence of change drivers. However, It could significantly reduce their impact (particularly during integrity assurance).						
Application of standards for development and support.	92%	Yes.  All of these technologies have the potential to improve software quality and hence reduce the number and severity of residual errors in software products.	Yes.  All of these technologies have the potential to improve the software development process and reduce the impact of change drivers.	Yes.	No.	Yes.  These technologies have the potential to improve documentation quality.	No.  This change driver is not affected by this technology, as code efficiency improvement is an intellectual process.	
Process Improvement.	92%							
Organisationally defined and standardised CASE tools.	92%							
Organisationally defined and standardised software management tools.	92%							
Organisationally defined and standardised Software Engineering Environment (SEE).	92%	Yes.  Standardising the format of mission data will reduce the work required to fulfil the needs of multiple platforms, this will reduce the potential for error introduction.	Yes.  Standardising the format and loading of mission data will reduce the work required to fulfil the needs of multiple platforms, this will educe change driver impact.	No.	This change driver is not affected by this technology.	Yes.  Procedural documents will become common to many activities.	No.  This change driver is not affected by this technology.	
Standardised approach to data formatting and loading.	51%							

Technology Opportunity (Refer to Appendix G for descriptions)	Potential Improvement (Sum change driver influence).	Potential to Reduce Incidence or Impact of Change Driver						
		Corrective (22%)		Adaptive (24%)		Enhancement (41%)	Perfective (10%) Not including 'Other 3%'	
		Emergency Program Fixes (12%)	Routine Debugging (10%)	Program Interface Changes (19%)	Hardware Changes (5%)	User Enhancements (41%)	Documentation Improvements (5%)	Code Efficiency Improvements (5%)
Application of Object Orientation (OO).	65%	No. The utilisation of Object Orientation will not reduce the number of errors introduced during implementation.	Yes. Ease of implementation should be optimised through the principles of modularisation. This will reduce the impact of these change drivers.		No. User documentation will still be required.	No. This change driver is not affected by this technology.		
Exploitation of automatic code generation.	87%	Yes. Automatic code generation could reduce the potential for human error.	Yes. The impact of change is reduced through the automatic generation of code.		No. User documentation will still be required.	No. Automation typically produces inefficient code.		
Utilisation of reuse and multiuse.	97%	Yes. Well defined and tested software products should reduce the introduction of errors.	Yes. Once a product is developed, tested and defined it could be usable across systems or even platforms, reducing the need for bespoke development.			Yes. Optimised code can be reused or utilised by multiple applications.		
Utilisation of Commercial Off The Shelf (COTS) software.	24%	No. COTS just like bespoke software will require corrective maintenance at some time during their life.	Yes. Operating systems can offer a degree of interface separation.	Yes. Operating systems can offer a degree of hardware separation.	No. Once a product requires modification it can no longer be classified as COTS.	No. User documentation will still be required.	No. Once a product requires modification it can no longer be classified as COTS.	
Utilisation of Open Standards.	65%	No. The utilisation of Open Standards will not reduce the number of errors introduced during implementation.	Yes. Open Standards will significantly standardise program and hardware interfaces.		Yes. Standardised interfaces should improve modularity and supportability.	No. User documentation will still be required.	No. This change driver is not affected by this technology.	
Utilisation of Integrated Modular Avionics (IMA).	24%	No. The utilisation of IMA will not reduce the number of errors introduced during implementation	Yes. IMA should reduce application-to-hardware and application-to-application interface change drivers.		No. Enhancements are often within the applications layer.	No. User documentation will still be required.	No. This change driver is not affected by this technology.	

**Table H1 – Technology Supportability Benefit**

## APPENDIX I

### CAPABILITY MATURITY MODEL COMMON FEATURES

Common Features of the Capability Maturity Model<sup>28</sup> are as follows:

- “Commitment to Perform. *Commitment to Perform describes actions that the organisation must take to ensure that process is established and will endure. Commitment to Perform typically involves establishing organisational policies and leadership.*”
- “Ability to Perform. *Ability to Perform describes the preconditions that must exist in the project or organisation to implement the software process competently. Ability to Perform typically involves resources, organisational structures and training.*”
- “Activities Performed. *Activities Performed describes the activities, roles and procedures necessary to implement a key process area. Activities Performed typically involves establishing plans and procedures, performing the work, tracking it and taking corrective actions as necessary.*”
- “Measurement and Analysis. *Measurement and Analysis describes the basic measurement practices that are necessary to determine status related to the process. These measurements are used to control and improve the process. Measurement and Analysis typically includes examples of the measurements that could be taken.*”
- “Verifying Implementation. *Verifying Implementation describes the steps to ensure that the activities are performed in compliance with the process that has been established. Verifying Implementation typically encompasses reviews and audits by management and software quality assurance.*”

*Collation Page*  
*(for two sided printing)*

## APPENDIX J

### TRANSPOSITION OF TESTING INTO THE SUPPORTABILITY DOMAIN

The transposition of testing into the supportability domain is documented in Table J1. In the absence of any established body of evidence, this work is based on the author's interpretation of academic theory.

Testing Technique	Transposition
Equivalence Partitioning	This technique directly relates to the production of the Support System Model. Within this model support functions have been grouped into well-defined and separate areas of concern.
Boundary Value Analysis (BVA)	BVA focuses on the fact that a large proportion of errors occur at the point at which program choice or action is decided. For process, this is taken to relate to the way that decisions are made on how products will move through the support system.
Top Down and Bottom Up Testing	Testing can be carried out for individual support functions, parts of the support system or the support system as a whole. Where inputs and outputs are unavailable they should be simulated to facilitate early testing.
Coverage	Within the support model for each function there are a number of sub functions. These functions and sub functions can be tested for their presence and effectiveness within the proposed support system. Examining evidence of intent i.e. documented policy, assigned budgets and allocated resources, may be carried out to gain early supportability confidence.
Interface Testing	Definition of support function operation will facilitate early interface development (how products will be passed from one function to another). The interfaces themselves can then be tested to ensure that they are complete, appropriate and correct.
Special Case Testing	Each support function performs a specific and different role but special types of testing can be common to all, for instance tolerance testing can be carried out to explore resultant outcome when: <ul style="list-style-type: none"><li>• Defined rates of demand are exceeded.</li><li>• Demands are made beyond the scope of each function.</li><li>• Demands are made beyond the capability of each function.</li></ul>

Table J1 – Testing Domain Transposition

*Collation Page*  
*(for two sided printing*

