RAND PROJECT AIR FORCE

CHILDREN AND FAMILIES
EDUCATION AND THE ARTS
ENERGY AND ENVIRONMENT
HEALTH AND HEALTH CARE
INFRASTRUCTURE AND TRANSPORTATION
INTERNATIONAL AFFAIRS
LAW AND BUSINESS
NATIONAL SECURITY
POPULATION AND AGING
PUBLIC SAFETY
SCIENCE AND TECHNOLOGY
TERRORISM AND HOMELAND SECURITY

The RAND Corporation is a nonprofit institution that helps improve policy and decisionmaking through research and analysis.

This electronic document was made available from www.rand.org as a public service of the RAND Corporation.

Skip all front matter: Jump to Page 1

Support RAND

Purchase this document
Browse Reports & Bookstore
Make a charitable contribution

For More Information

Visit RAND at www.rand.org
Explore RAND Project AIR FORCE
View document details

Limited Electronic Distribution Rights

This document and trademark(s) contained herein are protected by law as indicated in a notice appearing later in this work. This electronic representation of RAND intellectual property is provided for non-commercial use only. Unauthorized posting of RAND electronic documents to a non-RAND website is prohibited. RAND electronic documents are protected under copyright law. Permission is required from RAND to reproduce, or reuse in another form, any of our research documents for commercial use. For information on reprint and linking permissions, please see RAND Permissions.

This product is part of the RAND Corporation technical report series. Reports may include research findings on a specific topic that is limited in scope; present discussions of the methodology employed in research; provide literature reviews, survey instruments, modeling exercises, guidelines for practitioners and research professionals, and supporting documentation; or deliver preliminary findings. All RAND reports undergo rigorous peer review to ensure that they meet high standards for research quality and objectivity.

REPORT

Improving Air Force Depot Programming by Linking Resources to Capabilities

Don Snyder, Julie Kim, Manuel Carrillo,
Gregory G. Hildebrandt

Prepared for the United States Air Force

Approved for public release; distribution unlimited

RAND PROJECT AIR FORCE

The research described in this report was sponsored by the United States Air Force under Contract FA7014-06-C-0001. Further information may be obtained from the Strategic Planning Division, Directorate of Plans, Hq USAF.

Library of Congress Cataloging-in-Publication Data

Improving Air Force depot programming by linking resources to capabilities / Don Snyder ... [et al.].
p. cm.
Includes bibliographical references.
ISBN 978-0-8330-5967-3 (pbk. : alk. paper)
1. United States. Air Force—Appropriations and expenditures. 2. United States. Air Force—Equipment—Maintenance and repair—Costs—Evaluation. 3. United States. Air Force—Supplies and stores—Costs—Evaluation. I. Snyder, Don, 1962-

UG633.2.I63 2012
358.4'18—dc23

2012020058

The RAND Corporation is a nonprofit institution that helps improve policy and decisionmaking through research and analysis. RAND's publications do not necessarily reflect the opinions of its research clients and sponsors.

RAND® is a registered trademark.

© Copyright 2012 RAND Corporation

Permission is given to duplicate this document for personal use only, as long as it is unaltered and complete. Copies may not be duplicated for commercial purposes. Unauthorized posting of RAND documents to a non-RAND website is prohibited. RAND documents are protected under copyright law. For information on reprint and linking permissions, please visit the RAND permissions page (http://www.rand.org/publications/permissions.html).

Published 2012 by the RAND Corporation
1776 Main Street, P.O. Box 2138, Santa Monica, CA 90407-2138
1200 South Hayes Street, Arlington, VA 22202-5050
4570 Fifth Avenue, Suite 600, Pittsburgh, PA 15213-2665
RAND URL: http://www.rand.org
To order RAND documents or to obtain additional information, contact
Distribution Services: Telephone: (310) 451-7002;
Fax: (310) 451-6915; Email: order@rand.org

Preface

One of the more challenging goals in programming is to link Air Force budget program element investments to operationally relevant capability metrics, then to use these metrics to assess any risk incurred by a proposed program objective memorandum (POM). Previous RAND Project AIR FORCE work developed a set of metrics and framework of analysis for capabilities-based programming and assessment of risks of expeditionary combat support.[1] Because of the success of that work, the RAND Corporation was asked by the Director of Resource Integration, Office of the Deputy Chief of Staff for Logistics, Installations and Mission Support, Headquarters U.S. Air Force to apply and to extend (as needed) this capabilities-based approach to the depot maintenance business areas, specifically focusing on depot purchased equipment maintenance (DPEM) as a first step, and to do so with a methodology that is broadly applicable.

DPEM comprises numerous depot-level maintenance and inspection activities on capital assets, as well as storage and other supporting activities that span numerous program elements, budget programs, and appropriations. It funds the bulk of the work done at the Air Force depots that does not involve the repair of spares or the work on modification programs. This report presents an analysis of how to better program for DPEM.

The research reported here was completed in fiscal year (FY) 2008 under the project "Capability-Based Resourcing: Balancing Depot Purchased Equipment Maintenance (DPEM) Accounts to Achieve Desired Operational Performance." The work was sponsored by the Director of Resource Integration, Office of the Deputy Chief of Staff for Logistics, Installations and Mission Support, Headquarters U.S. Air Force, and conducted within the Resource Management Program of RAND Project AIR FORCE. This report is intended to help the Air Force better measure and quantify its capabilities and risks in DPEM programming. It should be of direct interest to those working in planning and programming, depot-level maintenance, and the assessment of capabilities and risks.

RAND Project AIR FORCE

RAND Project AIR FORCE (PAF), a division of the RAND Corporation, is the U.S. Air Force's federally funded research and development center for studies and analyses. PAF provides the Air Force with independent analyses of policy alternatives affecting the development,

[1] Don Snyder, Patrick Mills, Adam C. Resnick, and Brent D. Fulton, *Assessing Capabilities and Risks in Air Force Programming: Framework, Metrics, and Methods*, Santa Monica, Calif.: RAND Corporation, MG-815-AF, 2009.

employment, combat readiness, and support of current and future air, space, and cyber forces. Research is conducted in four programs: Force Modernization and Employment; Manpower, Personnel, and Training; Resource Management; and Strategy and Doctrine.

Additional information about PAF is available on our website:
http://www.rand.org/paf/

Contents

Figures

Tables

Summary

Current planning and programming policies with a capabilities-based focus have spawned an intense interest in how to discern, define, and quantify Air Force capabilities. Capabilities-based programming has meant different things to different organizations, but the core spirit of the policy on which we focus is that it aims to provide a programmed force ready to meet a range of possible contingencies given an uncertain future. Previous RAND Project AIR FORCE research presented a framework for defining capability metrics and, using these metrics and methods, developed models for expeditionary combat-resource programming and budgeting.[1]

This work builds on and expands that previous research to embrace programming for depot-level maintenance, as illustrated by DPEM commodities. DPEM is a budgeting area that comprises numerous depot-level maintenance and inspection activities on capital assets, as well as storage and other supporting activities that span numerous program elements, budget programs, and appropriations. It funds the bulk of the work done at the Air Force depots that does not involve the repair of spares or the work on modification programs.

The challenges of relating money spent to capabilities rendered for expeditionary combat support resources and depot maintenance are quite different. For the former, the challenge lies in identifying resources needed to carry out the various planning scenarios and then determining the appropriate resource levels to meet those plans. In contrast, most depot-level maintenance is done on large capital items. The *resource levels* are already determined; the challenge is to prioritize the *allocation of sustainment funds* across weapon systems and to accurately quantify the risks engendered by various funding levels.

A first step in resolving these challenges is to define appropriate capability metrics. We sought metrics possessing three attributes: (1) that they relate directly to U.S. Department of Defense (DoD)–level planning objectives; (2) that they relate to program elements, subsets of program elements, or sets of program elements in a clearly understandable manner; and (3) that they apply across a wide range of resources.

Two metrics that satisfy these criteria and span most of depot-level maintenance are the ability to generate aerospace expeditionary force (AEF) and training sorties and the availability of Minuteman III intercontinental ballistic missiles (ICBMs). These metrics are broad enough to encompass many program elements and yet still relate resources to DoD-level planning objectives in a clear, quantitative manner. These metrics can also embrace other useful, lower-level submetrics. One example is aircraft availability, which is a submetric of the ability to generate AEF and training sorties.

[1] Snyder et al., 2009.

We have developed a prototype model and associated software tools that use aircraft availability as a metric. This model is designed to give guidance to a programmer on how to best distribute offsets among the aircraft and engine commodities of DPEM.[2] This model and its associated tools are designed with a view to be expanded in the future to include broader metrics, such as sortie generation, and to embrace a wider range of commodities.

Central to the model is a means of prioritizing across disparate weapon systems. By selecting metrics that relate to DoD-level plans, the planning objectives provide a natural prioritization. Weapon systems with high demands in plans relative to the number in inventory have a high weighting in prioritization; those with low demands in plans relative to the number in inventory have a low weighting. The caveat is that prioritization solely according to the operator's priorities will lead to nearly all the risk being placed on a small number of aircraft types. For example, if there were one mission design series that had an inventory higher than called for in war plans, a model that considers only the war plans for prioritization might suspend nearly all the maintenance on that aircraft type. Such a distribution of risk is unacceptable because of the need for training to maintain readiness.

To address the competing needs to meet warfighter priorities and to meet training requirements, the model provides a sliding rheostat for the user to select a position in the trade space between a prioritization according to operational plans and one that is a proxy for meeting peacetime training needs. Other inputs to the model are the depot induction requirements for each item by year of the Future Years Defense Program (FYDP), the topline budget for each year of the FYDP for the assets of interest, weighting factors for weapon system prioritization extracted from planning scenarios, costs of production for each depot induction, a penalty cost for deferring work, the maximum proportion of the work that can be deferred for each item, air logistic center capacities for each production task for each year in the FYDP, and force structure for each year in the FYDP.

The output of the model gives three insights. The first is how to prioritize—based on the weighting factors for weapon systems—and distribute the allocated workload and deferrals across the FYDP to avoid aircraft groundings. These outputs are the production levels for each year, what work is deferred, how much capacity is used, and how much all of this costs for each year in the FYDP. The optimal allocation will be such that no aircraft are grounded and that all work is performed by the end of the FYDP.

A second insight is how much the budget can be reduced in one or more years in the FYDP and still be able to execute the depot work at some time within the FYDP without grounding an aircraft, subject to the user-specified constraints. In this exercise, the user can decrease the budgets available in one or more years of the FYDP until the model returns an infeasible result. An infeasible result means that the program cannot be executed without grounding an aircraft. The user can then explore which constraints cause this problem (e.g., insufficient budget, lack of capacity) and which aircraft are the first to be grounded.

A third insight elucidates the operational consequences of a proposed POM. Given the underlying data used to construct the weighting factors, the resulting aircraft availability from the proposed programmed depot work can be used to see how many of each aircraft are available relative to those needed to execute the plans. This calculation gives a direct impact of the depot work on aircraft and engines in terms of operational priorities. The programmer

[2] *Offsets* are funds moved from one approved program to another approved program because the benefiting program is considered to be a higher priority than the paying program.

can explore risk expressed in these terms while varying any of the above inputs, including via the sliding rheostat that adjusts in the continuum between operational plans and training priorities.

We designed this prototype set of software tools to assist the programmer in using capability metrics to make balanced trades across weapon systems and to express quantitatively the risk those decisions carry in terms of the ability to meet DoD-level planning objectives. Unfortunately, some depot-maintenance business areas elude expression using the above metrics. One such area, which costs roughly $600 million per year, is software maintenance.

In the course of examining the numerous ways in which software maintenance challenges conventional operational metrics, we identified a number of areas open for improvement and recommend (1) clearer policy guidelines and standards for the baseline for operational acceptance of functional capability and better documentation of the software baseline; (2) additional refinements, where practical, in classifying software maintenance activities on the basis of what operational impacts can be identified; (3) standardization in data documentation across weapon systems; (4) more rigorous statistical analysis of historical data; and (5) additional analyses to establish operationally relevant metrics where possible and to identify the full extent of when analysis based on such metrics is appropriate.

Software maintenance serves as an important case study, but it also reveals four attributes that make capabilities-based programming so challenging that are also characteristic of other sustainment areas, such as sustaining engineering and technical orders. These are (1) not being easily related to operational objectives, such as sortie generation; (2) having a long lag time between funding and any operational impact; (3) whatever impact occurs does so across the entire fleet, not individual tail numbers; and (4) possessing a certain ambiguity in what constitutes a requirement. We entertained three policy options for programming these difficult business areas. We used the first two to make broader points but advocate only the last.

The first option highlights that policy decisions often drive ostensible operational consequences of sustainment. The linkage between a task, such as an inspection in a programmed depot maintenance (PDM) task list, to an operational metric, such as aircraft availability, is no more obvious than the linkage of a software change request to this metric. The consequences that not performing some PDM task would have for aircraft availability ensue from bundling the tasks together as a PDM, to be all done or all deferred. If the task is deferred, policies dictate how long until the aircraft is no longer certified to fly. Would it make sense to bundle software change requests into a task list, such as a PDM, for up or down inclusion in the budget?

Two factors make such policies as grounding a fleet for not performing software maintenance requests untenable. One is that these requests originate from the users of the weapon systems, not from engineers certifying the safety of the fleet, and hence do not carry the weight to warrant groundings. But more importantly, the impact of deferring software and related maintenance cannot be attributed to individual aircraft but must be assigned to an entire fleet. It is a very coarse, severe judgment that declares an entire fleet either available or unavailable given a diminished, but difficult-to-define, ability to carry out the designed mission.

A second option is to introduce a metric for the overall long-term health of a weapon system to complement metrics that reflect the near-term readiness of the fleet, such as aircraft availability. Each task to be funded would be assigned a score depending on the probability of occurrence of the problem it is meant to rectify and the potential impact to the mission should that problem occur. Conceptually, such a long-term health metric would be informative; in practice, it is not clear how to estimate the probabilities of problem occurrence and criticality to the mission.

A third option, and the one that we find the most promising, is a radical departure from the current paradigm of capabilities-based programming. The idea emerges from a recognition that the ability to perform any type of software change at any time on any system is more critical than any individual routine software change request. Hence, rather than evaluating risk to a program by appraising the relative merits of unfunded requests, risk would be assessed by the impact of funding on the capabilities of handling any possible emergency or urgent request related to any part of the software or interface with hardware. Software, and related areas, would be viewed as capabilities themselves, and thereby assessed in terms of readiness.

Acknowledgments

This work was sponsored by Maj Gen Polly Peyer, the Director of Resource Integration, Office of the Deputy Chief of Staff for Logistics, Installations and Mission Support for the Air Force, and could not have been done without her advice and support. (All ranks and office affiliations are those at the time of our last interactions.) We especially thank Col Amy Bouchard, Lt Col Heather Buono, Col Joe Diana, Tony Mauna, Lt Col Jeff Meserve, Lt Col Mike Morreale, Lt Col Davis Oishi, Chris Puckett, and Lorie Wade on the Air Staff for support. Discussions and recommendations during briefings with senior leadership were very helpful; we thank in particular Lt Gen Kevin Sullivan, Mike Aimone, Grover Dunn, Maj Gen Robert McMahon, Maj Gen Art Morrill, Brig Gen Frank Bruno, Brig Gen Ken Merchant, Brig Gen Art Cameron, Timothy Dues, Kathe Graham, Michele Rachie, Mark Johnson, and Dee Reardon.

The support and help from the Air Force Materiel Command were indispensable. We thank Sam Cannon for tirelessly working to put us in touch with the right experts and organizing these meetings. Meetings and discussions with many experts were very helpful; among these, we especially thank Deborah Hileman, Susan Kirby, Don Kosco, Shawn Lyman, Kyle McKown, Kel Utendorf, and Capt Brian Wilken.

We had numerous meetings with a range of subject-matter experts at the Air Logistics Centers. Those whom we met are too many to acknowledge individually, but we wish to especially thank Rick Grisenthwaite at Warner Robins Air Logistics Center, Jamie Jeter at Oklahoma City Air Logistics Center, and Glen Brown at Ogden Air Logistics Center, who organized these meetings and put us in contact with the right subject-matter experts.

Several experts helped us on numerous occasions; among these, we especially acknowledge Alberto Ayala, Mike Bauman, Stewart Carr, Bob Flinn, Dorothy Tenczar, and Jim Peebles.

At RAND, we benefited from numerous discussions and feedback; in particular, we thank Mahyar Amouzegar, Laura Baldwin, Jean Gebman, James Masters, Ronald McGarvey, Patrick Mills, Raymond Pyles, the late C. Robert Roll Jr., and Robert Tripp. Susan Woodward helped us considerably in expressing the ideas more clearly in several of the chapters.

Official reviews by Victoria Greenfield, Richard Hillestad, and Shari Lawrence Pfleeger improved the manuscript.

That we received help and insights from those thanked above should not be taken to imply that they concur with all of the views expressed in this report. We alone are responsible for any errors or oversights that might remain.

Abbreviations

A/B/L	area support, base support, and local manufacture
ACC	Air Combat Command
ACE	airframe condition evaluation
ACI	analytical condition inspection
AEF	aerospace expeditionary force
AFMC	Air Force Materiel Command
AFSOC	Air Force Special Operations Command
ALC	Air Logistics Center
AMARC	Aerospace Maintenance and Regeneration Center
AMARG	Aerospace Maintenance and Regeneration Group
APOM	Amended Program Objective Memorandum
AR	attrition reserve
ASRRC	Avionics Systems Requirements Review Conference
BAI	backup aerospace vehicle inventory
BPOS	base peacetime operating stock
CAFDEx	Centralized Access for Data Exchange
CAM	Centralized Asset Management
CDM	contracted depot maintenance
CIRF	centralized intermediate repair facility
CLS	contract logistics support
COCOM	combatant command
DMAG	depot maintenance activity group
DMISA	depot maintenance interservice support agreement

DoD	U.S. Department of Defense
DP	depot possessed
DPEM	depot purchased equipment maintenance
DT/OT	development test/operational test
EEIC	element of expense/investment code
FY	fiscal year
FYDP	Future Years Defense Program
GAMS	General Algebraic Modeling System
GWOT	global war on terrorism
ICBM	intercontinental ballistic missile
ISP	Integrated Security Posture
J-8	Force Structure, Resources, and Assessment Directorate
JEIM	jet engine intermediate maintenance
JSCP	Joint Strategic Capabilities Plan
LIMS-EV	Logistics Installations and Mission Support, Enterprise View
LSR	logistics support requirement
MAJCOM	major command
MCRT	Multistage Improvement Process Cockpit Review Team
MDS	mission design series
MSD	materiel support division
NDS	National Defense Strategy
NMS	National Military Strategy
NSS	National Security Strategy
O&M	operations and maintenance
OCM	on-condition maintenance
OMEI	other major end item
OPLAN	operation plan
OSD	Office of the Secretary of Defense
PAF	Project AIR FORCE
PAI	primary aircraft inventory

PCN	program control number
PDAI	primary development/test aerospace vehicle inventory
PDM	programmed depot maintenance
PDS	Program Data System
PGM	program group manager
PMAI	primary mission aerospace vehicle inventory
POAI	primary other aerospace vehicle inventory
POM	program objective memorandum
PPC	possession purpose code
PTAI	primary training aircraft inventory
RDT&E	research, development, test, and evaluation
RGC	repair group category
SCR	software change request
SCU	software change upgrade
SPM	system program manager
SRA	software requirements application
SRRP	software requirements review process
SW	software
UPNR	unit possessed, not reported
WRE	war readiness engine

CHAPTER ONE

Introduction

Funding is often insufficient to meet all the stated requirements of the U.S. Air Force, so building the budget submission and executing the budget each year incur risk. Managing that risk necessitates prioritization, which, in turn, introduces several programming challenges. Especially on the combat support side, the ties between programming and the ultimate objective—to perform operational missions—for many areas of funding can be difficult to isolate and quantify. Also, reducing the funding, and thereby reducing the capability, of one programming area can negatively affect the capability of another when their capabilities are mutually dependent. On top of these challenges, the overarching guidance for prioritization in programming is to meet future security environments, requirements that are intrinsically uncertain.

Several years ago, the U.S. Department of Defense (DoD) adopted capabilities-based planning and programming in part to address these issues.[1] A central concept of capabilities-based programming is to discern and quantify the links between funding a program and the resulting operational capability delivered by that program. These insights furnish the programmer with the insights to program for a desired mix of capabilities against a set of plans, given budgetary guidelines.[2] An additional element of this programming paradigm is to build the program objective memorandum (POM) against a spectrum of possible future security environments. Unfortunately, these goals have proven easier to state than to fulfill.

Previous RAND research presented a methodology that accomplishes these goals for a wide range of deployed combat support resources.[3] The approach has three main components. The first is to develop a suitable set of capability metrics. These metrics relate programming directly to DoD-level planning guidance; are expressed in terms of programmable entities, such as program elements or identifiable subsets of program elements; and apply across a wide range of program elements. By expressing capabilities in this manner, the overall capability (or risk) of programming decisions is immediately expressible in terms of the ability to achieve national objectives, is usable by programmers, and enables programming trades among disparate programming areas. The set of metrics recommended in that monograph was the ability to conduct the various operational-level activities specified in the Defense Planning Scenarios.

The second component of the method is to quantify the resources needed to achieve the goals measured by those metrics. In the case of combat support resources, the objective is to

[1] U.S. Department of Defense, *Quadrennial Defense Review Report*, Washington, D.C., February 6, 2006.

[2] We use the term *programmer* in the sense of one involved in building the Air Force program, not in the software sense, unless otherwise noted.

[3] Don Snyder, Patrick Mills, Adam C. Resnick, and Brent D. Fulton, *Assessing Capabilities and Risks in Air Force Programming: Framework, Metrics, and Methods*, Santa Monica, Calif.: RAND Corporation, MG-815-AF, 2009.

determine which resources and how much of those resources are required by each contingency, operation, and vignette in the Defense Planning Scenarios. It was recommended that a set of rules be compiled to automate this process as much as possible and to enable rapid assessment of multiple scenario sets. The feasibility of this approach has been previously established.[4]

The third component is an analytical tool that combines the objectives set by the DoD plans with the requirements determined by a rule set to nominate a POM for the programmer. This tool also assesses that POM's success in achieving DoD-level plans and quantifies the risks incurred. The tool developed operates in multiple modes: It can determine the resources needed at minimal cost (procurement and sustainment) to achieve the planning objectives across the Future Years Defense Program (FYDP), or it can determine the allocation of money (procurement and sustainment) that will maximize capability (as measured against a set of planning objectives), given fiscal guidance across the FYDP.[5] These assessments can be done against a single, deterministic future or against a portfolio of possible futures. The latter yields a POM that is more robust in light of an uncertain future security environment.

The goal of this report is to extend that methodology for capabilities-based programming to a large subset of depot-level maintenance work and to illustrate this expanded methodology on depot purchased equipment maintenance (DPEM). DPEM and other depot funding areas present complications to analysis not confronted in the previous work. The goals remain as outlined above: Establish appropriate metrics, determine the requirements to meet plans in terms of those metrics, and develop appropriate tools to nominate programming strategies to the programmer to meet plans at minimal cost or to maximize capability against plans, given fiscal constraints.

An important difference between programming for expeditionary combat support resources and depot-level activities lies in the second component: requirements. For combat support resources, the requirements consist of a combination of the authorized levels to procure and the costs of sustaining those resources. Sustainment costs include operations and maintenance (O&M) costs and costs of reconstitution after use, which depend on the frequency of use specified in plans.

In contrast, depot-level activities are maintenance activities required for capital assets already procured. Maintenance activities include (but are not limited to) repairing broken parts, performing inspections, modifying and upgrading hardware and software, and sustaining engineering. These activities are performed on the full spectrum of sustained resources, including aircraft, missiles, munitions, software, and a wide range of ground equipment and vehicles. Not performing necessary maintenance, or performing that maintenance inefficiently, causes resources to be unavailable due to not being mission capable. For assets of high capital investment, such as aircraft and missiles, deferring maintenance is not a viable long-term programming option.

How much of these maintenance activities are expected in any year emanates from an extensive requirements process performed across all weapon systems and applicable business areas under the purview of the Centralized Asset Management (CAM) office at Air Force

[4] Don Snyder and Patrick Mills, *Supporting Air and Space Expeditionary Forces: A Methodology for Determining Air Force Deployment Requirements*, Santa Monica, Calif.: RAND Corporation, MG-176-AF, 2004; Don Snyder and Patrick Mills, "Air Force Deployments: Estimating the Requirement," *Air Force Journal of Logistics*, Vol. 30, No. 2, Summer 2006, pp. 4–9.

[5] Snyder et al., 2009.

Materiel Command. The focus of the analysis in this report is not to challenge the requirements identified in this process; the purpose is to explore how to prioritize these requirements once determined.[6] We follow the recent policy, instituted under the CAM initiative, of prioritizing sustainment tasks by weapon system rather than business area.[7] Prioritization occurs at two stages: first among tasks within the various weapon systems, and then among those weapon systems.

The first challenge in prioritization lies in linking dollars spent on these maintenance activities to the ability to carry out prescribed operations specified in DoD-level planning objectives. As in the case of other programming areas, this means tying logistics performance metrics to operational performance metrics. A natural metric for maintenance activities is the availability of the weapon system. However, linking expenditures on specific depot-level maintenance tasks, and even groups of tasks, to weapon system availability is problematic. It is often unclear how not performing some task might eventually affect a weapon system, or in some cases, which weapon systems.

The second challenge is how to adjudicate between the desire to maximize capabilities called for in operational plans, or required by a combatant commander, and the need to meet peacetime training requirements (and the political pressure to sustain a force structure composed of large capital investments). For example, plans might indicate a high priority for a frequently used aircraft of small inventory, such as an E-3 Sentry Airborne Warning and Control System. Yet, it may not be a viable option to cut the maintenance on an aircraft of less demand and higher inventory in order to meet the needs of the E-3, especially if that requires suspending or deferring all maintenance on those aircraft less demanded in plans. Some balance must be struck.

A third challenge is to pull all of these elements together in the form of a software tool, or set of tools, that a programmer can use to nominate and assess programming solutions. Given the time frame of some decisions, these tools should execute rapidly and be easy to use. Ideally, these tools would employ a model that is sufficiently transparent that decisionmakers have confidence in its assessments. In turn, it is preferable that these assessments describe the capabilities a program delivers, and the risks it incurs, in terms that relate to DoD-level planning priorities rather than Air Force missions.

Increased costs for maintaining aging legacy systems and increased costs for maintaining newer, more complex systems are likely to increase the near-future operations and maintenance costs of aircraft, and it is likely that fiscal constraints will strain the Air Force budget.[8] These circumstances make the resolution of these challenges of capabilities-based programming for depot-level activities acutely important. In this report, we examine these issues in considerably more detail.

[6] The identification of requirements has its uncertainties due to the imperfect ability to predict future usage (such as the flying hours for aircraft) and the maintenance that that usage will provoke. These uncertainties lie beyond the scope of this report. We assume the requirements identified in the requirements process to be valid and consider only their prioritization.

[7] Focusing on weapon systems rather than business areas enables a better sustainment plan for large capital investments, one that can produce better availability of those assets. Congress, nonetheless, appropriates money according to business areas, so the Air Force needs to continue to be able to analyze its expenditures both by weapon system and by business area. For this reason, and to confine the scope of the work to a manageable area, this report focuses on prioritization of weapon systems within a single business area: DPEM.

[8] Adam Talaber, *Long-Term Implications of Current Defense Plans: Summary Update for Fiscal Year 2008*, Washington, D.C.: Congressional Budget Office, December 2007.

We concentrate on four themes: (1) defining suitable metrics of capability that relate to DoD-level planning objectives, expressing them in terms that programmers can use, and applying them across program elements; (2) prioritizing among weapon systems; (3) presenting a prototype tool to inform programming decisions for depot-level activities; and (4) linking capabilities to resources and the programming of deeply problematic areas, such as software maintenance. Although these discussions will focus on the DPEM business area, we note when these concepts are easily extended to other depot business areas and when additional complications arise in such extensions.

Whatever the merits of the overall paradigm of capabilities-based programming as an instituted policy, the ultimate desideratum of linking expenditures for resources to the capabilities that they provide is quite useful. Defining and, to the extent possible, quantifying these linkages provides a firm foundation for reducing risk in programming trade-offs and for expressing the capabilities and risks of a proposed program.

But achieving this goal of linking resources to capabilities has one additional challenge: The practical need for the link occurs in two different contexts. The first need arises during the requirements and POM build each year. This process must be done in some detail so as to be as unassailable as possible when the build goes to DoD and Congress. The process includes defining capabilities, determining their costs, and balancing them across programs, and therefore occurs over a time line of many months. In contrast, once the Air Force POM/presidential budget reaches the latter stages of deliberation, many proposed cuts to programs are considered that arise external to the Air Force, creating a need to understand how programs may be mutually interdependent, and therefore how to balance cuts guided by assessments of capabilities and risks. These decisions must be made in hours or days.

Given the differing demands on fidelity and time, decision support tools to assist the decisionmaker in these two contexts may be distinct, but the framework in which they operate should be the same. The specific objective of the work described in this report is to develop the overall framework for depot-level programming for both of these contexts and to develop a prototype tool that would be most useful during the latter stages of POM deliberation.

The next chapter is a conspectus on depot-level activities that provides the background necessary to understand the discussion that follows on capabilities-based programming for DPEM and other depot-level activities. Chapter Three develops metrics for programming and discusses general issues regarding linking expenditures for sustainment to operationally relevant measures, such as meeting DoD-level planning objectives. Chapter Four addresses how to prioritize expenditures among weapon systems, both for programming and for execution. Chapter Five presents a prototype tool for programmers to reproducibly inform programming decisions with assessments of how programming decisions affect capabilities and risks, as measured against DoD-level planning objectives. Chapter Six discusses the area of software maintenance, with recommendations on how to improve programming in this area. Chapter Seven draws general inferences about troublesome attributes of software maintenance for capabilities-based programming and suggests policy options for these areas. This chapter also summarizes the report and presents some broader conclusions.

CHAPTER TWO

Depot Purchased Equipment Maintenance

We begin with an overview of depot-level maintenance, with a focus on DPEM and the associated requirements and prioritization processes. Readers familiar with these areas might wish to skip to the next chapter, in which we take up measures of capability and address issues of tying expenditures to capabilities.

Depot-level maintenance is largely performed in several business areas: DPEM, contract logistics support (CLS), sustaining engineering, technical orders, materiel support division (MSD) exchangeables, and modifications. Modifications are funded through procurement appropriations; all others are funded through O&M appropriations. In this chapter, we describe DPEM in some detail and then present briefer summaries of the other depot activities.

Depot Purchased Equipment Maintenance Overview

DPEM comprises numerous depot-level maintenance and inspection activities on capital assets, as well as storage and other supporting activities. DPEM spans numerous program elements, budget programs, and appropriations. It is defined as a set of Air Force element of expense/investment codes (EEICs), summarized in Table 2.1.[1] The eight commodity groups, or categories, are (1) aircraft, (2) engines, (3) software, (4) other major end items (OMEI), (5) non-MSD exchangeables, (6) missiles, (7) area support, base support, and local manufacture (A/B/L), and (8) storage.

Aircraft

The aircraft category of DPEM includes two types: programmed work (repair group category [RGC] A), and unprogrammed depot-level maintenance (RGC B). Most of the work for RGC A is composed of inspections and repairs of defects noticed during inspections scheduled on a calendar, as opposed to flying-hour, basis. These include structural inspections for fatigue and corrosion and inspections of wire bundles for wear. Examples of programmed work are programmed depot maintenance (PDM), which is the calendar-based inspection and correction of defects that cannot be performed outside of the depot; analytical condition inspections (ACIs), which are a set of inspections on a sample of the fleet to discover unknown defects beyond the work done during PDM; airframe condition evaluation (ACE), which is performed by depot field teams or contract field teams and collects engineering information on potential aircraft

[1] An EEIC is an accounting category and is essentially a subdivision of a Treasury appropriation code.

Table 2.1
Depot Purchased Equipment Maintenance Element of Expense/Investment Code Structure

	EEIC			
Commodity	**Contract**	**Organic**	**CDM Non-DMAG**	**DMISA**
Aircraft	54100	54101	56010	54102
Engines	54300	54301	56030	54302
Software	54000	54001	56000	54002
OMEI	54400	54401	56040	54402
Non-MSD exchangeables	54500	54501	56050	54502
Missiles	54200	54201	56020	54202
A/B/L	54600	54601	—	—
Storage	54800	54801	—	—

NOTE: CDM = contracted depot maintenance. DMAG = depot maintenance activity group. DMISA = depot maintenance interservice support agreement.

defects to establish when an aircraft should receive on-condition maintenance (OCM); and OCM, which is a program to repair specific defects discovered in ACE in selected aircraft.[2]

The DPEM aircraft category also includes unscheduled work in RGC B. Most of this work is done by depot field teams and includes work requested of the depot by the possessing unit and repair of major structural damage to aircraft.[3] Note that the DPEM aircraft category excludes depot-level modifications of aircraft, although modification-funded work might occur on an aircraft at depot simultaneously with DPEM-funded work, such as PDM.

Engines

Much engine and engine module repair at depots is done under DPEM funding. This work can be divided into two areas: scheduled and unscheduled maintenance. For scheduled maintenance, engines are overhauled either on a flying-time or calendar-based interval (funded in RGC E). Unscheduled maintenance is repair of engines that cannot be done by the possessing unit. The engine category also includes engine ACIs (funded in RGC F), two-level maintenance of engines, and other tests and inspections.[4]

[2] Note that not all aircraft undergo PDM, and those that do, do so under differing schedules. See Secretary of the Air Force, Technical Manual: Depot Maintenance of Aerospace Vehicles and Training Equipment, Technical Order 00-25-4, change 2, June 1, 2004.

Regarding ACIs, see Commander Air Force Materiel Command, Maintenance: Analytical Condition Inspection (ACI) Programs, Air Force Materiel Command Instruction 21-102, January 29, 2002

[3] Secretary of the Air Force, Technical Manual: Maintenance Assistance, Technical Order 00-25-127, January 15, 2008a.

[4] Secretary of the Air Force, Maintenance: Selective Management of Selected Gas Turbine Engines, Air Force Instruction 21-104, December 11, 2007; Secretary of the Air Force, Technical Manual: Aircraft Engine Operating Limits and Factors, Technical Order 2-1-18, change 4, June 15, 2006; Secretary of the Air Force, Technical Manual: Comprehensive Engine Management System Engine Configuration, Status and TCTO Reporting Procedures, Technical Order 00-25-254-1, change 11, July 1, 2008b.

Software

This DPEM category includes all software maintenance for fielded systems and is funded in RGC S. This maintenance is in the form of correcting, perfecting, and adapting existing fielded software; it does not include software changes performed during a modification, which are funded under procurement appropriations. This budget category also includes the necessary testing, evaluation, validation, and rehosting of the software on the associated platform. Major categories of software maintenance are operational flight program, automated test equipment, test program sets, electronic warfare, and mission planning system software. Software maintenance is classified as either emergency, urgent, or routine. Emergency and urgent changes are unscheduled changes that are typically performed rapidly. Routine maintenance is normally done in batches of changes called blocks, which are typically sets of individual software changes that are performed together in a single release on some calendar basis. How these are done differs across weapon systems. A block release each 18 months is typical.

Software maintenance practices, the challenges they present for capabilities-based programming, and potential policy changes are discussed in detail in Chapters Six and Seven.

Other Major End Items

This category includes all programmed (RGC G) and unprogrammed (RGC H) depot-level repair of hardware not included in the other categories. Some examples are cryogenic systems, hush houses, missile transportation and handling equipment, mobile command and control vehicles, railway equipment, shelters, space systems (such as antennas), and special-purpose vehicles.[5]

Non–Materiel Support Division Exchangeables

This category includes the repair, maintenance, and calibration of stock-fund exempt items, many of which are in the classified equipment computation system (RGCs J, K, and L). Examples are rocket motors for air-to-ground and air-to-air missiles, various optical equipment, and other specialty support equipment.[6]

Missiles

This category includes a wide range of depot-level work on Minuteman III intercontinental ballistic missiles (ICBMs), both programmed (RGC C) and unprogrammed (RGC D). Maintenance includes but is not limited to ACI, structural integrity testing, destructive tests, PDM on launch facilities and launch control facilities, costs of storage, input to storage, preparation for movement, and depot-level field teams at the missile sites called Rivet Minuteman Integrated Life Extension teams.

Area Support, Base Support, and Local Manufacture

The category of A/B/L spans four RGCs (M, N, P, and R). An example of area support is calibrating equipment in an Air Force precision measurement equipment laboratory. An example of base support is support given via a host-tenant support agreement, such as filling gas bottles

[5] Air Force Materiel Command, "The Depot Purchased Equipment Maintenance (DPEM) Process," *Financial Management Handbook*, May 2005, p. 92-43.

[6] AFMC, 2005, p. 92-44.

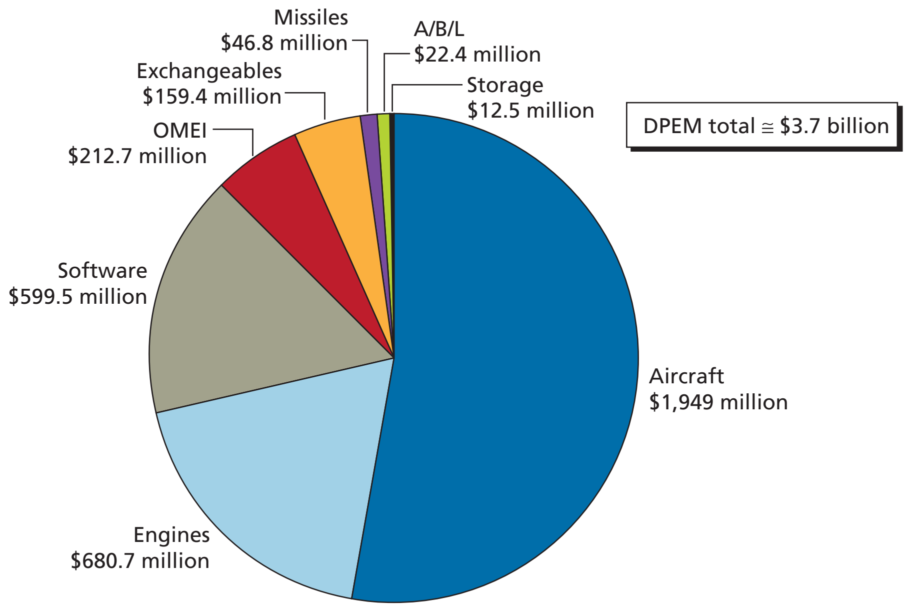
and other common tasks. Local manufacturing is authorized under certain conditions, such as to prevent work stoppages.[7]

Storage

DPEM storage is dominated by long-term storage of retired aircraft at the Aerospace Maintenance and Regeneration Group (AMARG) at Davis-Monthan Air Force Base in Tucson, Arizona.[8]

Figure 2.1 shows the spending on each of these DPEM categories for the fiscal year (FY) 2008 presidential budget submission. The total depot-related sustainment budget is approximately $16 billion, making DPEM roughly a quarter of the depot expenses. Within DPEM, aircraft is by far the largest category. Engines and software, which are comparable in size, make up the bulk of the remaining DPEM budget. For this reason, much of the discussion in the remaining portions of the report focuses on these three DPEM categories.

Figure 2.1
Depot Purchased Equipment Maintenance Commodity Groups in the FY 2008 Presidential Budget Submission



SOURCE: Personal communication from Weapons System Sustainment Division, Director of Resource Integration, Office of the Deputy Chief of Staff for Logistics, Installations and Mission Support, Headquarters U.S. Air Force.
RAND *TR905-2.1*

[7] AFMC, 2005, pp. 92-44–92-45.

[8] AMARG is part of the Ogden Air Logistics Center 309th Maintenance Wing. Prior to being subsumed under this wing, AMARG was formerly known as the Aerospace Maintenance and Regeneration Center. See Commander Air Force Materiel Command, Maintenance: AMARC Storage and Withdrawal of Aircraft and Equipment, Air Force Materiel Command Instruction 21-123, December 15, 2006.

Other Depot Activities

It is the goal of this study to apply capabilities-based programming across a range of depot business areas, using DPEM as a case study. Beyond DPEM, a large area of funding is MSD exchangeables. This includes the repair of spare parts not reparable at the base (organizational and intermediate levels). This business area has many characteristics in common with the engine category of DPEM. Engines can be viewed as spare parts that enable the operation of a weapon system. The only substantial difference is that engines are a larger capital investment than most other spares, and engines undergo considerably more scheduled maintenance than most spares. Neither of these attributes would impede the methods and metrics developed in this report for engines from being extended to MSD exchangeables.

Other major business areas have historically been managed under the Weapon System Management Support data system: CLS, sustaining engineering, and technical orders.[9] The work done under CLS is indistinguishable from work done under other depot business areas. It differs in being conducted according to a performance-based contract.[10] Each CLS contract is a separate legal instrument. Although the same concepts developed in this report might be applied to CLS programming, a general barrier to doing so is the terms under which each of these contracts are written and the visibility that the Air Force has into how the contractor executes the program. Prioritizing and performing enterprise-wide trading among business areas that include CLS are confronted with these impediments unless these contracts are written with terms that permit these types of programming trades.

The overall purpose of sustaining engineering is to ensure the long-term viability of the fleet and to mitigate long-term sustainment costs; this category includes expenses to conduct the necessary studies and tests to determine which inspections are needed for maintaining the structural integrity of the fleet. The technical order category includes the money needed to document procedures. As we argue elsewhere in this report, both of these categories share characteristics with some areas of DPEM—in particular, software maintenance.

Sustaining-engineering expenditures generally save money and reduce risk in the long term. Excessive cuts to this area will incur increasing costs in other expenditures in the future or could lead to a grounding or to early retirement of a fleet. Hence, the impacts of funding on capabilities have a delayed effect relative to many other funding areas. For technical orders, funding enables other sustainment activities, analogous to the A/B/L area in DPEM. These themes, especially that of delayed impacts, are taken up elsewhere in this report.

Modifications are not considered further in this report, as they are funded under procurement appropriations.

The Centralized Asset Management Initiative

Before entering the discussion in the next chapter on ways to improve measures and processes, we highlight some aspects of the current process. Much of the requirements and budget build is now overseen by the CAM office at the Air Force Materiel Command (AFMC). Besides

[9] CLS work comprises $3.8 billion in FY 2008, comparable to DPEM.

[10] See Defense Acquisition University, *Performance Based Logistics: A Program Manager's Product Support Guide*, Fort Belvoir, Va.: Defense Acquisition University Press, March 2005.

DPEM, CAM governance includes depot-level reparables; sustaining engineering; technical orders; CLS; aviation petroleum, oil, and lubricants; flying-hour consumables; and support equipment.

The CAM initiative is an outgrowth of the e-Log21 campaign that began in 2003. The goals of that campaign were to improve performance (as measured by such metrics as aircraft availability) and efficiency and reduce cost.[11] Before CAM, the major commands (MAJCOMs) led in the determination of depot maintenance requirements, constructed that portion of the POM for the Air Staff, and, most importantly, controlled the execution of those appropriations. The requirements build, the appropriation of funds, and the execution of those funds were compartmented by business area (e.g., DPEM, CLS, sustaining engineering) rather than by weapon system.

The focus on business area and MAJCOM control were perceived as weakening an enterprise-wide management of weapon systems, resulting in suboptimal performance, as measured by such metrics as aircraft availability. In December 2005, the Air Force Chief of Staff approved CAM.[12] For the themes pursued in this report, the principal changes with CAM were to subordinate the role of the MAJCOMs in the prioritization of requirements relative to the system program managers (SPMs) and program group managers (PGMs); centralize programming, budgeting, and execution in AFMC rather than the MAJCOMs; and embrace performance-based logistics, which is to say, to adopt goals of weapon system performance.

These changes are being instituted in a time-phased, spiral development that began in FY 2007. Individual business areas are being brought under CAM sequentially over a number of years. At the time of this writing, the Air Reserve Component (ARC) was not yet participating in this centralization effort.

The Centralized Asset Management Requirements and Prioritization Processes

Fundamental to sustainment funding is determining the anticipated requirements. These include both scheduled inductions to the depots, such as PDM, and unscheduled inductions, such as engine breaks that cannot be performed at the unit level. Generally, scheduled inductions are determined by engineering factors, including information gleaned from inspections and efforts done under sustaining engineering. Estimates of unscheduled maintenance are generally guided by historical break rates and the anticipated operational tempo given by the flying-hour program for each year in the FYDP. For such items as special-purpose vehicles, support equipment, and similar assets, estimates of depot-level inductions are made from a running average of historical data, often over the past three years. All of these requirements are also affected by any anticipated changes in the force structure.

Note that some requirements, such as PDM, apply to a single weapon system; these are called *weapon system requirements*. Others, such as vehicles, support equipment, and some electronic-warfare software, support a number of weapon systems; these are called *common requirements*. The distinction is important. In the former, a single system program manager

[11] Aircraft availability is the time-averaged fraction of the fleet that is mission capable or partial mission capable relative to the total aircraft inventory. An aircraft in depot maintenance counts against aircraft availability.

[12] At the time, the initiative was known as *Future Financials*.

can determine and prioritize requirements, whereas the latter is complicated by the need for metrics that apply across weapon systems and the imperative to resolve potentially differing priorities of multiple program managers. Some of these decisions might have strategic implications, such as whether a munition can still be carried by all relevant host platforms after it is changed.

Details of how requirements are defined and validated vary considerably across the different business areas. Elsewhere in this report, we discuss some of the areas in more detail as needed. Here, we focus on a top-level summary of how prioritization of requirements is currently done, both within and among weapon systems.

Within weapon systems, the prioritization policy is to use four criteria, in order of descending importance: safety, readiness, mandate, and sustainability. Safety is the extent to which the proposed work would decrease risk to life, health, property, and the environment. Readiness is the degree to which the requirement would enable the full mission capability of relevant weapon systems to be realized. Mandate is whether the work is mandated by an authority outside the program manager and outside the duration of the compliance period. Finally, sustainability is the degree to which the work would increase the reliability of the relevant weapon systems.

These assessments are made qualitatively by the program managers in consultation with the lead MAJCOMs. Much of this ranking is done at the program control number (PCN) level of work, ideally by a reproducible, transparent algorithm. Once this is accomplished, a further prioritization is done across weapon systems.

This enterprise prioritization continues to be developed and refined. It is conducted by AFMC, the lead MAJCOMs, and the Air Staff. The process as of FY 2008 treats only aircraft and ranks them into three tiers of priority. The highest tier gives highest priority. Still, placement of a weapon system in the highest tier has not forced full funding, nor has placement in the lowest tier forced zero funding. A fully reproducible methodology for performing this prioritization had not yet been accepted by all parties as this project was being executed. Much of the following discussion treats how to approach an enterprise prioritization, the strengths and weaknesses of differing approaches, and issues in implementation.

In the next chapter, we explore the issues surrounding the selection of capability metrics and how these metrics can be used to establish priorities and assess risk.

CHAPTER THREE

Metrics for Assessing Capabilities and Risks in Depot Purchased Equipment Maintenance

The necessary sequence of undertakings for capabilities-based planning and programming for depot-level activities is first to establish metrics for performance goals that relate to Office of the Secretary of Defense (OSD)–level planning objectives, then to define and validate anticipated requirements to meet those objectives, and finally to set priorities among these requirements. The latter activity can be greatly facilitated by analytical tools. This chapter discusses the definition of metrics. The following chapter discusses prioritization and is followed by development of the underlying formalism for programming tools in Chapter Five. We do not examine the requirements process in general in this report, but we touch on some aspects of the details of software requirements in Chapter Six.

In taking up the issue of the appropriate metrics for capabilities-based programming, the focus of the discussion and modeling is on the DPEM business area. Nevertheless, we strove to keep the methodology and concepts as comprehensively applicable as possible to the whole suite of depot-level business areas, as well as to base-level maintenance activities. In the following discussion, we point out where the concepts and models might not apply or where additional work might be needed to enhance their applicability.

Goals for Capability Metrics

We seek the same goals as argued in previous RAND work on capabilities-based programming:[1] Capability metrics are best expressed as impacts on operational performance, should be related directly to national-level planning guidance, and, as much as possible, should pertain across multiple weapon systems and business areas.

The latter goal should not be interpreted to mean that a single metric should be found for all possible weapon systems and business areas. The benefits of such a single metric would be illusory because its overreaching breadth would diminish its utility. Take the example of finding a metric flexible enough to express the operational impact of funding for PDM of C-5s and depot inspections of Minuteman III ICBMs, both of which fall under DPEM funding. These systems support such vastly different missions that any metric embracing both will inevitably introduce awkward linkages between funding and capabilities, and the measure would inadequately capture the essential elements of the mission fulfilled by either platform. Instead, we urge that a parsimonious set of metrics be sought, facilitating a maximal degree of potential

[1] Snyder et al., 2009.

trading among programming areas, while retaining a directness of linkages between the supported system and some operationally relevant performance measure.

Of these objectives, the least progress has been made on the first: developing metrics that relate to operational objectives. In selecting these metrics, it is helpful to bear in mind three challenges. The first is to identify the metrics that best characterize the operational performance. A metric currently used widely is weapon system availability. Availability is well defined for aircraft, and, in that context, it means the proportion of time that the total aircraft inventory is mission capable or partial mission capable. In practice, this means computing the time aircraft in the fleet were mission capable or partial mission capable and dividing this number by the total time that the aircraft were in this status or were not mission capable, were depot possessed, or were unit possessed and not reported.[2]

In isolation, weapon system availability expresses the proximal goal of sustainment, which is to maximize the time that a system can perform its designed operational roles. Availability does have some shortcomings, nonetheless: It fails to reveal the interdependencies of investments in various business areas. For example, what the operator needs might be the ability to deliver a precision-guided bomb to a particular desired mean point of impact. That an F-16CG is available to do this partially enables that mission, but, if other supporting capabilities—such as munitions, munitions uploaders, fueling support, and so forth—are not present, the mission will, nonetheless, fail. Availability metrics also present a challenge in linking many funding areas to the performance metric. In DPEM, software maintenance is most prominent; it has no clear link to any current measure of weapon system availability. Finally, availability, as defined for aircraft, does not always transfer adequately to other systems. One example is ICBMs, where the alert rate is a more apt measure.[3]

The second challenge is to balance sustainment funding in business areas that directly affect the operational performance with those that affect performance indirectly. Even though a metric may capture the linkage to operational performance, it is not helpful for programming if it gives an undesired bias toward some business areas at the expense of others. Funding tends to be more protected for business areas that lie ostensibly closer to the combat mission. Said another way, it is easier to defend spending on sustaining a combat aircraft than spending on sustainment for training aircraft, or for supporting systems, such as vehicles and software.

But this perceived higher proximity to the mission is temporal only. Failure to meet sustainment needs in any area will eventually affect the ability to perform missions. For example, failure to perform sustainment that is perceived to be indirect, or that might have impact only in the future, will eventually have a direct impact on the ability to perform the operational mission. A poignant example would be the failure to address aging-aircraft issues, such as the identification and repair of fatigue cracks and corrosion. This oversight might not affect any mission performance in the near term, but an unaddressed fatigue issue could ground the entire fleet in the future. Metrics should not unduly bias performance evaluation in favor of near-term effects to the detriment of addressing longer-term effects.

The third challenge is to determine the optimal target value for each metric. The next chapter takes up this theme, but here we note that, when defining the appropriate metrics, it is imperative that some guidance exists for prioritization. Merely attempting to maximize each

[2] Secretary of the Air Force, Maintenance: Equipment Inventory, Status and Utilization Reporting, Air Force Instruction 21-103, December 14, 2005.

[3] The alert rate is the percentage of mission-ready missiles relative to the number of deployed missiles.

performance measure risks improper balance of capabilities across systems and risks funding some areas at too high a level, where large increases in funding return only small marginal gains in capability.

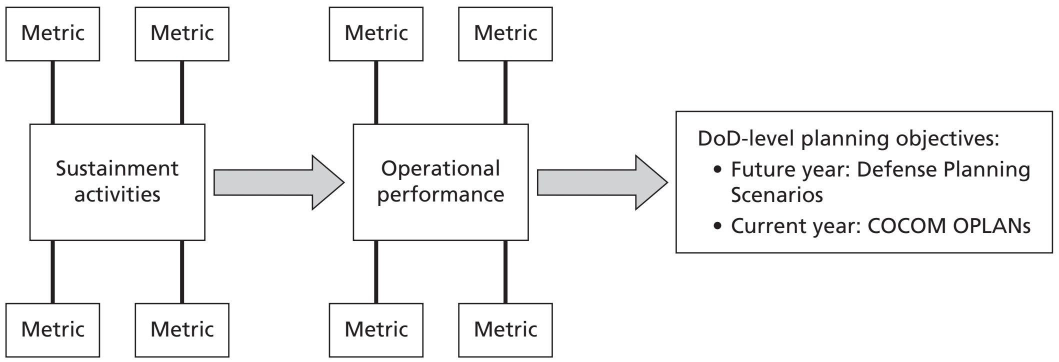
Options for Capability Metrics

We now turn to how to address these challenges. With a view toward eventual prioritization, Figure 3.1 schematically depicts the flow of activities from sustainment funding to performing missions specified in DoD-level plans. Although, in practice, effects flow from left to right in the diagram, the logical progression for defining metrics is to work from right to left. The national planning objectives, as expressed in DoD-level plans, provide the necessary direction for prioritization, and hence a foundation for defining performance metrics.

Accordingly, the process of defining metrics most naturally begins with DoD plans, moves on to determine the appropriate operational metrics that capture the Air Force contribution to these planned missions, and then seeks the appropriate manner in which sustainment activities affect these operational metrics. We pursue in detail in the next chapter the use of DoD plans for prioritization, but, for the purposes of defining metrics, we stress the need not only to link sustainment funding to operational performance but also to link the operational performance metric to DoD planning objectives.

In the next chapter, we take up planning objectives in more detail, but it is important to note here that DoD planning objectives are principally driven by the Quadrennial Defense Review, the Defense Planning and Programming Guidance, and the Guidance for the Employment of the Force.[4] These guidance documents engender two independent set of plans: current-year plans developed and maintained by the combatant commanders, and the future-year plans developed in partnership by the Office of Cost Assessment and Program Evaluation in OSD and the Force Structure, Resources, and Assessment Directorate of the Joint Chiefs of Staff. The future-year plans are to be used to guide programming by all services, whereas

Figure 3.1
Block Diagram of Linking Sustainment to DoD-Level Planning


Metric
Metric
Metric
Metric
Sustainment activities
Operational performance
DoD-level planning objectives:
• Future year: Defense Planning Scenarios
• Current year: COCOM OPLANs
Metric
Metric
Metric
Metric

NOTE: COCOM = combatant command. OPLAN = operation plan.
RAND *TR905-3.1*

[4] DoD, 2006.

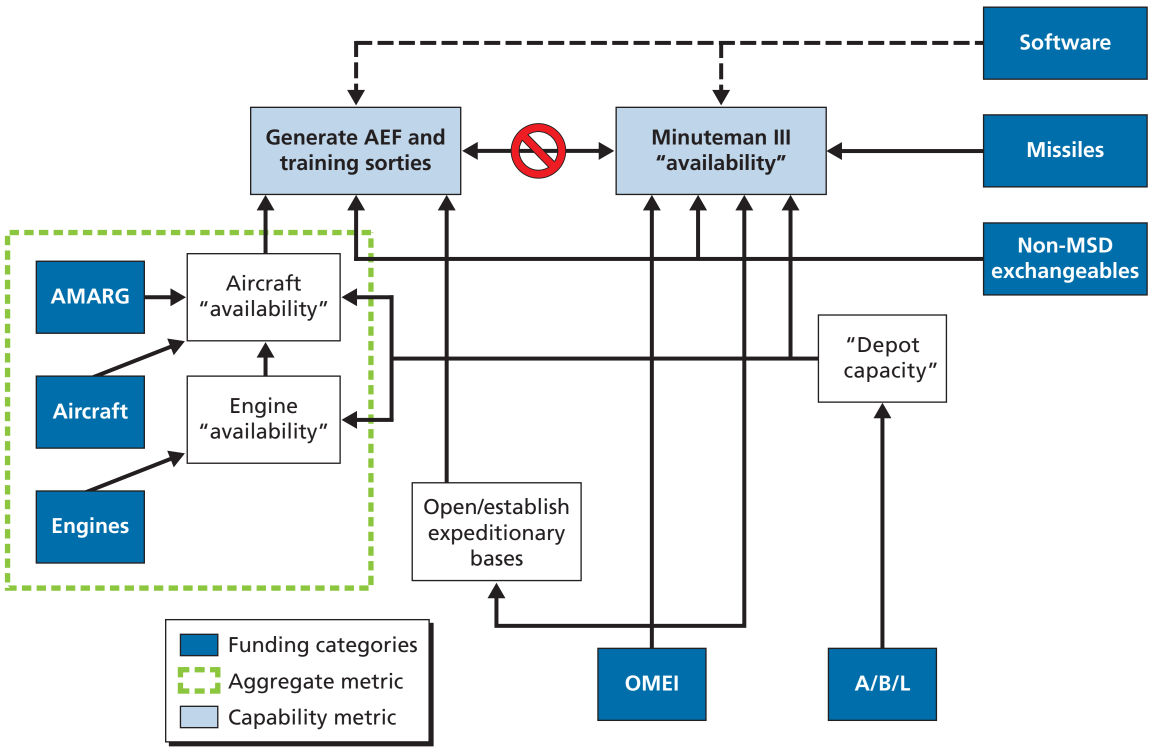
the current-year plans principally influence programming via integrated priority lists from the combatant commanders.[5]

This division of plans suggests that whatever metrics are selected apply across all plans. However, adopting common metrics does not preclude that a different set of target values for those metrics might be sought during prioritization, depending on the time frame. The future-year plans would then be used to set targets and priorities for programming, and the current-year plans for execution.

If we consider DPEM in this light, we note that plans call for generating aerospace expeditionary force (AEF) sorties, support to the nuclear mission, and training sorties for readiness. Working from right to left in Figure 3.1, we then seek a minimum number of metrics that capture these planning objectives. We have selected "generate AEF and training sorties" and "Minuteman III 'availability'" as overall metrics that embrace most of the functions supported by DPEM and needed in plans. We do not attempt to rigorously define the latter, as it does not play a significant role in our more detailed analysis. These goals are depicted in the center boxes in Figure 3.2.

The array of sustainment activities performed under DPEM render partial contributions to these two overarching capabilities. We look first at generating AEF and training sorties. Here, sustainment activities in the aircraft category clearly contribute to the ability to generate sorties. Engine repair directly supports aircraft readiness, so it, too, is a contributor. AMARG

Figure 3.2
Schematic Partial Mapping of Depot Purchased Equipment Maintenance Operational Capabilities



RAND *TR905-3.2*

[5] U.S. Department of Defense, Support for Strategic Analysis, Instruction 8260.01, January 11, 2007.

provides a reservoir of retired aircraft and parts that can be drawn on when needed and, in that sense, also supports this capability. Non-MSD exchangeables contain maintenance on some munitions, which directly support sortie generation. Finally, OMEI contains depot maintenance on special-purpose vehicles, which are needed to open and establish bases, which are, in turn, needed to generate sorties.

Similarly, for the Minuteman III availability (which might be defined as an alert status), missile maintenance directly affects the number of available missiles. Non-MSD exchangeables include sustainment of the MK12A re-entry vehicle, which also affects available missiles. Finally, OMEI includes special-purpose vehicles for extracting missiles from their silos and moving them, without which, missiles would not be available.

In both sortie generation and missile availability, A/B/L is an indirect contributor that is difficult to assess quantitatively. This category includes activities necessary for the long-term functioning of the facilities that enhance the operation of many of the other categories and, in this way, vitally facilitates other sustainment activities. For example, precision measurement equipment laboratories, which fall under A/B/L, enable a number of jobs across depot- and wing-level maintenance. That being so, A/B/L, although difficult to quantify, plays a vital role in rendering sortie generation and missile availability.

Figure 3.2 defines some intermediate metrics, such as aircraft availability, engine availability, and the capability to open and establish bases. These are convenient, useful metrics, but using them alone for capabilities-based programming suffers two weaknesses. First, without higher-level metrics, such items as special-purpose vehicles and engines are not readily compared for the purposes of programmatic trades. Second, it is not clear how to relate such metrics as aircraft availability to the DoD-level planning objectives, which ultimately provide the guidance needed to set priorities. Nevertheless, metrics like aircraft availability are collected and used in the Air Force and have merit in their own right. We use aircraft availability later as a cross-cutting metric but emphasize that it is best understood in the context of what a given availability rate means in terms of the ability to execute war plans.

Viewing the problem from this aggregate level of sortie generation and missile availability mitigates these limitations. By linking disparate funding areas—such as munition maintenance, aircraft inspections, and special-purpose vehicles—into one capability (sortie generation), it enables maximal trading during programming. By expressing capabilities in high-level, aggregate terms, the capabilities are naturally related to planning objectives and, hence, are easier to relate and prioritize and more transparent to justify.

We have deferred the discussion of software maintenance in Figure 3.2 and have used broken lines in the figure to connect that category to objectives. We take up the issue of software maintenance in Chapter Six, as it presents some peculiar difficulties, but ones shared by other resources. First, we focus on aircraft availability as a metric and how to link the aircraft and engine (or any other spare part) categories to form this common metric. We do not develop concepts for programming for missiles any further in this report.

Metrics for Aircraft and Engines

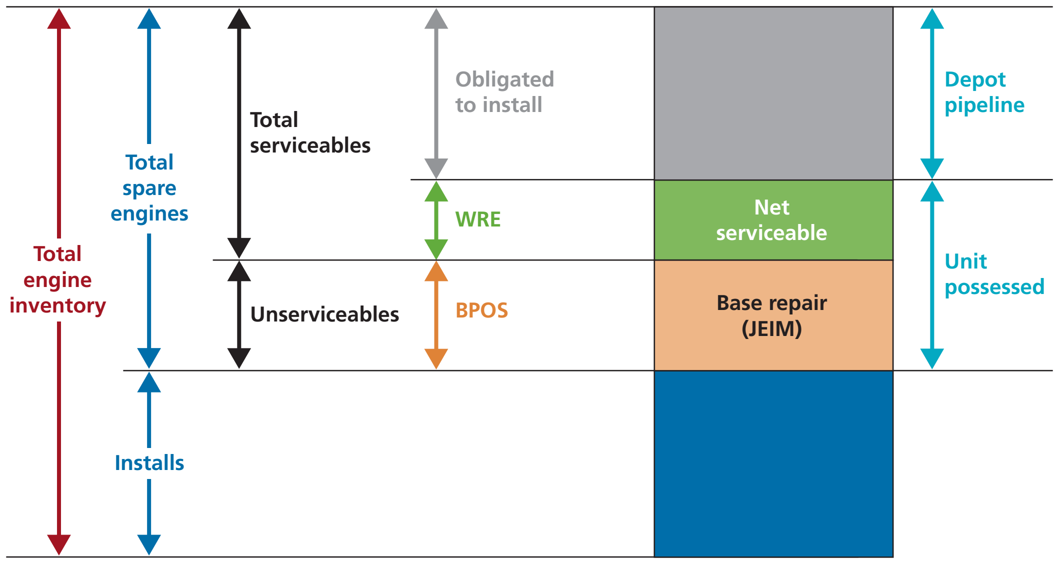
The bulk of the sustainment funded in the aircraft category of DPEM consists of mandatory inspections, such as PDM, that, if not performed, eventually lead to a grounding of aircraft by policy.[6] For such areas, policies cause aircraft sustainment to directly affect aircraft availability and, hence, sortie generation.

Considered as a spare part for aircraft, how might sustainment of engines combine with aircraft sustainment to affect aircraft availability? To quantify this, we keep in mind the penultimate objective, which is to generate sorties. A certain number of engines will be needed per aircraft for sortie generation. This number is the sum of the number of installed engines in aircraft plus some number of spare engines. This number of spare engines is determined by planning factors based on expected sortie rates for both peacetime training and wartime deployments, break rates, and service times.[7]

Figure 3.3 shows a simplified nomenclature of engine status for these categories of installed and spare engines. (Note that the diagram is not drawn to scale.) For a given engine type, the total number of spare engines can be divided into the base peacetime operating stock (BPOS), the war readiness engines (WREs), and those in the depot pipeline obligated for installation. Dividing the sum of the BPOS and the WRE for a unit by the number of combat-coded aircraft possessed by the unit gives the number of engines needed for a deployment per aircraft as enumerated by planning factors. This ratio forms the basis for the similitude of capability across these business areas and a quantitative anchor for programming trades.

Anecdotal information indicates that the current WRE levels for some engines might be higher than needed for current plans and operation tempi. We cannot confirm or disconfirm this supposition. Our argument does not hinge on whether or not the current numbers for these levels are unimpeachable but on the fact that readiness requires, in addition to installed

Figure 3.3
Engine Inventory Terminology



NOTE: Diagram not to scale. JEIM = jet engine intermediate maintenance.
RAND *TR905-3.3*

[6] TO 00-25-4.

[7] According to policies designated in Air Force Instruction 21-104 and planning factors that derive principally from the War Mobilization Plan and Technical Order 2-1-18.

engines, some number of spare engines. These are estimated according to planning factors and planning scenarios and are currently given by the BPOS and WRE. If plans and planning factors have changed substantially since the last update of these numbers, they should be revisited in order to align engine and aircraft force structure and should serve as the basis of programming priorities.

The reader might object that, during execution, some work-arounds might exist for accommodating lack of engines but that comparable work-arounds for the lack of aircraft are paltry. Therefore, equating aircraft and engines implies a misleading equivalency that provokes an inaccurate biasing toward engines. A similar theme arises whenever a programmer attempts to define measures of capability across weapon systems or business areas for the purposes of prioritization. Although we concur that insufficiency during execution in some resources is more critical than in others, we argue that—for programming purposes—programming trades and priorities should be set by the best estimates of requirements, not what might be accomplished by work-arounds during execution. Hence, the same logic used to shape the force structure should be used to set priorities in programming.

This correspondence for programming prioritization can be made more specific mathematically. For aircraft PDM and engines, if we define the ratio k_i to be the number of primary aircraft inventory (PAI)–coded aircraft for a unit divided by the number of BPOS plus WRE engines allocated to that unit according to plans (where i indexes across mission design series [MDS]), then the availability of aircraft i due to aircraft and engines is given by[8]

$$A_i = \frac{M_i \min\left[P_i - \delta_i, \left(B_i - \varepsilon_i \right) k_i \right]}{T_i}. \tag{3.1}$$

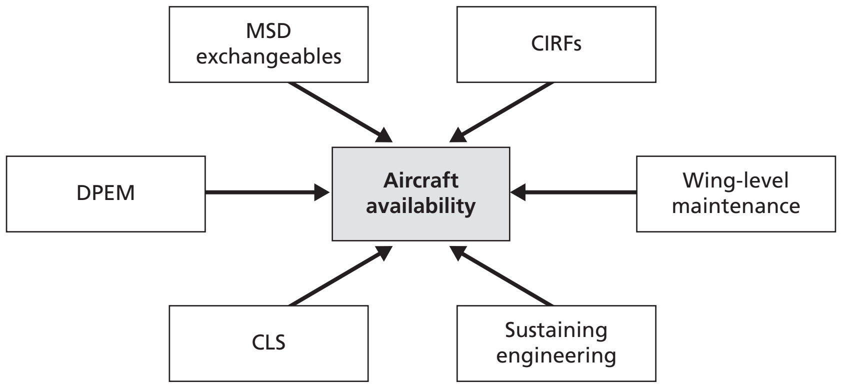
Here, M_i is the mission-capable rate, P_i is the number of combat-coded aircraft, δ_i is the number of deferred aircraft, B_i is the number of engines, ε_i is the number of deferred engines, and T_i is the total aircraft inventory. That some engine types can service more than one type of aircraft slightly complicates the analysis but does not alter the underlying correlation between aircraft and engine trades.

In Equation 3.1, the contribution of DPEM to the aircraft availability is leveraged entirely by depot possession, which is reflected in the decision variables of deferral, δ_i and ε_i. Many investment areas beyond DPEM contribute to the availability, most of which influence the mission-capable rate. If the mission-capable rate is low due to factors outside the scope of DPEM, DPEM funding has little leverage to improve the mission-capable rate. Figure 3.4 summarizes this phenomenon by indicating the multiple contributors to aircraft availability.

This observation highlights the need for a comprehensive view of funding that crosses business areas and, hence, metrics that apply as broadly as possible. Of the existing relevant Air Force metrics, the aircraft availability serves this role best.

[8] See Appendix A for a definition of the PAI code in the context of all possession-purpose codes for the total aircraft inventory.

Figure 3.4
Major Contributors to Aircraft Availability



NOTE: CIRF = centralized intermediate repair facility.

RAND *TR905-3.4*

CHAPTER FOUR

Enterprise Prioritization in Depot Purchased Equipment Maintenance Funding

The objective in enterprise prioritization is to find optimal target values for programming in each program element in the budget in terms of the relevant capability metrics—or, put more succinctly, to find the target values of the metrics.[1] An initial temptation is for each actor in the Air Force to push to increase the values of all metrics within that actor's purview as much as funds allow. Unfortunately, this local attempt at maximizing capabilities is globally inefficient, leading to an overall lower capability for the Air Force for a given funding level.

As a simple illustration, suppose the system program manager for weapon system A attempts to maximize the availability of weapon system A without regard to the availability of other weapon systems. That is, he amasses as much funding as he is able to secure in programming and allocates all of these funds toward increasing the availability of weapon system A. While higher availabilities are better for the operator than lower ones, in an environment of scarce resources, at some point, the funds expended on weapon system A will generate only a marginally higher availability. This increase will result in a significantly lower availability in one or more other weapon systems. Funds used for that marginal increase in A are hence misspent. A better goal is to balance capability among the weapon systems. That balance is achieved by having a well-selected common metric and a common goal (target value for that metric) across the weapon systems.

The key to finding where the preferred target values lie for each weapon system resides in two concepts. The first is the overall contribution of the weapon systems to operational goals, which we take up in the sections later in this chapter on weighting factors. The second is the form and position of the curves that relate the funds spent to capabilities realized for each weapon system (and component funding areas within the weapon system). These curves vary depending on program, weapon system, or business area, and we call these relationships *cost-capability curve*s.

Cost-Capability Curves

All other things constant, and assuming frugal spending, allocating more money to a funding area yields increasing capability. The form of the curve relating cost and capability will vary depending on the business area and capability metric. It is possible, nonetheless, to make some general statements about the form of this curve. If the capability of the system is very poor, any additional funds will increase capability greatly, as there are many opportunities to improve

1 By *optimal*, we mean that which yields the lowest risk across all programs.

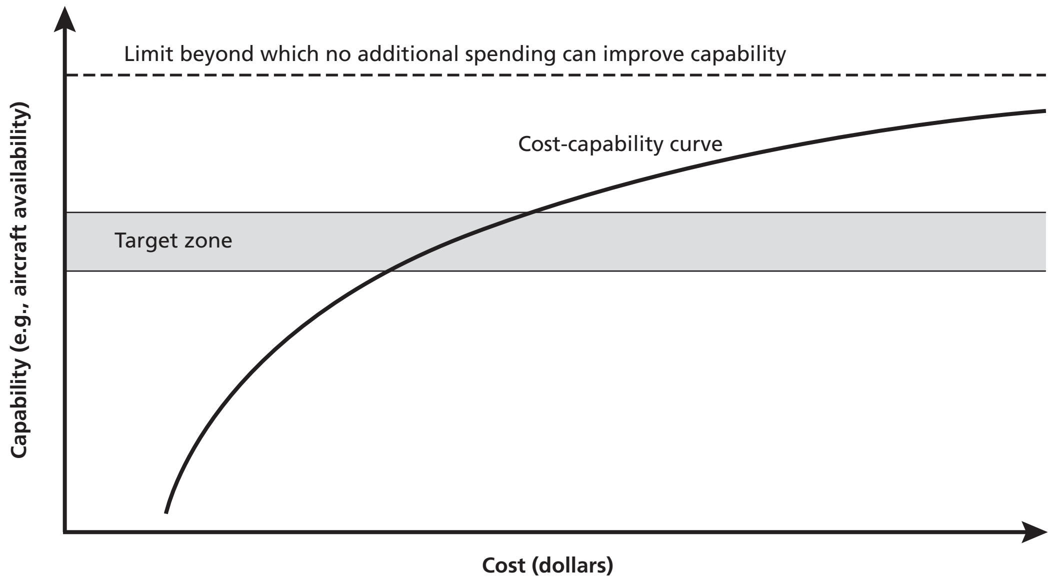
performance, some of which will be low cost. However, as perfection is neared in the capability, improvement becomes much harder to obtain. Indeed, at some point, any additional spending is pointless, as a maximum upper bound on capability is reached. Hence, the growth in capability with expenditure is not linear but shows decreasing returns to scale, as shown in the idealized curve in Figure 4.1. The shape of the curve is important for understanding trade-offs among various contributors to capability, both within and among weapon systems. Consider first trades within a weapon system.

To balance capabilities using cost-capability curves, it is easiest to visualize the problem by considering the simpler problem of balancing between just two contributors to a single weapon system. An example might be the contribution of spending on spare parts versus depot-level engine repairs on a B-1B aircraft. Figure 4.2 shows notional cost-capability curves for two such funding areas, labeled A and B.

In the case of funding areas within a single weapon system, the goal will be to achieve the same capability across all its funding areas. That is, if the goal is to achieve a target value of capability of 0.3 for funding area A, the target for funding area B should also be 0.3. Carrying through with this example, from the left panel of Figure 4.2, a capability of 0.3 would cost $6.1 million for funding area A, and, from the right panel, a capability of 0.3 would cost $5.3 million for funding area B. The total, balanced cost for the weapon system would then be $11.4 million.

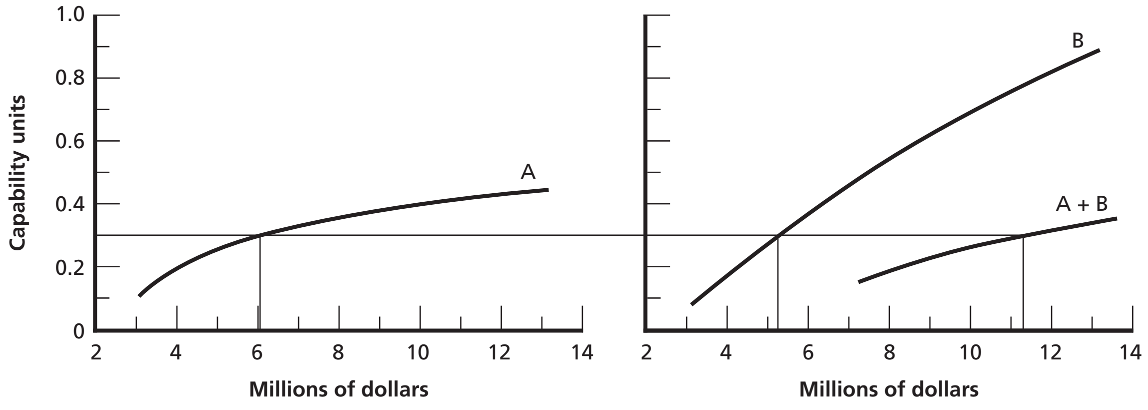
Now suppose that $11.4 million is not available but that $9.7 million is appropriated instead for the weapon system. We then need to find the level of capability on the curves labeled A and B in Figure 4.2 that gives costs that sum to $9.7 million. The solution is $4.9 million for A and $4.8 million for B, yielding a capability of 0.25. The curve labeled A + B on the right of the figure gives the locus of such points, which is the combined cost-capability curve of A and B. In practice, this exercise is quite tedious to do by hand even for the two-funding area example shown here. It becomes unworkable to do by hand when considering tens to hundreds

Figure 4.1
Notional Cost-Capability Curve



RAND *TR905-4.1*

Figure 4.2
The Merger of Two Notional Cost-Capability Curves



RAND *TR905-4.2*

of funding areas. Analytical, automated tools need to be developed to perform these calculations, and one approach to this problem is the subject of the next chapter.[2]

Before turning to the model that generates this solution curve, we need to consider the complications that arise in balancing capabilities among weapon systems, where the target values for the capability metric may not be common for all weapon systems. How can optimal values for that metric be determined, and how can these be balanced using cost-capability curves?

Goals for the Employment of Weapon Systems: Weighting Factors Among Weapon Systems

Within a weapon system, each proposed programmed task would ideally produce the same capability. If we use aircraft availability as a measure, for example, programmed work on spare parts and inspections for a single weapon system should each yield the same aircraft availability. Among weapon systems, however, this will no longer generally be the case. For example, if the capability metric of choice is aircraft availability, it might be desirable to try to achieve different target values of availability for different weapon systems. The target values for capability on cost-capability curves for each weapon system then depend on how the systems are to be employed, training needs, and any interdependencies of the weapon systems during employment.

For employment priorities, we turn to the DoD-level planning guidance. Planning guidance outlines various contingencies that form a basis for programming analyses by the services. These contingencies range from small operations, such as humanitarian relief, to large operations, both conventional and nuclear. Further guidance specifies how these various contingencies should be considered in time, including temporal overlaps.

The relative demand for weapon systems in these plans indicates the relative importance of the weapon systems to the warfighter. Each of these contingencies requires a certain number

[2] See also Snyder et al., 2009.

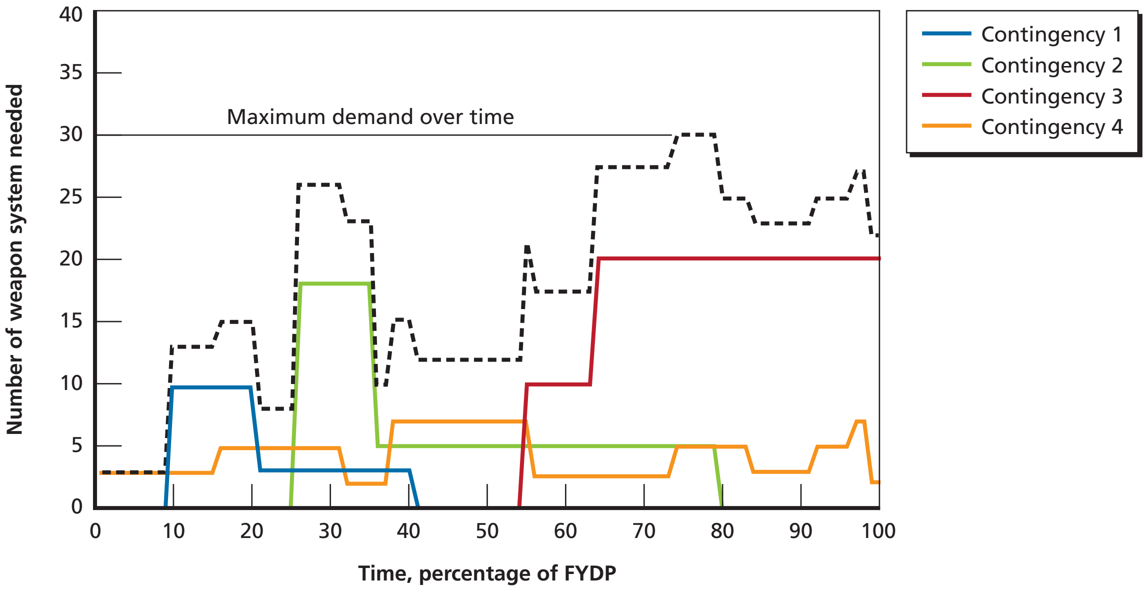
of the various weapon systems to achieve its objectives. Figure 4.3 illustrates a case constructed from notional contingencies. In the figure, the demand for some weapon system is indicated for four contingencies over the FYDP. The fourth contingency is a steady-state demand over time that might be a sum of small, regular deployments that can be met within normal AEF-type rotational policies. The heavy, dashed curve indicates the sum of the four contingencies over time, and the upper line shows the maximum demand for that weapon system over time, which, in this case, is 30.

For this set of contingencies, which we call a scenario, 30 of this particular weapon system are needed to handle the maximum demand.[3] Similar plots can be constructed for all other weapon systems called for in plans. These maxima for weapon systems form a guide for prioritization across weapon systems and a baseline for evaluating programming risk.

For each weapon system *i*, if the maximum number needed in plans is d_i, this value can be compared with the number of combat-coded aircraft of that type in the inventory, s_i. The ratio d_i/s_i then gives a measure of the degree to which weapon system *i* is in high demand in plans relative to its number in inventory.[4] We call this ratio a *weighting factor* for the weapon system, w_i. Assets with a high value of this weighting factor ($w_i >> 1$) have a higher priority for the operator than assets with a low value ($w_i << 1$).

These factors will depend on the nature of plans, both in types of contingencies and in their distribution in time. Two large groupings of plans are currently maintained. A partnership of the Office of Cost Assessment and Program Evaluation in OSD and the Force Structure, Resources, and Assessment Directorate defines the *future-year plans* for programming

Figure 4.3
Notional Demands for a Weapon System in Plans over the Future Years Defense Program



RAND *TR905-4.3*

3 This maximum requirement does not include reconstitution after deployments, which may be important for combat support resources. See Snyder et al., 2009, pp. 24–34.

4 Often called by the Air Force a *low-density, high-demand asset*.

purposes, to guide the force structure in the medium- to long-term time frames. The various combatant commanders maintain the *current-year plans* for near-term possible execution. These two sets of plans may have thematic overlap but differ in assumptions about objectives and force structure of the United States and possible adversaries.

Determining values of weighting factors for these plans presents several challenges. For some contingencies in the plans, the number of each weapon system needed has been determined in the course of the planning process. For the combatant commanders' OPLANs, this determination is part of the detailed time-phased force and deployment data. For the OSD plans, the determination is in the form of multiservice force deployment data. For a small number of contingencies, these estimates still need to be done to determine values of weighting factors.

More critically, requirements to execute the theater plans for combat aircraft (fighters, bombers, and intelligence, surveillance, and reconnaissance aircraft) are more easily estimated than the requirements for mobility aircraft (air lifters and tankers). The reason is that the role of the combat aircraft for the theater is restricted to operations in the theater and, hence, directly tied to the operational goals. Mobility aircraft have both a theater role and a strategic supporting role.[5]

The number of each type of mobility aircraft needed for the strategic role depends on several factors. For the lifters, one needs a fairly accurate estimate of the quantity and destination of all materiel and personnel that need to be moved by air. Developing this estimate requires a quite detailed plan. Tankers are needed for theater support, which depends on the basing, sortie schedule, and missions flown by supported aircraft, as well as to form the strategic air bridge to support all types of aircraft moving into and out of the theater.

Hence, estimates for requirements for combat aircraft can, in general, be easily extracted from theater plans. Estimating mobility requirements demands a deeper analysis not generally found in plans; these analyses are provided by periodic assessments of mobility capabilities requirements performed by U.S. Transportation Command.

Strengths of Weighting Factors

The clear benefit of the weighting-factor method of quantifying priorities among weapon systems is that it places the operator's needs foremost. Recently used target values for capability metrics have generally lacked this characteristic. Consider the target values used for aircraft availability, the most widely used measure of performance for aircraft.

Aircraft availability target values are derived from the annual flying-hour program. The process unfolds roughly as follows. Each year, operators determine the number of flying hours needed to satisfy training requirements for pilots to meet readiness standards. Given the number of aircraft in the fleet, mission-capable rates are then computed that would supply enough aircraft to meet the flying schedule. Finally, from the mission-capable targets, programmed depot inductions, and force structure, target aircraft availability values are computed that will permit the flying-hour program to be executed.

If done accurately, aircraft availability values below the target would mean that parts of the flying-hour program would not be executable; values above the target would result in

[5] Bombers and intelligence, surveillance, and reconnaissance aircraft also have enduring strategic missions that are additive to the theater requirements associated with any single plan. Any estimate of the forces required should include those assets assigned to these strategic missions.

misspent money, in the sense that levels of availability would be bought that are above those needed to execute the flying-hour program. The resources spent to purchase those levels of availability could be better spent elsewhere.

What these aircraft availability target values do not express is how important it is that a given weapon system meet its flying-hour program. Nor do these values reflect the priorities for these weapon systems for planned deployments—only for training. Hence, these targets do not properly inform the full spectrum of issues that determine priority trades across weapon systems, and specifically do not reflect operational priorities. The prioritization by weighting-factor approach avoids these weaknesses.

Limitations of Weighting Factors

All prioritization schemes have limitations, and the weighting-factor approach is no exception. While the method directly emphasizes operational priorities for weapon systems in plans, some shortcomings and caveats should be noted.

First, the fidelity of the weighting factors is only as good as plans on which they are based. Plans are imperfect estimates of potential future military operations. These estimates change as the political landscape changes, with a frequency that is rapid relative to the ability of most of the force structure to respond. By nature, then, some dimensions of the force structure will always be somewhat misaligned with plans.

A related issue is that placing all priorities with operational needs in plans can cause such a high priority to be placed on one system that it would then be fully funded, and such a low priority to be placed on another system that it would receive no funding. The latter is not tenable for major capital investments, such as aircraft. Some balance needs to be struck between the operational priorities and sustaining the existing force structure.

Second, these weighting factors do not incorporate training priorities. For combat-coded weapon systems, the lack of explicit treatment of training needs is related to the problem just mentioned. The training needs are directly proportional to the number of pilots, and the number of pilots is proportional to the number of aircraft of a given type. Therefore, the training needs are roughly proportional to the number of aircraft. Hence, there is a need for a balance between the training priorities, which are approximately that of sustaining the current force structure, and the operator's priorities, which are to have a fleet that is optimally ready to support war plans. We take up the treatment of this trade space for programming purposes in the next chapter.

But even acknowledging this trade-off does not resolve the issue of how to assign weighting factors to non–combat-coded weapon systems, such as the T-38 Talon training aircraft, and associated supporting components, such as the J85 turbojet engine that powers the T-38.

Policy Recommendations for Weapon System Prioritization

To partially resolve these issues, we propose a tiered system of weapon system prioritization that is based on the weighting factors derived from plans described in the previous section. Three tiers may suffice: high, medium, and low priority. The highest tier would be for weapon systems that are fewer in number than those needed in plans. The middle priority would be for training assets and systems with inventories roughly equal to the number needed for plans. The lowest tier would be for weapon systems for which the inventory exceeds the number needed for plans.

This grouping naturally avoids an excess of precision in the assignment of weights beyond that warranted by the plans, and it dampens the occasionally rapid temporal fluctuations in those planning objectives. Also, by using a tiered system, the non–combat-coded aircraft, such as the T-38, can be more easily handled by placing them together in the middle tier.

This tiered system would be similar to the method in use in FY 2008 with CAM, although it differs in being traceable to OSD objectives. As such, if the state of a weapon system does not meet planning goals, that deficiency can be expressed in terms of the degree to which that weapon system cannot perform plans, both the near-term combatant commanders' plans and the longer-term Defense Planning Scenarios.

In addition to weapon system prioritization anchored in planning objectives, the consequences of programming and execution decisions could be expressed in terms of the risk of not being able to execute plans. The next chapter shows how to use such weighting factors to inform the building of a program and expressing the risks that inhere in a program in terms of the inability to satisfy plans.

CHAPTER FIVE

A Prototype Model for Assessing Capabilities and Risks in Depot Purchased Equipment Maintenance Programming

In the previous chapters, we argued that given a set of depot-level production requirements and a limited budget to perform them, prioritization among weapon systems can be achieved via weighting factors derived from operational plans. The desired programming levels are then set to maximize a balanced capability among weapon systems via the appropriate funding along cost-capability curves. In this chapter, we develop an analytical model to guide programmers in making these trades. The model and related software tools to be described examine trades among the aircraft (specifically, PDM) and engine commodities of DPEM. The model uses aircraft availability as a metric and weighting factors from plans as a prioritization scheme to locate optimal positions along cost-capability curves.

During the building of the Air Force POM and president's budget, frequent decisions on how to absorb budget cuts in the various programs need to be made in a very short time frame. By *short*, we mean too short for the programmers to reproduce the process that led to the initial proposed POM and its inherent prioritizations. The goal of the tool described in this chapter is to provide programmers on the Air Staff a guide for distributing risk during these so-called POM drills and a means for articulating the magnitude of those risks in terms of the projected inability to execute planning objectives in the Defense Planning Scenarios and combatant commanders' plans.

We emphasize from the outset that models and software tools, such as those described in this chapter, provide guides to programming, not a turnkey solution to complex programming trades. A poignant and informative example of the dangers of overreliance on algorithms occurred recently when a six-year-old story about the declaration of bankruptcy by United Airlines was mistakenly posted on the website of the South Florida *Sun-Sentinel*.[1] Google's web crawler picked up the story, further disseminating it, and eventually led to selling of the stock by traders who had heard the story but not read it. Trading done automatically by algorithms that sell stock based on the movement of stock prices alone then took effect, and the value of United stock fell by more than $1 billion in 12 minutes before trading was halted—a distressing example of the perils of the lack of human attention.[2] No algorithm is fail-safe; all require some attention and intervention of humans. In the case of the models we describe here, the intent is that they provide guides to inform decisions, not a means to automate decisionmaking.

There are many possible models that could point Air Staff programmers to a budget that delivers a balanced set of depot production capabilities in the face of fiscal constraints. The one

[1] United had previously emerged from bankruptcy in 2006.

[2] Micheline Maynard, "A Mistaken News Report Hurts United," *New York Times*, September 9, 2008; Tim Arango, "I Got the News Instantaneously, Oh Boy," *New York Times*, September 14, 2008.

that we develop here determines an optimal set of depot induction deferrals in order to avoid grounding an aircraft due to lack of depot maintenance. We begin with a general description of the model, the required data inputs, and the underlying assumptions. This description is followed by a precise mathematical formulation.

Description of the Depot Purchased Equipment Maintenance Model

The requirements process designates, for each year in the FYDP, a number of depot inductions by DPEM commodity (aircraft, engines). It is a basic assumption in this model that these requirements are valid.[3] These requirements generate costs for each year in the FYDP. For a given production level, costs might sometimes be reduced by adoption of efficiencies and other process changes, some of which may incur up-front fixed costs to implement. We do not treat such cost-reduction efforts in this model, but these could be incorporated into more sophisticated models. Hence, we take as given a set of required depot tasks for each year of the FYDP. Each of these tasks has a cost. A first approximation of costs can be gleaned from average prices charged by the Air Logistics Centers (ALCs) in recent years for comparable work. Summing the costs for each year gives the overall requirement in terms of dollars.

If money is available to cover all of these costs, the programmer is not confronted with an allocation problem. We assume that some budget lower than this amount is available for one or more years of the FYDP. If costs exceed money available, some programmed work will not be able to be performed. If this programmed work is an aircraft PDM, we assume that work can be deferred for 12 months. Beyond that, the aircraft will be grounded. For engines, we assume that aircraft will become unavailable due to lack of spare engines when the number of engines falls below the threshold described in Equation 3.1 in Chapter Three.

The programming allocations are determined subject to a number of user-specified constraints; others could easily be added as needed. One constraint is that each production line in the ALCs has a finite capacity. This capacity might vary from year to year and, hence, places a bound on the ability to shift work from the desired year of induction to some future year. Another constraint is an option to cap the proportion of the fleet that can be deferred in any given year and an option to penalize deferrals by making them more expensive to do in deferral than the cost would have been had they been done in the desired year of induction.

One additional option is to explore the consequences of using the prioritization of the weighting factors as given by the requirements in operational plans versus the prioritization given by training needs. The former reduces the risk of not being able to fulfill planning objectives; the latter approximately distributes the risk proportionately across all weapon systems. The reason for incorporating this option is that some cases arise in which the model will suspend all maintenance on some weapon system of low weighting in favor of full funding for weapon systems of the highest weighting. Such an allocation solution is, in general, untenable

[3] Some requirements can be fairly accurately predicted, such as aircraft PDMs that are done on a calendar basis. Others, such as estimates of unscheduled engine inductions, carry greater uncertainty, especially for the out-years of the FYDP. Engine inductions will depend on the number of hours flown and factors that relate those hours to engine breaks. Both are difficult to predict years in advance.

for major capital assets.[4] The model provides a sliding rheostat to place emphasis on any position in this continuum between operators' and training priorities.

To summarize, the inputs to the model are

- induction requirements for each item by year of the FYDP
- the topline budget for each year of the FYDP for the assets of interest
- weighting factors for weapon system prioritization, extracted from planning scenarios
- costs of production for each depot induction
- penalty cost for deferring work
- maximum proportion of the work that can be deferred for each item
- ALC capacities for each production task for each year in the FYDP
- force structure for each year in the FYDP
- the place in the trade space between the operators' priorities and training priorities.

The output of the model gives three insights. The first is how to prioritize—based on the weighting factors for weapon systems—and distribute the workload and deferrals across the FYDP to avoid aircraft groundings. These outputs are the production levels for each year, what work is deferred, how much capacity is used, and the resultant total costs for each year in the FYDP. The optimal allocation will be such that no aircraft are grounded and all work is performed by the end of the FYDP.

A second insight is how much the budget can be reduced in one or more years in the FYDP and still be able to execute the depot work at some time within the FYDP without grounding an aircraft, subject to the user-specified constraints. In this exercise, the user can decrease the budgets available in one or more years of the FYDP until the model returns an infeasible result. An infeasible result means that the program cannot be executed without grounding an aircraft. The user can then explore which constraints cause this problem (e.g., insufficient budget, lack of capacity) and which aircraft are the first to be grounded.

A third insight is elucidating the *operational* consequences of a proposed POM. Given the underlying data used to construct the weighting factors, the resulting aircraft availability from the proposed programmed depot work can be used to see how many of each aircraft are available *relative to those needed to execute the plans*. This calculation gives a direct impact of the depot work on aircraft and engines in terms of operational priorities. The programmer can explore risk expressed in these terms while varying any of the above inputs, including the sliding rheostat that adjusts in the continuum between operational priorities and sustaining the force structure.

[4] As discussed in Chapter Four, the reasons are many: The sunk capital investment discourages not maintaining an aircraft, the political costs of doing so would be too high, and the consequences for training and the industrial base would be unacceptable.

Mathematical Formulation of the Depot Purchased Equipment Maintenance Model

We now express these concepts more precisely as a mathematical formalism. All notation for this section is summarized in Table 5.1. We follow the convention of using upper-case letters to describe decision variables.

The problem lends itself well to be formulated as a multiperiod dynamic production linear programming problem that can be solved via standard algorithms. In this type of model, each period has some production requirements. The multiperiod decisions of the model can be viewed as a multistage chronological process, in which the consequences of falling short in producing one period's requirements need to be compensated for in subsequent periods. In the

Table 5.1
Summary of Notation

Symbol	Meaning
a_{i1}	Total aircraft inventory of aircraft *i*
b_t	Total budget available in year *t*
c_{ijt}	Cost for aircraft *i* of production task *j* in year *t*
d_{ijt}	Production requirement for aircraft *i* of production task *j* in year *t*
e	Parametric weighting factor ($e \geq 0$)
f_{ijt}	Maximum fraction of d_{ijt} that can be deferred
g_{ijt}	Extra cost of executing a deferred task, expressed as a fraction of c_{ijt} ($0 \leq g_{ijt} \leq 1$)
h_{i1}	Minimum aircraft availability for aircraft *i* ($0 \leq h_{i1} \leq 1$)
i	Index of aircraft
j	Index of production tasks
m_{i1}	Mission-capability goal for aircraft *i* ($0 \leq m_{i1} \leq 1$)
p_{i1}	PAI for *i*
q_{ijt}	Utilization fraction of production capacity
s_{i2}	On-hand stock of engines for aircraft *i*
t	Index of year
T	Number of years in the FYDP
V	Maximum fractional production demand shortfall for all MDS
w_{i1}	Weighting factor for aircraft *i*
X_{ijt}	Production for aircraft *i* of production task *j* in year *t*
Y_{ijt}	Deferrals for aircraft *i* of production task *j* in year *t*
Z_{it}	Available aircraft *i* in year *t*
α_i	Number of BPOS engines for aircraft *i*
β_i	Number of war reserve engines for aircraft *i*

model, most parameters have been indexed by time, thus making it possible to have values that vary with time. Similarly, many parameters can vary with aircraft type and aircraft subsystem.

The objective function for the linear program is a balance between maximizing aircraft availability and sustaining the force structure. Hence, we maximize

$$\frac{1}{T}\sum_{i}\sum_{t}\left[\left(\frac{w_{i1}}{\sum_{n} w_{n1}}\right)\frac{m_{i1}Z_{it}}{a_{i1}}\right]-eV. \tag{5.1}$$

The term in the brackets expresses the aircraft availability for each weapon system and prioritizes them by the associated weighting factors in parenthesis. The term *eV* in Expression 5.1 provides an adjustable parameter to vary the relative importance of the aircraft availability and that of training. As $e \to 0$, the objective function becomes dominated by the term in brackets, the weighted aircraft availability. As *e* grows large, the right-hand term dominates, and the deferrals are increasingly distributed across the weapon systems proportionately, independent of weighting factors or aircraft availability, thus acting as a proxy for prioritizing according to training needs.

The various state and decision variables in Expression 5.1 are specified for each year of the planning horizon and for each category of interest. Currently, these categories include aircraft type (e.g., F-15, KC-135) and system or subcategory within aircraft type (e.g., aircraft PDM, engine overhauls).

This objective function is subject to numerous constraints, some imposed by policy decisions. Several parameters are defined as fractions and are therefore bounded by 0 and 1:

$$0 \le f_{ijt} \le 1, \tag{5.2}$$

$$0 \le g_{ijt} \le 1, \tag{5.3}$$

$$0 \le h_{ijt} \le 1, \text{ and} \tag{5.4}$$

$$0 \le q_{ijt} \le 1. \tag{5.5}$$

The amount of money spent must be less than or equal to that available in the budget. This expression reflects a penalty cost for deferred work:[5]

$$\sum_{i}\sum_{j} c_{ijt}\left(X_{ijt}+g_{ijt}Y_{ijt}\right) \le b_t, \tag{5.6}$$

[5] Note that a single expression, such as Inequality 5.6, presenting a constraint might, in fact, be multiple constraints. There is a constraint for each possible value of the variable subscript that is not a dummy index of a summation. In the case of Inequality 5.6, there is a constraint for each year *t* in the planning horizon.

and the production is constrained by capacity:[6]

$$X_{ijt} \le \frac{d_{ijt}}{q_{ijt}} \quad \forall t > 1. \tag{5.7}$$

The production decision variables and the backlog variables computed by the model make sense only if they are nonnegative; hence,

$$X_{ijt} \ge 0 \quad \forall i, j, t, \text{ and} \tag{5.8}$$

$$Y_{ijt} \ge 0 \quad \forall i, j, t. \tag{5.9}$$

The backlog state variable of one period affects the computation of the backlog of the next period. The linkage of backlog variables across time periods is represented by the following state variable accounting constraints:

$$Y_{ijt} = \sum_{t' \le t} \left(d_{ijt'} - X_{ijt'} \right). \tag{5.10}$$

We have also added constraints that limit the deferrals or backlog in meeting the production demands. On the one hand, we have added an upper bound on the deferrals so they do not exceed a policy-specified percentage of the demands. The upper bounds for the last year in the planning horizon indirectly specify a boundary condition for that horizon. On the other hand, we have added a lower bound so the deferrals from one year to the next are caught up with that same next year:

$$X_{ijt} \ge Y_{ijt-1} \quad \forall t > 1, \text{ and} \tag{5.11}$$

$$Y_{ijt} \le f_{ijt} d_{ijt}. \tag{5.12}$$

The following constraints on Z_{it}, together with the objective function (Expression 5.1) ensure that the Z_{it} reflect the number of aircraft available, constrained by the available airframes and engines:

$$Z_{it} \le p_{i1} - Y_{i1t}, \text{ and} \tag{5.13}$$

$$Z_{it} \le \frac{\left(s_{i2} - Y_{i2t} \right) p_{i1}}{\alpha_i + \beta_i}. \tag{5.14}$$

[6] We denote year 1 as the first year of the planning horizon.

The following constraints on *V*, together with the objective function (Expression 5.1), ensure that *V* is the maximum fractional production demand shortfall:

$$d_{ijt}V \geq d_{ijt} - X_{ijt}. \tag{5.15}$$

We allow the user to force a minimum aircraft availability, which is given by

$$\frac{m_{i1}Z_{it}}{a_{i1}} \geq h_{i1}. \tag{5.16}$$

Finally, we force the maximum fractional production demand shortfall to be nonnegative and the on-hand stock of engines is, by definition, nonnegative; hence

$$s_{i2} \geq 0, \text{ and} \tag{5.17}$$

$$V \geq 0. \tag{5.18}$$

Implementing the Depot Purchased Equipment Maintenance Model

We have developed a prototype software tool that implements the formalism described in the previous section. This tool and its use are described in more detail in Appendix B. The optimization is done using commercial off-the-shelf linear programming software called the General Algebraic Modeling System (GAMS). In addition, we developed a computational environment that interacts with GAMS to facilitate data input, executing multiple consecutive model runs and analyzing and presenting the results. This decision support system is also discussed more fully in Appendix B. Table 5.2 summarizes the data sources that we used as inputs to the model for feasibility testing. All values can be adjusted by the user as desired.

Using the Depot Purchased Equipment Maintenance Model to Make Policy Decisions

We advocate using modeling approaches such as this one to help programmers to reach decisions about funding levels and to understand and express the risks that those decisions incur. The simplest application is to specify the full range of input data, budgets, and other constraints and run the model. If a feasible solution is reached, a program plan is presented that will specify the workload by task for each year, any needed deferrals (without groundings), budgets used, capacities used, and other parameters. These results are then presented in expressions of risk, including the projected aircraft availability and the projected proportion of the various Defense Planning Scenarios that can be fulfilled.

The model has numerous constraints, including limited budgets, finite capacities, and minimum allowable aircraft availability. These can very easily forbid any solution that will not

Table 5.2
Data Sources for Depot Purchased Equipment Maintenance Model Parameters

Parameter	Symbol	Data Source
Production demand levels	d_{ijt}	Management-of-funds spreadsheet
Annual budgets	b_t	Management-of-funds spreadsheet
Cost of production	c_{ijt}	Management-of-funds spreadsheet
Extra cost or penalty rate (of normal cost) for production being delayed a year	g_{ijt}	User adjustable; default = 5%
Utilization fraction of production capacity	q_{ijt}	User adjustable; default = 93%
Engine on-hand stocks	s_{i2}	LIMS-EV
Authorized BPOS engines	α_i	LIMS-EV
War reserve engines	β_i	LIMS-EV
PAI	p_{i1}	PDS
Total aircraft inventory	a_{i1}	PDS
Mission-capability goal	m_{i1}	User adjustable
Aircraft availability minimum	h_{i1}	User adjustable

NOTE: LIMS-EV = Logistics Installations and Mission Support, Enterprise View. PDS = Program Data System.

force the grounding of aircraft and hence result in an infeasible solution. In this case, the user can still glean many insights.

The linear programming methodology provides a mechanism to identify the most binding constraints through the values of the linear programming dual variables or *shadow prices*.[7] These indicate which are the most binding constraints on the solution. When there is no feasible solution for a given set of inputs, one strategy to proceed from this point is to gradually relieve these constraints until a feasible solution is reached. In this way, the user can find a set of inputs for which a feasible solution exists. Another strategy is to begin the model run with relaxed values for the constraints, then proceed with multiple model runs, increasingly tightening the various constraints.

Note that, as the constraints (e.g., budgets, capacities) are tightened, the permissible region of the solution space becomes increasingly smaller, and the prioritization (weighting factors) plays less and less of a role in specifying the solution. The constraints predominantly determine the optimal solution. As the constraints are relieved, the prioritization plays a more dominant role in determining the optimal solution.

In all cases, the user will wish to explore the trade space between prioritization according to the operator's priorities (weighting factors from plans) and that of training (risk distributed proportionately across the weapon systems). The parameter *e* is used in the objective function to make this adjustment. If *e* is selected to be 0, the weighting factors dominate the objective function, but this does not mean that training is not considered; the various constraints that

[7] A *shadow price* for a constraint is the infinitesimal change in the objective function over the infinitesimal change in the corresponding constraint at the optimum. Shadow prices reflect the quantitative role of a constraint on the optimal solution and are 0 if the constraint is not binding.

reflect force structure (such as maintaining a minimum aircraft availability) will still impose some bounds on the solution. If *e* is selected to be unity, the objective function tends to spread risk evenly across the various aircraft types without regard to any shortfalls in meeting peacetime or wartime scenarios. By exploring various values of *e*, the user can see risk incurred in terms of projected aircraft availability and projected ability to fulfill the various Defense Planning Scenarios.[8]

Future improvements to this model might include making this trade-off between the operator's and training priorities. For example, the objective function could be modified to make the parameter *e* reflect more precisely the training priorities by linking it to the flying-hour program for each year in the FYDP.

The modeling approach described in this chapter and in Appendix B works effectively for commodities, such as aircraft and engines, which our prototype tool treats explicitly, as well as similar resources, such as spare parts. For resources that are less obviously linked to planning objectives, it is less useful. Software is one important such business area, which we explore in the next chapter.

[8] Based on some of our preliminary runs with sample data, we suggest a starting value of $e = 0.25$.

CHAPTER SIX

Software Maintenance Challenges for Capabilities-Based Programming

Trading among weapon systems and business areas necessitates a certain similitude, not only in metrics but in a range of other characteristics. Some of these include time lines of the requirements and programming process. The discussion of these details is best done with a concrete example. In this chapter, we discuss in depth one of the most challenging areas for capabilities-based depot-level programming, software maintenance.

After aircraft and engines, software constitutes the third-largest (by funding) DPEM commodity category, accounting for about 17 percent of total DPEM funding and representing about $634 million in the 2008 programmed budget.[1] Although overall requirements for DPEM have been chronically underfunded, underfunding has been most pronounced for the software category. For example, according to DPEM data from a recent POM cycle (i.e., POM/08), software is consistently funded at about 60 percent of the programmed requirements.[2] In comparison, aircraft maintenance is funded at about 75 to 85 percent. It is unclear how the lead commands solve the software funding shortage and deferral problems year after year. The long-term operational impacts and risks associated with such deferrals are currently not well understood. These deferral issues are discussed more fully in Appendix C.

Before discussing the software area of DPEM, it is important to understand what is and is not included in this funding area. This area funds software maintenance, which, for the most part, is the development, test, evaluation, and installation of the next version of software on fielded hardware systems. The software is generally real-time software that interacts with hardware, such as antennas and flight controls. The code is often written in languages no longer used elsewhere and no longer taught. Expertise to do this kind of software work requires long training—often more than a year—in the software language, code, and hardware interface. The software changes are installed in the entire fleet at once, not platform by platform, and are installed only after extensive test and evaluation. DPEM-funded software work does not include software changes necessary because of hardware modifications or because of new missions; these are funded out of other accounts. Nevertheless, DPEM funds the bulk of all software work.

In this chapter, we discuss the challenges involved in adopting a more capabilities-based programming process for software maintenance. The overarching difficulty, which is shared

[1] As a depot maintenance activity, software sustainment involves "production effort required to design, code, test, and produce embedded weapon system and associated test system software after establishment of an initial software production baseline." See Air Force Materiel Command, "The Software Requirements Review Process (SRRP) and Guide for National Security Systems (NSS) Software Funding Policy/Procedures," *Financial Management Handbook*, April 2004.

[2] POM/08 represents programming activities corresponding to the FYDP period extending from FY 2008 to FY 2013.

with some other depot-maintenance business areas, is the ambiguity in coupling maintenance requirements with operational capabilities. We discuss what these difficulties are in the first section of this chapter. That section is followed by a section detailing challenges that are more specific to software. Software maintenance in general has an added difficulty of sustaining a minimum level of resource capability, whether contract or organic, that can handle the Air Force's obsolete legacy software systems. There are other peculiar difficulties for programming software maintenance requirements, difficulties that render it an especially complex and elusive process. These are the

- diverse nature of software maintenance activities
- varied funding sources for software maintenance work
- varied practices in establishing software baselines
- lack of standardization in the software requirements process and documentation
- long lag times between the initial programming and execution phase.

To understand these challenges, we examine certain practices related to types of maintenance, software categories, scheduling, funding, baselining, documentation, and the requirements process. This discussion leads to some recommendations specific to improving software maintenance. In Chapter Seven, we discuss the general challenge that extends to business areas beyond software as it relates to the difficulty in coupling funding and capabilities, and we explore a range of policy options for capabilities-based programming for depot-level maintenance in general.

We note that there is an extensive literature dealing in general with change management in software development and modification. Most of this literature deals with the ability to establish appropriate software functionality and resulting changes in requirements in the software development process. This literature is an appropriate reference for software changes in the development (acquisition) phase or hardware-related software modifications. However, it has limited application in DPEM-funded software and thus is not included in our discussion. As is discussed in more detail in this chapter, DPEM software requirements constitute a composite of the miscellaneous software deficiencies and minor fixes that are discovered in fielded hardware during the operations phase. Developing requirements for each individual deficiency as it arises is, relatively speaking, not the most challenging task. The challenge lies in "packaging" these deficiencies, both from operational efficiency and cost-effectiveness standpoints, with the appropriate programming time line and funding combinations.

General Challenge: Loose Coupling Between Requirements and Operational Capabilities

For any program, the operational response to programming and funding can be clear and certain (tightly coupled) or it can be fuzzy and uncertain (loosely coupled).[3] Relatively speaking, aircraft and engines (and missiles and, to some extent, storage) have much clearer and certain

[3] We borrow and adapt this concept and terminology of coupling from Charles Perrow, *Normal Accidents: Living with High-Risk Technologies*, Princeton, N.J.: Princeton University Press, 1999. We thank Shari Lawrence Pfleeger for bringing to our attention that the concept of loose coupling in complex systems is originally from W. P. Stevens, G. J. Myers, and L. L. Constantine, "Structured Design," *IBM Systems Journal*, Vol. 13, No. 2, 1974, pp. 115–139.

operational response to programming, whereas software (and other DPEM categories, such as some OMEI and A/B/L) are fuzzier and uncertain. For example, funding below the aircraft PDM requirements would cause eventual deferrals and groundings, which would, in turn, decrease aircraft availability. Although some buffer is provided by the reserve stock in the case of engines, the impact of not having sufficient engines for a given weapon system is also clear and certain from an operational standpoint. For software, however, the operational response may be quite varied. Some software modifications might be deferred with minimal operational impacts, whereas others, such as the real-time embedded software within the weapon systems, might result in flight restrictions or reduced mission capability with more direct and immediate operational impact.

One of the attributes of the DPEM categories that makes the coupling "loose" is when there are multiple ways or work-arounds in which the capability can be provided. In the case of software, the work-arounds can apply in both the operational and sustainment senses. In the operational sense, if particular software (e.g., a cockpit display enhancement), for example, is not provided for a weapon system, pilots might be able to exercise work-arounds without materially affecting the mission capability.

Likewise, in the sustainment sense, the ALCs can exercise a variety of work-arounds, such as stretching the work with extended schedules or combining work to gain more efficiency through economies of scale to deal with funding shortages. For a category to have a "loose" coupling effect in terms of funding is not to say that its impact on operational performance is inefficient or not valuable. It is just more difficult to determine the impact due to the multiple processes that can provide the same capability.

One basic question that needs to be asked in dealing with the coupling effect is whether loosely coupled categories, such as software, can indeed be programmable from the capabilities-based programming standpoint. Based on the preliminary review presented in this chapter, it appears that some portion of software sustainment is so programmable, whereas others are not. A significant portion of the software sustainment activities are nonroutine urgent or emergency work. Some of this work might be unprogrammable as it is currently defined, in the sense that it is impractical to make the requirements process capabilities based. Ironically, this unprogrammable part can be more easily estimated based on the statistical analysis of the historical data and be "programmed" into the POM/presidential budget buildup process with relatively stable predictability.

The challenge is to gain sufficient understanding of the process to maximize the capabilities-based programmable part of the software sustainment. As this challenge is common to other business areas, including sustaining engineering and technical orders, we discuss possible mitigating policies in the next chapter. For the remaining part of this chapter, we focus on challenges, and possible solutions, that are idiosyncratic with software.

Specific Software Challenges

Maintaining Sustained Capability for Obsolete Systems

A lot of the Air Force's weapon system software is written in antiquated computer languages. And this software code must interface with hardware specific to Air Force platforms. One of the key challenges faced by Air Force software maintenance managers is hiring and keeping software personnel, both contract and organic, who can handle the obsolete legacy systems and understand the interfaces. There is significant competition for the same resources outside the Air Force with better pay and opportunities to work on the latest software systems. Replacing these people becomes a difficult and costly undertaking once they are lost. For electronic-warfare legacy software, for example, it takes at least three to five years for a new software programmer to gain the requisite experience and proficiency.

Some parts of the software community within the Air Force use, as a general rule, 60 percent of the stated requirement as the minimum threshold level needed to be programmed to sustain the basic software maintenance capability without the risk of losing the expertise permanently. This level is also believed to be the level that "kept airplanes flying" without having material impact on the operational capability. Although we could not verify it, this 60-percent threshold could be the rationale underlying the consistent underfunding of software at about 60 percent of the programmed requirements. Additional discussion is provided in Chapter Seven on how this issue of maintaining sustained capability for obsolete systems relates directly to capabilities-based programming.

Software Maintenance Is Multifarious

Software maintenance involves different categories of software and different types of maintenance activities that can be scheduled on a routine, urgent, or emergency basis.

Software categories include

- operational flight program
- automated test equipment
- test program set
- electronic warfare
- mission planning system
- communications-electronics
- other.

Software maintenance performed under these categories can be weapon specific or common across multiple weapon systems.[4] For example, operational flight program software is normally associated with a specific weapon system and generally included within normal, scheduled updates for that weapon system. On the other hand, electronic-warfare software, although it could be weapon specific, is, in general, common across multiple systems and is fairly evenly split between routine and nonroutine work. In this way, the software category can be helpful in identifying whether maintenance activities will be weapon specific or common or whether they are routine or nonroutine. It can also be helpful in identifying business areas other than DPEM that provide funding. For example, depending on the location of the test instrument requiring the software support, automated test equipment software and test program set software can be funded variably by DPEM or MSD business areas.

Software maintenance can also be classified into three different types of activities, i.e., corrective, perfective, and adaptive maintenance:

[4] For example, B-1 aircraft weapon electronics are weapon-specific software, whereas the installable software module is electronic-warfare software that is common across multiple weapon systems.

- *Corrective* maintenance encompasses changes designed to correct latent errors or deficiencies in software programs that otherwise meet the criteria for operational suitability and usefulness. Although the types of activities within corrective maintenance can vary, the common reference point for this type of maintenance activity is correction with respect to the initial software baseline (e.g., correcting latent problems that were discovered after initial baseline was accepted or correcting known deficiencies that were not fixed during the development phase due to premature fielding of the baseline).
- *Perfective* maintenance represents changes that improve performance or other attributes within the bounds of the existing system-level weapon system specification (e.g., cockpit display enhancement). Sometimes, however, perfective maintenance is also used to cover software changes that are part of the scheduled modification but were not funded due to an early depletion of modification funds for that weapon system.
- *Adaptive* maintenance involves changes designed to adapt a weapon system to a changing environment. This type of maintenance activity can include software changes to accommodate increased memory in the hardware or changes to accommodate a new threat in the operating environment.

In principle, these definitions could assist in assigning potential operational impacts of servicing or not servicing a software change request (SCR). In practice, however, these types of maintenance are often grouped together such that many change requests are said to be of all three types. It may be possible to improve the "binning" of software maintenance work in ways that are operationally meaningful. For example, to the extent practical, corrective maintenance that affects the baseline capabilities (and, hence, the minimum maintenance needed to sustain mission capability) can be separated out from perfective and adaptive maintenance work and tracked more rigorously.

Software maintenance is also diverse in that it can be scheduled on a routine or nonroutine basis. Routine maintenance represents those software changes designed to keep the weapon system mission capable and ready within the established operational baseline for that weapon system. Routine maintenance is implemented within a fixed schedule period called a *block cycle*.[5] Block cycles generally range from 18 to 36 months, but the actual durations vary depending on the weapon system, software category, the nature of maintenance work included in the block cycle, and other factors. For example, a block cycle for the F-16 can be as long as 42 months; for electronic-warfare software, around 21 months; and, for intercontinental ballistic missiles, from 15 to 18 months. The length of block cycles can pose a challenge for programming, which is discussed later in this chapter.

Software maintenance that is one time, nonroutine, or urgent is not part of scheduled block cycles but nevertheless must be accommodated in a timely manner. These "nonblock" changes can include, for example, "what-if" and "compatibility" investigations (e.g., interoperability verifications), documentation requests, engineering troubleshooting (e.g., inability to load an old version of software), site visits to investigate deficiencies, independent testing of user-generated data, and so forth. The terms *nonblock changes* and *urgent changes* are sometimes used interchangeably. Another category of unscheduled software maintenance is referred to

[5] *Block cycle* is a generic term. A standard nomenclature does not exist. For example, for the F-16, a block cycle is referred to as a *software change unit* (SCU); for A-10, they are called *suites*; for B-52, they are referred to as *software block changes*; and, for mission planning system software maintenance, there is no block cycle.

as *emergency work*, which is used predominantly within the electronic-warfare software community. Emergency work is typically labor intensive, requiring work around the clock and a 72-hour turnaround time. A significant portion of electronic-warfare work is of this type.

Some of the challenges associated with the diverse nature of software maintenance have been addressed to some extent by the CAM process discussed in Chapter Two. CAM realigned and aggregated software programming on a weapon system basis rather than a business area basis. By embedding software maintenance requirements within weapon systems, the CAM office has been able to better align the requirements with weapon system operational priorities, thereby facilitating software trades across weapon systems. However, a framework and tools to accomplish these programming trade-offs have not yet been developed.

Software Funding Can Be Difficult to Track

Software maintenance funding is a challenge for a capabilities-based approach because, although most funding is within DPEM, some funding crosses other business areas and is drawn from multiple sources. The DPEM business area represents the largest source of software maintenance activities that can be tracked. Software maintenance can be also drawn from CLS, sustaining engineering, and technical order business areas.[6] CLS, potentially another large source of software maintenance funds, can be problematic from a programming standpoint because the software component is not currently broken out from the rest of weapon system maintenance and not tracked independently.

Tracking funding for software maintenance is also difficult due to blurred transition points between development, maintenance, and modification activities. The typical funding sources for these activities are as follows:

- Development activities are funded from research, development, test, and evaluation (RDT&E) (3600 funds). RDT&E funds are used for all initial hardware and software development efforts up to the point where an operational baseline is accepted. RDT&E funds are also used for software changes that increase weapon system capability beyond the existing baseline.
- Maintenance activities are funded from O&M (3400, 3740, and 3840 funds). O&M appropriations can also fund software changes that increase weapon system capabilities, provided the changes are relatively minor in scope and cost and the work can be accomplished during regular software maintenance.
- Modifications are funded from procurement (3010, 3011, 3020, and 3080 funds).

Although various policies and guidelines are available to fine-tune the funding process, transition points between these three types of software activities are sometimes unclear. For example, a key transition point between development and maintenance occurs when the software emerges from the RDT&E phase and is accepted by the weapon system user (this point is referred to as the *baseline*). Often, however, it becomes necessary to test some of the software's key functionalities in the field after the operational acceptance. Testing is an RDT&E activity,

[6] For sustaining engineering and technical orders, the software component is relatively minimal. For DPEM/DMAG and MDS/supply management activity group, the sustainment requirements are defined across EEICs (540 and 56000), and, for CLS, sustaining engineering, and technical orders, they are defined across Weapon System Management Support business areas.

but, when it occurs after operational acceptance and fielding, it might be funded as maintenance. Consequently, software maintenance work sometimes cannot be easily distinguishable from that covered by RDT&E or procurement funds.

The two primary challenges associated with the varied funding sources from the capabilities-based programming standpoint are (1) if multiple funding sources are used to implement a set of software maintenance activities that are operationally linked, it would be difficult to capture the full operational impact of those activities if only a subset of funding sources were considered and (2) it would be difficult to distinguish specific operational impacts associated with specific software maintenance activities if various of types of work (i.e., RDT&E, maintenance, and modifications; corrective, perfective, and adaptive) are combined into the same work package.

Practices for Establishing Software Baselines Vary

Unlike maintenance requirements of aircraft and engines, which are based on performance-based engineering specifications, software maintenance requirements are based on loosely defined software production *baselines*. There are varying approaches to establishing baselines, depending on the needs of specific weapon systems (e.g., hardware and software integration), the type of software being developed, particular circumstances underlying the software procurement (e.g., funding situation and the competency of the software developer), and other factors. Other than guidelines associated with acquisition practices, it appears there are currently no policy guidelines or standards for establishing software baselines that could be applied across multiple weapon systems.[7]

Software baselines are critical in capabilities-based programming in two important aspects. First, and the more important, software baselines establish the reference point that defines maintenance requirements and hence serve as a benchmark for determining any operational impact if shortfalls in funding arise. Second, baselines also define when a particular software service becomes maintenance, as opposed to development, triggering the appropriate funding mechanisms.

Lack of Standardization in Data Documentation Compromises Data Integrity

The lack of standardization in defining, uniquely characterizing, and documenting software maintenance work makes it difficult to analyze the data in an operationally meaningful way. Before the CAM initiative, the basic guidelines for developing software maintenance requirements were established in the software requirements review process (SRRP).[8] One of the key emphases in the SRRP was a disciplined approach to software data documentation, most of which has now become part of the CAM Centralized Access for Data Exchange (CAFDEx) database. Currently, all data and assumptions used in developing the software maintenance requirements are input into the software requirements application (SRA) database, which generates AFMC Form 230/231. Form 230/231 provides all supporting documentation used for building the software maintenance requirements.

[7] See, for example, U.S. Department of Defense, The Defense Acquisition System, Directive 5000.01, May 12, 2003.

[8] See Commander Air Force Materiel Command, Acquisition: Software Requirements Review Process, Air Force Materiel Command Policy Directive 63-4, April 16, 2003b, rescinded, and Commander Air Force Materiel Command, Acquisition: Software Requirements Review Process, Air Force Materiel Command Instruction 63-401, April 16, 2003a, rescinded.

Form 230/231 includes, among other elements, a PCN, software maintenance type, impact statement associated with the operational impact of not funding the work, task description, software category, resources required (in man-hours), variance statements to describe how requirements changed from one year to the next within the FYDP period or between the POM cycles, and related funding sources other than the O&M funds. Also provided in the SRA database are the details on the methodology used to convert the requirements into man-hours.

PCNs are used as the basic unit for programming requirements for software maintenance. They also represent a line item in the DPEM database that has a unique six-digit designation of software maintenance that can be tracked individually.[9] There are no standard procedures in the PCN assignation, so the definition of PCN varies greatly depending on the weapon system, lead command, software category, and other factors.

For example, a single PCN is typically assigned for a single set of maintenance services (i.e., a single block cycle). However, for weapon systems associated with the Air Force Special Operations Command (AFSOC), multiple PCNs are assigned for a single block cycle. The same is true for most electronic-warfare software. In mission planning system software, where there is no block cycle, the same PCNs are used year after year.[10] In some instances, a PCN is retired completely after the associated block cycle is complete. In other instances, the same PCN is used again for a subsequent block cycle. The same last three digits are sometimes used to designate the same type of software maintenance work. For example, for AFSOC software, the same three sets of last three digits are used for flight testing, coding, and analysis, respectively, regardless of which ALC performs the work.

There is also a great deal of variation on how Form 230/231s are delineated in relation to PCNs. For example, a single PCN can have a single Form 230/231, but multiple PCNs can also be included in a single Form 230/231. Moreover, the same PCNs can appear in multiple Form 230/231s. This lack of standardization makes it difficult to perform a meaningful analysis of data that has common application across multiple weapon systems or programs.

Although some of the data items in the SRA database and Form 230/231 were specifically intended to be useful in linking software requirements to some type of operational capability, there has not been sufficient standardization or discipline to ensure data integrity. For example, the impact statement entries in Form 230/231, which describe the impact of not funding a particular requirement, are not very useful as they are currently entered. Although quite detailed in some cases, these statements are qualitative and are not linked to a common set of metrics that could be compared. Most of the statements describe the technical impacts on software functionality, but they illuminate little about the impacts on weapon system operational capability.

The most critical problem related to the lack of standardization is the criteria used in prioritizing the software maintenance requirements. These prioritization schemes ultimately help in making deferral and trade decisions. The CAM office's recent attempt to develop a new prioritization scheme that can be applied across multiple weapon systems is a move in the right direction. As mentioned before, the new scheme is based on a set of common metrics related

[9] By convention, the first three digits represent, respectively, the lead command, the DPEM commodity category, and the Air Logistics Command assigned to perform or administer the maintenance work, and the last three digits represent a unique identification of the work to be performed.

[10] For example, the same three PCNs are used, respectively, for UNIX, affordable Portable Flight Planning System, and Joint Mission Planning System year after year, regardless of the associated block cycles or weapon systems.

to safety, readiness, mandate, and sustainability.[11] Under this new scheme, the requirements within weapon systems are prioritized by assigning a numerical value to represent a composite effect of the four metrics identified. Although this prioritization scheme has been met with limited success by the DPEM stakeholders for several reasons, some of the attributes can serve as useful guidelines in the documentation and augmentation of the current prioritization process.[12]

Standardization in requirements programming and data documentation would help to facilitate the capabilities-based programming for software sustainment by providing an operationally meaningful basis to compare the requirements and to allow appropriate software deferral decisions and trades across DPEM commodities and weapon systems.

Long Lag Times Create Fluidity in Software Maintenance Requirements

One of the greatest challenges for capabilities-based programming for software maintenance is that the requirements estimates vary over time more than most other maintenance estimates. Software requirements change at a number of junctures, including when new SCRs that have higher operational priority come in from the field, when the overall weapon system prioritization changes, and when unexpected funding shortages occur.

Long lag times between the initial programming phase and the budget execution phase often compound the problem by creating opportunities to make such changes. The resulting fluidity in requirements makes it difficult to directly link software maintenance requirements with the operational capabilities. As a result, it is also difficult to identify meaningful capability metrics that can be applied generally across different software categories and weapon systems to facilitate important deferral and trade decisions.

The programming process for software maintenance requirements involves four major stages. Taking the F-16 as an example, these stages span over a 42-month period and include

1. SCR collection and compilation (12 months)
2. buildup of fiscally unconstrained requirements, leading to the publication of the unconstrained requirements, sometimes referred to as logistics support requirements (LSRs) (nine months)
3. buildup and finalization of fiscally constrained POM/presidential budget submittal to the Congress (14 months)
4. congressional budget review and final authorization (seven months).

Appendix D provides a more detailed description of these stages and of the software maintenance requirements programming process in general.

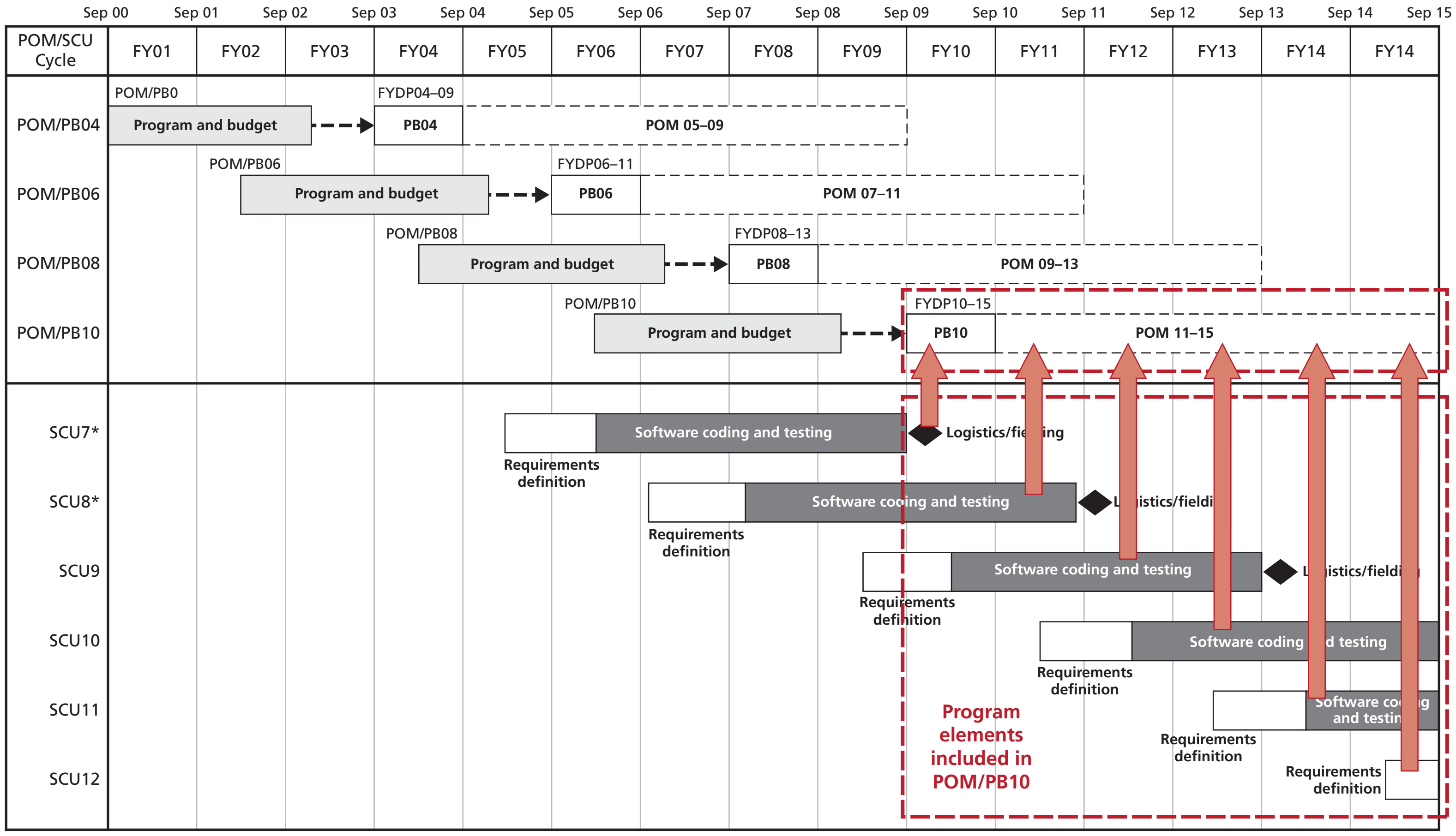
The process becomes complicated because the 42-month period extends over multiple POM cycles involving multiple block cycles. For illustrative purposes, Figures 6.1 and 6.2 present the programming cycles (i.e., POM cycles) juxtaposed against the execution cycles (i.e., block cycles) for F-16 software maintenance.[13] In these two figures, the top half represents

[11] These metric terms were defined in Chapter Two.

[12] Some of the reasons cited by the stakeholders include disagreements about the priorities, confusion about the methodology, and questions about the practicality and usefulness of the metrics and the approach.

[13] For the F-16, a block cycle is referred to as SCU, as shown in Figures 6.1 and 6.2. For consistency, we use the term *block cycle* instead of SCU in our F-16 discussions.

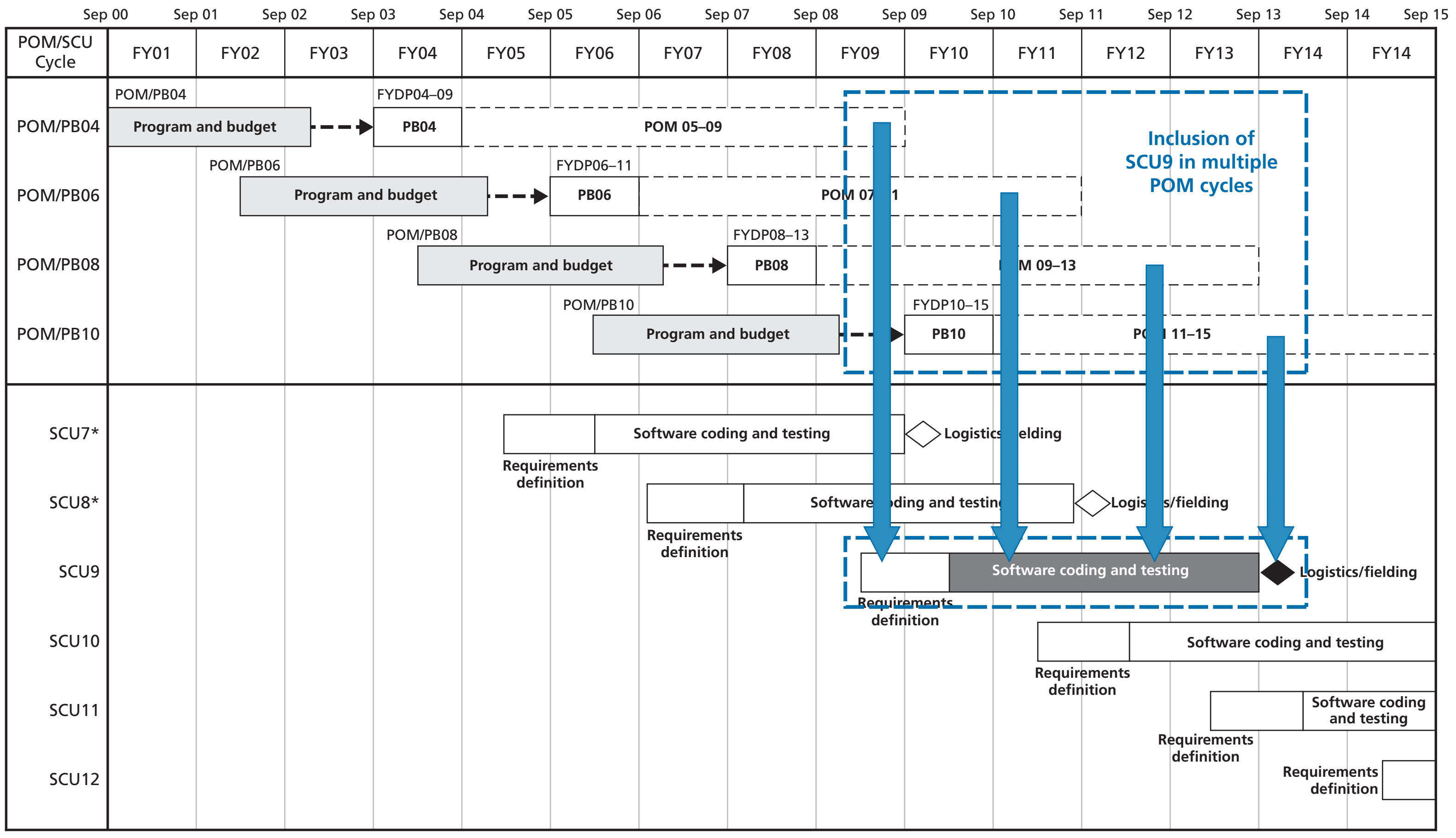
Figure 6.1
Multiple Block Cycles in a Single Program Objective Memorandum/Presidential Budget Cycle (F-16 example)



*Time lines are adjusted to reflect the actual execution dates.

RAND *TR905-6.1*

Figure 6.2
Multiple Program Objective Memorandum/Presidential Budget Cycles in a Single Block Cycle (F-16 example)



*Time lines are adjusted to reflect the actual execution dates.

RAND *TR905-6.2*

programming activities representing the 42-month POM cycles and the bottom half represents execution activities associated with each block cycle. Figure 6.1 illustrates the need to consider multiple block cycles in a given POM cycle, and Figure 6.2 illustrates the need for each block cycle to extend over multiple POM cycles in establishing the requirements.

In Figure 6.1, for the FY 2010 cycle (POM/PB10) where the FYDP period covers FY 2010 through FY 2015, for example, all block cycles that have one or more execution activities that fall within that FYDP period are included in the POM cycle (these activities are enclosed in the dotted box highlighted in Figure 6.1). This means that, when the programming and budgeting activities started in April 2006 for FY 2010, they had to consider the combined requirements for six different block cycles (SCU7 through SCU12), each having execution activities in varying stages of implementation during the FYDP. The software maintenance requirements scheduled for these block cycles would have different levels of fidelity depending on how far into the future the requirements are to be executed and on the time available to introduce changes in the requirements before the execution.

Conversely, a given block cycle is considered in multiple POM cycles, and the software maintenance requirements for the block cycle are changed and redefined progressively from one cycle to another for various reasons mentioned earlier. These peculiarities are illustrated in Figure 6.2 for block cycle 9 (SCU9). As shown, block cycle 9 (SCU9) is programmed and reprogrammed multiple times over four POM cycles, with the initial programming activities reflected in the FY 2004 POM cycle (POM/PB04) and the final programming activities in the FY 2010 POM (POM/PB10).[14] This means that the first set of software change requirements considered in block cycle 9 (SCU9) were collected starting in April 2000 (the beginning of the FY 2004 POM cycle or POM/PB04). These requirements, along with the additional requirements collected between April 2001 and April 2007, were reprioritized multiple times and inserted or deleted accordingly in the FY 2006 (POM/PB06), FY 2008 (POM/PB08), and FY 2010 (POM/PB10) POM cycles. The requirements definition will not be finalized for this block cycle until April 2010, 57 months after the beginning of the requirements process.[15]

Although the long lag times encourage significant changes in the content of requirements for software maintenance and complicate the requirements process, they also provide ample opportunities to refine and redefine the requirements more accurately. If an operationally meaningful and capabilities-based programming approach can be applied, this process would sharpen the estimates of operational impact of funding decisions. That is, the lag time itself is not an inherent challenge as long as the challenges related to any loose coupling between requirements and operational capabilities can be resolved.

Recommendations to Mitigate Challenges Peculiar to Software

There are a number of challenges confronting a capabilities-based programming approach for the DPEM software category beyond linking funding to operational capabilities. These challenges are peculiar to software and include (1) the multifarious nature of the software main-

[14] For simplification, the Amended Program Objective Memorandum (APOM) cycles are not included.

[15] It should be noted that, as shown in Figures 6.1 and 6.2, the final requirements for a given block cycle are determined as the first step in the execution phase. Appendix D provides additional detail on the process involved in the establishment of final requirements in the execution phase.

tenance activities, (2) the varied funding sources, (3) variation in software baselining, (4) the lack of standardization in documenting requirements and resulting data integrity concerns, and (5) fluidity in requirements due to long lag times. The fluidity in requirements represents the greatest of these challenges. It is at the core of all the other challenges identified, in that it can help to resolve most of the critical issues associated with the other challenges.

To deal with these challenges, we offer the following recommendations as potential remedies:

- Establish clear policy guidelines and standards in (a) the operational acceptance of functional capability and the establishment of the software baselines, both from the standpoint of the software developers and weapon system users, and (b) the documentation of the baseline, clearly linking functional technical requirements with the operational capabilities and impacts.
- Provide additional refinements, where practical, in "binning" the software maintenance activities on the basis of, for example, some ordinal ranking of the operational impacts and in exercising discipline in tracking these "bins" separately in the programming data documentation.
- Establish standardization in data documentation and in defining block cycles, PCNs, Form 230/231, and so on, in a more operationally meaningful way.
- Perform more rigorous statistical analysis of historical data (e.g., parametric as opposed to trend analysis) to guide the programming of nonblock changes.
- Conduct additional analyses to (a) better understand the coupling effect of scheduled block changes, (b) identify operationally meaningful metrics and attributes, and (c) ultimately establish and, where practical, standardize a capabilities-based requirements prioritization scheme across weapon systems to facilitate weapon system trades and deferral decisions for software maintenance.

With the exception of the first recommendation, which involves improved software baselining in the acquisition phase, these recommendations are, by design, process improvements with few cost implications. It should be possible to embed them within the current programming process without much effort, which could be facilitated through the CAM office.

CHAPTER SEVEN

Policy Options and Conclusions

Our discussion of quantitatively linking spending for depot-level maintenance to capabilities has focused on the utility of capability metrics, such as aircraft availability, and how to apply such a metric to balance capabilities across a range of commodities and weapon systems. The metric reflects important aspects of the utility of the weapon system to the operator, and, when tied to a higher-level metric, such as the ability to generate sorties, it can be related to DoD-level planning guidance and used to balance priorities among a diversity of resources and weapon systems. We also showed how optimization tools can then guide allocation of funds and indicate consequences of alternative funding options in terms of DoD-level planning objectives.

The key to this analysis is establishing a strategy for prioritization among weapon systems. We propose a weighting scheme based on the ratio of the maximum demand of each type of aircraft in the Defense Planning Scenarios or combatant commanders' plans divided by the number of combat-coded aircraft of that type. This measure yields a quantification of the degree to which an aircraft is a high-demand, low-density resource.

The measure has the strength of reflecting the priorities of national planning objectives and combatant commanders' needs. The more a weapon system is called for in plans relative to its supply, the higher the priority for its sustainment. A caveat is, of course, that these weighting factors are only as useful as the degree to which the plans reflect future demands on aircraft by combatant commanders, and they must be balanced by training needs, as estimated by the flying-hour program.

We also noted an additional weakness of this approach: It is not easily extended to noncombat aircraft—namely, training aircraft, such as the T-38 Talon. As these aircraft are for training only, they do not appear in plans and have no defined weighting factor. We suggested that this shortcoming might be overcome by assigning the training aircraft a weighting factor that is an average of the combat-coded ones or by clustering all weighting factors into tiers and assigning the training aircraft to the median tier.

But there is a further limitation to the approach that we applied to aircraft and engines, and that is that the method is not easily extended to business areas that are not easily linked to metrics, such as weapon system availability. One important such area is software maintenance. In Chapter Six, we noted a number of challenges that software maintenance presents to the capabilities-based programmer. In the remaining portion of this chapter, we discuss various policy options for addressing software maintenance and other sustainment areas that possess similar problematic attributes.

Salient Characteristics of Problematic Business Areas

The broader the scope of business areas that need to be embraced in budget trades, the broader the metrics needed to define their capabilities. For trades more inclusive than among aircraft and engines, we advocated a more comprehensive capability metric, such as the ability to generate AEF sorties. Yet, even this metric fails to fully capture the impacts of funding to all areas—in particular, the software maintenance area discussed in Chapter Six.

The most challenging characteristics of the software maintenance area that impede capabilities-based programming are these:

- The nature of the impacts of varying funding levels on operational objectives are varied and sometimes obscure.
- The lag time between funding and any operational impact can be very long.
- The impact occurs for the entire fleet, not individual tail numbers.
- A certain ambiguity exists in what constitutes a requirement.

These characteristics, although introduced in the context of DPEM software maintenance funding, are applicable to a wider range of business areas, e.g., funding of sustaining engineering and technical orders. Progress in developing approaches to capabilities-based programming centered on the above four characteristics should be widely beneficial in helping to set budget priorities and balance capabilities across disparate funding areas.

Policy Options

We explore three ways in which alternative policies might address these challenges. The first is to introduce policies to force funding decisions to affect existing metrics, such as aircraft availability. The second is to introduce a new metric that is responsive to the above four issues. The third is to define, in this case, software maintenance as a *capability*, not merely a supporting function. We discuss these ideas in turn.

Bundling of Tasks

Before discussing this first policy option, we note that we do not advocate it as one of the more fruitful of the alternatives. We introduce it nonetheless to highlight some of the general principles involved in programming and how deeply implicit policy decisions drive the process.

Consider software as an example. As a generalization, software requirements begin as a list of fiscally unconstrained SCRs that emerge out of a consultation among operators, often pilots, from the various MAJCOMs. These requests are prioritized, again generally by the operators, considering both the merits of each request and whether some requests can be addressed without much additional effort if others in the same area of code are being done regardless. With this list, a line is drawn that divides those that will be programmed from those that will not. In drawing this line, consideration is given to the existing capacity, especially organic capacity, to perform software maintenance.

During the year of execution, the funding available might be above or below this programmed level. If above, unprogrammed requests will be added to the block; if below, some of the lower-priority requests will be deferred to the next block cycle, which is a deferral that is typically around 18 months.

The process for determining PDM tasks is similar. Aircraft are due into depot for inspection on a calendar basis as determined by engineers. These engineers decide on a list of fiscally unconstrained tasks to perform on each aircraft due for induction. These tasks are bundled together into a single PCN. If the executed budget is less than programmed, rarely are individual tasks deferred until the next induction; rather, the entire PDM is deferred until the next fiscal year. The reason is that individual tasks would be costly to do at any time other than a PDM induction because of the efficiencies of doing these tasks together when the aircraft is considerably disassembled. Waiting until the next induction, which is often many years later, would incur unacceptable risk. Hence, the whole set of tasks is either executed or deferred until the next fiscal year.

It is not inherently easier to tie the execution or deferral of a stated PDM requirement (task) to operational impact than it is a software requirement (change request). For each task in a PDM inspection, an engineer would be hard pressed to justify the operational impact of doing, or not doing, that inspection. The very nature of inspections is that it is uncertain whether they will detect any problems, and the engineering assessments of the consequences of the flaw to be detected often involve considerable uncertainty.

The reason PDM is so easily tied to operational impact—in terms of aircraft availability—is that these tasks are bundled together, and a *policy decision* is made that they are all either executed or deferred as a whole, and, if they are not executed within a specified time, engineers will declare a lack of confidence in the safety of flight, with the result that the aircraft will be grounded.

SCRs—although often bundled in the form of blocks for a similar reason that inspection tasks are bundled into PDMs—have no similar policy that affects a discernible metric. So the potential option would be to create such a policy: to change the reporting status of the aircraft (weapon system) in some manner if bundled software changes are not executed within a specified time.

The strongest barriers to implementing such a policy for SCRs are two. First, SCRs originate with the operators, not the engineers. As such, they carry a perception of being a "wish list" more so than PDM inspections that, in principle, are tied to reproducible engineering analysis and can affect safety of flight. Software changes that involve safety of flight are treated as emergency and urgent, not block-cycle work. The impact of not executing block-cycle software maintenance is generally not that an aircraft will not be available but that its suitability and effectiveness to carry out its designed missions could be diminished.

Second, PDMs are done on individual aircraft, and software changes are done on large fleets (either the entire fleet of an MDS, a block series of an MDS, or, in the case of electronic warfare, across several MDSs). Hence, the impact of deferring software maintenance cannot be attributed to individual aircraft but must be assigned to an entire fleet. It is a very coarse, severe judgment that declares an entire fleet either available or unavailable given a diminished, but difficult-to-define, ability to carry out the designed mission. Hence, we entertain another policy option, one that attempts to assign some implication for deferred software maintenance to a novel, fleet-wide metric.

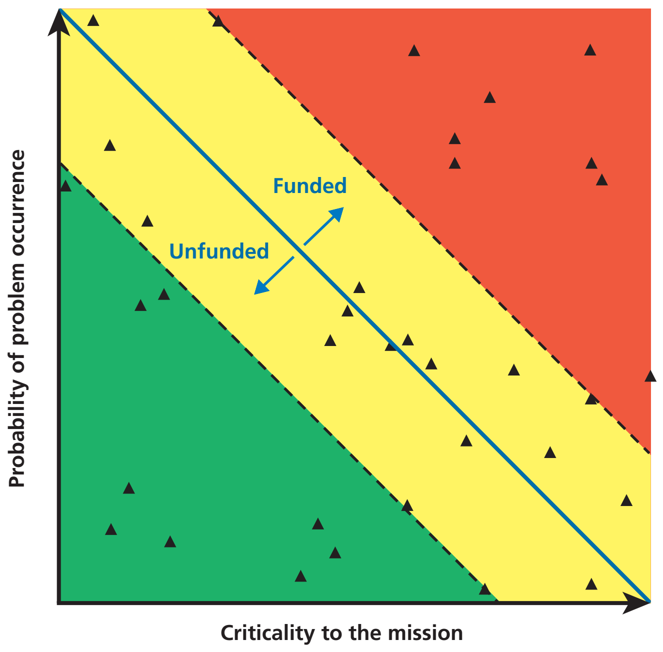
A Novel, Fleet-Wide Metric

There are sound reasons for not executing or deferring software blocks in their entirety, as is done in the case of PDM inductions. Nevertheless, real consequences do materialize if some fraction of those change requests are not serviced. These observations, in concert with the fact that the impacts are fleet-wide and not specific to particular aircraft, suggest a second, alternative policy option: to establish a novel, fleet-wide metric of the overall suitability and effectiveness of the fleet to carry out its designed missions or, more simply, a fleet (weapon system) health status. Readiness reporting then would be composed of two metrics: one like aircraft availability that is determined on each tail and gives an estimate of the immediate health status of the fleet, and a second that estimates the projected long-term health of the fleet by reporting on the funding of such areas as software maintenance, sustaining engineering, and technical orders.

The idea would be to determine a set of requirements in the requirements process as a baseline for the health of the fleet. Each of these change requests in the requirements can be thought of as a mitigation of a possible negative consequence to a mission. This risk can be assessed on a diagram of probability of occurrence of the problem and how critical to the mission the occurrence of that problem is. Figure 7.1 shows a schematic representation of such a process.

In the figure, each symbol represents an SCR. Associated with that request is a probability of occurrence of the event that the request is meant to mitigate and the impact to the mission should that event occur. The solid blue line, arbitrarily placed, that runs from the upper left to the lower right of the diagram indicates a notional funding level during execution. All change requests above this line are funded; all those below are unfunded. As additional funds become

Figure 7.1
Schematic Depiction of a Fleet-Wide Metric



RAND *TR905-7.1*

available, this curve shifts to the lower left. The fleet health of the software is then indicated by the region in which the funding line falls, shown as green, yellow, and red in the figure.

Ideally, the green region in the lower left should grade again to yellow, then red as funds are expended to address increasingly few, lower-priority change requests at the expense of funding other programs that may have more urgent needs. Hence, these funds, in general, would be better spent in another program. We have omitted these additional regions in the diagram for simplicity but also because of the enormous challenges in adequately defining these regions.

The central challenge with this second policy option is quantifying the probabilities and mission impacts in an objective, reproducible, and accepted process. It may be unclear what the probability of occurrence is, or the exact mission impact. For example, suppose the SCR is that a black-and-white display of information in the cockpit of a fighter is to be changed to a color display, where different colors indicate different sets of information. This change might improve the pilot's response time to a threat or might reduce fatigue during a mission. The probability of lack of sufficient response time or incidence of fatigue might be very difficult to assess, as well as the impact either might have, in general, on the ability to execute a mission.

Despite these limitations, the concept is an improvement on the concept of bundling the SCRs in the manner of PDMs. A fleet-wide risk metric handles a number of the important difficulties that are not well treated with metrics, such as aircraft availability and the methods described in earlier chapters. It, by nature, handles the impact of funding decisions on the fleet rather than individual aircraft, and it incorporates the lag times these effects have. Further, the concept can be used beyond software, to include other areas with the four problematic attributes, opening a programming trade space across a range of business areas. We conclude that the concept has strong merits, but the implementation presents some severe challenges.

Software Maintenance as a Capability

Software is now an integral part of weapon systems; every stage in the life cycle of these systems necessitates some level of software servicing. At the time of acceptance of a weapon system, when emerging from RDT&E, software defects are generally more difficult to detect than hardware defects. The number of software configurations to be explored and tested is of combinatorial size, and some permutations are encountered only after weapon system acceptance. Hence, even after a weapon system is accepted, it often retains unrevealed software defects. When these defects are eventually detected, they must be corrected using sustainment funds.

As the weapon system is used, other changes are needed as either the system or the environment in which it operates evolves. Many of these needed maintenance actions might be relatively benign, but some will continually arise that may be sufficiently critical that, until addressed, the entire weapon system is made unavailable. For aircraft, the consequence can be as severe as to *require grounding the entire fleet*.

To avoid such catastrophic consequences, the Air Force must retain the capability to address such crises when they arise. This ability is what we will call a *software maintenance capability*. In order to address these crises—as well as routine work—a cadre of software programmers is needed, with mastery of sometimes (otherwise) obsolete computer languages, intimate familiarity with specific blocks of code, and often deep knowledge of associated hardware and its interface with software. Such expertise—whether grown organically in the Air Force or supplied by a contractor—takes years to develop to the point at which an engineer is able to perform adequate software maintenance, especially servicing emergency and urgent requests.

This level of competency is in contrast to many other maintenance areas. For example, the expertise needed of a sheet-metal worker is more quickly developed, more available on the market, and more fungible across weapon systems and production lines. Hence, fluctuations in funding are more easily absorbed in traditional maintenance. For software, funding fluctuations that generate a layoff of certain software engineers could leave the Air Force without some capability to address certain software maintenance tasks for some time and could place the Air Force in a position of risk should emergency work in those areas be required.

So, the need to maintain software is integral to the operation of a weapon system; the ability to provide emergency and urgent software maintenance is part of the overall weapon system capability. This integrality leads us to entertain a radical reversal of the current requirements and programming process in a policy option that we consider the most promising for articulating software maintenance capability impacts: to program to provide a *software maintenance capability*.

The details of how this capability might be defined and reported remain to be resolved, but the overall contours of this capability perspective might be established as follows. For each weapon system or software category, the core component of this capability could be defined around stipulated areas of expertise that span the range of software maintenance. In other words, a set of skills that encompasses all possible maintenance tasks might include proficiency in certain languages, expertise in blocks of code, and expertise in hardware interfaces.

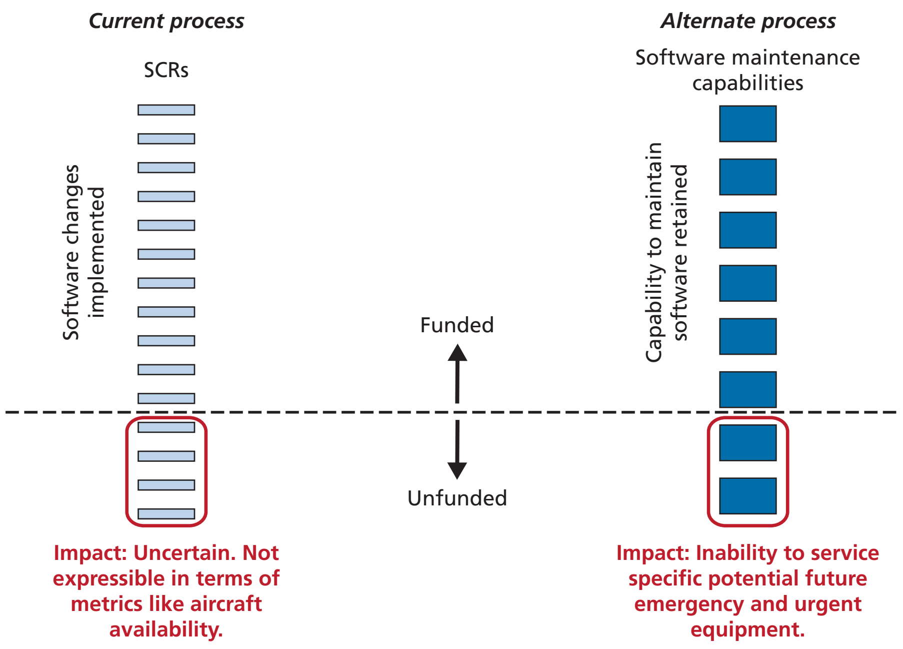
This exhaustive set of skills would define a critical level of capability. If the Air Force dropped below that level, it would risk not being able to service all potential kinds of software maintenance for that weapon system or software category (i.e., it would define the red zone in a "stop-light" chart). In addition to knowing where this level resides, the Air Force could also quantify the recovery time that would be needed to recuperate from falling below that level, based on the time needed to train new workers in these areas.

Capabilities above this critical level would then define increasing degrees of ability to work anticipated workloads. Workloads of the various types could be estimated from engineering data or from historical experience. These levels would then define increasing gradations in the readiness level of software maintenance capability. The final product for decisionmakers would be a readiness status of software maintenance.

This policy concept is depicted schematically in Figure 7.2. The dashed horizontal line in the figure represents the available funding (either during programming or execution), and anything below the line represents unfunded requirements. The column on the left of the diagram shows the current process. Requirements are defined by a list of SCRs. Unfunded requirements are then expressed as unfunded change requests. Their impact on the operator and their expression as unfulfilled capabilities remain unresolved problems. The column on the right expresses requirements in terms of skill sets that compose software maintenance capabilities. What is unfunded is expressed, under this policy option, as unfunded skills. These skills, once lost, are not readily regained, and hence the impact can be expressed in terms of the inability to service specific potential future emergency and urgent SCRs. As some of these requests may incur severe consequences on aircraft availability or degrade mission effectiveness, the impact is more easily expressed.

In viewing software maintenance as a capability, and defining a metric to measure it, a price is paid in not being able to readily do programming trades between software and incommensurable business areas—for example, those, such as aircraft and engines, that are more naturally expressed by such metrics such as aircraft availability. However, areas such as software

Figure 7.2
Schematic Depiction of a Software Capability Metric


Current process
SCRs
Software changes implemented
Alternate process
Software maintenance capabilities
Capability to maintain software retained
Funded
Unfunded
Impact: Uncertain. Not expressible in terms of metrics like aircraft availability.
Impact: Inability to service specific potential future emergency and urgent equipment.

RAND *TR905-7.2*

maintenance, sustaining engineering, and technical orders could share this common measure of capability, facilitating program trades among this family of business areas. There will also be challenges in building an exhaustive list of software maintenance capabilities needed and estimating the frequency of need for these, but these challenges are worth examining further to see whether they can be reasonably addressed. If these challenges can be overcome, this proposed policy option appears to be a considerable improvement over the current policies for addressing the impacts of the vicissitudes of funding in such areas as software maintenance.

Conclusions

In linking program funding for depot-level activities to operational capabilities rendered, we emphasize the importance of defining metrics that relate directly to objectives in operational plans and that are defined broadly enough to embrace a range of funding areas. Aircraft availability is a step in this direction, but the more general metric of generating training and AEF sorties is more inclusive of a wider range of resources (and thus enabling a wider range of programming trades), and it ties more directly to plans.

Using such a metric, programming trades can be prioritized according to weighting factors that express the ratio of the resource levels needed to meet the objectives in operational plans to those that are in the combat-coded inventory. This metric needs to be balanced by another metric that expresses the need to execute a flying-hour program to meet training needs (and the political pressures to sustain the force structure of large capital investments). The programmer needs tools that can explore how to balance resources in this trade space between the operator's priorities and the flying-hour priorities and must be able to express the risk in terms of the ability or inability to meet combatant commanders' plans.

Prototype tools demonstrate how to implement such a decision support system for the aircraft and engine commodity groups within DPEM and, by extension, to similar business areas, such as spare parts. Other business areas pose difficulties that are best treated with a different approach. These areas have the common attributes of (1) not being easily related to operational objectives, such as sortie generation; (2) having a long lag time between funding and any operational impact; (3) whatever impact occurs affecting the entire fleet, not individual tail numbers; and (4) possessing a certain ambiguity in what constitutes a requirement. Software maintenance, sustaining engineering, and technical orders share these characteristics.

These common attributes suggest grouping investment areas into categories that share common attributes. Each category would be evaluated according to a common metric, with trading first done within each category for a balanced capability as measured by that metric, then trading among the categories according to the judgment of senior leaders. For depot maintenance, we have entertained two categories: one measured by sortie generation (or its more limited proxy, aircraft availability), and the other a measure that reflects the longer-term health of the fleet. This latter measure would reflect the funding in such areas as software maintenance, sustaining engineering, and technical orders. Levels of capability could be defined that would express the ability to meet ongoing needs, rather than specific tasks.

We hope that, with these concepts, models, and prototype software tools, an additional stride has been contributed to the journey of linking a larger array of combat support funding to operational impacts and risks.

APPENDIX A

Aircraft Possession Purpose Codes

Table A.1
Possession Purpose Codes for Total Aircraft Inventory

Inventory Category		PPC	Description
PAI	PMAI	CA	Combat direct support
		CC	Combat mission
		IF	Industrial fund/Air Mobility Command airlift
	PTAI	TF	Training
	PDAI	CB	Combat development/equipment evaluation
		EH	Test support
		EI	Test and system evaluation
	POAI	CF	Combat auxiliary support
		ZA	Special mission
		ZB	Operational support/airlift
	Other	TJ	Ground instruction
		PJ	En route involving disassembly
		PL	En route transfer delivery
		PR	Operation-ready storage
BAI	DP	DJ	Awaiting depot maintenance
		DK	Contracted PDM at contractor site
		DL	Depot delivery flight
		DM	Depot work by unit or contractor at base
		DO	Organic PDM at depot
		DR	Postdepot awaiting delivery
	UPNR	BJ	Crash/battle damage waiting
		BK	Unit programmed maintenance
		BL	Extended transient maintenance
		BN	Crash damage, no AFMC assistance

Table A.1—Continued

Inventory Category		PPC	Description
		BO	Battle damage, no AFMC assistance
		BQ	Major maintenance, awaiting AFMC action
		BR	Major maintenance, awaiting parts
		BT	Not available due to transfer
		BU	AFMC authorized depot maintenance by unit
		BW	Weather damage, awaiting AFMC decision
		BX	Weather damage, no AFMC assistance
AR	UPNR	XJ	Excess awaiting disposition
		XW	Lost due to accident
		XZ	Lost or missing
	Other	XK	Inactive standby
		DN	MDS change upon depot maintenance
		EJ	Ground testing
		PP	New product await release

SOURCE: For inventory categories, Secretary of the Air Force, Operations Support: Aerospace Vehicle Programming, Assignment, Distribution, Accounting, and Termination, Air Force Instruction 16-402, August 1, 1997.

NOTE: PPC = possession purpose code. PMAI = primary mission aerospace vehicle inventory. PTAI = primary training aerospace vehicle inventory. PDAI = primary development/test aerospace vehicle inventory. POAI = primary other aerospace vehicle inventory. BAI = backup aerospace vehicle inventory. DP = depot possessed. UPNR = unit possessed, not reported. AR = attrition reserve.

APPENDIX B

Decision Support System Description

Description

The goal of the DPEM decision support system is to realize the model in Chapter Five in the form of a set of software tools. The decision support system is composed of three programming tools. The first is GAMS, which is commercial, off-the-shelf linear programming software that includes an optimization solver. The second component is a software application that facilitates the creation, maintenance, and manipulation of data, first to pass them to the optimization model, and then to summarize the model output data for the analyst. These tasks are performed with Microsoft Access. The third component facilitates the creation, maintenance, and execution control of graphical user interface panels and mechanisms. These are then used to orchestrate the loading of data to the linear programming model, the execution of a GAMS solver, and the summarization and display of model results. This component is written in Microsoft Visual Basic for Applications under Microsoft Access.

Figure B.1 shows the topmost control panel of the user interface of the prototype decision support system.

Figure B.2 shows the results in a typical data table, which can be edited by the user through one of the user interface panels.

At the time of execution, the Visual Basic for Applications interface calls and executes the appropriate GAMS code. A window appears (see Figure B.3) showing these steps, but the user does not need to have any proficiency with GAMS to use the software. All interactions with GAMS are automated.

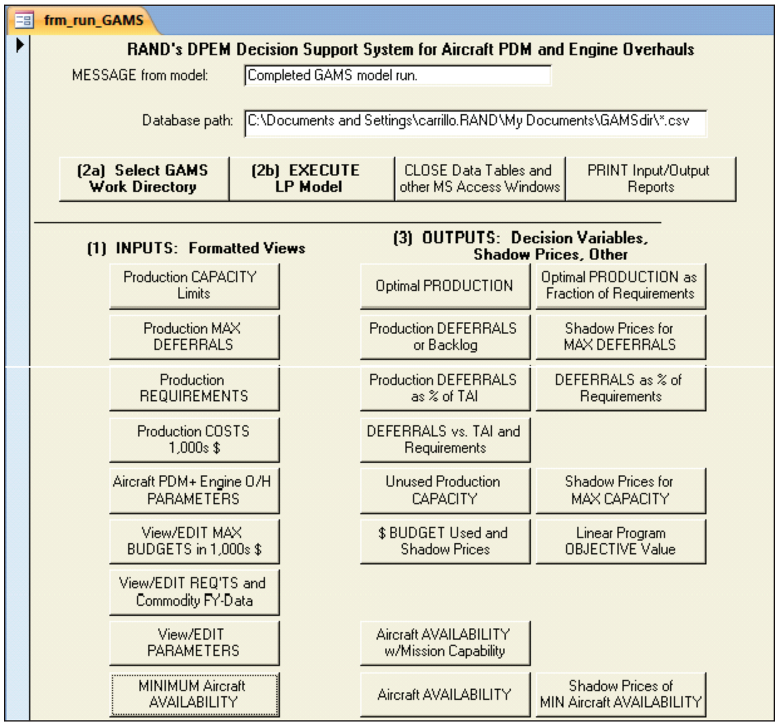
We now take the reader through a realistic tour of the use of the DPEM decision support system in looking for an optimal DPEM budget allocation. The following description presumes that all input data have already been entered in the various database tables, as described above. For the illustrative calculations below, these data values were extracted from the various databases indicated in Table 5.2 in Chapter Five; values for weighting factors are notional in order to keep the results unclassified.

Finding a Feasible Solution

Initial Steps

The calculation begins with a budget, specified for each year in the FYDP. Figure B.4 shows a typical starting point for the budget.

Figure B.1
The Decision Support System User Interface Control Panel

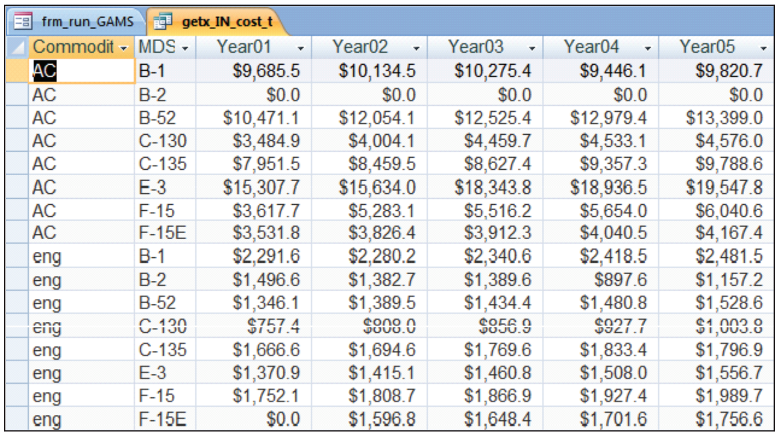

RAND's DPEM Decision Support System for Aircraft PDM and Engine Overhauls

MESSAGE from model: Completed GAMS model run.

Database path: C:\Documents and Settings\carrillo.RAND\My Documents\GAMSdir\*.csv
frm_run_GAMS
(2a) Select GAMS Work Directory
(2b) EXECUTE LP Model
CLOSE Data Tables and other MS Access Windows
PRINT Input/Output Reports
(1) INPUTS: Formatted Views
Production CAPACITY Limits
Production MAX DEFERRALS
Production REQUIREMENTS
Production COSTS 1,000s $
Aircraft PDM+ Engine O/H PARAMETERS
View/EDIT MAX BUDGETS in 1,000s $
View/EDIT REQ'TS and Commodity FY-Data
View/EDIT PARAMETERS
MINIMUM Aircraft AVAILABILITY
(3) OUTPUTS: Decision Variables, Shadow Prices, Other
Optimal PRODUCTION
Optimal PRODUCTION as Fraction of Requirements
Production DEFERRALS or Backlog
Shadow Prices for MAX DEFERRALS
Production DEFERRALS as % of TAI
DEFERRALS as % of Requirements
DEFERRALS vs. TAI and Requirements
Unused Production CAPACITY
Shadow Prices for MAX CAPACITY
$ BUDGET Used and Shadow Prices
Linear Program OBJECTIVE Value
Aircraft AVAILABILITY w/Mission Capability
Aircraft AVAILABILITY
Shadow Prices of MIN Aircraft AVAILABILITY

RAND *TR905-B.1*

Figure B.2
Representative Cost Data Table

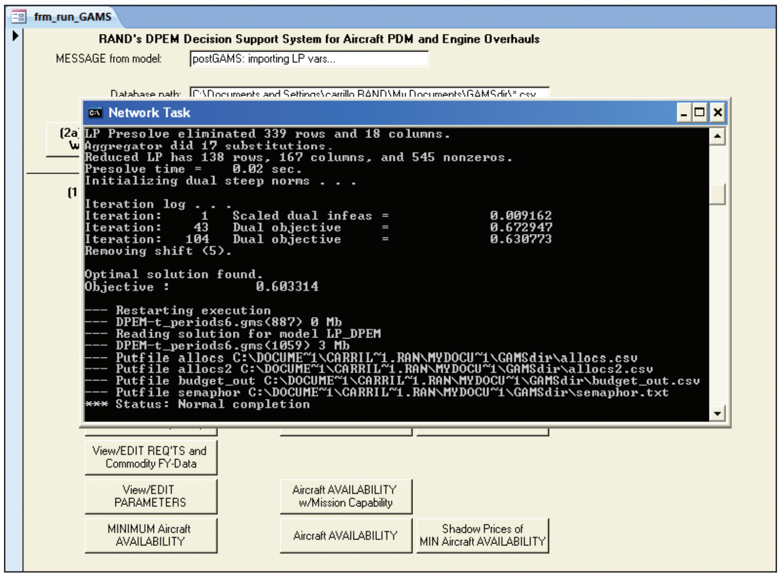

frm_run_GAMS
getx_IN_cost_t

Commodit	MDS	Year01	Year02	Year03	Year04	Year05
AC	B-1	$9,685.5	$10,134.5	$10,275.4	$9,446.1	$9,820.7
AC	B-2	$0.0	$0.0	$0.0	$0.0	$0.0
AC	B-52	$10,471.1	$12,054.1	$12,525.4	$12,979.4	$13,399.0
AC	C-130	$3,484.9	$4,004.1	$4,459.7	$4,533.1	$4,576.0
AC	C-135	$7,951.5	$8,459.5	$8,627.4	$9,357.3	$9,788.6
AC	E-3	$15,307.7	$15,634.0	$18,343.8	$18,936.5	$19,547.8
AC	F-15	$3,617.7	$5,283.1	$5,516.2	$5,654.0	$6,040.6
AC	F-15E	$3,531.8	$3,826.4	$3,912.3	$4,040.5	$4,167.4
eng	B-1	$2,291.6	$2,280.2	$2,340.6	$2,418.5	$2,481.5
eng	B-2	$1,496.6	$1,382.7	$1,389.6	$897.6	$1,157.2
eng	B-52	$1,346.1	$1,389.5	$1,434.4	$1,480.8	$1,528.6
eng	C-130	$757.4	$808.0	$856.9	$927.7	$1,003.8
eng	C-135	$1,666.6	$1,694.6	$1,769.6	$1,833.4	$1,796.9
eng	E-3	$1,370.9	$1,415.1	$1,460.8	$1,508.0	$1,556.7
eng	F-15	$1,752.1	$1,808.7	$1,866.9	$1,927.4	$1,989.7
eng	F-15E	$0.0	$1,596.8	$1,648.4	$1,701.6	$1,756.6

RAND *TR905-B.2*

We begin by not allowing any deferrals in the last year of the planning period. This set of constraints is too tight for the given demands, and no feasible solution can be found (Figure B.5). Relaxing this constraint to allow deferrals of up to 20 percent of production requirements still does not yield a feasible solution.

Figure B.3
General Algebraic Modeling System Execution Triggered by Decision Support System Programs


frm_run_GAMS
RAND's DPEM Decision Support System for Aircraft PDM and Engine Overhauls
Network Task
View/EDIT REQ'TS and Commodity FY-Data
View/EDIT PARAMETERS
MINIMUM Aircraft AVAILABILITY
Aircraft AVAILABILITY w/Mission Capability
Aircraft AVAILABILITY
Shadow Prices of MIN Aircraft AVAILABILITY

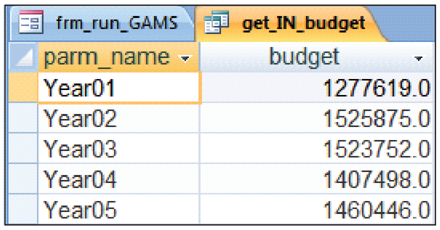
MESSAGE from model: postGAMS: importing LP vars...

Database path: C:\Documents and Settings\carrillo.RAND\My Documents\GAMSdir\*.csv

```
LP Presolve eliminated 339 rows and 18 columns.
Aggregator did 17 substitutions.
Reduced LP has 138 rows, 167 columns, and 545 nonzeros.
Presolve time =    0.02 sec.
Initializing dual steep norms . . .

Iteration log . . .
Iteration:     1   Scaled dual infeas =            0.009162
Iteration:    43   Dual objective     =            0.672947
Iteration:   104   Dual objective     =            0.630773
Removing shift (5).

Optimal solution found.
Objective :           0.603314

--- Restarting execution
--- DPEM-t_periods6.gms(887) 0 Mb
--- Reading solution for model LP_DPEM
--- DPEM-t_periods6.gms(1059) 3 Mb
--- Putfile allocs C:\DOCUME~1\CARRIL~1.RAN\MYDOCU~1\GAMSdir\allocs.csv
--- Putfile allocs2 C:\DOCUME~1\CARRIL~1.RAN\MYDOCU~1\GAMSdir\allocs2.csv
--- Putfile budget_out C:\DOCUME~1\CARRIL~1.RAN\MYDOCU~1\GAMSdir\budget_out.csv
--- Putfile semaphor C:\DOCUME~1\CARRIL~1.RAN\MYDOCU~1\GAMSdir\semaphor.txt
*** Status: Normal completion
```

RAND *TR905-B.3*

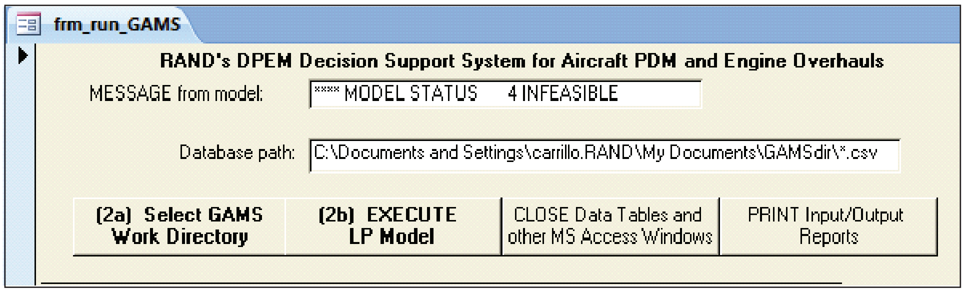
Figure B.4
Initial Tentative Annual Budgets
($ thousands)



parm_name	budget
Year01	1277619.0
Year02	1525875.0
Year03	1523752.0
Year04	1407498.0
Year05	1460446.0

RAND *TR905-B.4*

Figure B.5
Depot Purchased Equipment Maintenance Model Output: Infeasible Solution


frm_run_GAMS
RAND's DPEM Decision Support System for Aircraft PDM and Engine Overhauls
(2a) Select GAMS Work Directory
(2b) EXECUTE LP Model
CLOSE Data Tables and other MS Access Windows
PRINT Input/Output Reports

MESSAGE from model: **** MODEL STATUS 4 INFEASIBLE

Database path: C:\Documents and Settings\carrillo.RAND\My Documents\GAMSdir\*.csv

RAND *TR905-B.5*

A strategy we recommend in such a case is to begin by relaxing all policy and budget constraints. In this case, we increased the annual budgets to \$5 billion, the capacities to 80 percent, and the allowed deferrals to 30 percent of production requirements. With these settings, a feasible solution was found. By checking the outputs of the models (obtained through the control panel—see Figure B.2), one finds that the actual budgets used by the model in obtaining an optimal solution did not exceed \$2.00 billion in any one year.

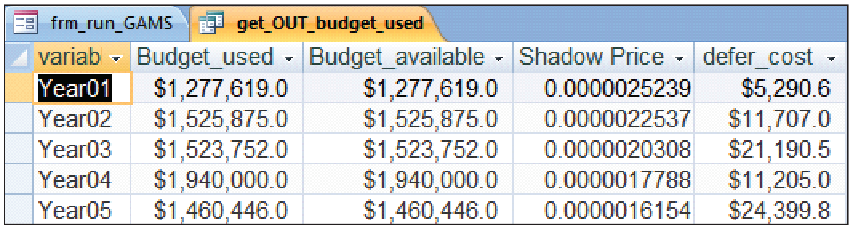
We now try to get back as close to the realistic constraints as possible and still retain a feasible solution. Reverting back to the proposed initial annual budgets, we find that these are fine for years 1, 2, and 3, but the initial proposed budget for year 4 causes the problem to become infeasible again. Therefore, we relax the budget for year four to \$5 billion and retain the original budget for year 5. These choices allow the problem to remain feasible. By trial and error, we were able to attain a feasible solution with the annual budgets proposed, except on year 4, where the achievable budget was \$1.7 billion.

Guidance Provided by the Shadow Prices

Figure B.6 shows that the model used the entire budget allowed during each budget year. Consistent with this observation, note that the respective shadow prices are positive (not zero). The ranking of the shadow prices for the annual budgets also gives an indication of which budget-year constraint is the most binding for the set of inputs chosen so far (where capacity and deferral constraints have been relaxed). In this case, the year 1 constraint is the most binding. The analysis of shadow prices should focus on constraints of the same type (e.g., budget constraints). The absolute value of a shadow price for a constraint in a group is not as important as the value relative to the others. The column in the far right of Figure B.6 shows the portion of the budget to pay for the increased costs attributed to deferrals (in this case, an assumed extra 5 percent).

It appears that we cannot tighten up the budget constraints to coincide with the proposed budget implicit in the funding spreadsheet. Continuing with this strategy of tightening up the constraints previously relaxed, we observe that the capacity shadow prices are mostly zero (only four out of the 60 entries are positive), indicating that we can tighten those constraints further.[1]

Figure B.6
Depot Purchased Equipment Maintenance Model Output: Analysis of Annual Budget Constraints (budgets in \$ thousands)



variab	Budget_used	Budget_available	Shadow Price	defer_cost
Year01	\$1,277,619.0	\$1,277,619.0	0.0000025239	\$5,290.6
Year02	\$1,525,875.0	\$1,525,875.0	0.0000022537	\$11,707.0
Year03	\$1,523,752.0	\$1,523,752.0	0.0000020308	\$21,190.5
Year04	\$1,940,000.0	\$1,940,000.0	0.0000017788	\$11,205.0
Year05	\$1,460,446.0	\$1,460,446.0	0.0000016154	\$24,399.8

RAND *TR905-B.6*

[1] The 60 entries arise by multiplying 4 years times the 15 aircraft or engine entries. The first year does not carry capacity constraints because it is assumed that any deferrals that are carried onto the first year are included as the first-year production demands.

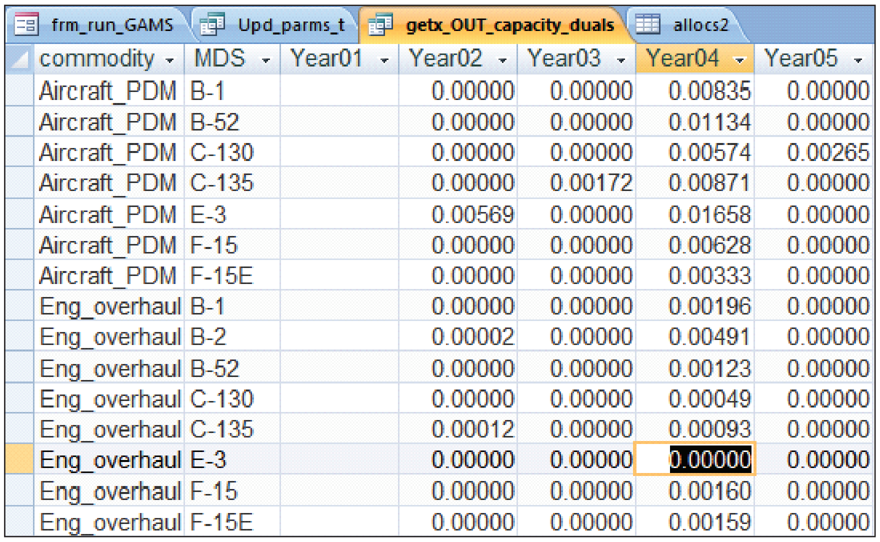
We then tighten those constraints to 89 percent of capacity; with this setting, an additional 15 shadow value entries become positive, indicating binding constraints. Figure B.7 shows those capacity shadow prices. Note that all capacity-constraint shadow prices for year 4 are positive except for the E-3 engine overhauls. This is consistent with the model having to produce as much as possible in year 4 to meet year 5 production requirements and any production backlog.

At this point, we try to tighten up the deferral limits but are unable to do so and also find a feasible solution. In our strategy, we tightened the budget and production constraints first, then the deferral limits were implicitly given less priority and could not be tightened any further. We loosened some of the production capacity constraints to 80 percent and then tightened the deferral constraints to 29 percent. With those settings, we were still unable to find a feasible solution. This indicates that, for our production demand, the ability to defer the meeting of production requirements is foremost in obtaining problem feasibility.

Analyzing Results

Figure B.8 shows a display of the deferrals selected by the model expressed as a percentage of the production requirements. Although some production is deferred, sufficient capacity and budget are allocated to prevent any aircraft from being grounded due to policy. From Figure B.8, one can see that the deferral limits are not binding in year 4, with the exception of B-2 engine overhauls.[2] The data shown in Figure B.7 indicate that it is the capacity constraints that were among the most binding constraints for year 4.

Figure B.7
Depot Purchased Equipment Maintenance Model Output: Analysis of Capacity-Constraint Shadow Prices

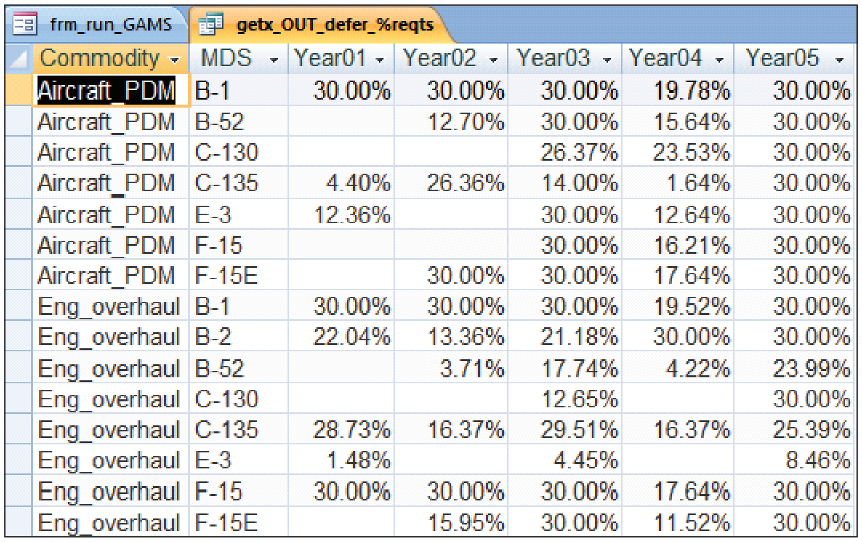


commodity	MDS	Year01	Year02	Year03	Year04	Year05
Aircraft_PDM	B-1		0.00000	0.00000	0.00835	0.00000
Aircraft_PDM	B-52		0.00000	0.00000	0.01134	0.00000
Aircraft_PDM	C-130		0.00000	0.00000	0.00574	0.00265
Aircraft_PDM	C-135		0.00000	0.00172	0.00871	0.00000
Aircraft_PDM	E-3		0.00569	0.00000	0.01658	0.00000
Aircraft_PDM	F-15		0.00000	0.00000	0.00628	0.00000
Aircraft_PDM	F-15E		0.00000	0.00000	0.00333	0.00000
Eng_overhaul	B-1		0.00000	0.00000	0.00196	0.00000
Eng_overhaul	B-2		0.00002	0.00000	0.00491	0.00000
Eng_overhaul	B-52		0.00000	0.00000	0.00123	0.00000
Eng_overhaul	C-130		0.00000	0.00000	0.00049	0.00000
Eng_overhaul	C-135		0.00012	0.00000	0.00093	0.00000
Eng_overhaul	E-3		0.00000	0.00000	0.00000	0.00000
Eng_overhaul	F-15		0.00000	0.00000	0.00160	0.00000
Eng_overhaul	F-15E		0.00000	0.00000	0.00159	0.00000

RAND *TR905-B.7*

[2] No B-2 PDM entries appear throughout our model outputs because the financial budget spreadsheet we used as the basis for the model input showed no B-2 PDM DPEM requirements.

Figure B.8
Depot Purchased Equipment Maintenance Model Output: Analysis of Deferrals of Production Requirements

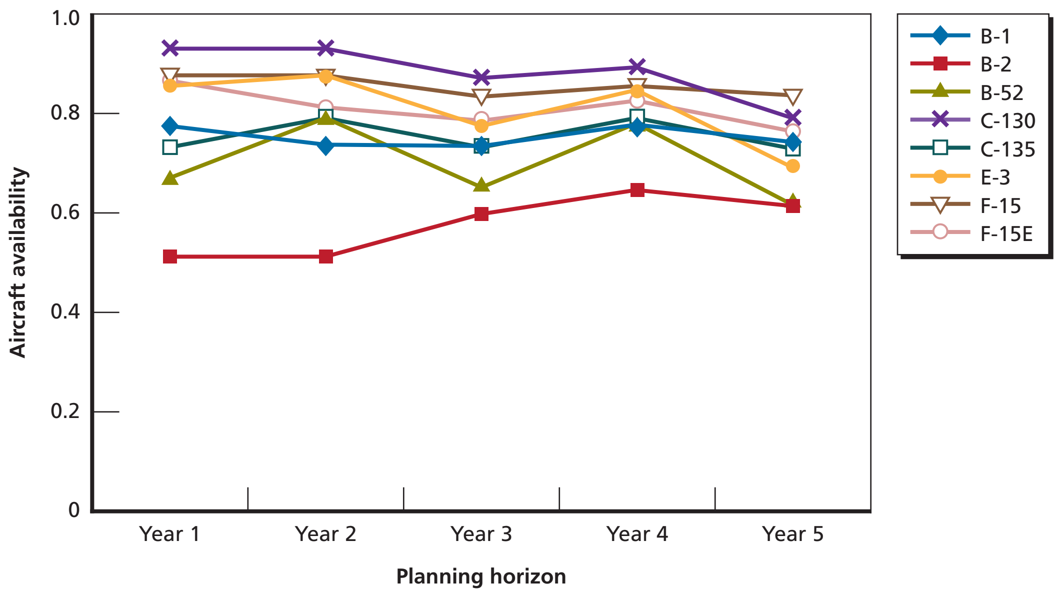


Commodity	MDS	Year01	Year02	Year03	Year04	Year05
Aircraft_PDM	B-1	30.00%	30.00%	30.00%	19.78%	30.00%
Aircraft_PDM	B-52		12.70%	30.00%	15.64%	30.00%
Aircraft_PDM	C-130			26.37%	23.53%	30.00%
Aircraft_PDM	C-135	4.40%	26.36%	14.00%	1.64%	30.00%
Aircraft_PDM	E-3	12.36%		30.00%	12.64%	30.00%
Aircraft_PDM	F-15			30.00%	16.21%	30.00%
Aircraft_PDM	F-15E		30.00%	30.00%	17.64%	30.00%
Eng_overhaul	B-1	30.00%	30.00%	30.00%	19.52%	30.00%
Eng_overhaul	B-2	22.04%	13.36%	21.18%	30.00%	30.00%
Eng_overhaul	B-52		3.71%	17.74%	4.22%	23.99%
Eng_overhaul	C-130			12.65%		30.00%
Eng_overhaul	C-135	28.73%	16.37%	29.51%	16.37%	25.39%
Eng_overhaul	E-3	1.48%		4.45%		8.46%
Eng_overhaul	F-15	30.00%	30.00%	30.00%	17.64%	30.00%
Eng_overhaul	F-15E		15.95%	30.00%	11.52%	30.00%

RAND *TR905-B.8*

An important aspect of the approach and model for capabilities-based programming is to be able to quantify risk. We quantify risk in terms of the capability metric. When tied directly to plans, this risk is expressed in terms of the fraction of the various plans that are projected to be able to be met for each year of the FYDP. The results here are unclassified, so we restrict the presentation to the projected aircraft availability for each year, shown in Figure B.9.

Figure B.9
Depot Purchased Equipment Maintenance Model Output: Projected Aircraft Availability



RAND *TR905-B.9*

APPENDIX C

Software Deferrals

Fluidity in requirements makes effective programming trades and deferral decisions difficult. To understand the extent of this problem, we assessed the recent historical level of software deferrals and examined how the MAJCOMs have managed without funding for a significant portion of the software requirements year after year.

What Has Been the Extent of Software Deferrals?

We discussed these issues with several Air Force personnel who are part of the software sustainment community and key stakeholders in the requirements process, including representatives from the MAJCOMs, the ALCs, the community of program group managers and system program directors, the CAM office, and the Air Staff.[1] In our discussions, a majority of the stakeholders reported that software deferrals are characteristically high, often as high as 40 percent of the stated requirements and consistently much higher than the deferral levels for aircraft and engines. They also reported that the long-term operational impacts of these chronic deferrals are poorly understood.

Our analysis of data derived from the DPEM database for POM/PB08 confirmed that software deferrals are consistently high, at about 40 percent. Tables C.1 and C.2 summarize these results, providing a general picture of the deferral levels for software and other major DPEM categories. Table C.1 presents the estimates of deferrals for the three major DPEM commodity categories (aircraft, engines, and software) and for DPEM as a whole. Also shown is an "Other" category that combines all other remaining DPEM commodities (including OMEI, non-MSD exchangeables, missiles, A/B/L, and storage). The data, drawn from the POM/PB08 DPEM database, cover the execution year (FY 2006), the budget year (FY 2007), and the program years (FY 2008 through FY 2013), which comprise the six-year FYDP period.

As shown, the deferral levels for the execution year and the budget year are generally much lower than those for the program years. For example,

- for the execution year (2006), the deferrals are 7 percent for aircraft, about 20 percent for software, and about 13 percent for DPEM as a whole
- for the budget year, the deferrals are about 21 percent for aircraft, about 28 percent for software, and about 25 percent for DPEM as a whole

[1] These meetings took place between September 2007 and June 2008.

Table C.1
Deferrals by Depot Purchased Equipment Maintenance Categories for Program Objective Memorandum/Presidential Budget 2008 Cycle

Year Type	Year	Budget Category	Aircraft	Engines	Software	Other	Total
Execution year	2006	LSR requirement ($)	2,070,286	758,565	710,569	516,926	4,056,346
		Funded ($)	1,915,314	620,278	570,621	428,734	3,534,947
		Deferral (%)	–7	–18	–20	–17	–13
Budget year	2007	LSR requirement ($)	2,029,475	811,604	830,121	565,990	4,237,190
		Funded ($)	1,607,685	539,171	601,776	450,218	3,198,850
		Deferral (%)	–21	–34	–28	–20	–25
Program year	2008	LSR requirement ($)	2,243,953	1,186,300	932,117	602,692	4,965,062
		Funded ($)	1,883,447	718,249	585,030	446,493	3,633,219
		Deferral (%)	–16	–39	–37	–26	–27
	2009	LSR requirement ($)	2,383,275	1,188,401	1,104,410	655,846	5,331,932
		Funded ($)	1,837,625	767,372	670,676	522,965	3,798,638
		Deferral (%)	–23	–35	–39	–20	–29
	2010	LSR requirement ($)	2,543,519	1,256,950	1,185,754	714,535	5,700,758
		Funded ($)	1,999,817	830,162	773,788	555,473	4,159,240
		Deferral (%)	–21	–34	–35	–22	–27
	2011	LSR requirement ($)	2,592,608	1,292,594	1,229,747	756,714	5,871,663
		Funded ($)	1,933,072	805,381	757,271	523,346	4,019,070
		Deferral (%)	–25	–38	–38	–31	–32
	2012	LSR requirement ($)	2,530,148	1,343,321	1,313,368	767,525	5,954,362
		Funded ($)	1,718,339	934,589	796,078	533,356	3,982,362
		Deferral (%)	–32	–30	–39	–31	–33
	2013	LSR requirement ($)	2,666,642	1,402,153	1,353,945	824,735	6,247,475
		Funded ($)	1,937,122	958,231	812,674	576,263	4,284,290
		Deferral (%)	–27	–32	–40	–30	–31

SOURCE: DPEM data from POM/PB08.

NOTE: For each year, the fiscally unconstrained LSR and the funded amounts, which are equivalent to the budget authorizations, are presented. (Strictly speaking, the funded requirements are equal to program authorizations, which may deviate slightly from the budget authorizations. For all practical purposes, we assume that program authorizations are equal to budget authorizations.) The deferrals are estimated with respect to the fiscally unconstrained LSR based on the assumption that the unconstrained requirements represent a true measure of the operational capability needed by the lead commands without the fiscal constraint (but with the depot maintenance capacity and resource constraints as embedded within the existing force structure).

- over the six-year FYDP period (rows 3–8 in the table), deferrals for DPEM as a whole are fairly consistent, averaging about 30 percent, much higher than the 13 and 25 percent for DPEM as a whole during the execution and budget years, respectively. The aircraft category has the lowest deferrals compared with the other DPEM categories, but the deferrals fluctuate significantly between 16 and 32 percent during the FYDP period. Software has similar, consistently high deferral levels, at about 40 percent, the deferral level echoed by many software sustainment stakeholders.

In Table C.2, we present analyses focused on deferral data associated with software only from the same POM/PB08 DPEM database. For each year (execution, budget, and program/FYDP), the deferral levels are presented for organic (EEIC 54001) and for CDM non-DMAG (56000). The other two EEICs in the DPEM software category, i.e., contract (54000) and interservice (54002 for DMISA), are not presented because their share of the total software budget is negligible. As before, the deferral levels for the execution year and the budget year are generally much lower than those for the program years.

Why Have the Software Deferrals Been So High?

One of the factors that contributed to these high deferral levels was the now-superseded software requirements process, which was not sufficiently rational to allow a reasonable picture of true software deferrals.

Historically, before CAM introduced a weapon system focus approach in the programming process, the DPEM requirements buildup activities followed a rough guideline that involved prioritization by DPEM commodity categories. After the Air Staff established the total DPEM topline budget, the old process involved (1) giving first priority to aircraft, missiles, engines, and storage (AMARG) DPEM categories by funding these categories at 100 percent; (2) deferring the requirements from the other four DPEM categories (i.e., software, OMEI, non-MSD exchangeables, and A/B/L) until a funding level of 60 percent of the requirements was achieved (this would be accomplished by spreading the cuts proportionally across these business areas, often referred to as a *peanut butter spread*); and (3) if the top line was still short of the requirements, deferring the requirements for the first four categories (i.e., aircraft, mis-

Table C.2
Software Deferral Levels for Program Objective Memorandum/Presidential Budget 2008 Cycle

Year Type	Year	Organic EEIC 54001 (%)	CDM Non-DMAG EEIC 56000 (%)	Total (%)
Execution	2006	–26	–12	–20
Budget	2007	–32	–25	–28
Program	2008	–37	–37	–37
	2009	–37	–41	–39
	2010	–34	–35	–35
	2011	–35	–40	–38
	2012	–36	–42	–39
	2013	–36	–42	–40

SOURCE: DPEM data from POM/PB08.

siles, engines, and storage) far enough into the future that the top line is met.[2] In deferring the high-priority commodity requirements in step 3, there were some additional general priority guidelines (e.g., deferring engines before aircraft); however, within a given category, the deferrals were accomplished again through proportionate cuts.

There was some logic behind this old approach. First, there was (and is) a general belief that the four lower-priority DPEM commodities are more adaptable and fungible in the sense that (1) the requirements are much more fluid and (2) the nature of the software sustainment work is such that there are sufficient work-arounds to deal with various funding shortages. A general rule of thumb and assumption underlying this belief was that the 60-percent level represented a minimum threshold capacity that retained the ability to maintain a minimum level of resources in the work force (both organic and contract) without permanently losing the expertise needed.[3] This threshold was also considered as the level that "kept airplanes flying" without having material impact on the operational capability.

Several work-around methods were also used frequently to deal with the funding shortages. Examples of software work-arounds included (1) spreading the work and extending the schedule; (2) eliminating the requirements or providing a partial solution, including limiting the testing efforts or combining two or more tests together to gain some efficiency; and (3) realigning workload between organic and contractor or within program element code for similar systems for potential economies-of-scale benefit.

It is unclear where the 60-percent rule originated and whether some of the work-arounds may eventually have material operational impacts. The adaptability and fungibility of the lower-priority DPEM commodities can be viewed as both its strengths and weakness. In many regards, the requirements for these commodities in the old programming process were considered more level-of-effort than necessity, and the operational impacts of underfunding the requirements were viewed as less apparent or immediate.

How Have the Major Commands Managed with Chronically Unfunded Requirements for Software Maintenance?

To understand the how the MAJCOMs have dealt with chronically unfunded requirements, we met with many of the key software sustainment stakeholders and functionals from the MAJCOMs and obtained detailed historical data associated with actual execution and distribution of the software sustainment funds for FYs 2006 and 2007.[4]

Table C.3 explains how the fallouts during the execution phase have allowed the MAJCOMS to recover some of the requirements that were deferred or unfunded during the

[2] Our description of the old DPEM programming process is based on interviews and discussions with some of the key stakeholders in the software sustainment community.

[3] Because many of the legacy systems are becoming obsolete, it is often difficult to replace software sustainment personnel (both organic and contract) once they are lost. For electronic-warfare legacy software, for example, it takes at least three to five years for a new software programmer to gain the necessary experience.

[4] For each weapon system, these actual data included, among others, unconstrained LSRs, initial distributions following the budget authorizations, the end-year revised distributions (obligations), and additional details regarding the sources of additional funds recovered from various fallouts, including supplemental appropriations associated with operations in Iraq and Afghanistan.

programming phase.[5] The table summarizes the actual execution data for all DPEM commodity categories from Air Combat Command (ACC) for FY 2006. For each DPEM commodity and corresponding EEICs, the table presents (1) the LSR unconstrained requirements, (2) the initial distributions, (3) the revised distributions at the end of the year, (4) the percentage of deferrals associated with initial and revised distributions with respect to the LSR unconstrained requirements, and (5) the percentage of these deferrals recovered during the execution year from all fallouts combined and those recovered from supplemental appropriations only.

As shown, the revised distributions indicate that a significant recovery of the deferrals took place during execution. For software, more than 25 percent (or 12 percent of the 41 percent) of the initial deferrals were recovered during the year and none came from supplemental appropriations.[6] For ACC as a whole, the total deferrals associated with the initial distributions were about 24 percent; about one-third (or 8 percent of the 24 percent) was recovered during the execution year and about 3 percent of this 8-percent recovery was due to the supplemental appropriations.

To determine whether the level of recovery varied significantly across different weapon systems or programs, we traced the progression of the DPEM funding levels from the unconstrained LSRs in the programming phase to the revised distributions at the end of execution year for a selected weapon system and program. Table C.4 presents a summary of our analysis of these data for four selected weapon systems and programs that were representative of both weapon-specific and common programs: B-1, F-16, electronic warfare, and mission planning system software. The data presented for B-1 and F-16 include multiple DPEM categories, whereas electronic warfare and mission planning system are primarily software.

The table reveals that both the deferral levels and the end-year recovery levels for software maintenance requirements vary greatly across different weapon systems and programs, and from year to year. For example, based on the two-year data, the funded amount for F-16 generally exceeded the LSR unconstrained requirements; a significant level of additional funds was also allocated to this weapon system during the execution phase. B-1, on the other hand, was funded at 20 to 30 percent below the unconstrained level, but a significant portion of this unfunded amount was recovered during execution. For electronic warfare and mission planning system, most of which is software, the deferred levels remained quite high except for what appears to be an anomaly in FY 2007 for mission planning system.

From the data analysis presented above, it appears that the "true" deferrals as reflected in the execution phase might be lower than the deferral levels that can be inferred from the programming phase. As discussed in Chapter Six, the reason for this difference might be due to the inherent fluidity in the nature of software sustainment work and the challenges associated with developing the requirements that are capabilities based. The results of the data analysis do demonstrate, however, that there is some foundation for concerns about the fidelity of the requirements for software maintenance. There could be some utility in tracking the actual execution data on a continuing basis and analyzing these data in some detail to gain additional insights for purposes of ultimately guiding the requirements efforts.

---

[5] *Fallouts* are defined as surplus funds that are not used for the program for which they were obligated during the execution year and that are made available for other programs.

[6] Note that the software category does not qualify for global war on terrorism (GWOT) funding. Our discussions with lead commands indicated that the software recovery sometimes came from the reallocation of the aircraft funding, which effectively cascaded down from the GWOT funding allocated to the aircraft category.

Table C.3
Software Deferral Recovery During Execution Phase for Air Combat Command

			Actual Execution ($)		Deferral (%) (LSR)		Recovered (%)	
Commodity	**EEIC**	**LSR Unconstrained ($)**	**Initial Distributions**	**Revised Distributions**	**Initial Distributions**	**Revised Distributions**	**From All Sources**	**From GWOT**
SW	540/56000	472,863	278,738	334,560	–41	–29	12	0
Aircraft	541/56010	739,100	652,563	691,644	–12	–6	6	4
Missiles	542/56020	7,586	7,681	8,553	1	13	12	0
Engines	543/56030	236,377	189,826	201,720	–20	–15	5	2
OMEI	544/56040	43,681	26,783	35,060	–39	–20	19	5
Exchange	545/56050	66,592	41,501	50,248	–38	–25	13	4
A/B/L	546	3,230	2,353	2,646	–27	–18	9	0
AMARG	548	105	105	40	0	–62	–62	0
Subtotal		1,569,534	1,199,550	1,324,471	–24	–16	8	3

SOURCE: Initial and revised distribution data from ACC, FY 2006.

NOTE: SW = software.

Table C.4
Recovery of Deferrals for Selected Weapon Systems

		MDS[1]			
Fiscal Year	Budget Type	B-1	F-16	Electronic Warfare	Mission Planning System
2006	LSR unconstrained ($)	247,207	94,942	26,381	44,973
	Funded (initial distribution, $)	192,122	95,448	13,971	24,997
	Deferral (%)	–22	1	–47	–44
	Obligations (closeout, $)	233,620	114,511	14,925	24,997
	Deferral (%)	–5	21	–43	–44
2007	LSR unconstrained ($)	289,454	75,460	36,612	23,268
	Funded (initial distribution, $)	201,167	94,393	23,623	42,607
	Deferral (%)	–31	25	–35	83
	Obligations (closeout, $)	228,985	118,001	25,729	39,140
	Deferral (%)	–21	56	–30	68

SOURCES: DPEM data from POM/PB06, APOM/PB07, POM/PB08; initial and revised distributions from ACC (FY 2006 and FY 2007).

The detailed execution data, such as those presented in the above discussions, have been maintained by the lead commands that have had the responsibility in the past for the execution of the DPEM funds. It appears that some level of knowledge has been accumulating in the lead commands about the execution of the funds such that this knowledge, including the level and nature of potential recovery of unfunded requirements during execution, has been reflected in the requirements process. Under CAM, the fund execution responsibility has been transferred to the CAM office, and the lead commands are no longer keeping track of the detailed execution data, such as those presented in the above discussions. In establishing the process under CAM, consideration should be given to the DPEM software programming and execution life cycle so that this knowledge is not lost in the transition.

APPENDIX D

The Software Requirements and Execution Process

As mentioned in Chapter Six, in contrast to aircraft and engines, one of the key challenges in establishing capabilities-based programming for software sustainment is that the requirements change constantly over the multiple POM cycles from their initial programming stage to execution. This fluidity in requirements makes it difficult to identify meaningful capability metrics that can be applied generally across different software categories and weapon systems. Some of the key software challenges associated with this "fluidity factor" can be best understood by reviewing the software sustainment requirements process in some detail.

Programming Phase

Figure D.1 provides an overview of the software sustainment requirements process. For illustration purposes, the flow diagram focuses on the programming exercise for the PB10 budget buildup and the POM cycle associated with FYDP 2010 through 2015.[1] The entire exercise spans more than a three-year period, beginning effectively in April 2006 and ending in September 2009.[2]

This time span consists of the following four major programming and budgeting stages (these stages are highlighted by the bold broken lines in Figure D.1):[3] (1) SCR collection and compilation (12-month period extending from April 2006 through March 2007), (2) buildup of fiscally unconstrained requirements leading to the publication of the unconstrained requirements, sometimes referred to as LSRs (nine-month period extending from April 2007 to December 2007), (3) buildup and finalization of fiscally constrained POM/PB budget leading to the OSD budget submittal to the Congress (14-month period extending from January 2008 through February 2009), and (4) congressional budget review and final authorization (seven-month period extending from March 2009 to September 2009).

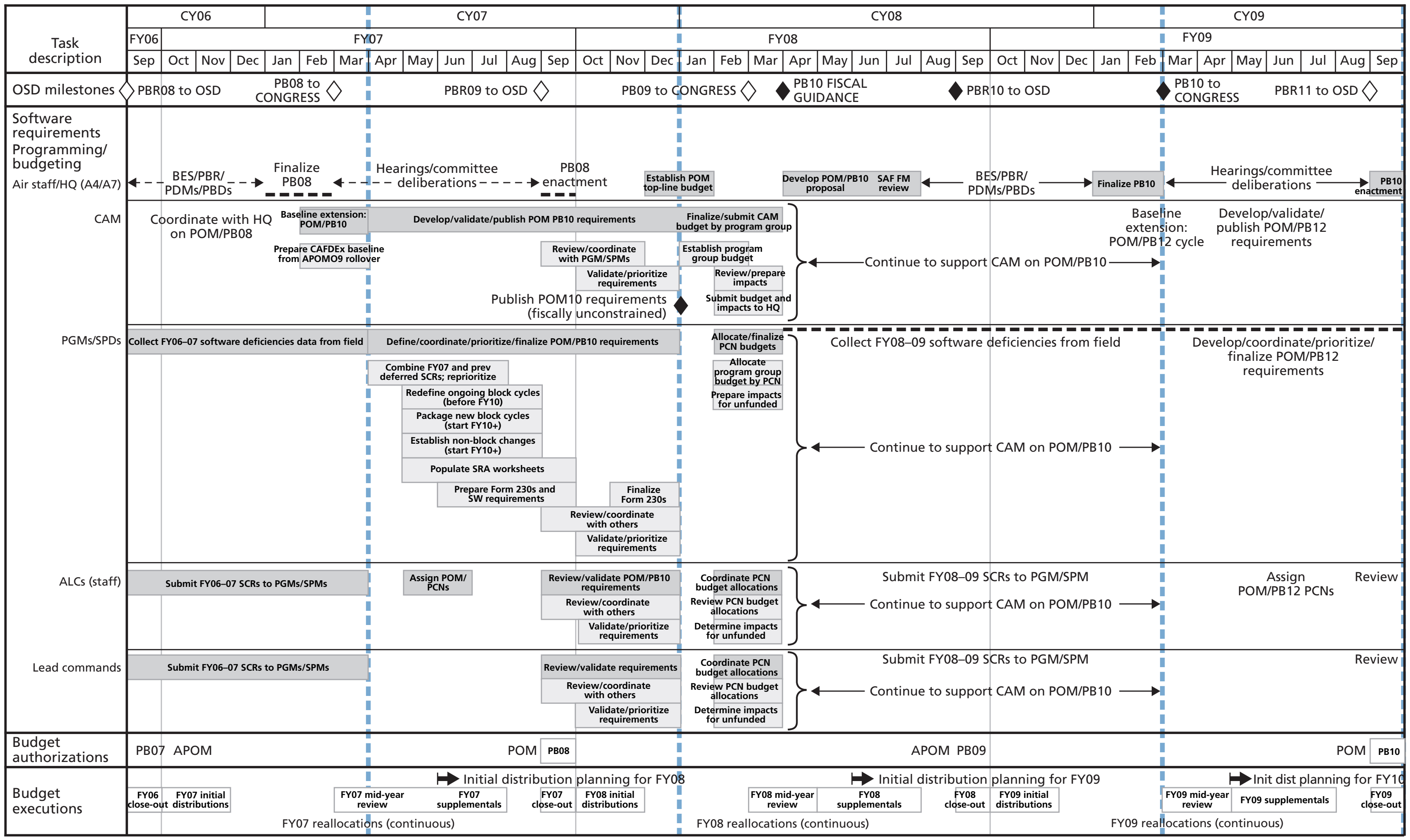
The primary stakeholders responsible for software requirements, programming, and budgeting are the Air Staff (AF/A4/7), the CAM office (AFMC/A4F), the PGMs and SPDs, the ALCs, and the lead MAJCOMs. The core activities of these stakeholders pertaining to the PB10 POM cycle are provided in Figure D.1 and are highlighted in shades or bold letters. As

1 For this POM/PB10 cycle, FY 2008 represents the actuals, FY 2009 is the execution year, and FY 2010 is the execution plan, financial plan, or budget year.

2 The April 2006 start date reflects the lead time required to collect the SCRs from the field before the POM cycle starts in September. For simplification, the process is truncated to start in September 2006 in Figure D.1.

3 The specific time periods given are notional for the POM/PB10 cycle and may vary from year to year.

Figure D.1
Overview of the Software Sustainment Requirements Process


CY06
CY07
CY08
CY09
FY06
FY07
FY08
FY09
Task description
Sep Oct Nov Dec Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar Apr May Jun Jul Aug Sep
OSD milestones
PBR08 to OSD
PB08 to CONGRESS
PBR09 to OSD
PB09 to CONGRESS
PB10 FISCAL GUIDANCE
PBR10 to OSD
PB10 to CONGRESS
PBR11 to OSD
Software requirements Programming/ budgeting
Air staff/HQ (A4/A7)
BES/PBR/ PDMs/PBDs
Finalize PB08
Hearings/committee deliberations
PB08 enactment
Establish POM top-line budget
Develop POM/PB10 proposal
SAF FM review
BES/PBR/ PDMs/PBDs
Finalize PB10
Hearings/committee deliberations
PB10 enactment
CAM
Coordinate with HQ on POM/PB08
Baseline extension: POM/PB10
Develop/validate/publish POM PB10 requirements
Finalize/submit CAM budget by program group
Prepare CAFDEx baseline from APOMO9 rollover
Review/coordinate with PGM/SPMs
Establish program group budget
Validate/prioritize requirements
Review/prepare impacts
Publish POM10 requirements (fiscally unconstrained)
Submit budget and impacts to HQ
Continue to support CAM on POM/PB10
Baseline extension: POM/PB12 cycle
Develop/validate/ publish POM/PB12 requirements
PGMs/SPDs
Collect FY06–07 software deficiencies data from field
Define/coordinate/prioritize/finalize POM/PB10 requirements
Allocate/finalize PCN budgets
Collect FY08–09 software deficiencies from field
Develop/coordinate/prioritize/ finalize POM/PB12 requirements
Combine FY07 and prev deferred SCRs; reprioritize
Allocate program group budget by PCN
Redefine ongoing block cycles (before FY10)
Prepare impacts for unfunded
Package new block cycles (start FY10+)
Establish non-block changes (start FY10+)
Continue to support CAM on POM/PB10
Populate SRA worksheets
Prepare Form 230s and SW requirements
Finalize Form 230s
Review/coordinate with others
Validate/prioritize requirements
ALCs (staff)
Submit FY06–07 SCRs to PGMs/SPMs
Assign POM/ PCNs
Review/validate POM/PB10 requirements
Coordinate PCN budget allocations
Submit FY08–09 SCRs to PGM/SPM
Assign POM/PB12 PCNs
Review
Review/coordinate with others
Review PCN budget allocations
Continue to support CAM on POM/PB10
Validate/prioritize requirements
Determine impacts for unfunded
Lead commands
Submit FY06–07 SCRs to PGMs/SPMs
Review/validate requirements
Coordinate PCN budget allocations
Submit FY08–09 SCRs to PGM/SPM
Review
Review/coordinate with others
Review PCN budget allocations
Continue to support CAM on POM/PB10
Validate/prioritize requirements
Determine impacts for unfunded
Budget authorizations
PB07 APOM
POM PB08
APOM PB09
POM PB10
Budget executions
FY06 close-out
FY07 initial distributions
FY07 mid-year review
FY07 supplementals
Initial distribution planning for FY08
FY07 close-out
FY08 initial distributions
FY08 mid-year review
FY08 supplementals
Initial distribution planning for FY09
FY08 close-out
FY09 initial distributions
FY09 mid-year review
FY09 supplementals
Init dist planning for FY10
FY09 close-out
FY07 reallocations (continuous)
FY08 reallocations (continuous)
FY09 reallocations (continuous)

NOTE: CY = calendar year. HQ = headquarters.

RAND *TR905-D.1*

points of reference, the time lines associated with the OSD milestones are shown at the top, and those associated with the congressional budget authorizations and subsequent (and continuous) budget executions are shown at the bottom. Where possible, the activities associated with previous and future POM cycles are also shown in the background as points of reference.[4]

The Air Staff

The Air Staff is the process owner, and its involvement in the process starts in midstream after the fiscally unconstrained requirements were published by the CAM office in December 2007. The Air Staff is responsible for (1) establishing the initial overall topline budget for DPEM and other depot maintenance programs (January 2008 time frame); (2) based on the PB10 fiscal guidance from OSD (March–April 2008 time frame) and in close coordination with the CAM office, developing the POM/PB10 budget proposal for review and approval by the Secretary of the Air Force for Financial Management (July 2008 time frame); (3) submitting the Secretary of the Air Force for Financial Management–approved PBR10 budget to OSD (September 2008 time frame); and (4) in coordination with the OSD and CAM office, refining and finalizing the POM/PB10 budget and overseeing the process through the PB10 final budget enactment and authorization by Congress.

The CAM Office

The CAM office (AFMC/A4F) is the process coordinator and central clearinghouse for the overall requirements process. Its involvement starts with the preparation of the baseline extension of the APOM09 data (i.e., extending FY 2013 programmed data from APOM/PB09 to FY 2014 and FY 2015) in CAFDEx for the POM/PB10 cycle in early spring 2007. This baseline database in CAFDEx is used by all primary stakeholders to coordinate and update all relevant programming information throughout the programming and budgeting process.

Starting with this baseline, the CAM office is responsible for (1) developing, validating, and finalizing the requirements in close coordination with the PGMs, SPDs, and other stakeholders as necessary, which culminates in the publication of the fiscally unconstrained requirements (December 2007 time frame); (2) using the topline DPEM budget established by the Air Staff as the initial fiscal constraint, allocating the budget for each program group (i.e., the topline budget for each PGM and SPD), coordinating with PGMs/SPDs regarding the PCN-level allocations, and submitting the final allocated requirements to the Air Staff, including the impact assessment of the unfunded deferrals (March 2008 time frame); and (3) supporting the Air Staff in the preparation of the POM/PB10 budget proposal consistent with the OSD fiscal guidance and in the finalization of the PB10 budget (April–December 2008).

One of the emphases of the CAM office (in a departure from the previous SRRP process) has been the shift from the business area focus to the weapon system focus. Among other effects, this shift resulted in a reallocation and, in some instances, higher aggregation of the programs previously established under the SRRP system. The requirements process under CAM is now carried out based on "program groups" under the oversight of PGMs and SPDs. Each "program group" broadly represents a weapon system and is an aggregation of all DPEM commodity categories that pertain to that weapon system. Other operationally relevant systems that are common across multiple weapon systems are also aggregated into pro-

[4] For simplification, APOM09 activities are not shown in some cases.

gram groups based on the nature of work performed. For example, weapon-specific software, such as operational flight program software for F-16 and B-1, are folded under F016 and B001 program groups, respectively. For electronic-warfare software, on the other hand, those that are weapon specific are folded under the respective weapon system program groups, and those that are common across multiple weapon systems are combined together into a separate system program.

Program Group Managers and System Program Directors

The PGMs/SPDs are responsible for each CAM program group and act as the primary focal points in ensuring that all requirements are validated and cost, schedule, and operational impacts are understood by the lead commands, the ALCs, and the CAM office. The software requirements process effectively starts with the submittal of the SCRs by the lead commands, ALCs, and other various field locations identified earlier. The collection and compilation of these SCRs are carried out by the PGMs/SPDs for each program groups for which they are responsible. For the POM/PB10 cycle, the freeze date for these SCRs was around March 2007, which means that the compilation of the SCRs, and hence the beginning of the POM/PB10 cycle for software sustainment, started in April 2006.

Once all relevant SCRs are compiled for the year, the PGMs/SPDs are responsible for coordinating closely with the lead commands and the ALCs in defining, prioritizing, and finalizing the fiscally unconstrained requirements so that the CAM office can publish the requirements for the POM/PB10 cycle at the end of the year (April to December 2007 time frame). It is important to note that the *fiscally unconstrained* requirements represent requirements that are fiscally unconstrained explicitly but capacity constrained (organic and contractor) implicitly.

More specifically, these steps involve (1) incorporating the unfunded deferred SCRs from the previous POM cycle (i.e., APOM/PB09) with those collected from April 2006 through March 2007, and rack-and-stacking and reprioritizing the combined SCRs; (2) redefining the block cycles that were defined in the previous POM cycles by adding new SCRs based on new prioritization (e.g., inserting new SCRs that must be implemented on a priority basis into the previously defined block cycles); (3) defining new block cycles to be programmed as part of POM/PB10 cycle based on the prioritization of the new combined SCRs; (4) defining non-block changes to be programmed together with the scheduled block cycles for POM/PB10 submission; (5) documenting and populating the CAFDEx database using the SRA and Form 230/231 mentioned earlier; and (6) reviewing, coordinating, validating, and finalizing the requirements data and documentation through close coordination with the lead commands and the ALCs, and supporting the CAM office in the publication of the fiscally unconstrained requirements.

The steps involving the block-cycle definition described above can have varied levels of fidelity depending on the lag time between the programming and the execution of the block cycle. A more detailed discussion is provided in the next section to address the issues related to the long lag time and its impact on requirements determination.

After the unconstrained requirements are published, the PGMs/SPDs receive topline allocations from the CAM office for each program group or weapon system. In close coordination with the lead commands, the ALCs, and the CAM office, the PGMs/SPDs then use these top lines to (1) allocate individual budgets to specific PCNs included within their program groups and system programs, (2) prepare impact statements for those requirements that are not funded, and (3) submit the budget proposal to the CAM office (January to March 2008 time frame). After this submittal, the PGMs/SPDs provide supporting documentation and continue to coordinate with the CAM office, as necessary, while the budget proposal is being fine-tuned and until the presidential budget is finalized. At the same time, the PGMs/SPDs also start a new POM cycle and start compiling the new SCRs from the field.

The Air Logistics Centers

The ALCs, also referred to as software support organizations, are the suppliers and administrators of the software maintenance services. Their primary roles and responsibilities are ensuring that sufficient organic and contractor software maintenance capacity is available to carry out the programmed and budgeted requirements. Throughout the entire programming and budgeting process, the ALCs work closely with the PGMs/SPDs and the lead commands to provide SCR inputs, assign PCNs, review and validate the fiscally unconstrained requirements, and review and validate the PCN budget allocations and impacts of the unfunded requirements.

The ALCs deliver the software maintenance service. Thus, they are often the primary source for the analysis of SCRs, for the identification and validation of software deficiencies or proposals for enhanced software performance, and for the analysis and determination of method of accomplishment (organic, contract, or interservice). They are responsible for assigning appropriate PCNs and estimating resources required to carry out the work included in the PCNs. They also confer with lead commands on the relative operational impacts of deferral decisions if sufficient capacity is not available.

The Lead Major Commands

The lead commands are the customers or users of the software maintenance service.[5] They establish required system operational capabilities and help to identify and prioritize software deficiencies and improvements for each weapon system based on assessed operational impacts. The ALCs can provide technical performance impacts of software changes, while the lead commands have a better understanding of the true operational impacts of these changes. The requirements focal points within the lead commands for each weapon system, also referred to as the *functionals*, consolidate and prioritize weapon system software maintenance requirements for all users under the lead command. Throughout the entire programming and budgeting process, the requirements focal points work closely with the PGMs/SPDs and the ALCs to provide SCR inputs, review and validate the fiscally unconstrained requirements, review and validate the PCN budget allocations, and provide input in determining the impacts of the unfunded requirements.

Other Time Lines: Office of the Secretary of Defense Milestones, Budget Authorizations, Execution Time Lines

In Figure D.1, the time lines for the OSD milestones, the congressional budget authorizations, and the actual budget executions are also provided as reference points. Of particular interest are the budget executions, which involve constant reallocations of the initial distributions throughout the execution year as shown. The actual executions can vary significantly from the congressional budget authorization, and this is particularly true for software sustainment. The

[5] For the DPEM software commodity category, the lead commands include ACC, AMC, AFSOC, Air Force Space Command, and AFMC and do not include the Air National Guard or the Reserve components.

true deferrals can be better determined if the fiscally unconstrained requirements can be compared with the actual executions. There is currently no formal mechanism that allows tying the execution data with the initial requirements data. Appendix C provides additional discussion of software deferrals.

Execution Phase: F-16 Example

As mentioned in Chapter Six, software requirements for a given block cycle are not finalized until the beginning of the execution of that block cycle.

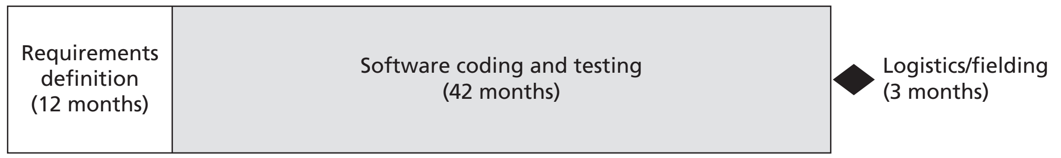
The F-16 currently has by far the most rigorous and streamlined approach to the block cycle (or SCU) execution.[6] The execution of a new F-16 SCU starts every two years, and each SCU execution spans a 57-month period, which is composed of the following three sets of activities: (1) pre-SCU activities associated with detailed requirements definition (typically 12 months), (2) SCU-specific activities that include software coding and testing (typically 42 months), and (3) post-SCU activities for the preparation of the software fielding and associated logistics (typically three months). The basic execution building blocks are shown in Figure D.2 (the same building blocks were presented in the bottom half of Figures 6.1 and 6.2 in Chapter Six).

These execution cycles are guided by a general time line that starts in April every two years. For example, the planned execution time line for SCU10 is (1) requirements definition (April 2011–March 2012), (2) software coding and testing (April 2012–September 2015), and (3) logistics and fielding (September 2015–December 2015). These execution time lines are often out of phase with the POM cycles, making the requirements process more complex and difficult.

The pre-SCU requirements definition activities, occurring immediately prior to the actual execution of each SCU, are intended to further refine the requirements developed in the programming and budgeting phase. The finalization of the requirements for a given SCU ultimately occurs at this stage, which extends over a 12-month period. This process involves four major reviews, each four months apart, with different levels of participation by the stakeholders from the operational side. Figure D.3 shows this requirements definition process for SCU10.

As shown in Figure D.3, the first review is referred to as the Avionics Systems Requirements Review Conference (ASRRC) and involves (1) reviewing all the new SCRs that were collected over a two-year period since SCU9 (note that these new SCRs might be different

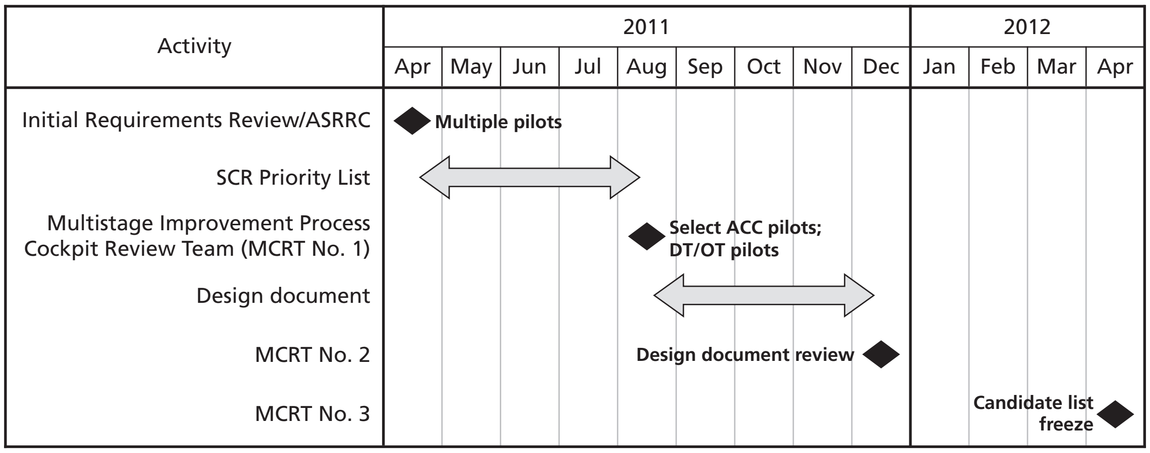
Figure D.2
F-16 Software Change Unit Execution Cycle (57 months)


Requirements definition (12 months)
Software coding and testing (42 months)
Logistics/fielding (3 months)

RAND *TR905-D.2*

[6] This discussion refers to the block 30 F-16 software maintenance done organically by the Air Force.

Figure D.3
Pre–Software Change Unit Requirements Definition Activities (F-16 SCU10)



NOTE: MCRT = Multistage Improvement Process Cockpit Review Team. DT/OT = development test/operational test.
SOURCE: Arango, 2008.
RAND *TR905-D.3*

from those considered in the programming and budgeting phase for SCU10) combined with all the deferred SCRs from SCU9 that made the candidate list but were not funded previously and (2) establishing a priority list of the combined SCRs based on the inputs from the operators.[7] In addition to the PGM/SPD representative, participating in the review are a multitude of pilots from the field who understand and have a stake in the operational outcomes of all the change requests identified in the combined SCRs.

The remaining three reviews are performed by the MCRT. The MCRT is composed of selected pilots from the commands (total force), pilots from the DT/OT community, and other field pilots and weapon officers who are particularly knowledgeable about the group of change requests under consideration.

The first of the three MCRT reviews is conducted four months after the ASRRC review. It involves the MCRT participants reviewing the SCRs, further refining the prioritization based on the weapon system operational impacts, and developing sufficient details and inputs to allow the development of a detailed software design document in the following months. The design document is intended to provide specific design activities and issues associated with the implementation of the SCRs. The second MCRT involves the review of the design document developed in the previous months and further refinement of the priority list. The final MCRT review is intended to identify and define the final list of SCRs to be included in SCU10.[8]

As demonstrated in the above discussions, the requirements process associated with block cycles is complex and illusive. The long lag time and resulting changes in requirements contribute to the difficulty in directly linking the requirements with the operational capabilities

[7] For F-16 software sustainment, the SCRs that are deferred more than one POM cycle are subsequently dropped permanently from the candidate list for further consideration.

[8] Although not covered in our study due to time limitations, additional insights could be gained from the weapon system reviews, such as the MCRT reviews, in determining whether meaningful performance metrics can be identified for use in capabilities-based software sustainment programming.

in the programming phase and in facilitating the important trade and deferral decisions for software sustainment.

References

AFMC—*See* Air Force Materiel Command.

Air Force Materiel Command, "The Software Requirements Review Process (SRRP) and Guide for National Security Systems (NSS) Software Funding Policy/Procedures," *Financial Management Handbook*, April 2004.

———, "The Depot Purchased Equipment Maintenance (DPEM) Process," *Financial Management Handbook*, May 2005.

Arango, Tim, "I Got the News Instantaneously, Oh Boy," *New York Times*, September 14, 2008. As of September 22, 2010:
http://www.nytimes.com/2008/09/14/weekinreview/14arango.html

Commander Air Force Materiel Command, Maintenance: Analytical Condition Inspection (ACI) Programs, Air Force Materiel Command Instruction 21-102, January 29, 2002.

———, Acquisition: Software Requirements Review Process, Air Force Materiel Command Instruction 63-401, April 16, 2003a, rescinded.

———, Acquisition: Software Requirements Review Process, Air Force Materiel Command Policy Directive 63-4, April 16, 2003b, rescinded.

———, Maintenance: AMARC Storage and Withdrawal of Aircraft and Equipment, Air Force Materiel Command Instruction 21-123, December 15, 2006.

Defense Acquisition University, *Performance Based Logistics: A Program Manager's Product Support Guide*, Fort Belvoir, Va.: Defense Acquisition University Press, March 2005. As of September 22, 2010:
http://purl.access.gpo.gov/GPO/LPS60465

DoD—*See* U.S. Department of Defense.

DoDD 5000.01—*See* U.S. Department of Defense, 2003.

DoDI 8260.01—*See* U.S. Department of Defense, 2007.

Maynard, Micheline, "A Mistaken News Report Hurts United," *New York Times*, September 9, 2008. As of September 22, 2010:
http://www.nytimes.com/2008/09/09/business/09air.html

Perrow, Charles, *Normal Accidents: Living with High-Risk Technologies*, Princeton, N.J.: Princeton University Press, 1999.

Secretary of the Air Force, Operations Support: Aerospace Vehicle Programming, Assignment, Distribution, Accounting, and Termination, Air Force Instruction 16-402, August 1, 1997.

———, Technical Manual: Depot Maintenance of Aerospace Vehicles and Training Equipment, Technical Order 00-25-4, change 2, June 1, 2004.

———, Maintenance: Equipment Inventory, Status and Utilization Reporting, Air Force Instruction 21-103, December 14, 2005.

———, Technical Manual: Aircraft Engine Operating Limits and Factors, Technical Order 2-1-18, change 4, June 15, 2006.

———, Maintenance: Selective Management of Selected Gas Turbine Engines, Air Force Instruction 21-104, December 11, 2007.

———, Technical Manual: Maintenance Assistance, Technical Order 00-25-127, January 15, 2008a.

———, Technical Manual: Comprehensive Engine Management System Engine Configuration, Status and TCTO Reporting Procedures, Technical Order 00-25-254-1, change 11, July 1, 2008b.

Snyder, Don, and Patrick Mills, *Supporting Air and Space Expeditionary Forces: A Methodology for Determining Air Force Deployment Requirements*, Santa Monica, Calif.: RAND Corporation, MG-176-AF, 2004. As of September 22, 2010:
http://www.rand.org/pubs/monographs/MG176/

———, "Air Force Deployments: Estimating the Requirement," *Air Force Journal of Logistics*, Vol. 30, No. 2, Summer 2006, pp. 4–9. As of September 22, 2010:
http://www.aflma.hq.af.mil/shared/media/document/AFD-100111-122.pdf

Snyder, Don, Patrick Mills, Adam C. Resnick, and Brent D. Fulton, *Assessing Capabilities and Risks in Air Force Programming: Framework, Metrics, and Methods*, Santa Monica, Calif.: RAND Corporation, MG-815-AF, 2009. As of September 22, 2010:
http://www.rand.org/pubs/monographs/MG815/

Stevens, W. P., G. J. Myers, and L. L. Constantine, "Structured Design," *IBM Systems Journal*, Vol. 13, No. 2, 1974, pp. 115–139.

Talaber, Adam, *Long-Term Implications of Current Defense Plans: Summary Update for Fiscal Year 2008*, Washington, D.C.: Congressional Budget Office, December 2007. As of September 22, 2010:
http://purl.access.gpo.gov/GPO/LPS93522

TO 00-25-4—*See* Secretary of the Air Force, 2004.

TO 00-25-127—*See* Secretary of the Air Force, 2008a.

TO 00-25-254-1—*See* Secretary of the Air Force, 2008b.

TO 2-1-18—*See* Secretary of the Air Force, 2006.

U.S. Department of Defense, The Defense Acquisition System, Directive 5000.01, May 12, 2003. As of September 22, 2010:
http://www.dtic.mil/whs/directives/corres/pdf/500001p.pdf

———, *Quadrennial Defense Review Report*, Washington, D.C., February 6, 2006. As of September 22, 2010:
http://www.defenselink.mil/qdr/report/Report20060203.pdf

———, Support for Strategic Analysis, Instruction 8260.01, January 11, 2007.