Tutorial
Software Supportability
Methods and Techniques to Improve Life Cycle Costs

Part of a Three-Day Course

on

Software Supportability

by
Dr. David E. Peercy
Sandia National Laboratories
Chair, Society of Automotive Engineers
Reliability, Maintainability, Supportability Logistics G-11-SW Software Committee

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

Sandia National Laboratories

Software Supportability-Methods and Techniques to Improve Life Cycle Cost

Presentation by Dr. David E Peercy

About the Presentation

This tutorial will provide the practicing engineer with an appreciation of software, the software engineering process, and detailed insight into software supportability concerns. Software is a major component of all new systems and many of the existing systems. The role of software in the life cycle of most modern systems is significant, especially in regards to sustaining the system. This tutorial will describe an approach for addressing the support of software as an integral part of the system design, engineering, and logistics support processes. The approach is based on the software supportability and reliability guidelines recently published by SAE.

Participants will be given a basic understanding of how to plan for software support and develop a software support concept. Recently developed international standards and guidelines in software supportability and software reliability will be discussed. Other possible topics of interest include software life cycle process, software maintainability characteristics, software maintenance, using software reliability to predict corrective change activity, support scenarios, supportability factors, supportability analysis tasks necessary to evolve a software support capability, supportability analysis documentation, and Off-The-Shelf software support issues.

Presentation Outline (subject to some change)

02:45 - 02:50pm	Session 00: Introduction
02:50 - 03:15pm	Session 01: Life Cycle Activities in a Systems Context
03:15 - 04:00pm	Session 02: Developing a Support Concept, Plan and Case
04:00 - 04:45pm	Session 03: Technologies to Improve Life Cycle Cost

Who Should Attend

This tutorial is targeted to engineers and management who develop or support products that have software components and require a longer-life . Applicable areas of responsibility might include: acquisition management, project management, logistics, systems/software engineering, quality engineering, or specialty engineering areas of Reliability, Maintainability, Logistics, and Support. It is expected that tutorial participants are somewhat familiar with the life cycle development and support of systems and/or software. Commercial, aerospace, ground vehicle, automotive, or military systems experiences are all applicable. There are no special educational requirements for participants above the indicated general knowledge and experience.

About the Presenter

Dr. Peercy is the chairman of the Society of Automotive Engineers (SAE) G-11SW Software RMSL Committee developing International standards and guidelines for software supportability and software reliability. Dr. Peercy has numerous publications in software engineering areas such as software maintenance, software supportability, software reliability, and software process improvement.

Dr. Peercy has been a reviewer for several standards including the Ada language standard, the National Computer Security Center Trusted Network Evaluation Criteria and various IEEE standards. He has taught short courses/tutorials on a variety of software subjects including software supportability, software reliability, Ada Programming Language, software logistics, and software computer security. Dr. Peercy developed the methodology and software support tools for the Risk Assessment Methodology for Software Supportability developed for the USAF.

Dr Peercy is currently a Distinguished Member of the Technical Staff with Sandia National Laboratories (SNL) in Albuquerque, New Mexico. His duties with SNL include qualification of nuclear weapon and weapon-related software, development of software and system quality engineering practices, and assistance with specification of corporate policies on software that ensure adherence to National Nuclear Safety Administration (NNSA)/ Department of Energy (DOE) requirements.

Dr. Peercy received his Ph.D. in Mathematics from New Mexico State University in 1971 and his Bachelors in Applied Mathematics from the University of Colorado in 1966. Dr. Peercy has been an ASQ Certified Software Quality Engineer since 1996.

Sandia National Laboratories

Session 00
Introduction

Sandia National Laboratories

Introductory Topics

- Instructor
  - Background and experience
- Tutorial Participants
  - What do you expect to learn from this tutorial? What would make this tutorial a success for you?
- Course
  - Outline & Schedule
  - Goals & Objectives

Instructor

- Dr. David E. Peercy (David/Dave)
  - Ph.D. Mathematics from New Mexico State University, 1971
  - 28 years experience and publications in software engineering, methodology, supportability, reliability, quality; over a period of 12 years developed Risk Assessment Methodology for Software Supportability for the USA Air Force Operational Test & Evaluation Center
  - ASQ Certified Software Quality Engineer
  - Chair of SAE RMSL G-11SW Software Committee
  - Member of SOLE, ACM, IEEE CS, ASQ
  - Distinguished Member of the Technical Staff at Sandia National Laboratories
- Contact Information

Dr. David E. Peercy
Sandia National Laboratories
P.O. Box 5800, MS-0638
Albuquerque, NM 87185-0638
USA
Voice: 505-844-7965
Fax: 505-844-3920
E-mail: depeerc@sandia.gov

G-11SW Software Committee Homepage
http://www.sae.org/technicalcommittees/g11soft.htm

Sandia National Laboratories

Participants

- Your Backgrounds and Interests?
  - Topics
  - Logistics/Supportability Roles
  - Tutorial Expectations
- Tutorial Evaluation
  - Please complete at the end of the tutorial

Outline & Schedule

- 02:45 - 02:50pm
  - Session 00: Introduction
- 02:50 - 03:15pm
  - Session 01: Software Supportability: Life Cycle Activities in a Systems Context
- 03:15 - 04:00pm
  - Session 02: Software Supportability: Developing a Support Concept, Plan and Case
- 04:00 - 04:45pm
  - Session 03: Software Supportability: Technologies to Improve Life Cycle Cost
    - Selection of one/two topics based on interest from among:
      - Risk Assessment Methodology for Software Supportability (RAMSS)
      - Software Block Release Profile
      - Software Support Cost Estimation
      - System/Software Reliability
      - Software Logistics Identification
      - Off-The-Shelf Technologies and Support Issues

Goals and Objectives
Participant Capabilities

Given a system for which you must determine an approach to software support, you should be able to:

- Identify important software supportability issues
  - related to the software engineering life cycle, system engineering life cycle, and the system logistics support activities
- Plan for & provide evidence of software supportability
  - Understand methods for acquiring, engineering, and sustaining a software support capability
  - Determine performance-based software supportability requirements
- Derive elements of a software support concept
  - Characteristics of product, process, and environment supportability
  - Profiles for who, where, and what of support activity
  - Field, intermediate infrastructure, and centralized support functions

Some Questions To Answer

- Why should software support/maintenance be emphasized?
- How much does software support cost?
- What are software support performance measures?
- In what system engineering phases is it important to consider software support issues?
- Is software maintenance just another iteration of software development?
- What are software support activities?
- Who conducts software support activities?
- Where are software support activities conducted?
- When are changes to software released?

Sandia National Laboratories

Session 01

Life Cycle Activities in a Systems Context

Sandia National Laboratories

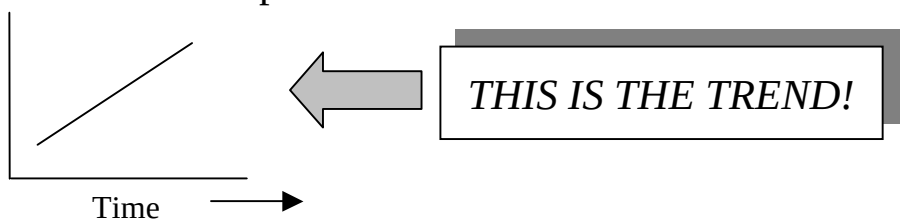
Session Objectives

- Motivation
  - Provide motivation for the importance of a comprehensive life cycle engineering approach to software support
- Terminology
  - Introduce terminology that will show the relationships of software engineering, software maintenance, software support, systems engineering, and systems integrated logistics support
- Standards
  - Establish initial understanding of some of the available standards and guidelines that are useful in this area

Motivation: Support Cost

- Support Cost
  - 1992
    - Industry - $30B supporting software; 60 % -80% of SW budget
    - US DoD - $20B supporting software; 70% of SW budget
  - 1995
    - Industry - $45B supporting software; 80%-90% of SW budget
    - US DoD - $25B supporting software; 70% of SW budget
  - 1999 (conservative estimates)
    - Industry - $60B supporting software; 80%-90% of SW budget
    - US DoD - $28B supporting software; 70% of SW budget
  - US DoD
    - 80% Air Force systems in development are software-intensive
    - 50% of software systems in development are not fielded

- % Software Functions
- Size of SW in Applications
- Cost of SW Support
- % SW to HW cost



Motivation: Support Failures

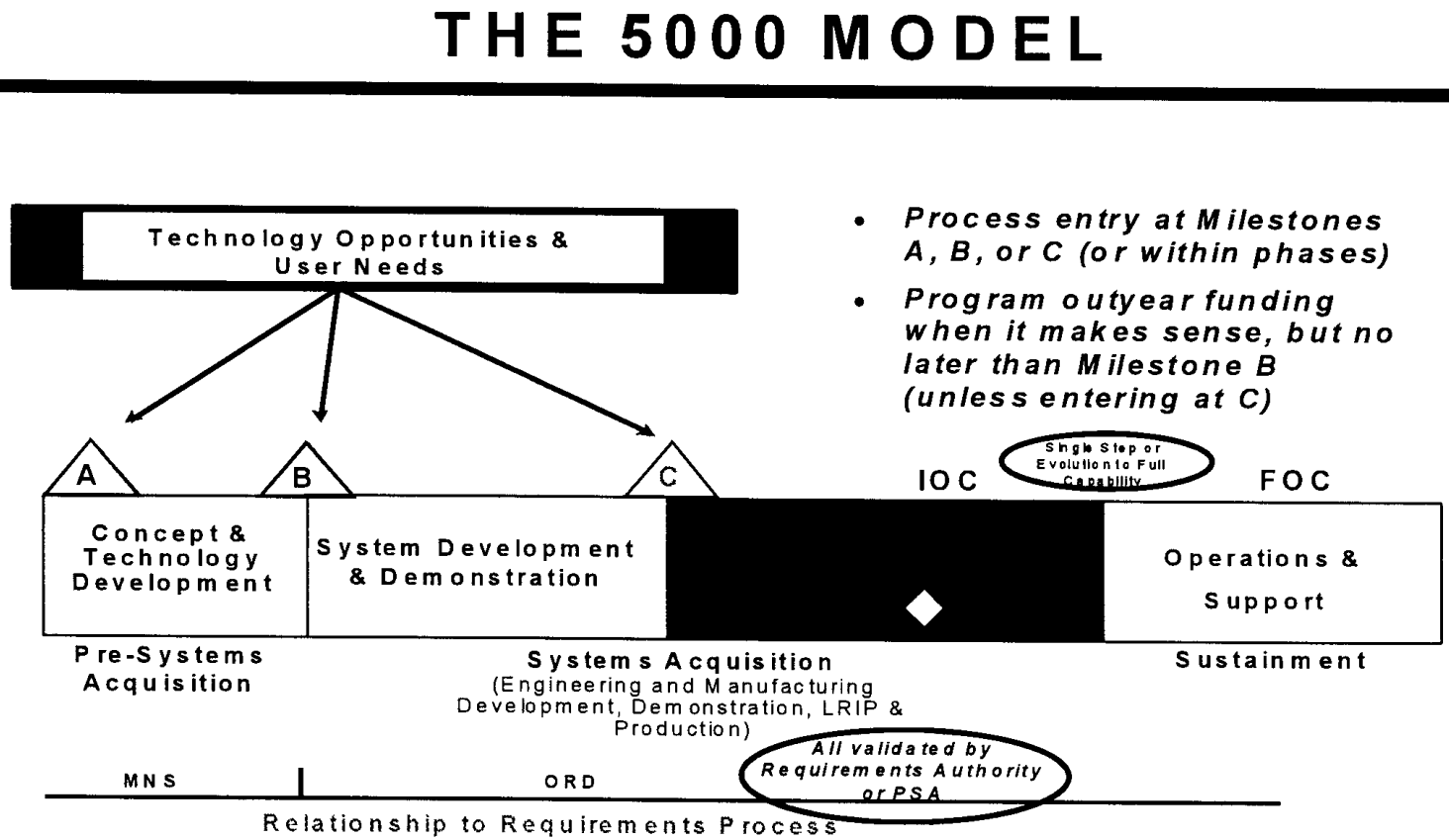
- Catastrophes Due To Modification Inadequacies
  - THERAC 25 - radiation kills several patients due to SW defect
  - ARIANE 5 - $7B rocket is destroyed due to inadequacy of support process to prevent insertion of a defect from reuse of previous code
- Technology Future Shock - how far is the future?
  - Y2K and Y2038
- Where's the Vision

  - Software maintenance and support have been issues since late 1970s
  - Development emphasis
    - do it right the first time
    - just need the right tools
  - Maintenance emphasis
    - Make process same as development
    - Re-engineering, reverse engineering, reuse methodologies

Some Basic Terms

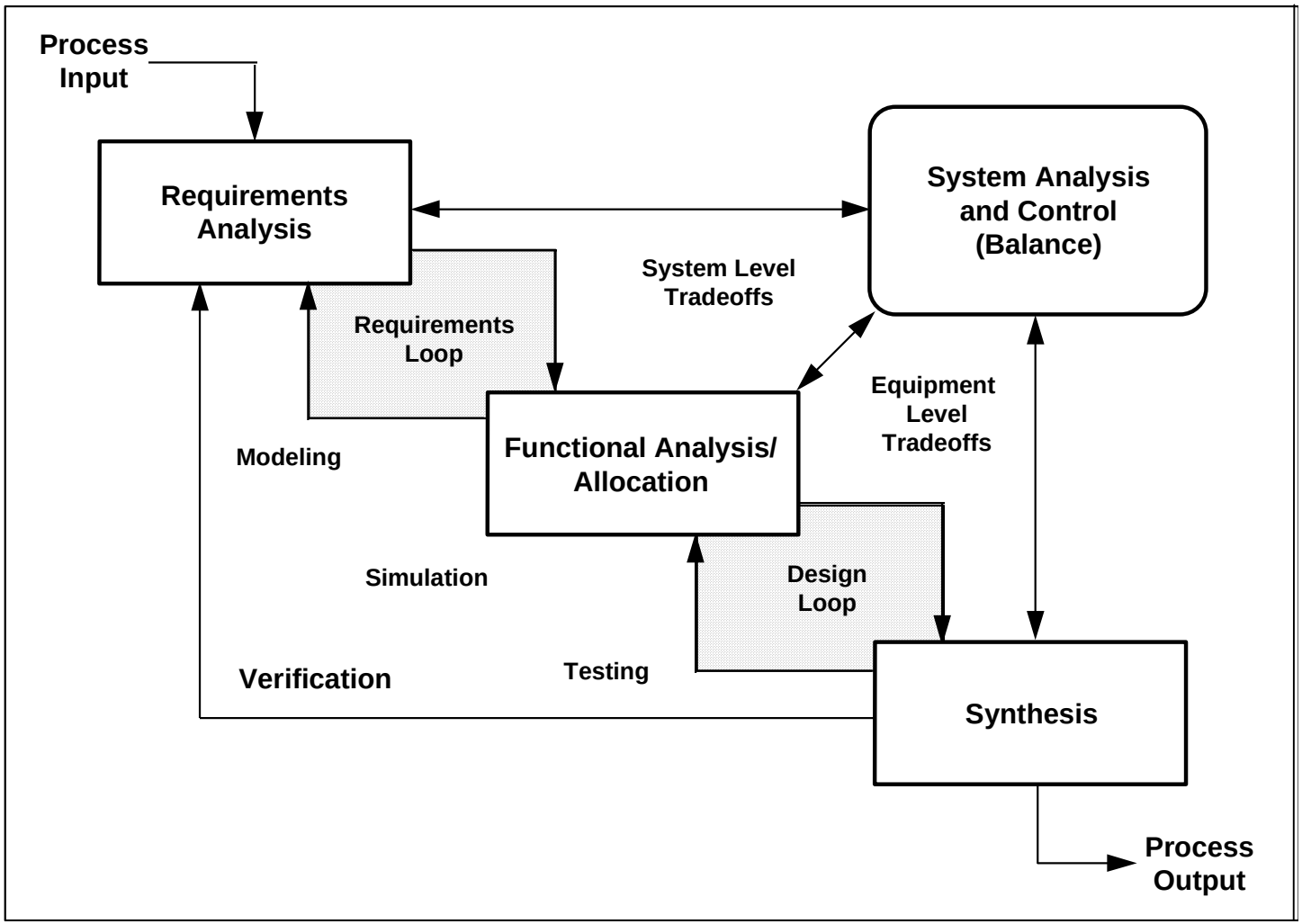
- Systems Infrastructure
  - Systems Engineering
  - Integrated Logistic Support
  - ILS Elements
  - Logistic Support Analysis
- Software Specific
  - Software Engineering
  - Software Support & Supportability
  - Software Support: Modification, Operational, Logistics Management
  - Software Maintainability, Maintenance

Systems Acquisition Phases



The framework is divided into three activities (e.g., Systems Acquisition). Activities are divided into phases (e.g., System Development and Demonstration). Phases are divided into work efforts (e.g., System Integration).

Systems Engineering Process



Integrated Logistics Support (ILS)

- A disciplined, unified, and iterative approach to the management and technical activities necessary to:
  - Integrate support considerations into system and equipment design
  - Develop support requirements that are related consistently to readiness objectives, to design, and to each other
  - Acquire the required support
  - Provide the required support during the operational phase at minimum cost

Logistic Support Analysis (LSA)

- The selective application of scientific and engineering efforts undertaken during the acquisition [and sustainment] process, to assist in:
  - Causing support considerations to influence design
  - Defining support requirements that are related optimally to design and to each other
  - Acquiring the required support
  - Providing the required support during the operational phase at minimum cost

Software LSA and Logistics

- Software Supportability Analysis / Software LSA
  - The selective application of the LSA tasks to the software components within the context of the system and its application domain.
- Software Logistics
  - The selective application of the integrated logistics support process to the system software element.

"selective" implies the use of:
- *tailored Logistics Support Analysis analysis tasks to assist the decision process*
- *analysis of dependencies upon application type and implementation languages*
- *Off-The-Shelf and developed software; and implementation languages, and*
- *application system and program life cycle phases*

Software Support

- Software
  - Programs, procedures, rules, data and any associated documentation pertaining to the operation of a computer system
- Software Support
  - The set of activities necessary to ensure that an operational software system or component fulfills its original requirements and any subsequent modifications to those requirements. Activities include all processes, resources and infrastructure required in order to provide support throughout the software's operational life.

Software Supportability

Software Supportability

- A measure of the adequacy of software personnel, product, resources, and procedures to facilitate the following support activities:
  - operational use
    - pre-mission; operation; post-mission
  - logistics management
    - user support; release replication and distribution; COTS installation; field configuration control
  - modification
    - problem reporting; maintenance activities; configuration management; logistic release
- a set of attributes of software design, the associated development tools and methods, and the support environment infrastructure that enable the software support activities to be accomplished.

Software Maintenance/Modification

- Software Maintenance
  - Modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a changed environment.
- Software Modification Support
  - The software support activities of change analysis, implementation, test and release of software products. Changes may be termed corrective, perfective and adaptive, and may also embrace modifications that are designed to prevent foreseeable future software operating problems.

Software Support Issues

- General Support Issues
  - What support activities will be conducted & in what sequence
  - Where is support going to be conducted
  - Who will be doing the support activities
  - When and how quickly/often will the support activities be performed
  - How will the support activities be accomplished
  - ===> oh, and is there any cost involved?
- Application Specific Software Support Issues
  - Aerospace
  - Automotive
  - Commercial
  - Military

Standards of Primary Interest

- *System/software Engineering Guidance*
  - *ISO/IEC 12207, Software life cycle processes*
  - *IEEE/EIA 12207.0, Software life cycle processes*
  - *IEEE/EIA 12207.1, SLCP - Life cycle data*
  - *IEEE/EIA 12207.2, SLCP - Implementation considerations*
- *System/software ILS and LSA Guidance*
  - *Defence Standard 00-60, ILS, Part 3, Software LSA*
  - *Mil-Hdbk-502, Acquisition Logistics*
  - *MIL-PRF-49506, Performance Specification Logistics Management Information*

Standards of Primary Interest

- *Software Supportability and Reliability Guidance*
  - *SAE G-11SW Standards*
    - *JA1002, Software Reliability Program Standard*
    - *JA1003, Software Reliability Implementation Guide*
    - *JA1004, Software Supportability Program Standard*
    - *JA1005, Software Supportability Implementation Guide*
    - *JA1006, Software Support Concept*
  - *IEEE Standards*
    - *IEEE 1219-1998, Software Maintenance*
  - Defence Standard 00-42 (PART 2)/Issue 1, Reliability and Maintainability Assurance Guides
- *Software Process Improvement*
  - *SW-CMM, Software Capability Maturity Model*
  - *ISO 15504 - Software Process Improvement Capability Determination (SPICE)*

Systems/SW Life Cycle Process Tailored ISO/IEEE 12207

ISO/IEC 12207

The International Standard ISO/IEC 12207 establishes a common framework for the life cycle of software in terms of the processes that can be employed to
(1) acquire, supply, develop, operate, and maintain software;
(2) manage, control, and improve the processes; and
(3) provide the basis for world trade in software.

ISO/IEC 12207 places requirements upon the characteristics of a designated set of life cycle processes, but does not specify the detailed implementation of those processes.

Primary Life Cycle Processes

Primary Life Cycle Processes define the software product processes from conceptualization through retirement. A primary process is used by a party that initiates the need for the software product or service and by the party that performs the software development, operation or maintenance. The primary processes are:

- **Acquisition process**
  - Defines the activities of the acquirer, the party that initiates the need for and acquires the system, software product or service.
- **Supply process**
  - Defines the activities of the party that provides the system, software product or service.
- **Development process**
  - Defines the activities of the developer, the party that defines and produces the software product.
- **Operation process**
  - Defines the activities of the operator, the party that provides the service of operating a computer system in its actual environment for its users, or the user of a system containing software.
- **Maintenance process**
  - Defines the activities of the maintainer, the party that provides the service of maintaining the software product; that is, managing modifications to the software product to keep it current and in operational fitness. This process includes the migration and retirement of the software product.

Maintenance Process

The Maintenance Process contains the activities and tasks of the maintainer. *This process is activated when the software product undergoes modifications to code and associated documentation due to a problem or the need for improvement or adaptation*. The objective is to modify existing software product while preserving its integrity. This process includes the migration and retirement of the software product. The process ends with the retirement of the software product. The activities provided in this clause are specific to the Maintenance Process; *however, the process may utilize other processes in this International Standard*. If the Development Process is utilized, the term developer there is interpreted as maintainer. ... *When the maintainer is the supplier of the maintenance service, the maintainer performs the Supply Process.* This process consists of the following activities:

- 1) Process implementation
- 2) Problem and modification analysis
- 3) Modification implementation
- 4) Maintenance review/acceptance
- 5) Migration
- 6) Software retirement

Supply Process

The Supply Process contains the activities and tasks of the supplier. The process may be initiated either by a decision to prepare a proposal to answer an acquirer's request for proposal or by signing and entering into a contract with the acquirer to provide the system, software product or software service. The process continues with the determination of procedures and resources needed to manage and assure the project, including development of project plans and execution of the plans through delivery of the system, software product or software service to the acquirer. ... This process consists of the following activities:

- 1) Initiation
- 2) Preparation of response
- 3) Contract
- 4) Planning
- 5) Execution and control
- 6) Review and evaluation
- 7) Delivery and completion

Note: The tutorial terminology for Software Supportability is slightly broader than the ISO terminology for Maintenance, e.g., the aspects of Supply Process for software are included.

Performance Specification DODD 5000, 2001

Notice 1, January 4, 2001: DODD 5000.1: Clause 4.2.4 Performance-Based Acquisition.
In order to maximize competition, innovation, and interoperability, and to enable greater flexibility in capitalizing on commercial technologies to reduce costs, *performance-based strategies for the acquisition of products and services shall be considered and used* ***whenever practical***. For products, this includes all new procurements and major modifications and upgrades, as well as the reprocurement of systems, subsystems, and spares that are procured beyond the initial production contract award. When using performance-based strategies, contractual requirements shall be stated in performance terms, limiting the use of military specifications and standards to Government-unique requirements only. Configuration management decisions shall be based on factors that best support implementation of performance-based strategies throughout the product life cycle.

June 2001: DODD 5000.2-R: Clause C2.8 Support Strategy
As part of the acquisition strategy, the PM shall develop and document a *support strategy for life-cycle sustainment* and continuous improvement of product affordability, reliability, and supportability, while sustaining readiness. This effort shall ensure that *system support and life-cycle affordability considerations are addressed and documented as an integral part of the program's overall acquisition strategy*. The support strategy shall define the supportability planning, analyses, and trade-offs conducted to determine the optimum support concept for a materiel system and strategies for continuous affordability improvement throughout the product life cycle. The support strategy shall continue to evolve toward greater detail, so that by Milestone C, it contains sufficient detail to define how the program will address the support and fielding requirements that meet readiness and performance objectives, lower TOC, reduce risks and avoid harm to the environment and human health. (TOC - Total Ownership Cost)

Bottom Line: *For the US Military, since 1996, direction has weakened toward "performance-based only". Direction has strengthened for a "support strategy" including software within that strategy (including support concept terminology).*

Software Support
Department of Army, AR 700-127

November 1999: Army Regulation 700-127: prescribes Department of the Army (DA) policies and assigns responsibilities for the management of Acquisition Logistics

Clause 1.4 Application of the ILS process
The ILS process applies to all materiel systems procured under the provisions of AR 70-1 including associated software, procured or modified for use by Army units.

Clause 3.14 Software.
Software associated with a materiel system is an integral component of that system, and software support will be addressed through the ILS program. System modernization involves software upgrades or changes and post deployment software support costs can be significant over the system's life. The effectiveness of system software has a direct impact on system readiness. Planning related to software management and support will be detailed in the Computer Resources Life Cycle Management Plan (CRLCMP) and summarized in the Computer Resources section of the supportability strategy. Interrelationships with the other ILS elements will be addressed through the SA process.

Bottom Line: *For the US Military Army, software support/supportability will be addressed as an integral part of Integrated Logistics Support.*

Software Support Life Cycle Activities

Concept

- Customer and supplier personnel are aware of software support impact
- Initial proposal for system requires software support analysis and deliverables

Development

- Customer develops basic requirements and top level support plan (e.g., Software Supportability Plan or equivalent)
- Software supportability is an integrated part of supplier Integrated Logistics Support strategy; hardware and software tradeoffs are conducted
- Software support concept is developed for all software items tailored appropriately by level of significance
- Application and Off-The-Shelf software items are all identified
- Software is designed to be supportable
- Estimates for software support costs, expected level of support activities, and risks are well-defined with documented rationale
- Detailed software support plan (e.g., Software Supportability Plan or equivalent) is developed for significant software items -- software support concept

Software Support Life Cycle Activities

Transition

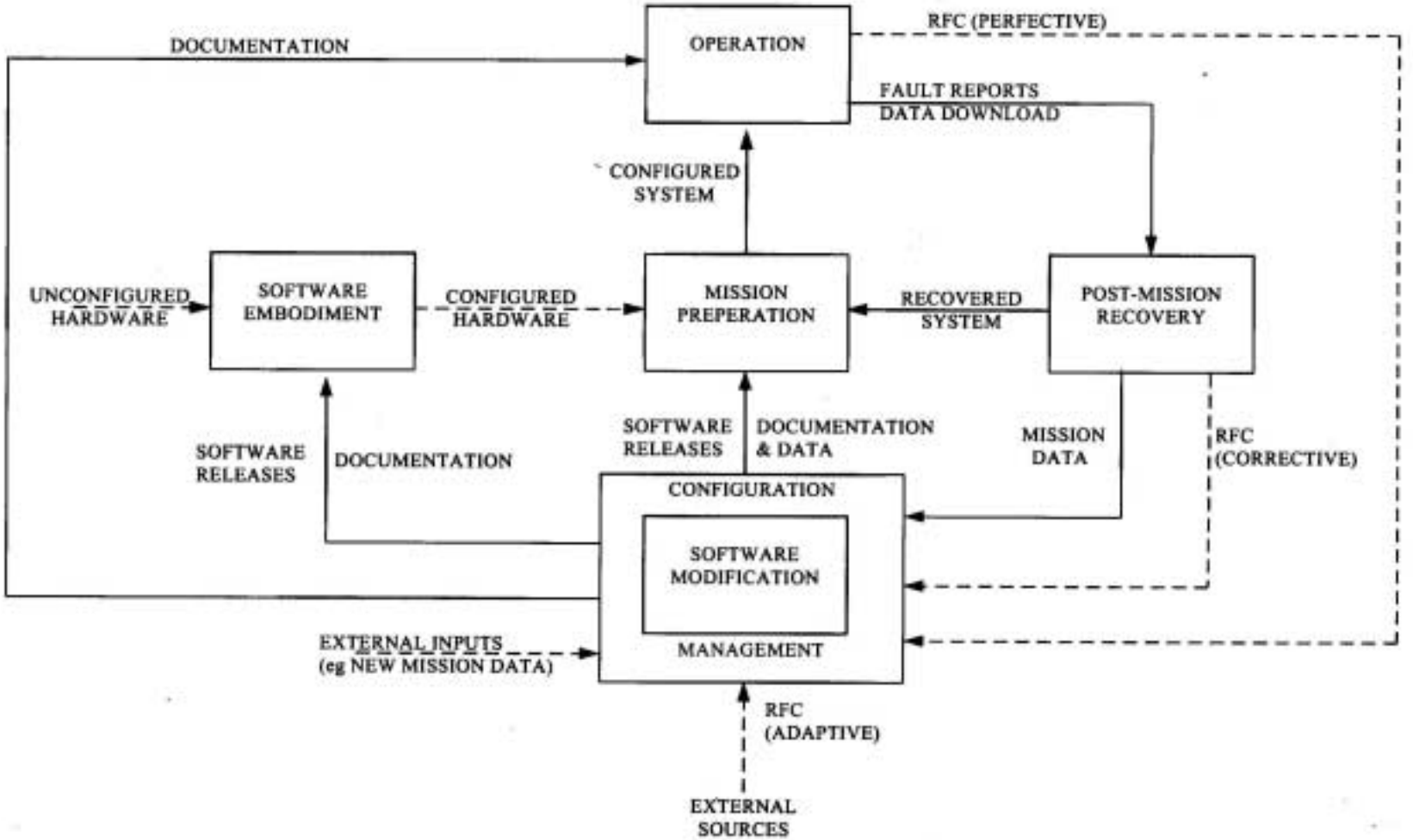
- System transition is iterated with appropriate software support resources and tasks put in place over time
- Support Center Capability (e.g., extension of developer duties) is in place to facilitate transition
- Support measurement system is in place to determine how well support processes are being accomplished
- Transition from Developer to Support Activity is well-defined

Operation and Support

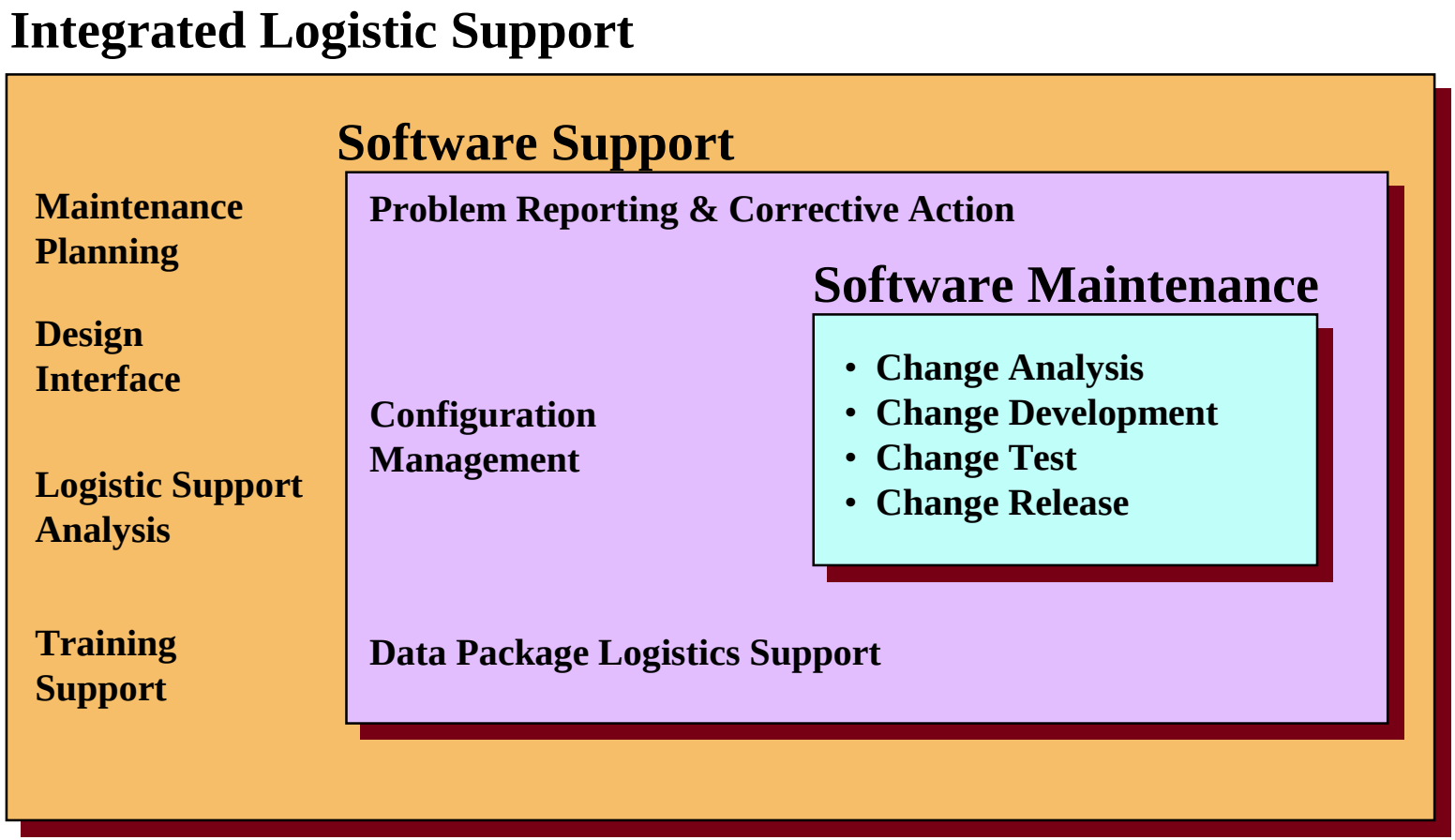
- Transition to Support Activity is completed
- Support concept is implemented: support resources are in place; support facilities are in place; support tasks for modification, logistics management, and operations are well-defined for all application, Off-The-Shelf, reuse software
- Estimate of support activity and support cost is within 10-15% of actuals
- Sustainment capability is operational

Software Support Process

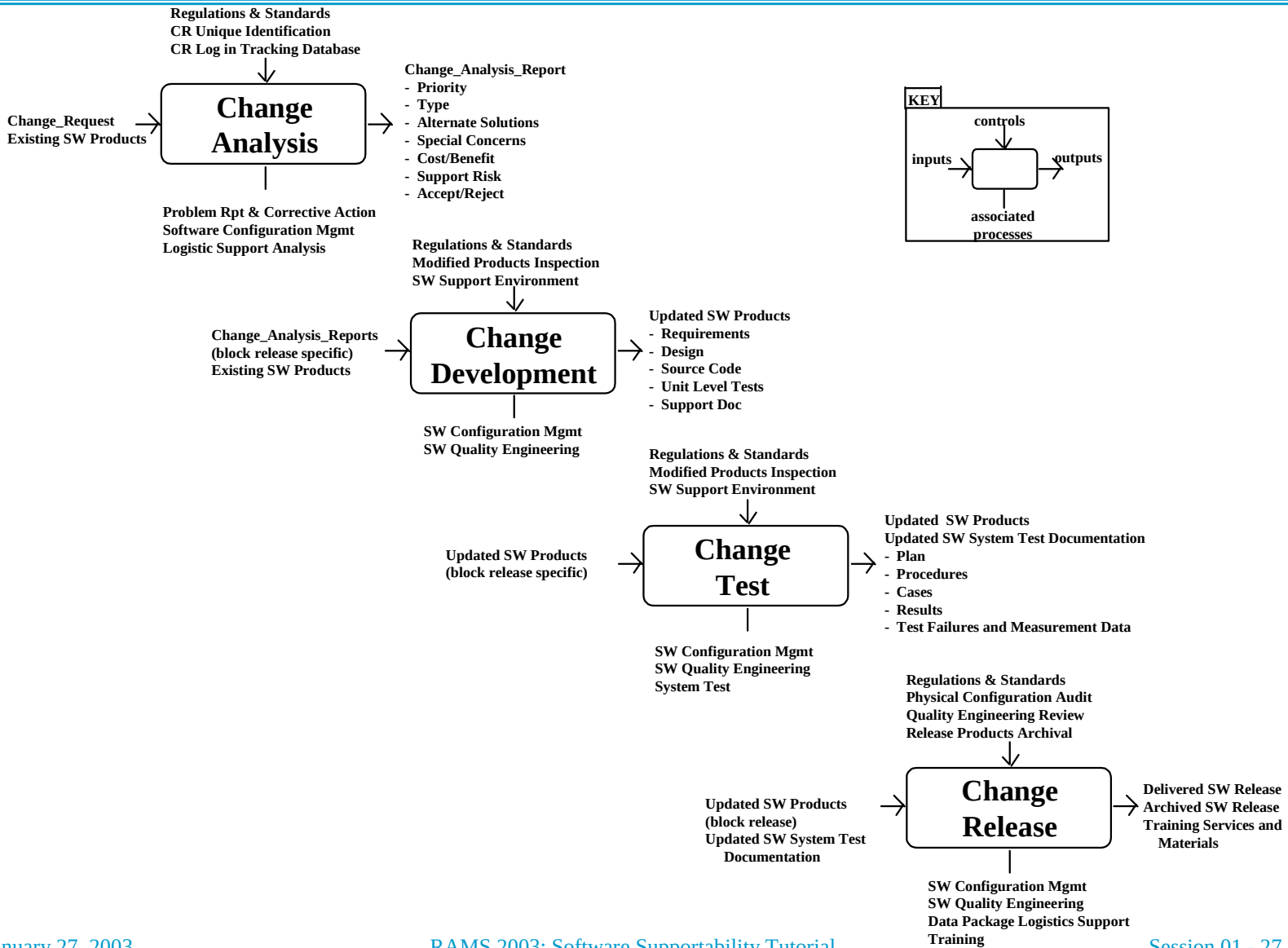
Generic Model (DEF STAN 00-60 Part 3)



Software Modification/Maintenance Task Infrastructure



Software Modification/Maintenance SubTask Activities



Sandia National Laboratories

Session 02

Developing a Support Concept, Plan and Case

Session Objectives

- Introduce Software Supportability Plan/Case
  - JA1004, Software Supportability Program Standard
    - Structure of a Plan and Case mechanism to achieve software supportability
    - Performance-based requirements
    - Walk through the general steps of a software supportability process to drive out a software support concept
    - Acquiring and sustaining software support capability
- Introduce Software Support Concept Dimensions
  - JA1006
    - Support classes: product, process, environment characteristics
    - Support profiles: agent, level, scenarios
    - Support functions: operational, logistics management, modification activities

Session Objectives

- Discuss Case Study Applications
  - Automobile Anti-lock Braking System
  - Missile Guidance System
  - Gateway Server System
  - Ariane 5 Disaster - Code Reuse
  - Software in Future Systems: Advertisement

SAE G-11SW Projects

- SAE G-11SW RMSL Committee
  - International committee within G-11 RMSL Division
  - Software Supportability and Software Reliability focus
- Software Supportability Task Guide
  - AIR5121, Software Supportability - An Overview, published by SAE, January 1997
  - JA1004, Software Supportability Program Standard, published by SAE, July 1998
  - JA1005, Software Supportability Implementation Guide, published by SAE, April 2001
  - JA1006, Software Support Concept Recommended Practice, published by SAE, June 1999

SAE G-11SW Projects

- Software Reliability Task Guide
  - JA1002, Software Reliability Program Standard, published by SAE, July 1998
  - JA1003, Software Reliability Implementation Guide, Committee Draft, November 1999
- Other
  - Reviewed UK MOD Standard: 00-60, Part 3, Software LSA
  - Reviewed UK MOD Standard: 00-42, Software Reliability and Maintainability
  - Reviewed US Documents: MIL-HDBK-502 and MIL-PRF-49506
  - Tentative liaison with ISO/IEC TC56 (dependability)
  - Tentative liaison with ISO 12207
  - Tentative liaison with NATO

JA1004: Software Supportability Program Standard

- Software Support Plan
  - Determining customer requirements
  - Meeting customer requirements
  - Demonstrating customer requirements have been met
- Software Supportability Analysis
  - Designing for supportability
  - Integrating with system support concepts
- Software Support Case
  - Collecting evidence (case) that software support capability has been or will be achieved
  - Assessing that software support capability is adequate

Software Supportability Plan & Case

- Plan and Case Framework
  - The framework for management of a software supportability program is built around two key components: the Software Supportability Plan and the Software Supportability Case. These artifacts are used to satisfy these primary principles:
    - Determine Customer Support Requirements
    - Meet Customer Support Requirements
    - Demonstrate Customer Support Requirements are Met

Software Supportability Plan & Case

- Software Supportability Plan
  - The Plan addresses software aspects of the System Supportability Plan, and describes the activities that are to be undertaken to achieve software supportability objectives. The Plan also describes activities that are to be undertaken to demonstrate that software supportability objectives have been achieved.
- Software Supportability Case
  - The Case provides a justification of the approach and documents evidence which verifies that the software meets its supportability requirements.

Software Supportability Plan

Determining Customer Requirements

- Factors to be Considered
  - Support performance measures
  - Criticality of software, specialty engineering
    - reliability, security, safety
  - Commercial vs bespoke development
  - Location, agents, scenarios
  - Support functions
  - Supportability characteristics: processes, products, environments
  - Supportability analysis tasks: use studies, trade-off analysis
  - Transition from development to support: interim to full
- Use These Methods
  - Historical evidence, existing systems
  - Analytical studies: supportability analysis (e.g., field replaceable SW)
  - Customer/user interactions

SW Supportability Analysis

Meeting Customer Requirements

The selective application of scientific and engineering efforts undertaken during the acquisition [and support] process, as part of the systems engineering process, to assist in:

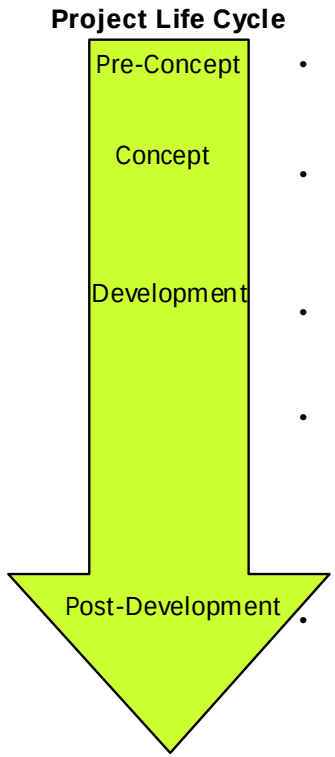
- Causing software support considerations to influence design
- Defining software support requirements that are related optimally to design and to each other
- Acquiring the required software support resources
- Providing the required software support during the operational phase at minimum cost

Software Supportability Case

Demonstrating Customer Requirements

- Types of Evidence
  - Quantitative evidence
  - Qualitative evidence
  - Historical/comparative evidence
  - Demonstrations of support capability and performance
- Structure and Content of Case
  - Supportability requirements
  - Supportability analysis results
  - Support concept derivation
    - Classes, Profiles, Functions

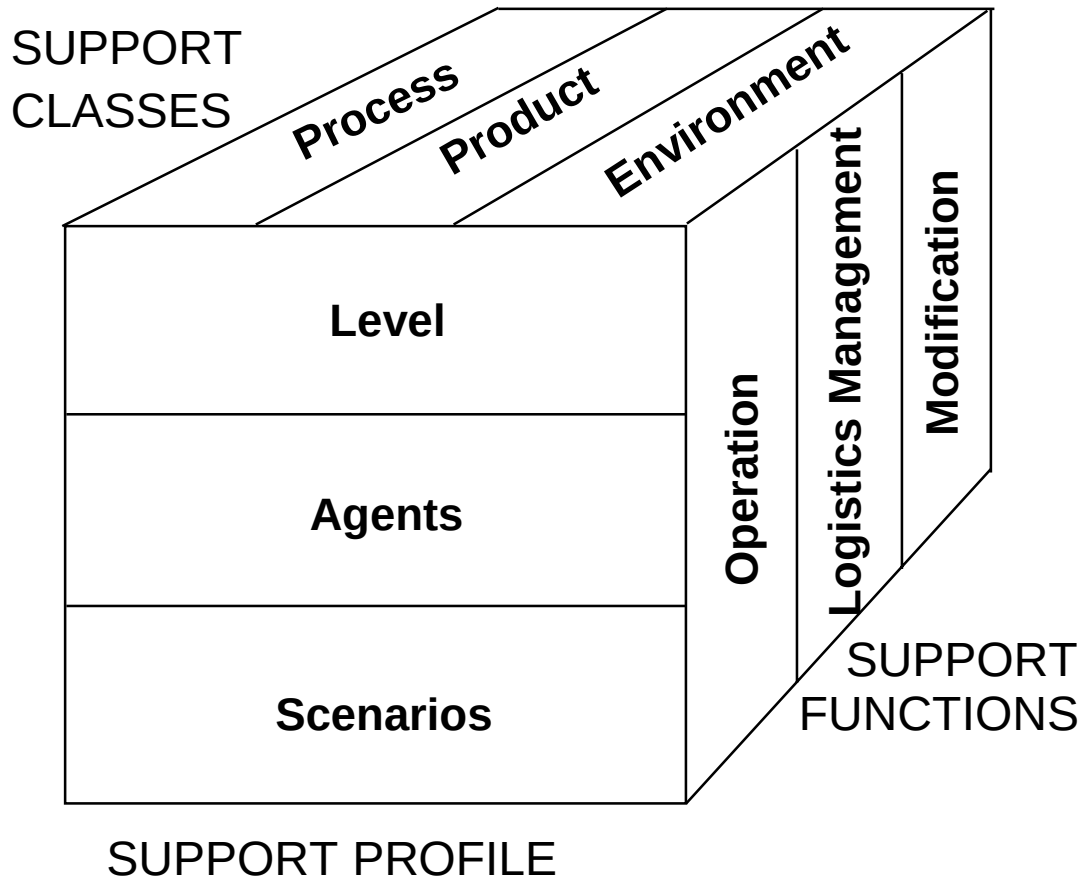
Supportability Analysis Tasks



- **PROGRAM PLANNING AND CONTROL**
  - Development of an early SW Supportability Strategy (***PLAN***)
  - Program and Design Reviews
- **MISSION, DEVELOPMENT & SUPPORT SYSTEMS DEFINITION**
  - Use Study
  - Standarization
  - Comparative Analysis
  - Technological Opportunities
  - Design Factors
  - Integration of ADP Systems
- **PREPARATION AND EVALUATION OF ALTERNATIVES**
  - Functional and Non-Functional Requirements
  - Support System Alternatives
  - Evaluation of Alternatives & Trade-Offs
- **DETERMINATION OF SUPPORT REQUIREMENTS**
  - Operational Task Analysis
  - Software Exception/Problem Support Analysis (FRACAS)
  - Software Modification Analysis
  - Software Transition Analysis
  - Post-Deployment Software Support (PDSS)/Logistics Management Analysis
- **SOFTWARE SUPPORTABILITY ASSESSMENT (CASE)**
  - Operation – Modification – Problem Reaction – Logistics Management
  - Lessons Learnd

NOTE: *Such analyses can be initiated and should be continued during the service phase as well!*

JA1006
Software Support Concept



Operational Support Example Tasks

- Support Initiator: preparation for operation
  - Load/Unload software/firmware element
  - Setup software parameters and data elements
- Support Initiator: software-induced failures during use
  - Reload/Restart Software
  - Recover/Backup Software/Data
  - Initiate Software Problem Report Process
    - Record information about system condition/configuration
- Support Initiator: end of operation
  - Remove/Fit Portable Data Media
  - Fault Investigation Request
  - Security Sanitization

Logistics Management Support Example Tasks

- Support Initiator: System Failure
  - Workaround
    - Help Desk Contact
    - Backup/Recovery
  - Problem Reporting and Corrective Action
    - Problem Report / Change Request
    - Tracking and Status Reporting
- Support Initiator: Software Update Release
  - Data Package Logistic Support
    - Packaging, Handling, Storage
  - Delivery
    - Installation
    - Checkout/Acceptance

Modification Support Example Tasks

- Support Initiator: Change Request
  - Change Analysis
  - Change Development
  - Change Integration and Test
  - Change Release
- Support Initiator: Change Release Complete
  - Configuration Management Release Build
  - Configuration Management Release Control

Software Support Factor Classes/Characteristics

- Software Product
  - Internal characteristics
- Software Process
  - Project and configuration management
  - Development, transition, and support procedures
- Software Support Environment
  - Personnel
  - Support Systems
  - Facilities

Evaluations of characteristics can be conducted as part of Software Process Improvement programs using tools as described in [SEICMM], [SPICE], [PEERCY1], or [PEERCY2].

Sandia National Laboratories

Product Characteristics

- External Interface Characteristics
  - load form, replication and installation features, interfaces for loading mission data to parameterize software functionality, and software/hardware interface mechanisms that have been established to facilitate the logistics management functions.
- Internal Quality Characteristics
  - modularity, simplicity, descriptiveness, consistency, expandability, and instrumentation
  - size or count is generally related to simplicity; number of independent paths, number of lines source code, number of modules, and number of operators and operands, number of function/feature points
  - software product documentation can influence supportability of the software product; complete, consistent, and descriptive documentation will assist software modifications to be understood and correctly implemented without inserting more faults

Product Characteristics

Off-The-Shelf Product Supportability Characteristics

- Supportability Focus for Customer and Supplier Varies
  - Supplier: provide cost-efficient upgrades - essentially the internal quality characteristics
  - Customer: capability to easily integrate and validate the OTS software within its operational use - essentially the external interface characteristics
- Frequency of defects
- Help-desk response capability
- Compatibility of product with other products
- Capability for backward compatibility
- Release frequency
- Intellectual Property Rights / escrow agreements
- Previous release support policy

Process Characteristics

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

Process Characteristics

- Development Process Characteristics?
  - Project Management
  - Engineering
  - Configuration Management and Quality Engineering
  - Integrated Logistics Management
- Support Process Characteristics?
  - Mission Preparation
  - Operation
  - Post-Mission Recovery
  - Problem Reporting and Corrective Action
  - Modification
  - Configuration Management
  - Packaging and Handling (Embodiment)

Environment Characteristics

- Personnel
  - type: management, technical, administrative support, contractor or in-service
  - skill level: computer science, application experience
  - function: manager, analyst, programmer, tester
  - number: count
- Support Systems
  - software engineering tools, configuration management support tools, and test subsystems
  - communication networks, problem reporting, and configuration control must assist logistics management support of product distribution and on-call support for the User.
  - site-specific subsets of software and user manuals as well as advanced communication capabilities (high-speed links, file transfer, e-mail, web site, secure interface) or an extended hot line/help desk

Environment Characteristics

- Facilities
  - office physical facilities;
  - support system physical facilities; and
  - personal convenience physical facilities.
    - Size, lighting, air conditioning, heating, organization of office and support system physical facilities will affect software support productivity. Cafeterias, workout facilities, child care, and other personal convenience facilities enable support personnel to be more productive and less likely to change jobs.

Support Level

- Field
  - operational site where SW is actually used
- Intermediate
  - subordinate site that coordinates SW support for operational purposes
- Central Facility/Depot
  - site where full scale modification of SW is accomplished
- Vendor

What software support levels are implemented? logical vs physical sites?
- support resources, support agents, interfaces, service costs
What tasks are accomplished at each SW support level and by which support agents?
- frequency, problems/defects addressed, procedures

Support Agents

CUSTOMER

- Buyer
  - initial support specific requirements
  - software support personnel and staffing profile
  - life cycle costing support parameters and metrics
  - quality achievement supportability parameters & metrics
- User
  - manuals, training, skill levels
  - problem resolution role
  - SW delivery methods (media, data formats)
  - HW/SW equipment platforms
  - SW installation, recovery, mission support, SW configuration control process

Support Agents

SUPPLIER

- Supporter
  - SW engineering information, support equipment
  - staffing profile, skill level, budget information
  - failure reporting collection
  - response to User problems
- Vendor
  - process improvement measures (CMM level)
  - engineering, quality assurance methods/techniques (requirements, testing)
  - support methods, maintenance fees

Scenario Framework

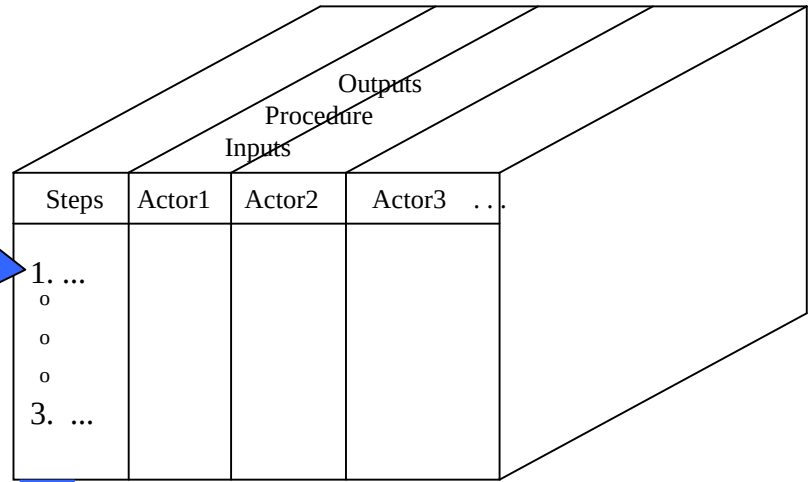
Description:

Pre-Conditions:

Steps:
1.
2.
...

Post-Conditions:

Variations:



Scenario Sequence Diagram

Scenario Case

Step/node Chart

NODE NUMBER: <node id>			
AGENT TYPE: <e.g., user, operator, supporter>		**LEVEL**: <e.g., field, intermediate, depot, vendor>	
SUPPORT FUNCTION NAME: <short name for support function>		**SUPPORT FUNCTION TYPE:** <e.g., operational, logistics management, modification>	
INPUTS		**OUTPUTS**	
From Node	**Description**	**To** Node	**Description**
<previous node id>	<list/description of input data, forms, actions>	<next node id>	<list/description of output data, forms, actions>
TASKS A. < List of actions, activities, procedures conducted as part of this node's function.>			

Example Support Scenario

Vendor Announces Update to Integrated COTS Product

- 1. Initiate Change Request: *Logistics Management Support*
  - initiate change request based on updated COTS product
  - conduct initial change analysis with vendor to determine general impact/cost
  - conduct Change Control Board review of change analysis information
  - send change request to Modification Support for analysis & test
- 2. Analyze Change Request: *Modification Support*
  - analyze change request for integration, transition, fielding impact
  - conduct integrated tests to determine impact of COTS change
  - provide analysis report to CCB
- 3. Allocate Change Request : *Logistics Management Support*
  - conduct CCB review of detailed analysis and change impact
  - allocate change to specific block release
  - send implementation authorization to Modification Support

Example Support Scenario

Vendor Announces Update to Integrated COTS Product

- 4. Implement Block Release: *Modification Support*
  - implement change in allocated block release
  - conduct formal block release integrated testing
  - build and release block to configuration management
- 5. Distribute Block Release: *Logistics Management Support*
  - provide configuration control of block release transition and fielding
  - distribute, install, train field users as necessary
  - release block to the field support
- 6. Install Block Release: *Operational Support Support*
  - conduct appropriate field tests to ensure fielded system is operational
  - send close out problem report/change request to next level
- 7. Closeout Change Request: *Logistics Management Support*
  - close problem report/change request

Example Support Scenario

Field Failure - Workaround

- 1. Conduct Problem Resolution: *Field User/Support*
  - observe failure (user)
  - recognize problem exists (user)
  - conduct preliminary checks iaw training: result - no fix (user)
  - report problem to field support (field support)
  - provide user with possible temporary workaround solutions and runs diagnostic checks per training: result - equip fault, can't isolate problem to software or hardware (field support)
  - remove faulty equipment, report problem to next level (field support)
- 2. Conduct Problem Review: *Logistics Management Support*
  - process faulty equipment problem report
  - retrieve faulty equipment from the field
  - test faulty equipment: result - equip SW fault, needs firmware replication, distribution, installation (service repair vehicle)
  - notify field support of possible equip workaround and field replacement

Example Support Scenario

Field Failure - Workaround

- 3. Replicate, Install, and Distribute Replacement Equipment: *Logistics Management Support*
  - generate replacement request - assume it can be processed by repair vehicle (service repair vehicle)
  - conduct replication and installation of firmware in replacement equipment
  - distribute equipment replacement
- 4. Conduct Field Replacement Checkout: *User/Field Support*
  - test replacement
  - send close out problem report/change request to next level
- 5. Close out Problem Report/Change Request: *Logistics Management Support*
  - close problem report/change request

Example Support Scenarios

Field Failure - No Workaround

- 1. Conduct Problem Resolution: *User/Field Support*
- 2a. Conduct Problem Review: *Logistics Management Support*
  - process faulty equipment problem report (log support)
  - retrieve faulty equipment from the field (log support)
  - test faulty equipment: result - equip SW fault, needs firmware change, replication, distribution, installation (software support center, can't be done by service repair vehicle)
  - notify field support of possible equip workaround and field replacement possibilities (log support)
  - send change request to CCB for review and allocation
  - conduct CCB review of detailed analysis and change impact
  - allocate change to specific block release
  - send implementation authorization to Modification Support

Example Support Scenarios

Field Failure - No Workaround

- 2b. Conduct Software Support Center Modification of Software: *Modification Support*
  - normal change analysis, implementation, test, release
- 2c. Distribute Software Block Release: *Logistics Management Support*
  - configuration control of block release to firmware to Replication, Installation, Distribution function
- 3. Replicate, Install, and Distribute Replacement Equipment: *Logistics Management Support*
- 4. Conduct Field Replacement Checkout: *User/Field Support*
- 5. Close out Problem Report/Change Request: *Logistics Management Support*

Some Risk Issues

- Systems Complexity
  - Numerous Application Software Packages
  - Numerous Off-The-Shelf Packages
  - Operational System Use for Multiple Customers
- Technology Innovation
  - Current vs Future Changes in Hardware and Support Systems
  - Communication Mechanisms
- Transition and Fielding
  - Numerous Field Scenarios
  - Incremental Implementation
  - Software Support Impacts
- Specialty Engineering: Security, Reliability, Safety

Issues of Concern

- Standards/Guides for Software Support
- Software Supportability Analysis Tasks
- Integration of Software Support with System Support
- Software Support Cost

What experience do we have with the above issues?

The Bottom Line

USE PLAN AND CASE FRAMEWORK FROM JA1004

- Define Software Support Concept
  - Support Functions;Support Class Characteristics; Support Profiles
- Determine Risk Areas and Tradeoffs
  - Emergency Support (War/Peace - Middle of Desert/Middle of Town)
  - Support Contractor vs In-House Support Organization
  - Transition of Initial/Update Software Releases
- Estimate Software Support Cost
  - Software Support Initiators; Software Support Processes/Tasks
  - Support Task Frequency and Distribution
  - Task Cost Estimation; Software Support Cost Estimation
- Implement and Evolve Software Support Concept

Sandia National Laboratories

Case Study Fragments

Automobile Anti-lock Braking System

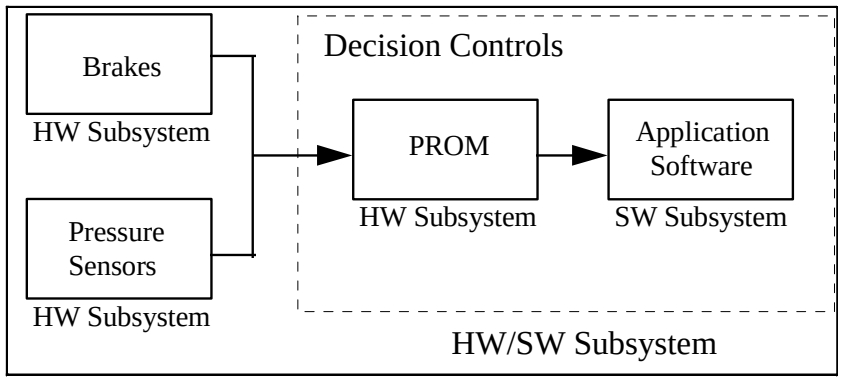
Missile Guidance System

Gateway Server System

Ariane 5 Disaster - Code Reuse

Software in Future Systems: Advertisement

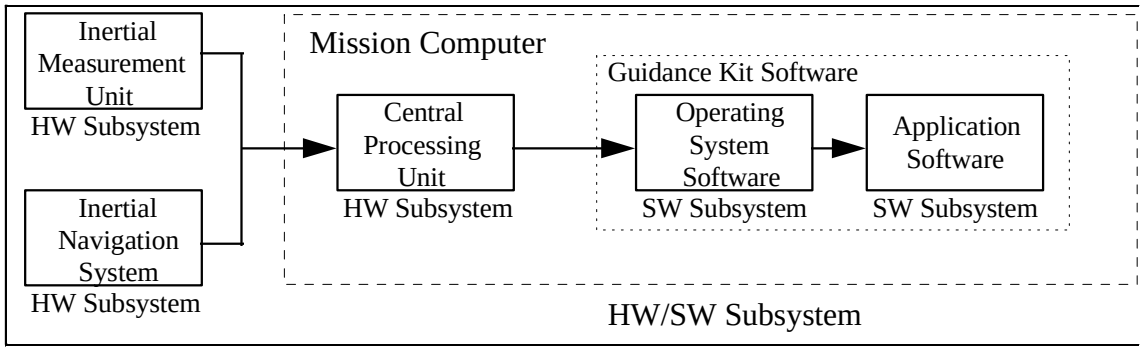
Automobile Anti-lock Brake System



- System consists of the brakes, sensors and actuators to control the applied brake pressure
- Smart system for controlling the rapid application and release of pressure many times per second
- Software is stored on a hardware device called a Programmable Read-Only Memory (PROM) chip

- SW Product
  - What, Where?
  - Criticality? Safety?
- Support Agents
  - Driver?
  - Service Garage?
  - Car Manufacturer?
- Supportability
  - SW BIT Tests?
  - SW support integration?
  - SW on Chip, Plug Replaceable?
  - Complexity?

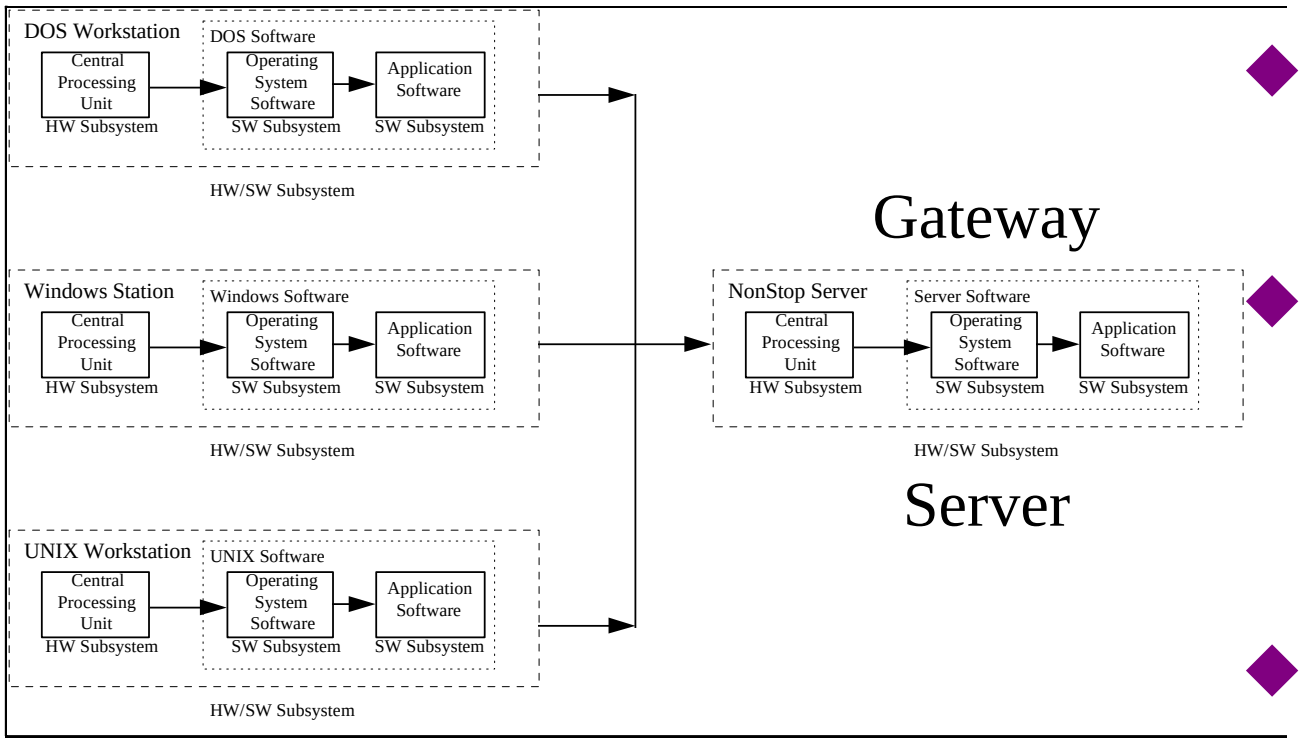
Missile Guidance System



- The Inertial Measurement Unit (IMU) and Inertial Navigation System (INS) are hardware-only components
- Mission computer contains both hardware and software elements
- Operating System (OS) software is operating continuously while the application software only operates when it is commanded to do so

- SW Product
  - What, Where?
  - Criticality? Safety?
- Support Agents
  - User - Military
  - Buyer-Govt?
  - Supporter-Govt/ Contractor?
  - Vendor?
- Supportability
  - SW Mission Load?
  - SW on PROM? Replaceable?
  - HW/SW Logistics?

Gateway Server System



- Gateway server feeds data from three types of workstations to external users
- Each subsystem is composed of HW&SW
- Server only subsystem operating at all times

- SW Product
  - What, Where?
  - Criticality? Security?
- Support Agents
  - User?
  - Buyer?
  - Supporter?
  - Vendor?
- Supportability
  - SW support integration?
  - SW on Chip, Plug Replaceable?

Ariane 5 Disaster - Code Reuse

- 10 years, $7 billion, Ariane giant rocket for launching 3-ton satellites - commercial space business for Europe
- 39 seconds after launch, altitude of 2 1/2 miles, self-destruct mechanism destroyed Ariane 5 & its payload of 4 scientific satellite; aerodynamic forces were ripping the boosters from the rocket.
- Spacecraft swerved off course under pressure three powerful nozzles in its boosters and main engine; rocket made unneeded abrupt course correction, compensating for a wrong turn not taken.
- Steering controlled on-board computer, thought the rocket needed a course change because of numbers from the inertial guidance system. Numbers were actually diagnostic error message. The guidance system had shut down (@36.7 sec) when the guidance system's computer tried to convert one piece of data, the sideways velocity of the rocket, from a 64-bit format to a 16-bit format. The number was too big, an overflow error resulted, the guidance system shut down and passed control to an identical, redundant unit, there to provide backup in case of just such a failure. The second unit had failed in the identical manner a few milliseconds before. It was running the same software.
- Programmers had decided this particular velocity figure would never be large enough to cause trouble. Unluckily, Ariane 5 was a faster rocket than Ariane 4. The calculation containing the bug served no purpose once the rocket was in the air - only function was to align the system before launch; should have been turned off but engineers chose long ago, in an earlier version of the Ariane, to leave this function running for the first 40 secs of flight -- to make it easy to restart the system in the event of a brief hold in the countdown.

Official Report @ http://www.esrin.esa.it/htdocs/tidc/Press/Press96/ariane5rep.html

- Issues for Reuse and Modification
  - Ariane 4 SW reused in Ariane 5 without retest; integration?
  - Modification issues?
  - Requirements defects?
  - Note figure pointing?
  - Lessons learned for cost impact of support?
  - Lessons Learned for COTS?

Software in "Future" Systems

Advertisement: The explosion is seventeen minutes away.

- A gasket is about to blow. A chip in the engine alerts an electronic service that the driver signed up for when he bought the car. This service locates the nearest garage, books an appointment, provides directions to the driver, pushes his next business meeting back an hour and arranges for a rental car to meet him at the garage. But that's only the beginning. Meanwhile, the car manufacturer is notified of the gasket problem (it isn't the first time!) and while the auto giant considers retooling the line, the service drafts a recall notice and engages the help of a PR firm. Wow. A lot is happening here. Car, gas station, rental car company, auto giant are all seamlessly linked to the internet. Not as a collection of websites or in a battle for eyeballs. But as a catalyst for the service-based economy. The next chapter of the internet is about to be written. And it will have nothing to do with you working the Web. Instead, the Internet will work for you, The next E. E-service.

- *Ad by Hewlett Packard in Wired Magazine, Apr 1999.*

- SW Product
  - What, Where?
  - Criticality?
- Support Agents
  - Driver?
  - Service Garage?
  - Car Manufacturer?
  - PR Firm?
  - Service Center?
- Supportability
  - SW Bite Tests?
  - SWHW support integration?
  - SW on Chip, Plug Replaceable?

Sandia National Laboratories

Session 03

Technologies to Improve Life Cycle Cost

Session Objectives

- Introduce Several Important Technologies
  - Risk Assessment Methodology for Software Supportability (RAMSS) - 20 slides
  - Software Block Release Profile - 16 slides
  - Software Support Cost Estimation - 26 slides
  - System/Software Reliability - 29 slides
  - Software Logistics Identification - 11 slides
  - Off-The-Shelf Technologies and Support Issues - 27 slides
- The Bad News
  - There is really only time for one/two - slides are included for all
  - Participants can select!

Sandia National Laboratories

Risk Assessment Methodology for Software Supportability (RAMSS)

Sandia National Laboratories

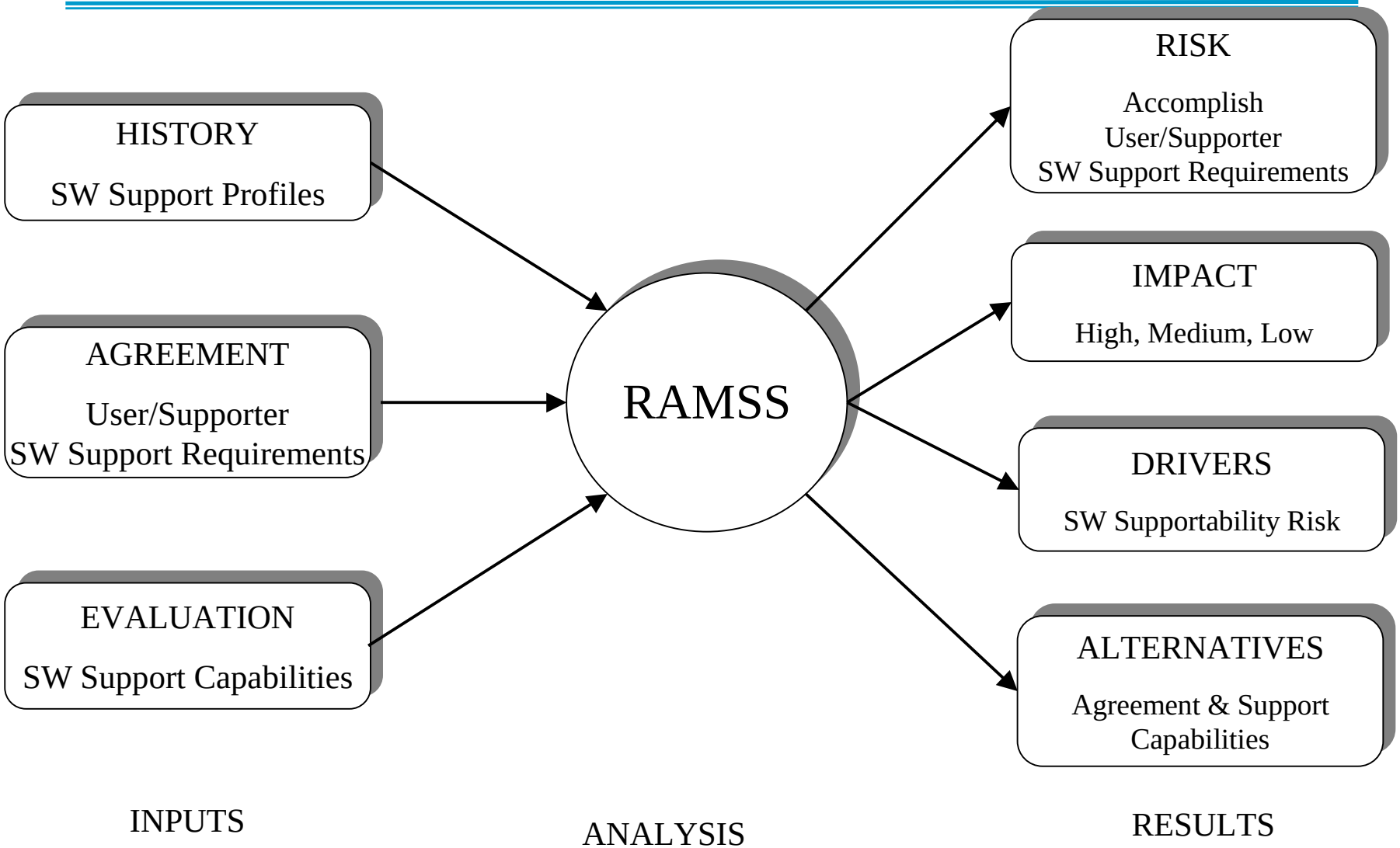
RAMSS

- RAMSS Background / High Level View
  - Software Supportability Classes
  - Methodology Hierarchy
- RAMSS Data Base
  - Evaluated Systems & Counts By Site
  - Summary Background Data
  - Summary Release Statistics
  - Example Raw Data
  - Historical Maintenance Profile
  - Factor Analysis Groups & Error Variance
- Case Study Example
- Demonstration of Program

RAMSS Background

- Development
  - Model, procedures, support tools developed for the Air Force operational Test and Evaluation Center, 1977-1990
- General Application
  - Provides AFOTEC with a structured method for determining the supportability of software acquired for mission critical systems
- Supportability Application
  - Software support product, process, environment characteristics
  - Block release profiles, and personnel required for the modification engineering activity to meet support risk goals; cost-estimation
- Benefits
  - Pinpoint areas of risk and risk mitigation/reduction targets

RAMSS
High Level View



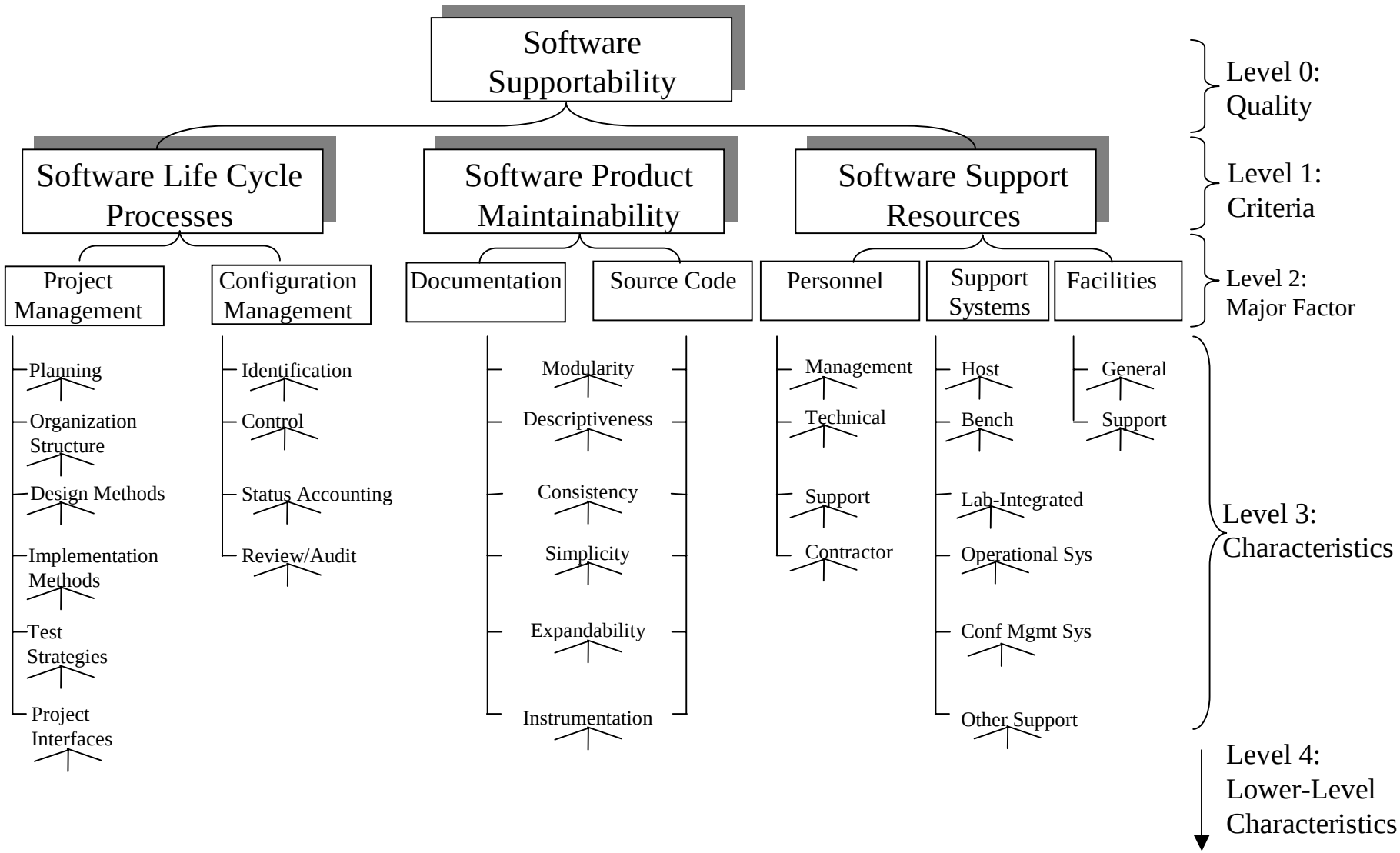
Sandia National Laboratories

RAMSS
Software Support Classes

- Life Cycle Processes
  - Project Management
  - Configuration Management
- Products
  - Documentation
  - Source Code
- Support Resources
  - Personnel
  - Support Systems
  - Facilities

Extensive characteristics of the above classes are defined in the maintainability model.

RAMSS
Methodology Hierarchy



Sandia National Laboratories

RAMSS Data Base Information

- Background Data
  - System Type, Size, Languages, Development Information
  - Personnel Effort, Life Cycle Impacts, Support Systems, Problems
- Maintenance Activity
  - Block Release Data
  - Available Personnel
  - Personnel Expertise
  - System Type
  - Expected Risk Regression Equation
- Supportability Evaluations
  - Factor Analysis
  - Evaluated Risk Regression Equation

Sandia National Laboratories

RAMSS
Example Systems Evaluated

- NORAD
  - CSS, MDS, MEBU, NCS, SSC
- WR-ALC
  - ALR-46, ALR-69, AN/ALQ-131. ALQ-131, APR-38, B-52 EVS ATE
  - E-3A Avionics ATE, F-15(CC, Radar), F-15 Avionics ATE
  - JTIDS, Pave Tack
- Castle AFB
  - A T-4, B-52 (CPT, WST), KC-135
- OO-ALC
  - F-16 (All), F-4/4E/4G, Minuteman

RAMSS
Summary Release Statistics

	Software Type							
Site	ATD	ATE	C-E	EW	OFP	SIM	SUP	TOTAL
NORAD			110					110
WR-ALC		9	2	24	6	2	1	44
SM-ALC					17			17
CASTLE	6							6
OO-ALC	3				34	3		40
OC-ALC	1	7			24		5	37
TINKER			19				14	33
LANGLEY	1		40				8	49
TOTAL	11	16	171	24	81	5	28	336

Sandia National Laboratories

RAMSS
Summary Release Data Elements

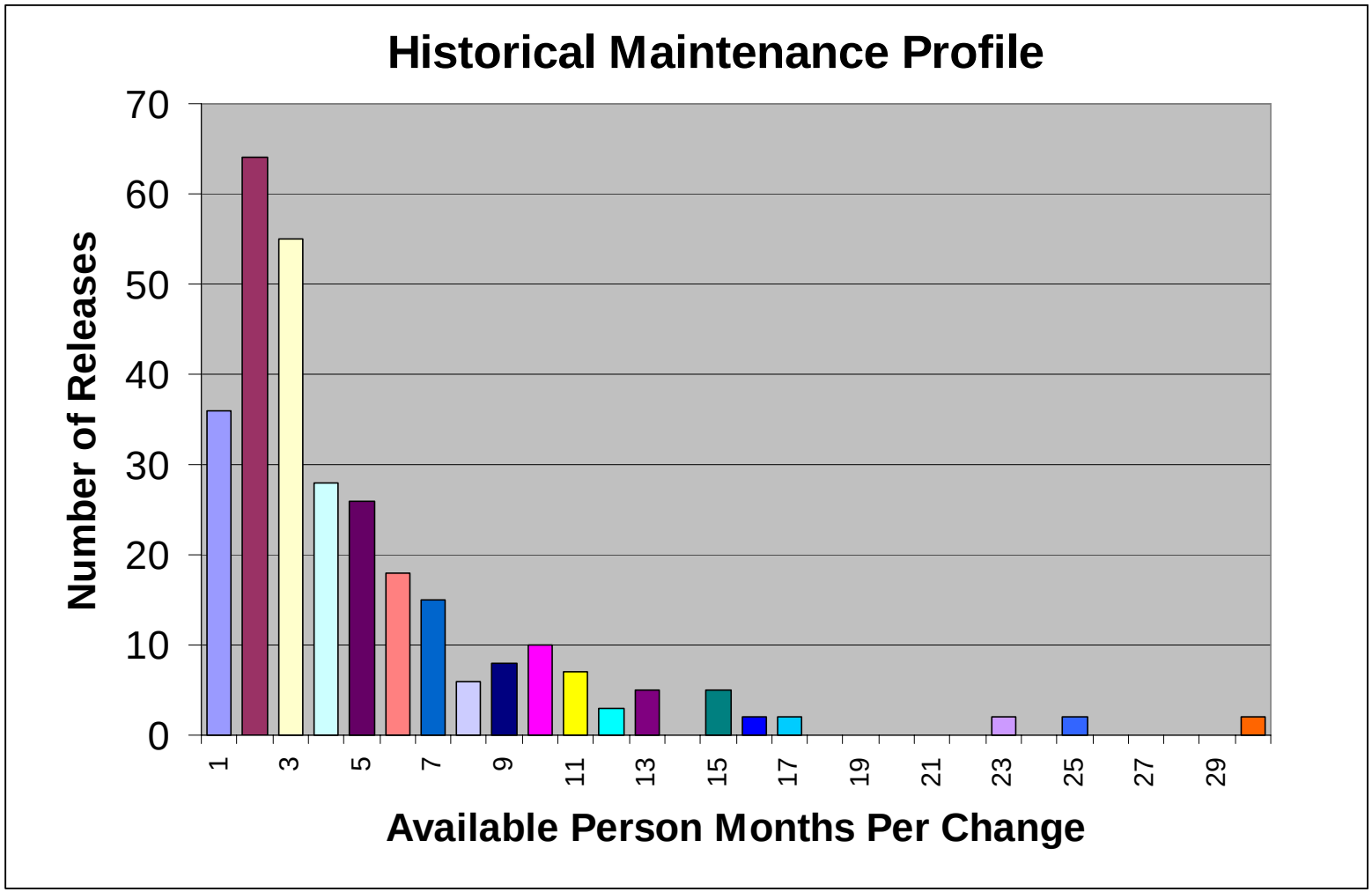
- Name
  - System, Subsystem, Software
- Block Release Schedule and Effort
  - ID
  - Start Date
  - Engineering Complete Date
  - Release Length in Months
  - # Persons on Project
  - % Persons Dedicated to SW
  - % Persons Dedicated To Release
  - Person Months Estimate
  - Person Months Actual
- Block Release Changes
  - Total Number
  - Type Changes
    - # Corrective
    - # Perfective
    - # Adaptive
  - Complexity Changes
    - # High Complexity
    - # Medium Complexity
    - # Low Complexity
  - Priority Changes
    - # Emergency
    - # Urgent
    - # Normal

Sandia National Laboratories

RAMSS
Summary Release Statistics

- Systems
  - Total Number - 81
  - #OFP - 32
  - #EW - 5
  - #C-E/C3I - 13
  - #ATD/OFT - 7
  - #SIM - 6
  - #SUP - 11
- Block Releases
  - Total # - 336
  - # with good profiles - 278
  - # with Type - 308
  - # with Complexity - 175
  - # with Priority - 311
- Changes
  - Total Number - 12,789
  - Type Changes
    - Total Number - 12769 : 100%
    - # Corrective - 9,982 : 78%
    - # Perfective - 2,735 : 22%
    - # Adaptive - 52 : 0+%
  - Complexity Changes
    - Total Number - 7627 : 100%
    - # High Complexity - 857 : 11%
    - # Medium Complexity- 2832 : 37%
    - # Low Complexity - 3938 : 52%
  - Priority Changes
    - Total Number -
    - # Emergency -
    - # Urgent -
    - # Normal -

RAMSS
Historical Maintenance Profile



RAMSS Case Study Example

- Baseline Estimate
- Questions/Objectives
- Estimated Supportability Risk
- Evaluated Supportability Risk
- Estimated and Evaluated Risk Integration
- Trade-Off Analysis

On-Line Demonstration of the Case Study will be Conducted Depending on Available Time

Baseline Estimate

System Profile

- System: F-16
- SW System: Comnet Terminal
- SWType: C-E
- Supporter: WR-ALC
- User: HQ-TAC

General Support Concept

- Release Schedule: 9 Mo Block Release with 3 Month Overlap
- Support Staff: 15 Persons, 19% Dedicated, Avg Skill 3.0 (1Lo-5Hi scale)
  9 Persons, 90% Dedicated, Avg Skill 3.0 (1Lo-5Hi scale)

Baseline Release Change Profile (1st 3 releases)

Block	Total # Changes	Type (C, H, V)	Complexity (H,M,L)	Priority (E,U,N)
1	15	(15,0,0)	(0,0,15)	(0,0,15)
2	20	(15,5,0)	(1,4,15)	(1,4,15)
3	20	(13,7,0)	(1,6,13)	(1,6,13)

Questions Addressed by RAMSS

- What is the Estimated Risk for the Baseline?
  - Risk computed from regression equation that estimates the person months per change from regression parameters derived from the Maintenance Data Base:
    - Average Skill, Block Release Change Profile, Software Type
- What is the Evaluated Risk for the Baseline?
  - Supportability metrics from an AFOTEC Evaluation
  - Risk computed using regression equation of supportability metrics across Hierarchy Framework using Maintenance Data Base
- What Trade-Offs Exist to Reduce the Risk?
  - Vary baseline estimate
  - Improve supportability evaluation metrics
  - Consider variance/confidence range in the historical baseline data

Estimated Supportability Risk

- Available Productivity (Person Months Per Change)
  - (Full Time Equivalent * Release Cycle Length) / Number Changes
- Estimate Productivity (Person Months Per Change)
  - "Optimum" Level of Effort
  - Derived from Regression Equations
    - PMPC $= e^{L\wedge}$, where

      $L\wedge = B_0 + B_1 * AVGSKILL + B_2 * \%CORR + B_3 * \%LOW + B_4 * \%HIGH + B_5 * \%NORM + SUM\{i = 6 \text{ to } 10, B_i * TYPE_i\}$

      B_i = correlation coefficient for each application system type and

      $TYPE_i$ = 1 for the same system type and 0 otherwise
- Estimated Risk
  - Derived from area under normal curve
  - Normal curve has mean (estimated productivity) and variance (standard error of estimate) from the Regression Equation

Estimated Supportability Risk

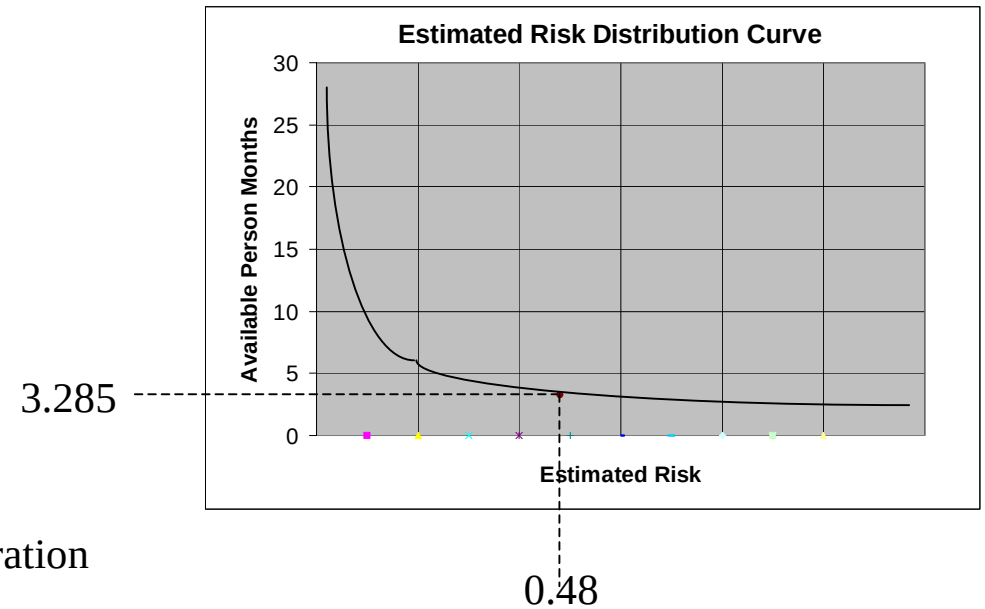
- Estimated Risk for Block 2 (demonstration)
  - Available Productivity
    - (15*0.19 + 9*0.90) (0.667)(9)/20 = 3.285 person months per change
  - Estimated Productivity
    - AVGSKILL = Average skill level of personnel = 3.00
    - %CORR = Percentage of corrective changes = 0.75
    - %LOW = Percentage of low complexity changes = 0.75
    - %HIGH = Percentage of high complexity changes = 0.05
    - %NORM = Percentage of normal priority changes = 0.75
    - TYPE = C-E
    - L^ = 1.13
    - PMPC ^ = Estimated Person Months Per Change = 3.11
    - Estimated Risk = (from equations) = 0.48
  - Evaluated Risk (quality of product, process, resources)
    - Based on existing evaluation data, evaluated risk is 0.55 (which approximately means 55% of the systems in the data base have a lower evaluated risk than the subject system)

Evaluated Supportability Risk

- Evaluated Risk for Block 2 (demonstration)
  - During the next session, a possible demonstration to illustrate evaluated risk using the RAMSS model is a possible option
  - Reference publications on RAMSS in the notes of this session

Estimated & Evaluated Risk Integration

Distribution Function for Block Release 2



Risk Comparison & Integration for All Block Releases

	Block 1	Block 2	Block 3
Estimated Risk	0.14	0.48	0.54
Evaluated Risk	0.55	0.55	0.55
Additional Risk Due to Quality	Estimated risk is probably >> 0.14	Estimated risk is probably > 0.48	Estimated risk is probably not affected

Trade-Off Analysis

Trade-Off Category	Example Trade-off	Risk Impact (Block 2)	Effective?
Estimated Risk			
➢ Available Persons	9 (90%) => 12 persons (90%)	Estimated Risk: 0.48 => 0.39	Somewhat
➢ Block Release Schedule	9mo/3mo overlap => 12 mo/no overlap	Estimated Risk: 0.48 => 0.22	Yes
➢ Change Request Profile	20: (15,5,0): (1,4,15) : (1,4,15) => 20: (18,2,0) : (1,2,17) : (0,2,18)	Estimated Risk: 0.48 => 0.39	Somewhat
Evaluated Risk			
➢ Supportability Metric Accuracy	Variance of 10% in five factors Product: 4.15 => (4.57,3.74) Personnel: 3.53 => (3.88, 3.18) System: 3.72 => (4.09,3.35) Facilities: 4.58 => (5.04,4.12) Life Cycle: 3.32 => (3.65,2.99)	Evaluated Risk: 0.55 = (0.43,0.67)	Yes
➢ Supportability Risk Drivers	Case 1: SCM 3.17 => 4.58 Case 2: SLCP 3.32 => 4.58 Case 3: Personnel 3.53 => 4.58 Case 4: Cases 1,2, & 3	Case 1: 0.55 => 0.43 Case 2: 0.55 => 0.35 Case 3: 0.55 => 0.52 Case 4: 0.55 => 0.31	Yes Yes No Yes

Sandia National Laboratories

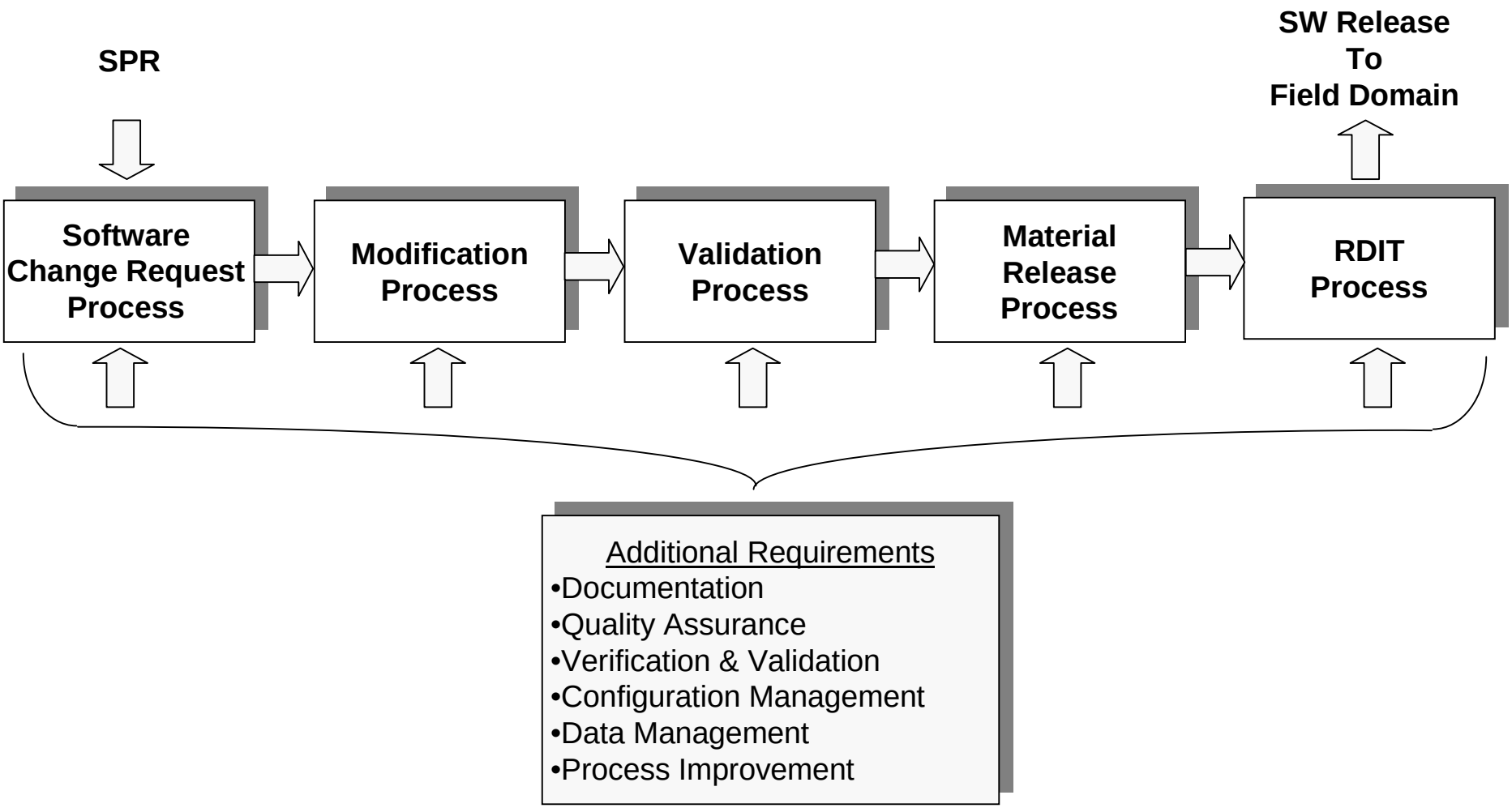
Block Release Process

Block Release Process

- Software Support Process Flow
- Change Request Process
- Normal block release
- Block Release Example Data Elements
- Conditional/Interim block release
- Time Critical/Emergency release
- Iterative Block Release Implementation Method
- Example Normal Block Release Content

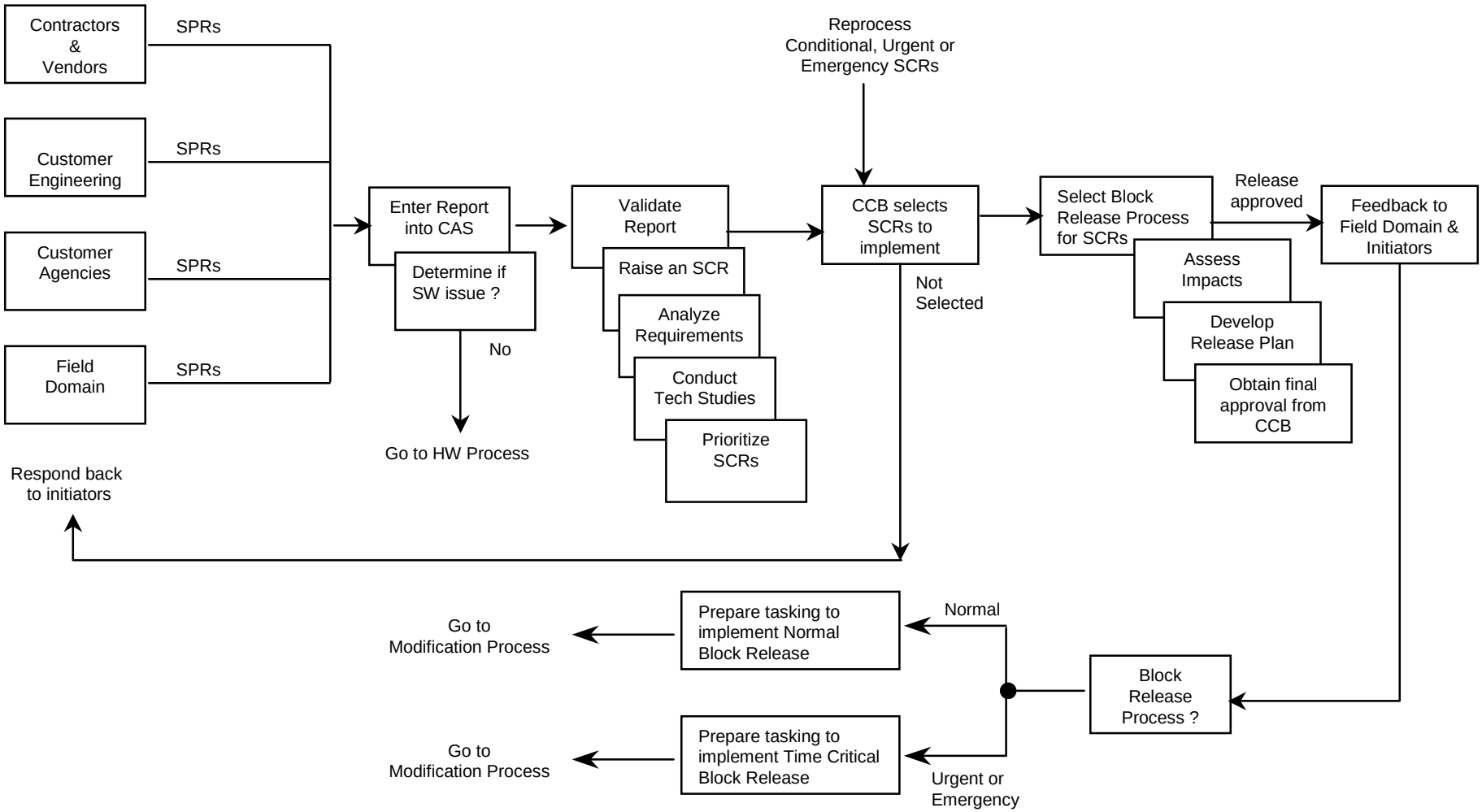
See session written notes about Block Releases

Software Support Process Flow



SPR - Software Problem Report
RDIT - Replication, Distribution, Installation and Training

Change Request Process

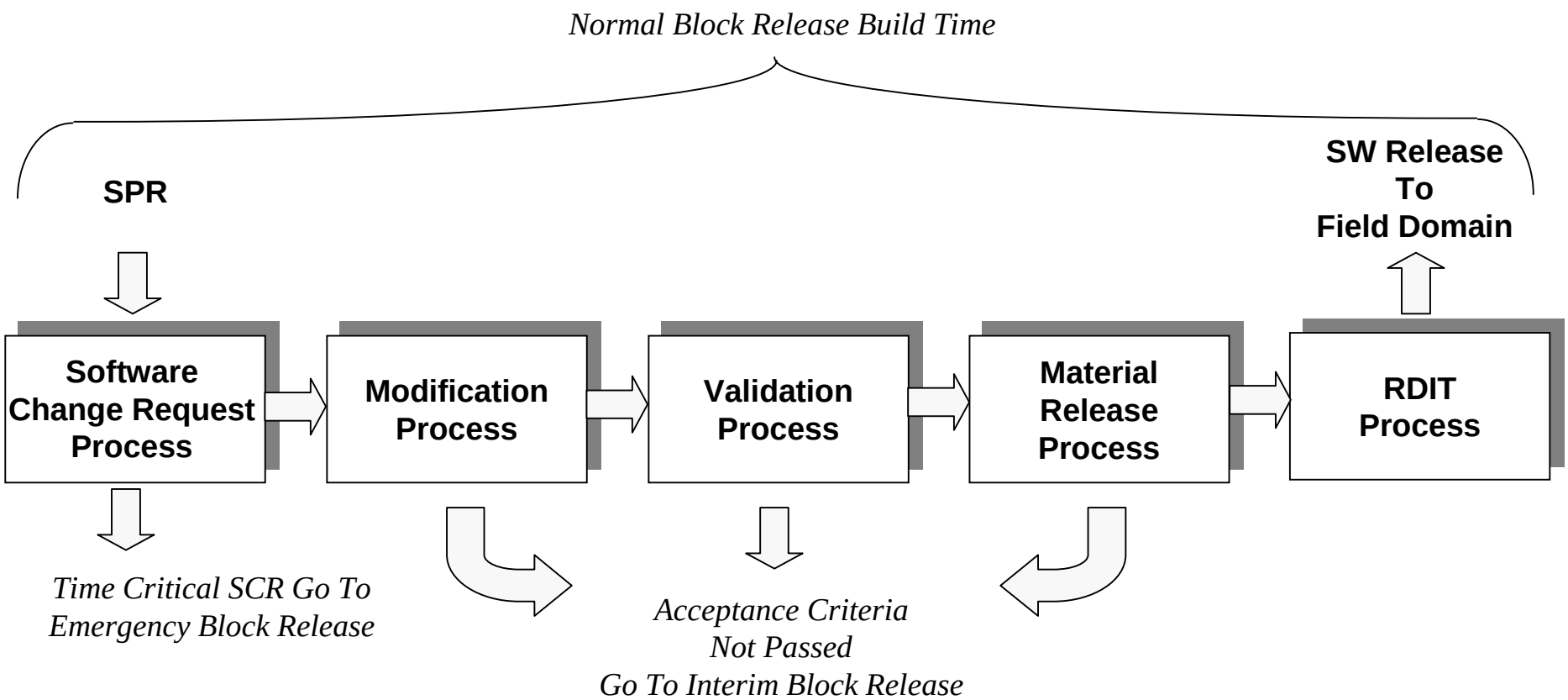


Normal Block Release

- Purpose: provides release of normal changes to the software for corrections, enhancements, and adaptations
- Executes all software support process flow steps
  - Implements non-time-critical SCRs that solve many problems
- Schedule
  - Depends on time to execute all the steps in the Process Flow, and time for Field Domain to assimilate previous Normal Block Release. An example of timing for block release might be:

Process Step	Timing
Software Change Request (package of requests)	3 months
Modification	9 months
Validation	3 months
Material Release	1 month
RDIT	1 month
Totals	17 months

Normal Block Release Process Flow



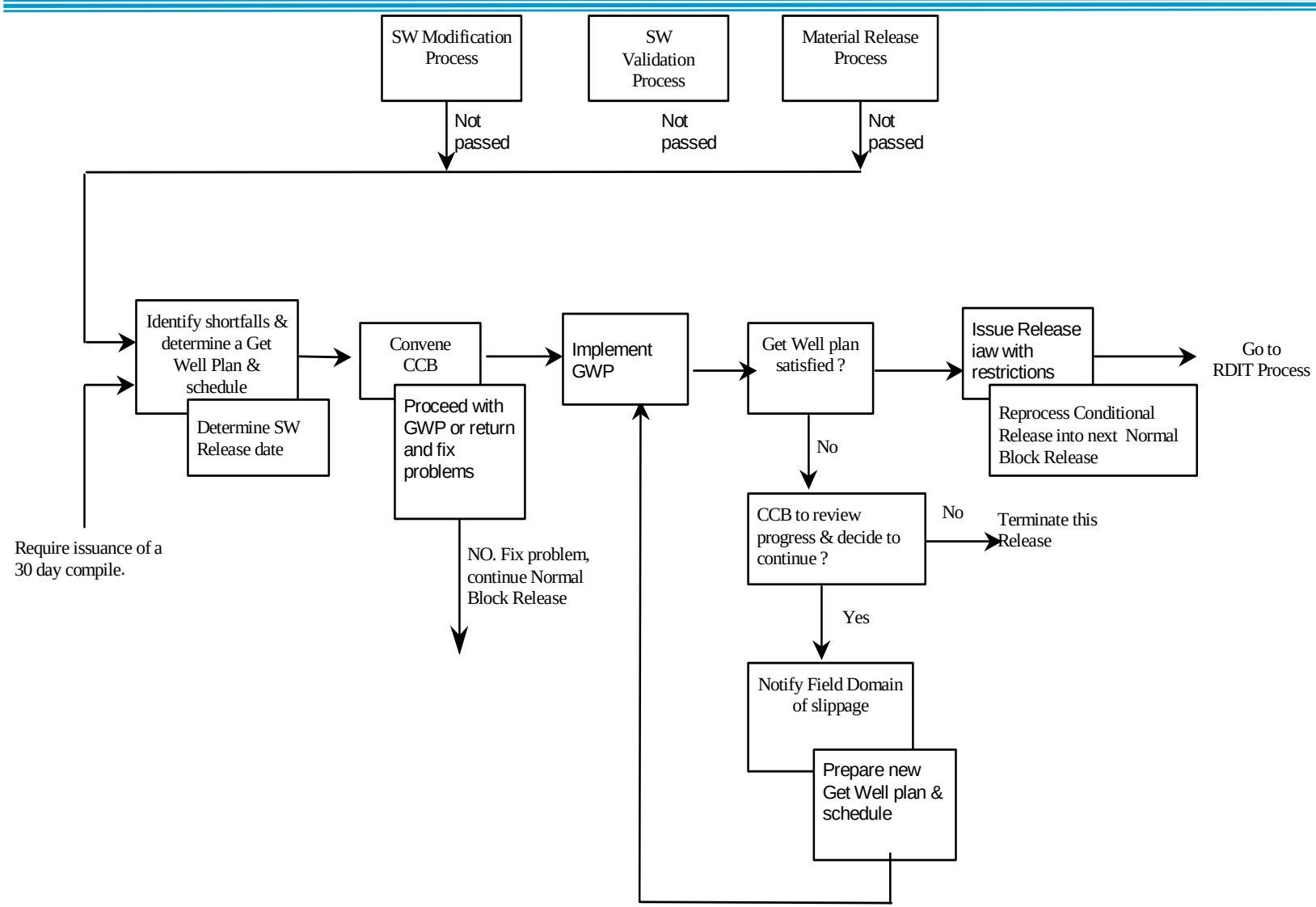
Block Release Example Data Elements

- Example Block Release Data Elements
  - Type Release – Normal, Interim, Urgent/Emergency
  - Total # CRs – Total Number of Change Requests implemented by the block release
  - # CR by Type – Number of Change Requests by Type (Corrective, Perfective, Adaptive)
  - # CR By Complexity – Number of Change Requests by Complexity (High, Medium, Low)
  - # CR by Priority – Number of Change Requests by Priority (Normal, Urgent, Emergency)
  - Personnel Load by Function – Personnel time (e.g., person hours, person months, equivalent FTEs) on the release by functional area (e.g., change analysis, modification, validation, material release, RDIT)
  - Personnel Skill Level – Measure of overall personnel capability to complete release by functional area (e.g., change analysis, modification, validation, material release, RDIT)
  - Changed Source Size – Measure of the amount of software changed by the implemented Change Requests (e.g., Function Points, Source Lines of Code)
  - Schedule – Breakdown of block release schedule timing by functional area (e.g., change analysis, modification, validation, material release, RDIT); includes specification of overlap time with other releases

Interim Block Release

- Purpose: provides an *acceptable* operational responsiveness to a requested software change when:
  - One or more Process Flow steps not successfully passed; and/or
  - Immediate interim Block Release is required to overcome difficulties experienced by the Field Domain.
- Within time period (e.g., 30 days)
  - Customer Engineering has access to all implemented SCRs to-date through the issuance of an Interim Block Release
  - Customer Engineering will issue a Interim Block Release that contains the implemented SCR solutions
    - to overcome difficulties experienced by the Field Domain and a solution has already been implemented as part of the 30 day Interim Block Release
- Interim Block Release changes are reprocessed into the next available Normal Block Release

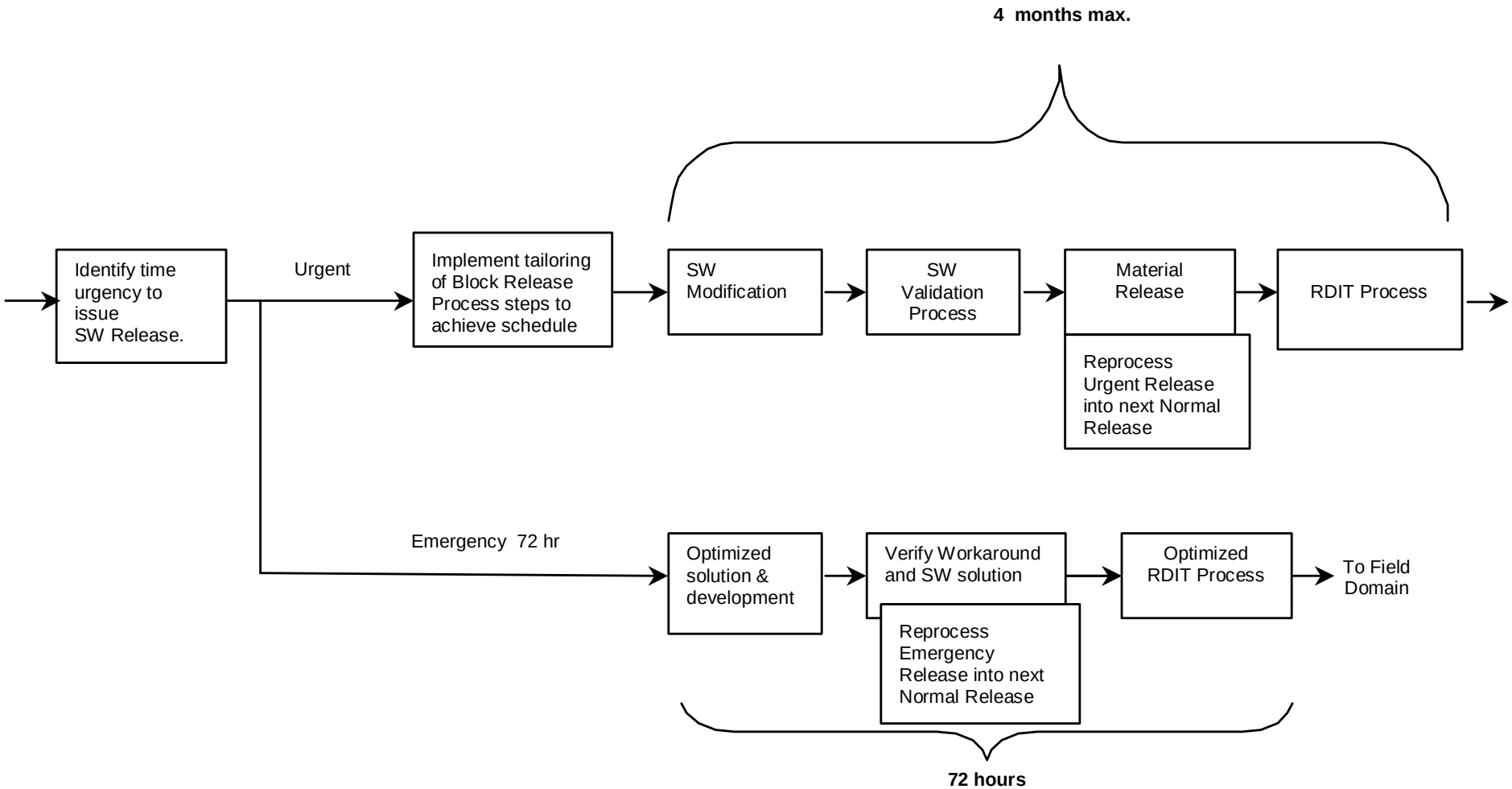
Interim Block Release Process Flow



Emergency Block Release

- Purpose: provides mechanism to overcome a serious difficulty with the System within an allocated response time constraint.
- Two types of Emergency Block Releases
  - Urgent, this type of Release solves a *specific problem* within a maximum response time interval of (for example) 4 months
  - Emergency, this type of Release implements a single SCR within a maximum response time interval of (for example) 72 hours.

Emergency Block Release Process Flow



Note: times indicated are examples

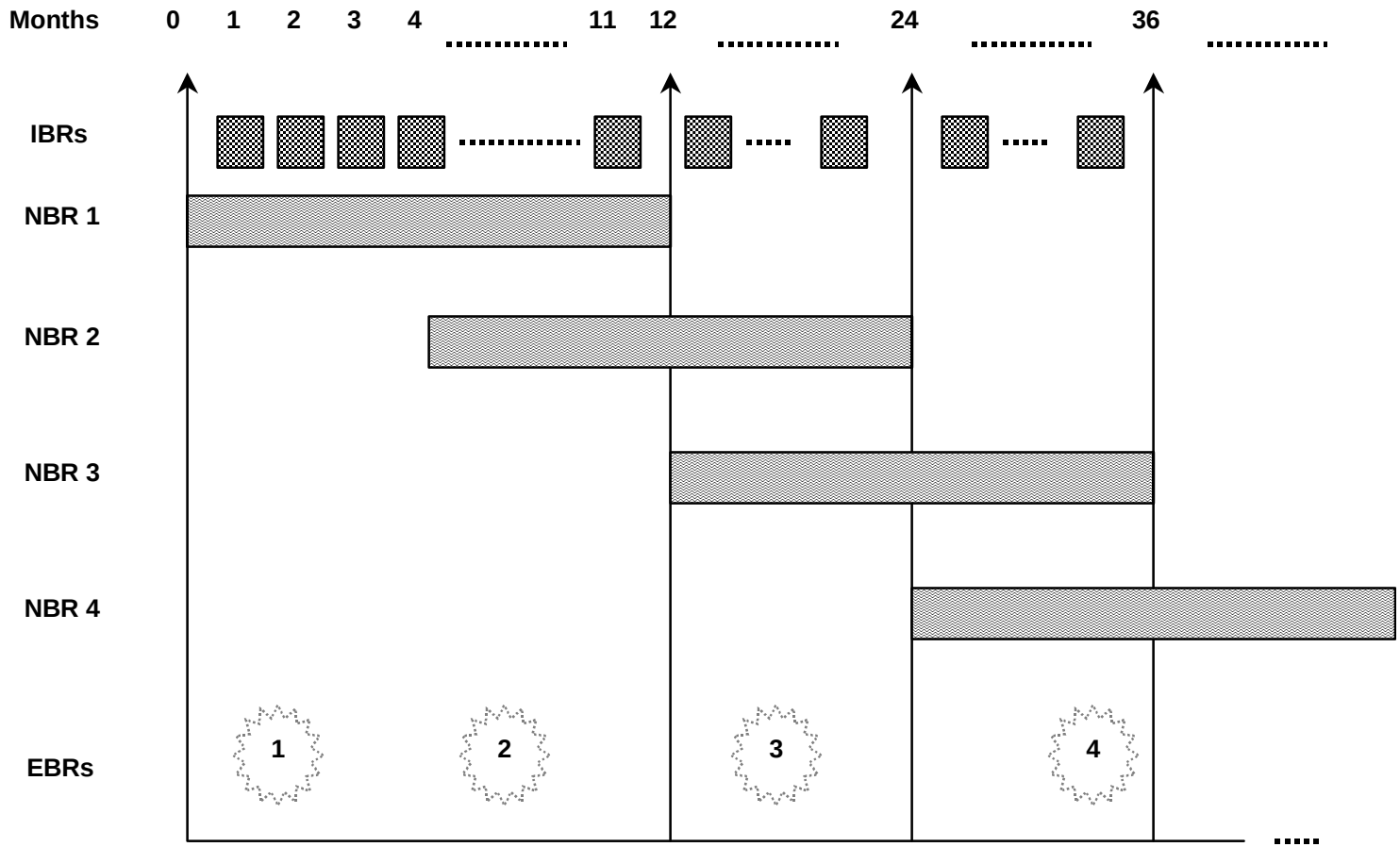
Iterative Block Release Implementation Method

- Purpose: provides method that is responsive and flexible to Block Release change schedule perturbations. Can accommodate:
  - short turnaround engineering cycles
  - inclusion of urgent change requests in the next Normal Block Release even though the Block Release is in progress
  - emergency situations where a quick fix to a software item in a limited number of Field Domain equipment may be required.
- Tailoring
  - The Block Release Implementation Method would need to be tailored to specific Customer/Supplier organizations based on the Software Support Concept derived to meet Customer performance requirements.

Iterative Block Release Features

- Features
  - individual changes are being implemented in smaller overlapping build cycles
  - maximum effective use of scarce and costly software engineering personnel resources
  - reduction in the Block Release schedule as compared to producing sequential Block Releases.
  - schedule-bound process
    - Normal Block Releases (NBRs) are released on a periodic basis, and contain as many implemented SCRs as resources permit.
  - During specified period of time (e.g., 30 days)
    - Customer Engineering can issue Interim Block Release (IBR) if necessary to solve difficulties experienced by the Field Domain
  - In interrupt mode, Emergency Block Releases (EBRs) can be issued when time responsiveness is critical.

Iterative Block Release Example Schedule



As an example, in order to achieve periodic NBRs on a desired 12 month cycle, an approximate work period of 20-24 months is estimated to be required for each NBR. The overlap of each 20-24 month NBR results in a 12 month apparent NBR cycle.

Normal Block Release Example Content

Period	Description	Profile	Type	Complexity	Priority
0-3 years	One Block Release every 6-8 months; may be overlapped; primary focus on corrective	22 SCRs	17 corrective (75%) 4 perfective (20%) 1 adaptive (5%)	2 high (10%) 9 med (40%) 11 low (50%)	19 normal (85%) 2 urgent (10%) 1 emergency (5%)
4-7 years	One release every 9-12 months; may be overlapped; balanced focus on corrective and perfective	44 SCRs	20 corrective (45%) 15 perfective (35%) 9 adaptive (20%)	4 high (10%) 18 med (40%) 22 low (50%)	38 normal (85%) 4 urgent (10%) 2 emergency (5%)
8-14 years	One release every 9-12 months; may be overlapped; primary focus on perfective and adaptive	44 SCRs	9 corrective (20%) 26 perfective(60%) 9 adaptive (20%)	9 high (20%) 22 med (50%) 13 low (30%)	38 normal (85%) 4 urgent (10%) 2 emergency (5%)

Sandia National Laboratories

Software Support Cost Estimation

Sandia National Laboratories

TOPICS

- Software Support Cost Estimation Basics
- Software Cost Estimation Methods/Models
- Example of Software Support Cost Estimation
- Using Activity Based Models with Software Support Profiles

Software Support Cost Estimation Basics

- Support Elements
  - C_D = Cost for Development Phase Activities
    - Software supportability planning, analysis, concept, case
  - C_S = Cost for Operational Support Phase Activities
    - Operational Support
    - Logistics Management Support
    - Software Modification Support
  - Relative Benefits for C_D and C_S
    - Schedule, Cost, Performance Trade-offs
- Software Products
  - Developed for Operational Use
  - Acquired as COTS/NDI for Operational Use
  - Acquired/developed for Support/Test of Operational Products

Software Support Cost Estimation Basics

- Basic Algorithm (Cost During Support Phase)
  - Software Support Cost Per Year = Sum of:
    - Cost for Operational Software Support
    - Cost for Logistics Management Software Support
    - Cost for Software Modification Support
    - Cost for Infrastructure Support
      - Support Staff, Support Systems Repairs, Maintenance Service Contracts, Travel and Indirect Costs, Training (operational and support staff), CM/QA, Delivery
- Typical models
  - Only compute cost for **software modification** & use a derivative of the basic software development cost model:

Person Months = a (size)^b x factors (product, platform, personnel, project)
Time to Develop = c(PM)^d
Staffing Profile sometimes is provided for distribution of
person months across:
personnel job-function categories
life cycle task activity categories
calendar time

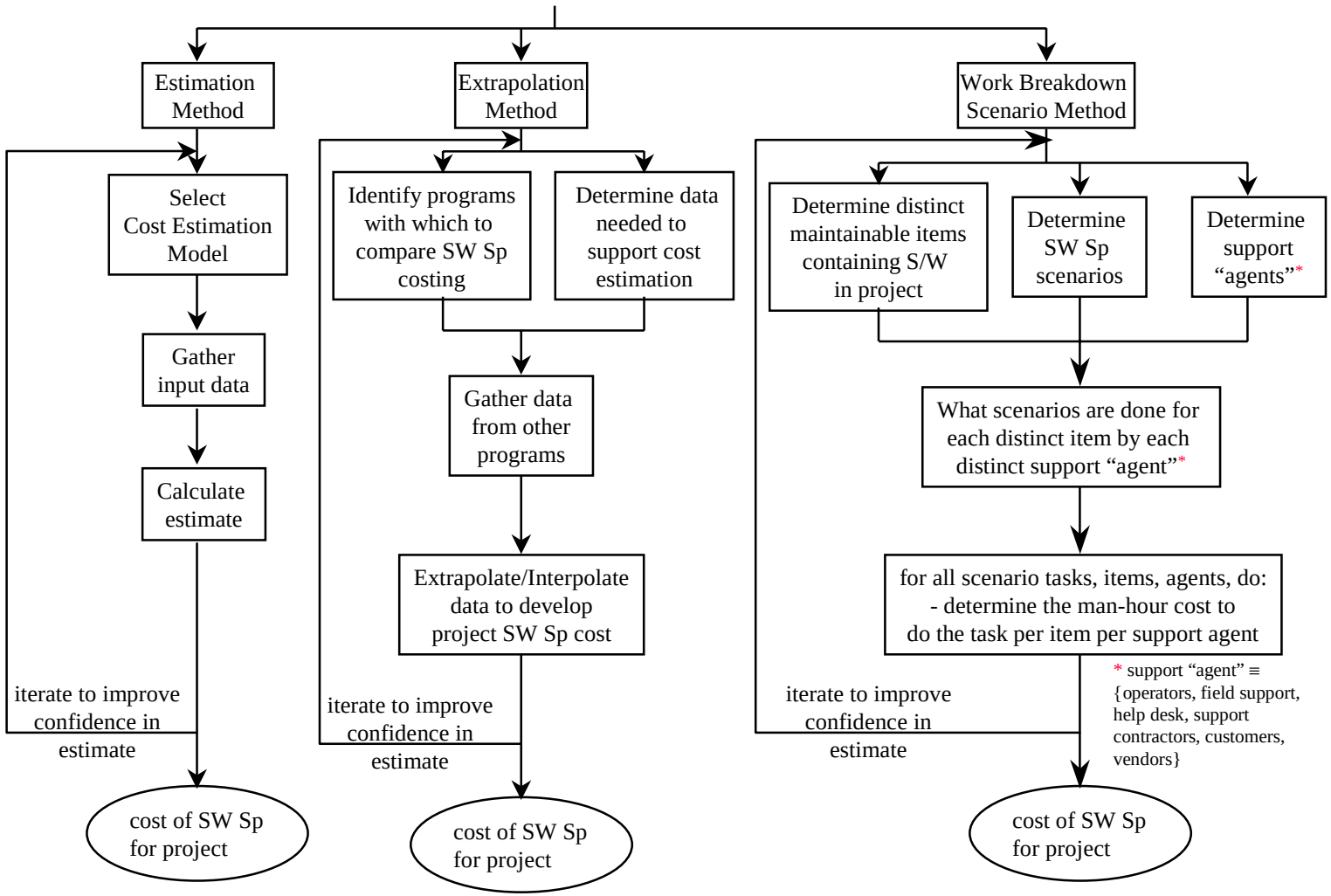
Software Support Cost Estimation Basics

- Support Cost is Driven by Activity and Load
  - Activity is determined by support initiator events and the subsequent profile support scenarios that result
  - The initiator events may result in an“assist” (reboot), a query for assistance at any of the support profile levels, and/or change request for the software product with eventual software release to the field
- Example Software Support Change Profile
  - First Four Years of Support: 75% corrective; 24% perfective; 1% adaptive
  - Next Four Years of Support: 35% corrective; 55% perfective; 15% adaptive
  - Next Four Years of Support: 20% corrective, 60% perfective; 20 % adaptive
  - Rest of Support Years
    - corrective increases, adaptive has slight increase, perfective decreases

Software Support Cost Estimation Basics

- Example Software Support Block Release Profile
  - 6 to 18 months per block release
  - block releases overlapped ~ 3 months
  - profile of change type, change complexity, change priority along with product, process, and environment quality characteristics affect the supportability risk of each block release
  - ~ 3.5 person months per change request in block release
  - Typical block release (early) composition
    - 15 corrections, 4 enhancements, 1 adaptation
    - 19 normal, 1 emergency
    - 1 high complexity, 4 medium complexity, 15 low complexity
  - Typical block release K ncsloc changed
    - 2000 per high complexity, 700 per medium complexity, 75 per low complexity
    - For Example: 5,925 ncsloc changed ; ~ same as 46 Function Points

Software Support Cost Methods



Software Cost Models

- Rules of Thumb
- Constructive Cost Model (COCOMO)
- Ray's Enhanced Version of Intermediate COCOMO (REVIC)
- PRICE-S / ForeSight
- Software Lifecycle Methodology (SLIM)
- System Evaluation and Estimation of Resources - Software Estimation Model (SEER-SEM)
- Activity Based Cost (ABC) - Similar to Scenarios

Sandia National Laboratories

Model Summary

Model	Model Approach	Support Model	Estimates	Strengths	Weaknesses	Comments
ROT	• Historical Data • Similar Systems	• Size / PROFAC	• Effort • Schedule	• Accurate for similar systems • Easy to use	• No calibration • Wide variance of accuracy • No visibility into support tasks	• Provides a quick bound for support LOE
COCOMO	• Basic, Inter, Detail Models • Organic, Semi, Emb, Ada Modes • LOC for Size • exponential with factors • Historical Data Base	• ACT for Size	• Effort • Schedule • Phase Distribution • Activity by Phase	• Established Model • Well Known • Easy to Use • Fairly accurate for similar systems	• LOC basis • Calibration is dated • New technologies not represented in data base • No visibility into maintenance tasks	• Provides a quick profile of development phases. • Provides a quick bound for support LOE by year.
REVIC	• COCOMO Inter • Adds Req & DT&E • Expanded DOD Data	• Same COCOMO	• Same COCOMO	• Same as COCOMO	• Same as COCOMO	• Same as COCOMO
PRICE-S	• PROFAC per CSCI • WBS/Appl Domains • FP & LOC Size • Development, Purchased, Furnished • Historical Data Base	• Life Cycle Model • Correction, Enhancement,Growth estimates • Four Block Releases	• Appl Model - program costs - schedules • Life Cycle Model - program costs - each sup type • Risk Analysis - graphical - min, mean, max - std dev - coef of var	• Real Time, CC, Embedded Appl • Acq, LifeCycle, Risk Analysis • LOC & FP Size • Good Calibration • GUI • PC & Unix • Exports to MS Office & Project	• Proprietary Model • Data Base Not Avail • Sensitive to PROFAC	• Pretty good at estimating software modification support cost • Accuracy depends on validity of historical data base

Sandia National Laboratories

Model Summary

Model	Model Approach	Support Model	Estimates	Strengths	Weaknesses	Comments
SLIM	• Dynamic, macro-level model • Development effort and schedule approximated by staff-buildup to a Rayleigh curve • Estimates by interleaving size, effort, and schedule • 4500 projects, 117.1M LOC, 78 languages, mostly business in historical data base	• Life Cycle estimation	• Effort and cost plan • schedule • staffing plan • MTTF • Risk assessment • Documentation	• Ease of use via GUI • Risk analysis function • Sensitivity analysis features • Report output capability	• Schedule compression factor is exaggerated • Staff buildup not like a Rayleigh curve • No stat evidence for productivity index • Project data base not published	• Pretty good at estimating software modification support cost • Accuracy depends on validity of historical data base
SEER-SEM	• FP & LOC Size • New SW, Reuse SW, COTS • Maintenance - years of support - # operational sites - size growth - life - min & max staff - % SW supported	• Life Cycle Model	• Effort and cost • Schedule • Staffing profile • Risk assessment • Sensitivity and trade-off analysis • Defects • Reliability	• Extensive use in real-time, embedded, C&C, MIS, defense • Flexibility for domains • Apply standards such as 2167A, 498, 9001, J-Std-016 • Planning Support • Variety of Mgmt Rpts • Can calibrate with historical data or organization data • Powerful Tool for - Tradeoff evaluations - Risk analysis - Project Planning	• Proprietary model; algorithms and data not available • unclear if data statistically supports all input data • independent results from using model are not published • Requires calibration to organization's data to be effective	• Has good possibilities for estimating software modification efforts

Sandia National Laboratories

Model Summary

Model	Model Approach	Support Model	Estimates	Strengths	Weaknesses	Comments
ABC	• Basic cost estimation by WBS activity analysis • Can be effectively used for development and support • Activity effort and schedule are typically normalized to software size metric (FP, LOC)	• Same for development or support model	• Bottom up estimate - schedule - effort - staff profile - activity by phase	• Enables managers to assign quantified measurements to specific sw development and / or support activities • Provides visibility into process activities • Enables communication among managers, developers, users, and supporters • Identifies improvement opportunities	• Requires a mature measurement capability • Assigning support cost to individual activities is an art • Unit cost per activity must be calibrated to organization	• Pretty good process to estimate software support costs

Rule of Thumb Macro Estimates

- % Life Cycle Method
  - Software Support Cost = % Total Software Life Cycle Cost
  - % modeled as 60%, 70%, 80% for low, medium, high change rate
  - Assumptions are added to separate out Field and Infrastructure Domain software support costs
- Size Per Person Year Method
  - Software Support Cost = software size / (support size per person)
  - Support Size Per Person modeled as variation
  - 10, 20, 30 KLOC per person for application software
  - 20, 40, 60 KLOC per person for test software
  - Assumptions are added to separate out Field and Infrastructure Domain software support costs

Rules of Thumb

- Historical Based
  - Software support costs based on previous similar projects
- Example Application of “Rules”
  - % Life Cycle Cost 30-70 Rule
    - support = 70% SW LCC; development = 30% SW LCC
    - Example: If Development cost = $157M then Support cost = $366M
      - Over a 20 year support life cycle, Support cost ~ $18.3M per year
  - Size per person year
    - Range: one person per 0.3K to 200K ncsloc
    - Average: one person per 23K ncsloc
    - Example: for a 3M ncsloc program, need ~130 support personnel

Example % Life Cycle Model
REVIC

- Description
  - REVIC is a parametric model for estimating software development and support costs based on the Constructive Cost Model (COCOMO) developed by Barry Boehm [BOEHM81]. It provides cost estimates for the basic four COCMO software development phases (preliminary and detailed design, implementation, integration test) and the system requirements engineering, system test, and support phases. In addition, this model provides special equations for software developed in the Ada programming language
- Features
  - Based on COCOMO; parametric model based on Delivered Source Lines of Code, project difficulty, and 19 parameters
  - Calibrated against a large data base (63 COCOMO projects and > 600 projects in Rome Laboratory data base)

Example % Life Cycle Model
REVIC

- PROS
  - Non-proprietary; used by Air Force as macro model
  - Calibrated against data base; simple to use; free automated tool support
- CONS
  - Based on Delivered Source Instructions
  - Parameter estimation is very subjective; predictive accuracy is still in the “Ball-Park” category and can’t be tied back to actual task activities
  - Assumes requirements specification does not change
  - Support cost is derived from development perspective; basically is a % life cycle cost model based on annualized change rate (# sloc changed)

REVIC
Definitions and Assumptions

- Intermediate COCOMO Model is Basis
  - **Organic:** Stand-alone small program, few interfaces, stable development environment, few constraints
  - **Semidetached:** Combination of Organic & Embedded
  - **Embedded:** Complicated and/or large, considerable interfaces, tight constraints, high criticality
- Life Cycle Phases (* - COCOMO Phases)
  - Phase 1: Systems/Requirements Engineering
  - Phase 2: *Preliminary Design
  - Phase 3: *Detailed Design
  - Phase 4: *Code & Unit Test
  - Phase 5: *Integration Test
  - Phase 6: Development Test & System Test
  - Phase 7: *Support

Sandia National Laboratories

REVIC
Definitions and Assumptions

- Productivity Units
  - Delivered Source Lines of Instructions
  - DSLIs do not include comment lines; SLOC is another term
  - Person Day - 8 hours
  - Person Month - 152 person hours
  - Loaded Person Hour Cost - $73
- Other Assumptions/Comments
  - Applies to WBS Activities shown on the next slide
  - Assumes Requirement Specification from Phase 1 is stable
  - REVIC equations produce more person month effort & longer schedule than COCOMO

Sandia National Laboratories

REVIC
Work Breakdown Structure (WBS)

Management	System Engineering	Design	Programming	Implementation	Test & Evaluation	Maintenance
Cost/Sched Mgmt	SW Requirements development, validation, SRR, tools, updates	Prel Design, V&V, PDR, updates, tools	Code & Unit Test	Product test plans, procedures, test reports	Installation plans, activities, test reports	Software update
Contract Mgmt	SW Config Mgmt Pgm Supp Lib	Detailed Design, V&V, CDR, updates, tools	Integration Test	Acceptance test plans, procedures, test reports	Conversion plans, activities, programs, data base documents, test reports	Corrective maintenance
Subcontract Mgmt	Acceptance Plan	Prototype code & unit test		Test support labs, tools, data	Training for operation & support	Perfective maintenance
Customer Interface	Quality Engr, Standards Eval	Prototype integration				Adaptive maintenance
Branch Office Mgmt	Studies					Data base administration
Mgmt Rev & Audit						

Sandia National Laboratories

REVIC Equations

Ada Mode

- Person Months of Effort (Nominal) = $PM_n = 6.8000 * (KDSI)^{0.94}$
- Person Months of Effort (Adjusted) = $PM_a = PM_n$ * Multiplier (see next slide)
- Time to Develop = TDEV = $4.376 * (PM_n)^{0.32}$

Organic Mode

- Person Months of Effort (Nominal) = $PM_n = 3.4644 * (KDSI)^{1.05}$
- Time to Develop = TDEV = $3.650 * (PM_n)^{0.38}$

Semidetached Mode

- Person Months of Effort (Nominal) = $PM_n = 3.9700 * (KDSI)^{1.12}$
- Time to Develop = TDEV = $3.800 * (PM_n)^{0.35}$

Embedded Mode

- Person Months of Effort (Nominal) = $PM_n = 3.3120 * (KDSI)^{1.20}$
- Time to Develop = TDEV = $4.376 * (PM_n)^{0.32}$

Person Months Per Year of Maintenance

- $PM_M = PM_n$ * ACT * K_m : ACT = % source lines changed; K_m = Multiplier

REVIC
Input Parameters

Three Model Control Parameters

- Estimated Delivered Source Lines: Low, Expected, and High
- Project level of difficulty mode: Organic, Semidetached, Embedded, Ada
- Average per hour labor cost: Includes rates, benefits, overhead, fee

Nineteen Multiplier Elements

Personnel

- Analyst Capability Experience (ACAP)
- Programmer Capability Experience (PCAP)
- Applications Experience (AEXP)
- Virtual Machine Experience (VEXP)
- Programming Language Experience (LEXP)

Product

- Required Reliability (RELY)
- Data Base Size (DATA)
- Complexity of Code (CPLX)
- Written for Reuse (RUSE)

Project

- Modern Programming Practices (MODP)
- Use of Software Tools (TOOL)
- Security Requirements (SECU)
- Management Reserve for Risk (RISK)
- Required Schedule (SCED)

Environment

- Execution Time Constraints (TIME)
- Main Memory Constraints (STOR)
- Virtual Machine Volatility (VIRT)
- Computer Turnaround Time (TURN)
- Requirements Volatility (RVOL)

Sandia National Laboratories

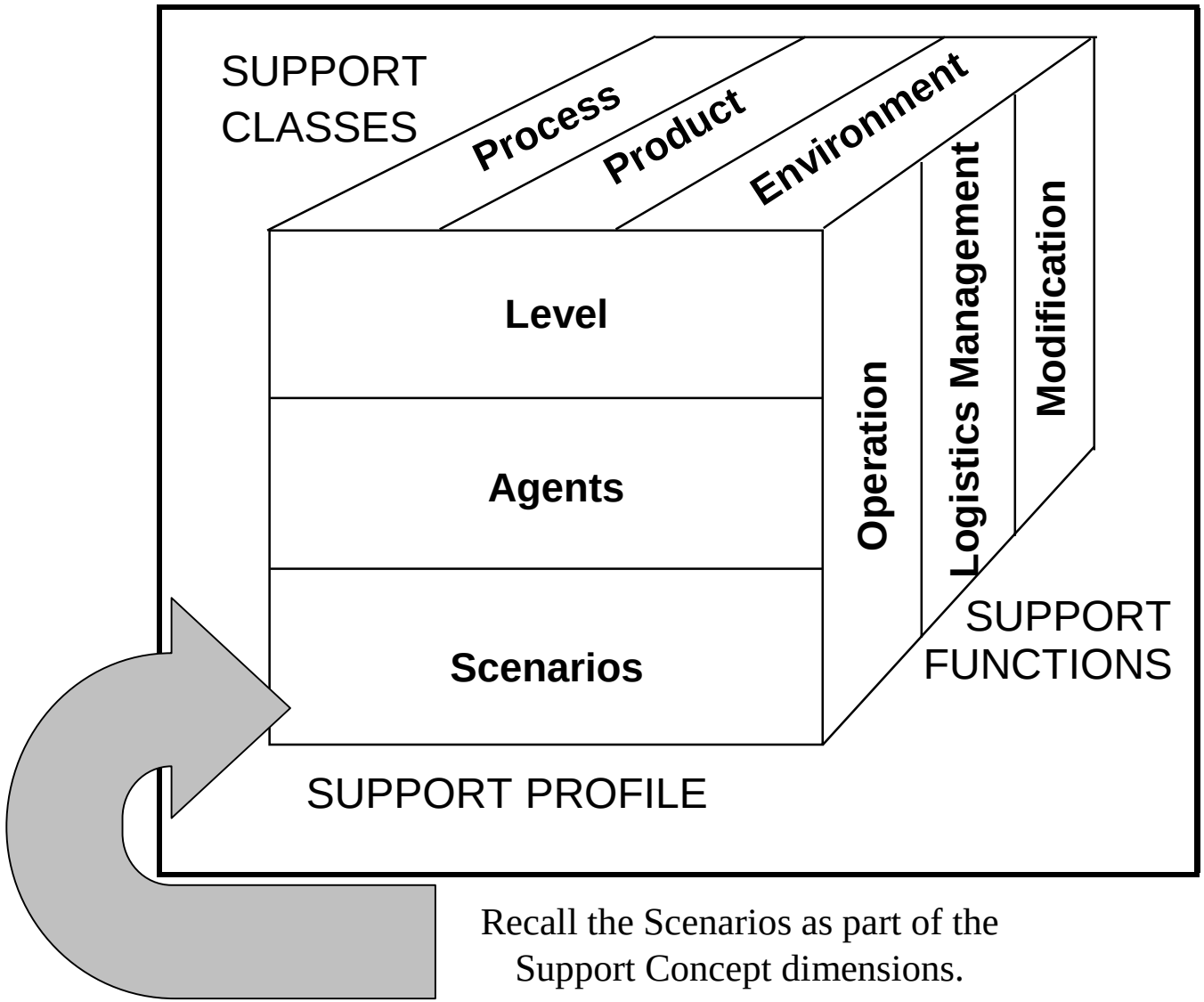
REVIC
Multiplier Table

Rec#	FactName	VL	LO	NM	HI	VH	XH	XX
1	ACAP	1.46	1.19	1.00	0.86	0.71	0.71	0.71
2	PCAP	1.42	1.17	1.00	0.86	0.70	0.70	0.70
3	AEXP	1.29	1.13	1.00	0.91	0.82	0.82	0.82
4	VEXP	1.21	1.10	1.00	0.90	0.90	0.90	0.90
5	LEXP	1.14	1.07	1.00	0.95	0.95	0.95	0.95
6	TIME	1.00	1.00	1.00	1.11	1.30	1.66	1.66
7	STOR	1.00	1.00	1.00	1.06	1.21	1.56	1.56
8	VIRT	0.87	0.87	1.00	1.15	1.30	1.49	1.49
9	TURN	0.79	0.87	1.00	1.07	1.15	1.15	1.15
10	RVOL	0.91	0.91	1.00	1.19	1.38	1.62	1.62
11	RELY	0.75	0.88	1.00	1.15	1.40	1.40	1.40
12	DATA	0.94	0.84	1.00	1.08	1.16	1.16	1.16
13	CPLX	0.70	0.85	1.00	1.15	1.30	1.65	1.65
14	RUSE	1.00	1.00	1.00	1.10	1.30	1.50	1.50
15	MODP	1.24	1.10	1.00	0.91	0.82	0.82	0.82
16	TOOL	1.24	1.10	1.00	0.91	0.83	0.73	0.62
17	SECU	1.00	1.00	1.00	1.10	1.10	1.10	1.10
18	RISK	1.00	1.20	1.00	1.60	1.80	2.00	2.50
19	SCED	1.23	1.08	1.00	1.04	1.10	1.10	1.10

Example Using REVIC

- Example
  - New Ada Application Software for Existing System
- Assumptions
  - Estimated Delivered Source Instructions is 150K
  - As a first approximation for the multiplier, use nominal value = 1.0
  - Set the Annual Maintenance Change Traffic at 15%
- Objectives (use your calculator & equations)
  - 1. Estimate Development Person Months
  - 2. Estimate Months to Develop Software
  - 3. Estimate Yearly Maintenance Person Months
  - 4. Check values against “rules of thumb” using 30-70 and 23K

Software Support Concept
Use of Scenarios for Cost Estimation



Recall the Scenarios as part of the Support Concept dimensions.

Support Scenario Cost Model

- Step 1: Determine Cost-Significant Support Scenarios (CSSSs)
  - Scenarios are part of the Software Support Concept derivation
- Step 2: Determine Frequency of Each CSSS
  - Frequency of Scenarios is determined from the estimate of support load for each of the scenarios and associated scenario task frequencies
- Step 3: Determine the Personnel Effort Associated With Each CSSS
  - Each CSSS is a threaded sequence of tasks that may cross all levels of support and all support functions. The persons involved in the support task activities and level of support involved is determined through supportability analysis tasks and is part of the software support concept.

Support Scenario Cost Model

- Step 4: Determine Stratified Software Support Cost Estimates
  - Total = sum of effort/cost for all CSSSs
  - Infrastructure Total = sum of all infrastructure thread fragments across all CSSSs
  - Modification Total = sum of all modification activity thread fragments across all CSSSs
  - Field/Operational Total, Logistics Management Total, COTS, and so forth
- Step 5: Conduct Steps 1-4 as Integrated Part of Supportability Analysis
  - Integrate software support cost approach with ILS activities and decision process.
  - Use cost model and supportability analysis as through-life methods

Software Support Cost Extended Example: Objectives

Provide for the TCCCS Iris System

- Heuristic Analysis of Support Cost for Software Maintenance/Modification
  - Include perspective of total Support Cost
- Include Stratifications
  - Low, Medium, High Number of Change Requests
  - Corrective and Perfective/Adaptive Maintenance Costs
  - Yearly Cost Estimates
    - Variations from year to year
- Assumptions for Analysis Results
  - Identify basic assumptions

NOTE: Example is real and used the Activity Based Model & RAMSS to compute support cost. Modification scenarios for L, M,H level of change requests were used. Results are proprietary.

Sandia National Laboratories

Software Reliability

Software Reliability

- Terminology
- System and Software Reliability Relationships
  - Design & Operational Reliability
  - Defect class, severity, CMM levels
  - Example methods and techniques: FMECA, FTA, FRACAS
- Software HW/SW Reliability Integration
  - Block diagram
  - Series and parallel components
  - Time issues and other differences
- Software Reliability Models
  - Musa Exponential and Logarithmic models and exercise

See session written notes about System/Software Reliability

What is Software Reliability? Terminology

- System Reliability:
  - the probability that a system, including all hardware and software subsystems, will perform a required task or mission for a specified time in a specified operational environment
- Software Reliability:
  - the probability that software will not cause the failure of a system for a specified time under specified environment conditions
  - a set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time
- Failure:
  - an event in which a software component does not perform a required function within its specified limits
- Fault/Defect:
  - an accidental condition that causes a software functional unit to fail to perform its required function

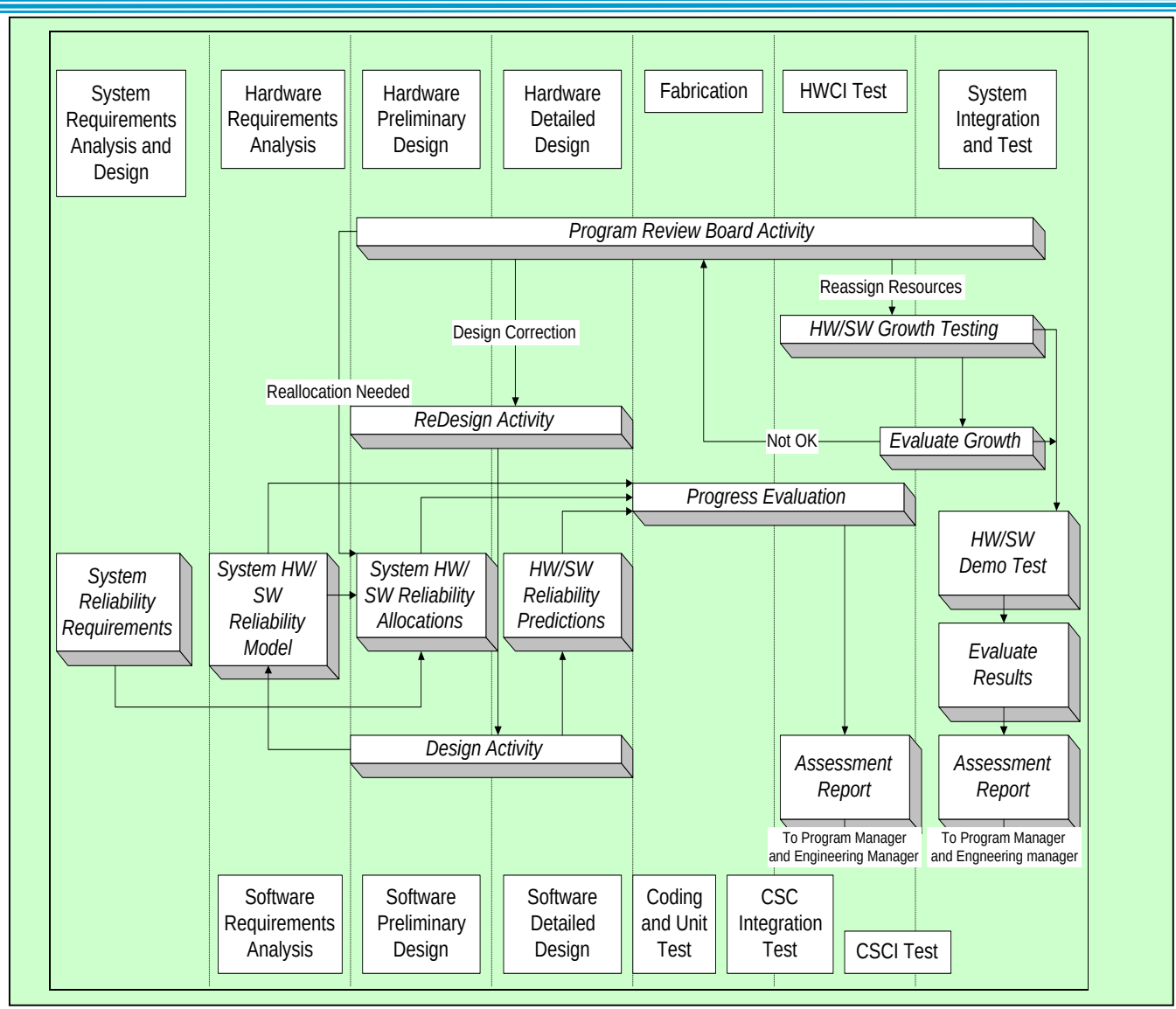
What is Software Reliability?
Terminology

- Error:
  - a human action or lack of action that results in the inclusion of a fault in a software product or the way it is used.
  - the variance between expected and actual results.
- Operational Profile
  - the set of functions a computer program is required to perform - further broken down by input data when it affects execution - along with the probabilities of occurrence
- Failure Intensity:
  - the number of failures occurring in a given time period (e.g., 1000 CPU hours, 1000 calendar hours)
- Fault Density:
  - number of faults per unit; unit might be source lines of code, function points, components, etc.

How is Software Reliability Used?

- Ensure product reliability meets user needs
  - estimate/predict software and integrated system reliability
- Improve time to market for products
  - detect and prevent propagation of development/support defects
  - improve test process and reduce extraneous testing time
- Reduce product cost
  - reduce defects, time to develop, and corrective maintenance
  - improve productivity
- Improve customer satisfaction
  - reduce defects, target specific high priority test areas
- Reduce/mitigate risks
  - target specific high risk functional areas - safety, security
  - reduce likelihood of defects being delivered to the customer

System Reliability Tasks



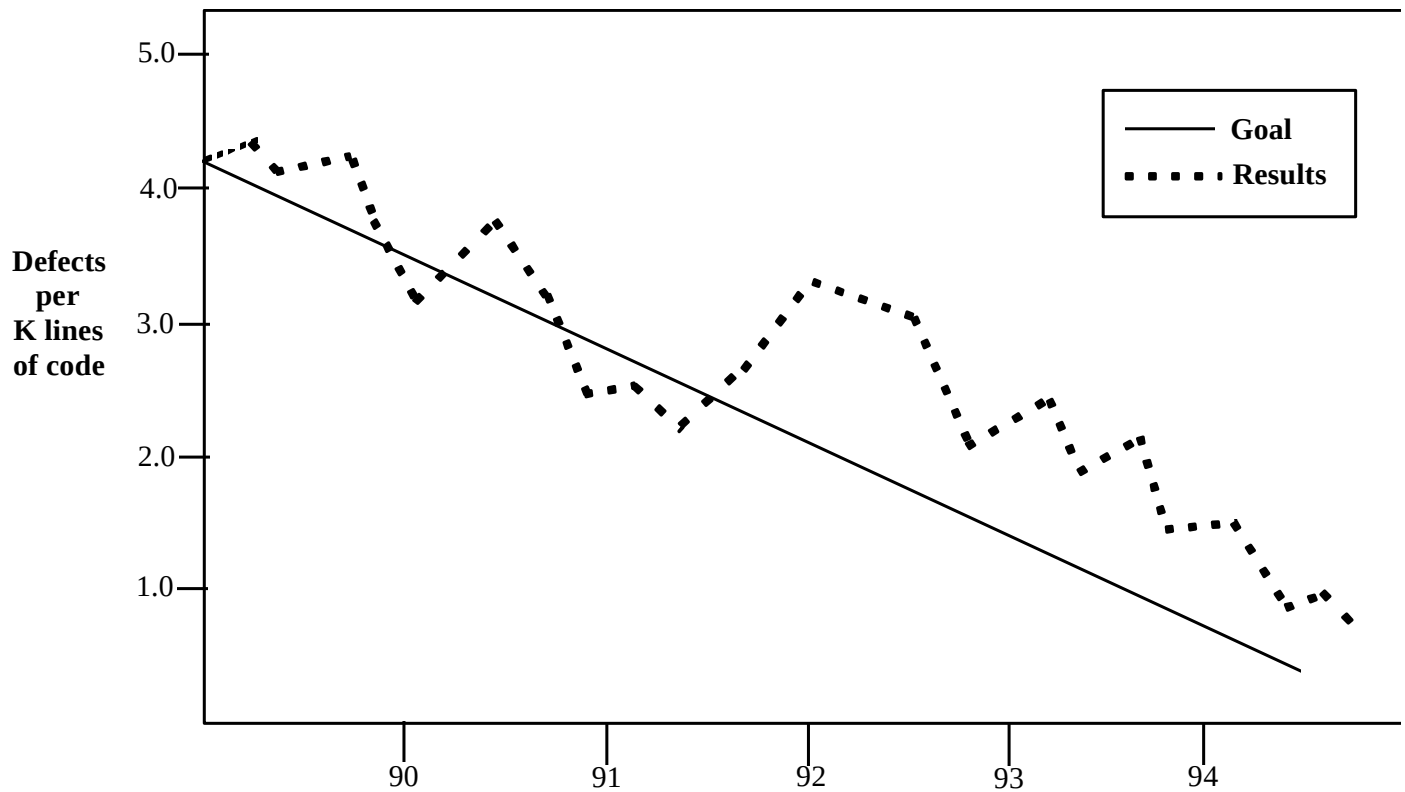
Sandia National Laboratories

Design Reliability

- Improve Software Design Characteristics
  - the set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time
- Implement System and Software Engineering Practices
  - life cycle activity defect reduction and prevention
    - requirements, design, implementation, test, operation, support
  - process improvement to reduce likelihood of product defects
    - SEI SW-CMM, Software Capability Maturity Model
    - ISO SPICE, Software Process Improvement Capability Determination
    - Practice standards and guidelines: Software Reliability Plan and Case
    - Software Operational Reliability Engineering Program
- Track Measurement Data
  - inspection defect data
  - defect density
  - percentage defect removal efficiency for each life cycle phase

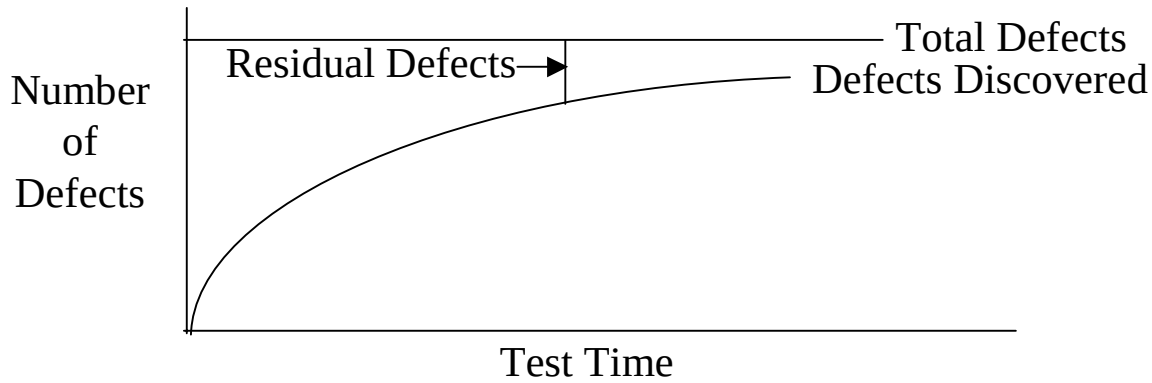
Defect Density by Release

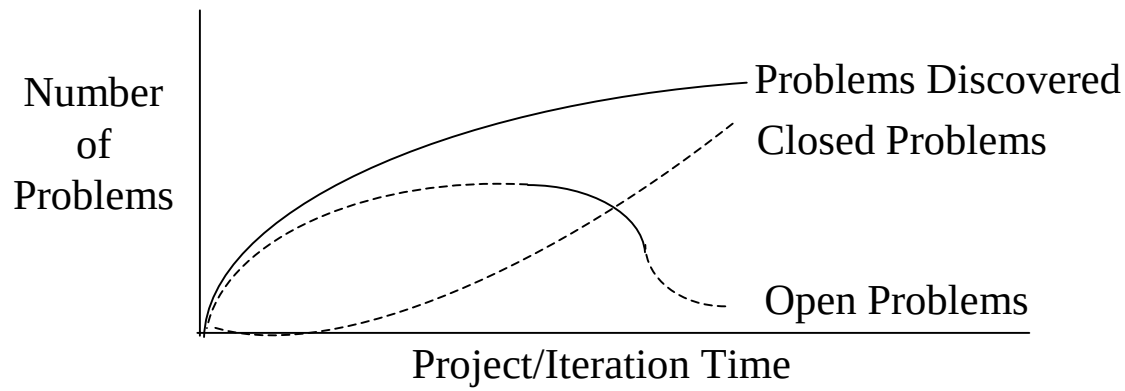
- Defects Over Time - Defect Tracking
  - Example: Tracking Defect Density by Release over Time



Tracking Defects and Problems

- Defects Over Time - Defect Tracking
  - Example: Cumulative Defects





Delivered Defects

Defects Over Time - Defect Tracking

- Example: Tracking Delivered Defects by Maturity Level [JONES]

SEL CMM Level	RTCA DO178B Level	Injected Defects/KSLOC	Delivered Defects/KSLOC
5	A	7.8	0.39
4	B	15.6	1.09
3	C	31.2	2.1
2	D	62.4	3.4
1	E	124.8	5.8

- Example: Defect Removal Efficiency by Life Cycle Activity

Defect Removal Activity	Defect Removal Efficiency
Informal design reviews	25% to 40%
Formal design inspections	45% to 65%
Informal code reviews	20% to 35%
Formal code inspections	45% to 70%
Unit test	15% to 50%
New function test	20% to 35%
Regression test	15% to 30%
Integration test	25% to 40%
System test	25% to 55%
Low-volume Beta test (< 10 clients)	25% to 40%
High-volume Beta test (> 1000 clients)	60% to 85%

All Phases Total:
Military Average: 0.96
System Software: 0.94
Commercial Software: 0.90
MIS: 0.73

Operational Reliability

- Improve Software Operational Characteristics
  - the probability that software will not cause the failure of a system for a specified time under specified environment conditions
- Implement Software Operational Reliability Engineering Program
  - Tailor program to application domain and organization
  - Reliability-based test architecture: operational profiles
  - Reliability-based measurement: defect collection, analysis during test, operation, and support; failure rates; confidence limits
  - Reliability-base risk decision system: when to ship, risks
- Apply Software Reliability Methods
  - System reliability allocation and prediction
  - Software reliability estimation (data) and prediction (models)
  - FMECA, FRACA, FTA

Orthogonal Defect Classification

Classification type	Description
Detection method	What activity detected the defect? Each detection method must be orthogonal in that there is no overlap between them. Some examples include: • Reviews • Inspections • Audits • Internal test • External test • External use
Types	This is directly related to what is repaired in the software. These types include but are not limited to: • Requirements defects • Design defects • Coding defects • Defects due to a corrective action Ideally, the development organization should tailor this list to those most commonly related to the defects on a specific application.
Trigger	This is an orthogonal list of activities that caused the defect to be discovered or observed. Some examples are: • Installation • Startup • Normal operations • Change in configuration • Corrective action • Change in input domain • Error handling • Environmental influences such as loss of power or communication • Change in customer requirements • Operator error
Source	The place in the source code where the defect exists or existed.

Sandia National Laboratories

Software Failure
Example Severity Categories

Category	Severity	Description
1	System Abort	A software/firmware problem that results in a system abort or crash
2	System Degraded No Workaround	A software/firmware problem that severely degrades the system and no alternative work-around exists: restarts not acceptable
3	System Degraded Workaround	A software/firmware problem that severely degrades the system and an alternative work around exists: process can continue with more operator action: restarts not acceptable
4	System Not Degraded	A software/firmware problem that does not severely degrade the system or any essential function: restart acceptable
5	Minor	All other problems or non-functional faults

Example SEI CMM Level Defect and Effort Data Relationships

Maturity Level	Calendar Months	Effort in Person Months	Defects Found	Defects Shipped	Total $ Median Case
I	29.8	593.5	1348	61	$5,440K
II	18.5	143.0	328	12	$1,311K
III	15.2	79.5	182	7	$ 728K
IV	12.5	42.8	97	5	$ 392K
V	9.0	16.0	37	1	$ 146K

Krasner Consulting, 1991, typical 200 kncsloc project

Example Methods & Technique

I. Analysis Techniques
change impact analysis
common cause failure analysis
formal scenario analysis
FRACAS
Petri nets
reliability block diagrams
reliability estimation modeling
response time, memory, constraint analysis
FMECA
FTA
sneak circuit analysis

II. Design Techniques
block recovery
degraded mode operations
defensive programming
diversity
error detection/correction
fault tolerant design
information hiding
reliability allocation

III. Verification Techniques
boundary value analysis
cleanroom
equivalence class partitioning
formal code inspections (Fagan)
functional testing
interface testing
peer reviews
performance testing
probabilistic testing
regression testing
reliability growth testing
root cause analysis
stress testing
testability analysis, fault injection, failure assertion
usability testing

Software FTA/FMECA

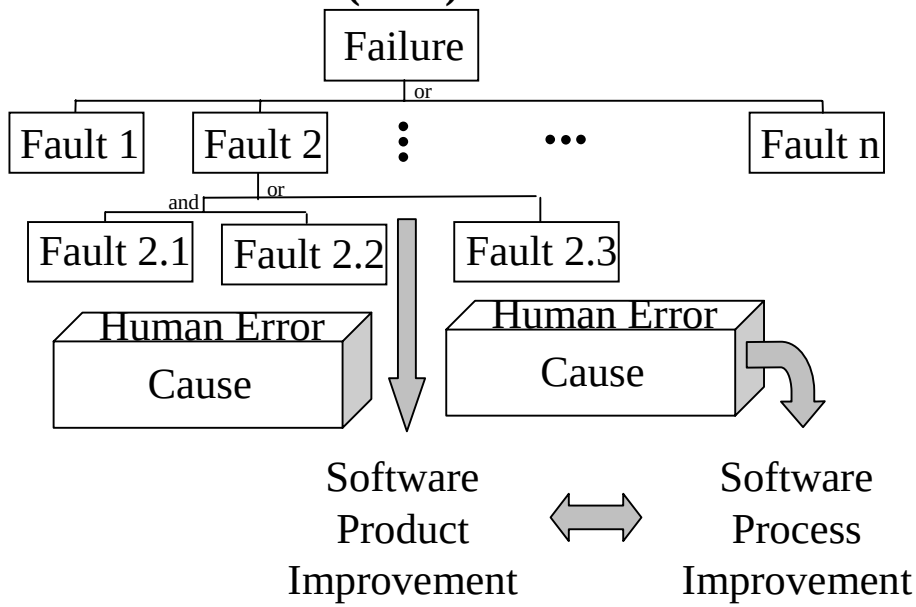
- FMECA
  - *Proactive approach* used for determining the *potential failure modes* of a system/equipment (including software), all likely ways in which a component or equipment can fail, causes for each failure mode, and effects/criticality of each failure mode.
- FTA
  - An extension of the FMECA activity in that the identified potential failure modes are analyzed in terms of what potential software faults (single point of failure) or multiple faults (multiple points of failure) might result in the potential failure mode
- FRACAS
  - *Reactive* approach used for determining the *real failure modes* of the system/equipment.

Software FTA/FMECA

Fault Type

Logic
Timing
Interface
Data
Documentation

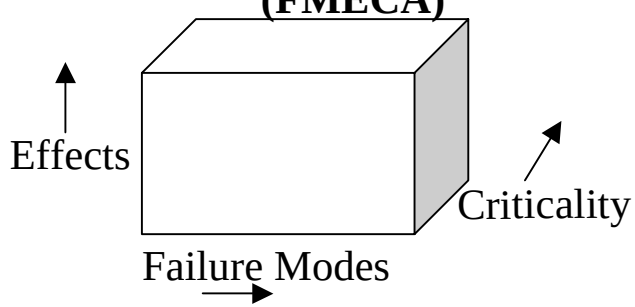
Fault Tree Analysis (FTA)



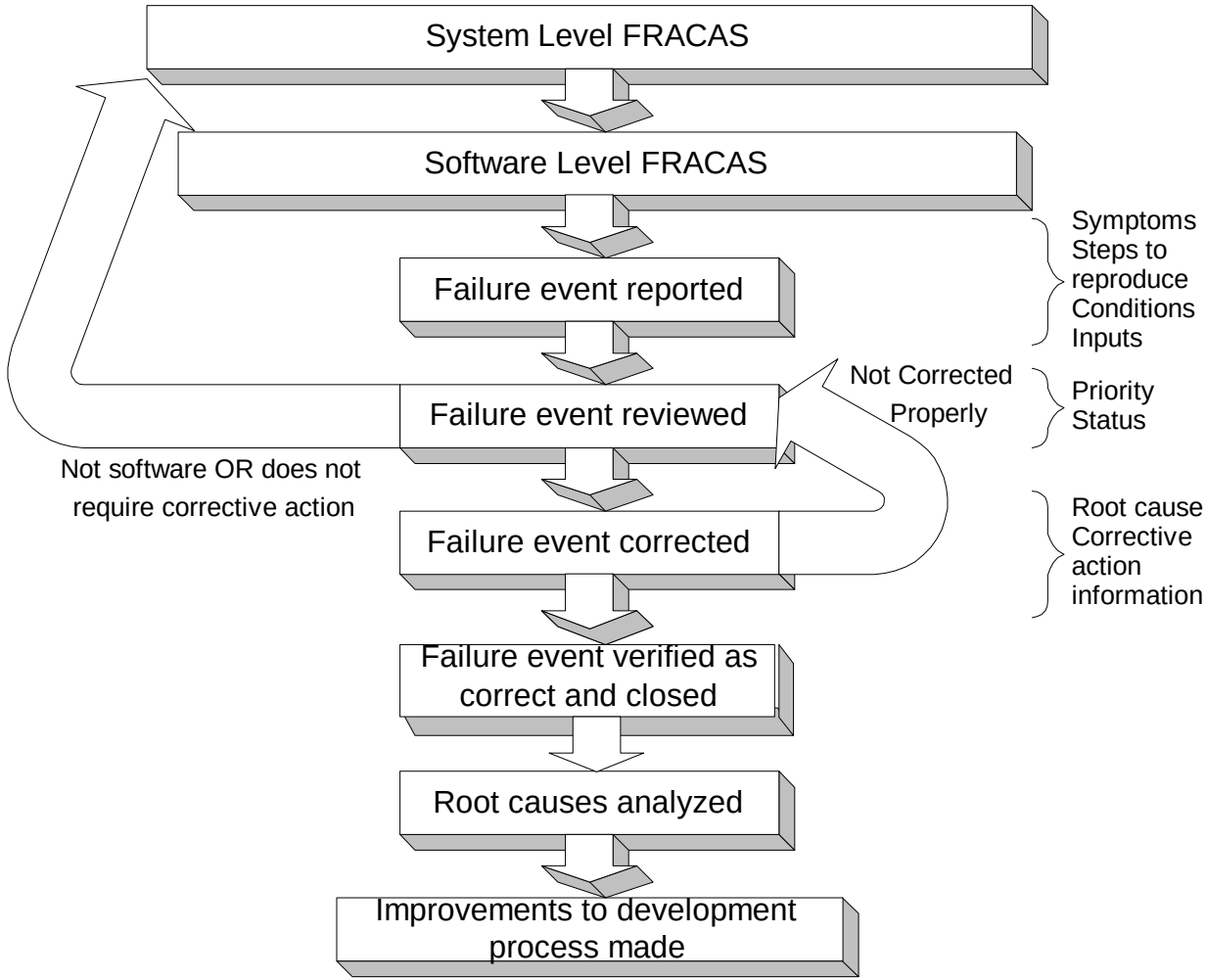
Failure Criticality

Category	Severity	Description
1	System Abort	A software/firmware problem that results in a system abort or crash
2	System Degraded No Workaround	A software/firmware problem that severely degrades the system and no alternative work-around exists: restarts not acceptable
3	System Degraded Workaround	A software/firmware problem that severely degrades the system and an alternative work around exists: process can continue with more operator action: restarts not acceptable
4	System Not Degraded	A software/firmware problem that does not severely degrade the system or any essential function: restart acceptable
5	Minor	All other problems or non-functional faults

Failure Modes and Effects Criticality Analysis (FMECA)



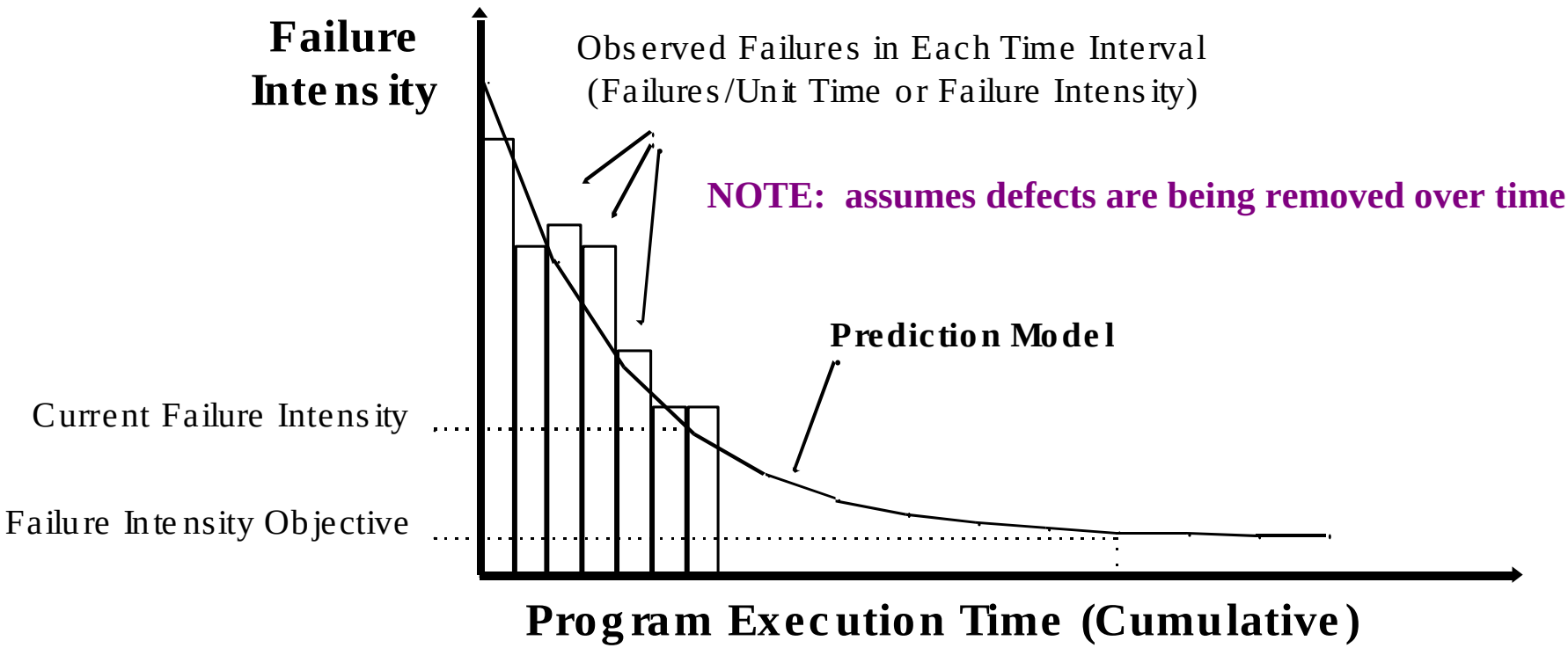
System/Software FRACAS



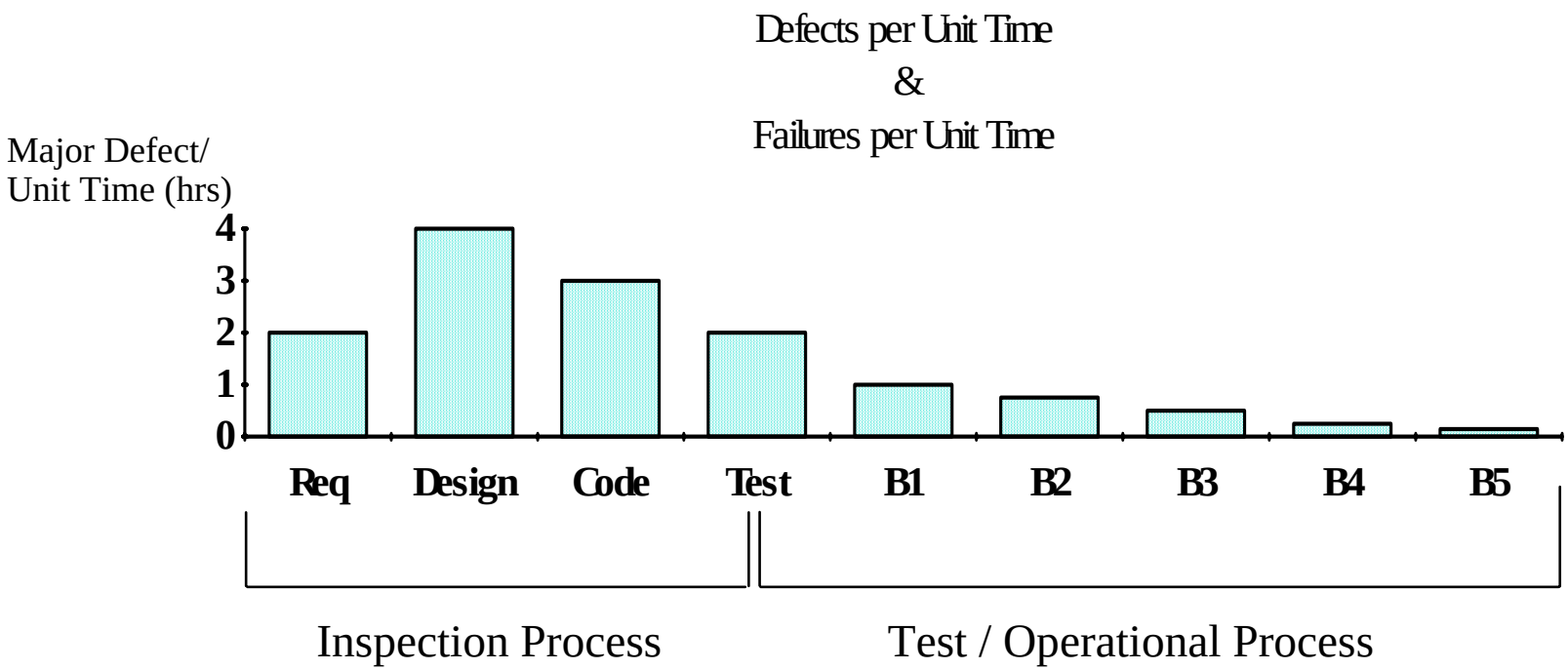
FRACAS - Failure Reporting Analysis and Corrective Action System

Failure Intensity vs Execution Time

- Failure Intensity
  - The number of failures occurring in a given time period
  - Example: 1 failure per 1000 operational hours



Defects & Failures per Unit Time Across the Life Cycle

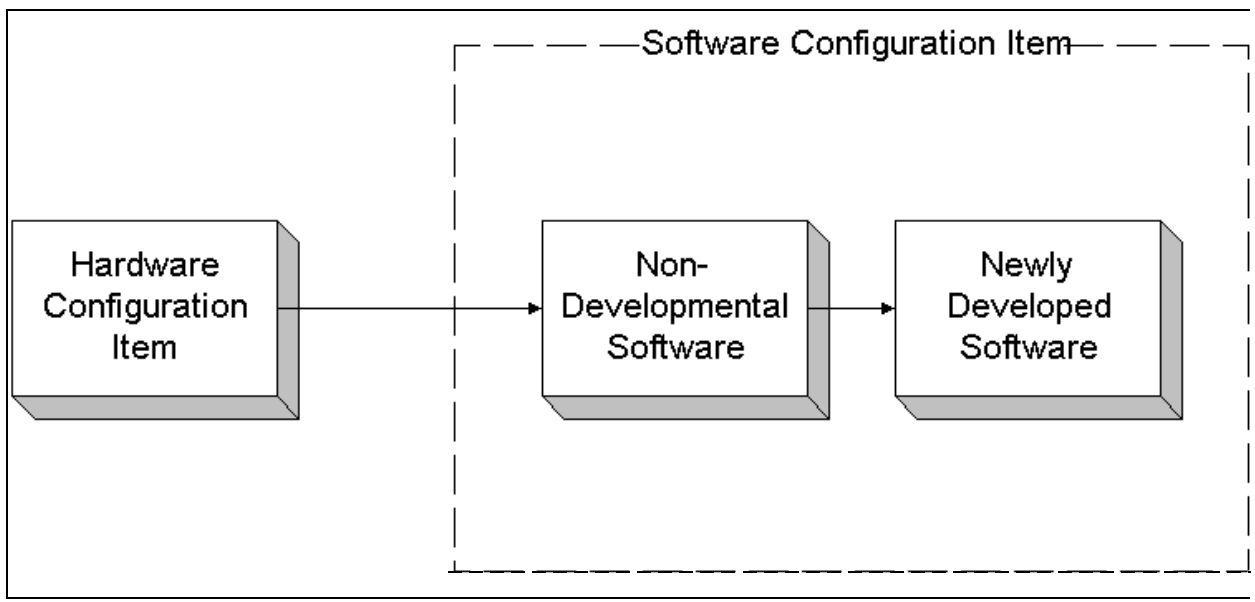


System/Software Reliability
Integrating SW and HW

- Step 1: Block Diagram
  - Divide system into block components for reliability analysis
- Step 2: Allocation
  - Allocate system reliability objectives to components by operational scenarios
  - Allocate to HW/SW components using parallel/series models
- Step 3: Predication
  - Conduct trade-offs with HW/SW to determine best approach to meeting reliability allocations
  - Select/develop HW/SW components to satisfy reliability allocations; use fitted data models for estimation and reliability growth models for prediction

Combining HW/SW Components to Compute Reliability

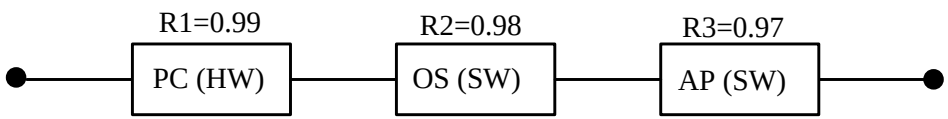
- Computational Component
  - Component with both hardware and software parts of which the software part may be composed of non-developmental and newly developed parts
  - Component parts are considered to be serial elements for reliability computation purposes; if any part fails, the component fails
  - More complicated versions can be constructed



Combining HW/SW Components to Compute Reliability: "AND" "OR"

- "AND" Configuration
  - "AND" part functions only when ALL components are functioning: SERIAL
  - R = R1*R2*R3*…*Rk
  - Example

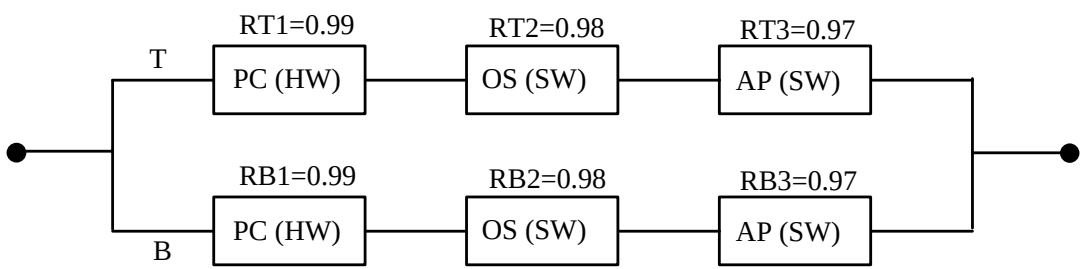
"AND" Configuration



R = R1*R2*R3 = 0.99*0.98*0.97 = 0.94

- "OR" Configuration
  - "OR" part functions when any of the components are functioning
  - R = 1-F = 1-F1*F2*F3*…*Fk
    = 1 - (1-R1)*(1-R2)…(1-Rk)
  - Example

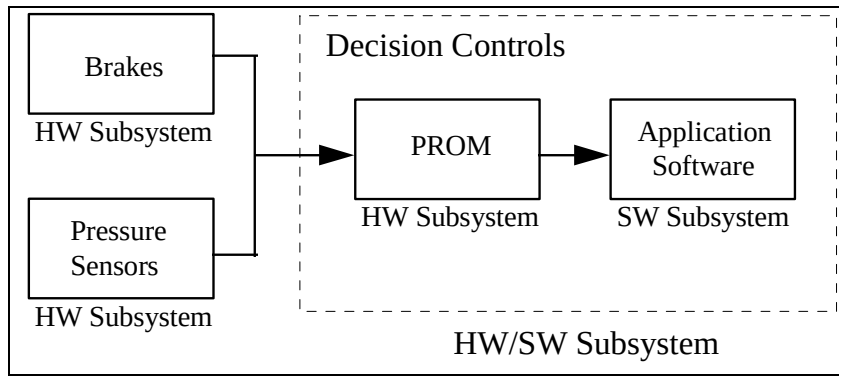
"OR" Configuration



R = [1- FT*FB] = [1 - (1-RT)(1-RB)] =[1-(1-.94)(1-.94)] = [1-0.06*0.06] = [1-0.0036] = 0.996

Simple Example/Exercise

- Given Block Diagram for Automobile Anti-lock Braking System (ABS)
- Suppose the following reliability values are known;
  - R (brakes) = R_b = 0.94
  - R (pressure sensors) = R_s = 0.96
  - R (PROM) = R_p = 0.91
  - R (application software) = R_a = 0.90
- Compute System R



Software Reliability Models

- History
  - 1970s: Jelinski-Morana, Shooman, Schick& Wolverton models
  - 1973: John Musa, Bell Laboratories, began his work
  - 1980s: Glory years of model research: Littlewood, Goel, etc
  - 1990s: The hard part - trying to apply models & get results
- Illustrate Musa's Models
  - Musa: Non-Homogeneous Poisson Process Exponential
    - Effective for System Test use
  - Musa: Non-Homogeneous Poisson Process Logarithmic
    - Effective for Field Operational use
- Modeling Cost [MUSA99]
  - 0.1-0.2% of project cost
  - Includes ALL SRE activity, not just Supportability-specific uses
  - Includes training, data collection, analysis

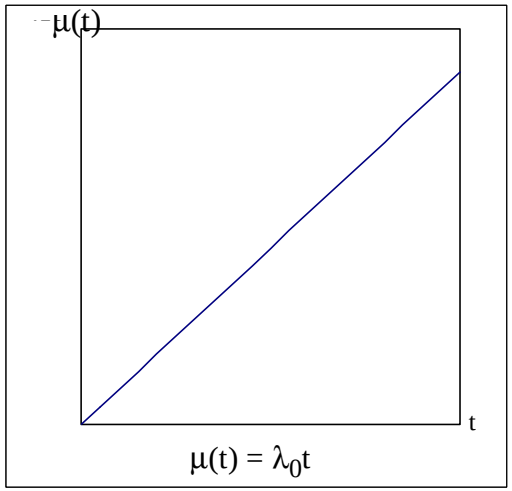
Model Types and Equations

	Exponential	Logarithmic
Failures Experienced (Expected)	$\mu(\tau) = \nu_0 (1 - e^{\frac{-\lambda_0}{\nu_0}\tau})$	$\mu(\tau) = \frac{1}{\theta} \ln(\lambda_0 \theta \tau + 1)$
Present Failure Intensity (Function of Failures)	$\lambda(\mu) = \lambda_0 (1 - \frac{\mu}{\nu_0})$	$\lambda(\mu) = \lambda_0 e^{-\theta\mu}$
Present Failure Intensity (Function of Time)	$\lambda(\tau) = \lambda_0 e^{\frac{-\lambda_0}{\nu_0}\tau}$	$\lambda(\tau) = \frac{\lambda_0}{\lambda_0 \theta \tau + 1}$
Additional Time to Failure Intensity Objective	$\Delta\tau = \frac{\nu_0}{\lambda_0} \ln(\frac{\lambda_P}{\lambda_F})$	$\Delta\tau = \frac{1}{\theta}(\frac{1}{\lambda_F} - \frac{1}{\lambda_P})$
Additional Failures to Intensity Objective	$\Delta\mu = \frac{\nu_0}{\lambda_0}(\lambda_P - \lambda_F)$	$\Delta\mu = \frac{1}{\theta}\lambda\nu(\frac{\lambda_P}{\lambda_F})$
Reliability (No Failures Corrected)	$R(\tau) = e^{-\lambda\tau}$	

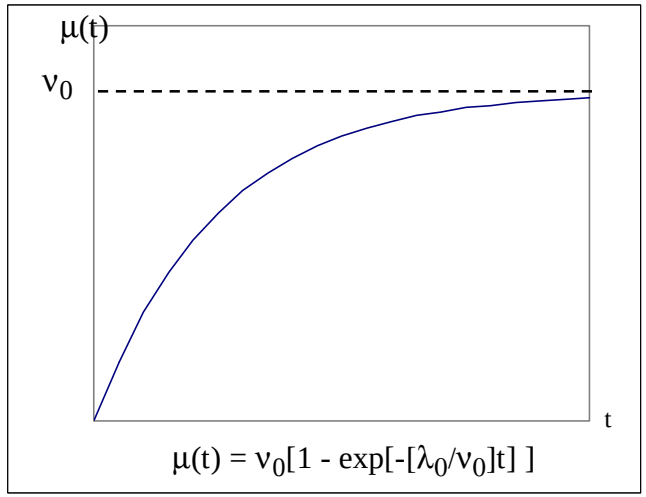
Basic Formula Terms

μ	=	failures experienced (expected)
$\Delta\mu$	=	additional failures (expected)
τ	=	execution time
$\Delta\tau$	=	additional execution time
$\lambda 0$	=	initial failure intensity - (must be estimated)
λ_P	=	present failure intensity
λ_F	=	failure intensity objective
ν_0	=	total failures (expected) - (must be estimated)
θ	=	failure intensity reduction (decay) rate - (must be estimated, nominal values = .02/failure - .05/failure)

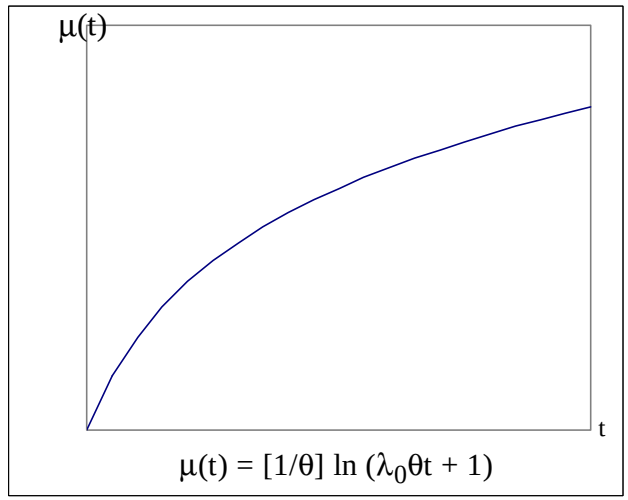
Example Graphs for Models



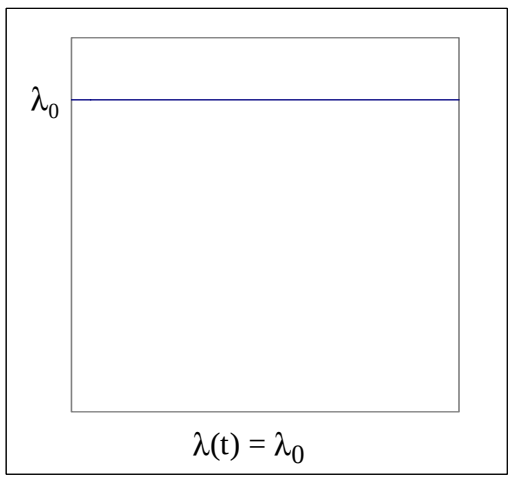
$\mu(t) = \lambda_0 t$



$\mu(t) = \nu_0[1 - \exp[-[\lambda_0/\nu_0]t]]$

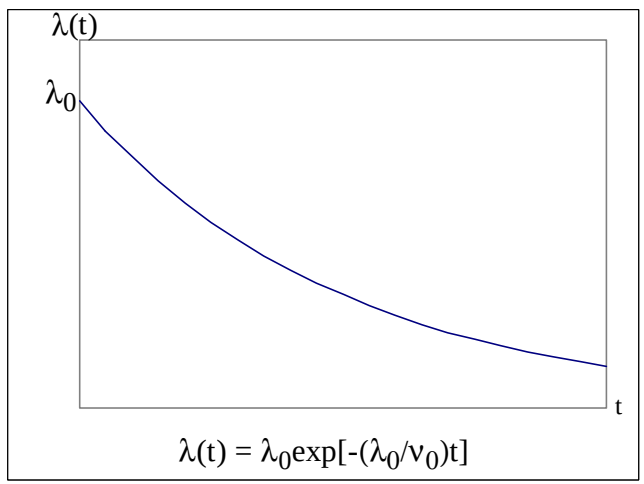


$\mu(t) = [1/\theta] \ln (\lambda_0 \theta t + 1)$



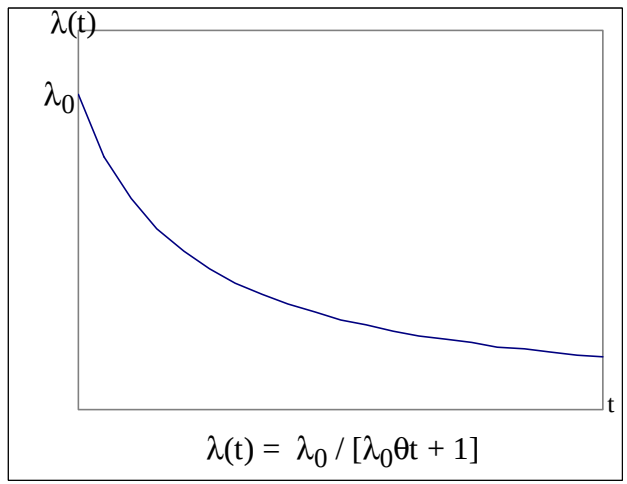
$\lambda(t) = \lambda_0$

Static



$\lambda(t) = \lambda_0 \exp[-(\lambda_0/\nu_0)t]$

Exponential



$\lambda(t) = \lambda_0 / [\lambda_0 \theta t + 1]$

Logarithmic

Computational Exercise

Given that $\nu_0 = 500$ $\lambda_0 = 27/hr$ $\lambda_P = 15/hr$ $\lambda_F = 0.5/hr$ and $t = 10$

Compute (Exponential Model)

$\mu(t) = \nu_0[1 - \exp[-\lambda_0 t / \nu_0]]$ = expected number of current failures experienced at time t

$= 500[1-\exp[-27*10/500]] = 208.62587 \sim 209$ failures

$\lambda(t) = \lambda_0 [1 - \mu/\nu_0]$ = expected failure intensity/rate at the current time t

$= 27 [1 - 209/500] = 15.714$ failures per hour at t=10 hours

$\Delta\mu = [\nu_0/ \lambda_0] [\lambda_P - \lambda_F]$ = expected additional failures to reach failure intensity objective

$= [500/27][15 - 0.5] = 268.51852 \sim 269$ more failures to reach objective of 0.5/hr

$\Delta t = [\nu_0/ \lambda_0] \ln [\lambda_P/\lambda_F]$ = expected additional time to reach failure intensity objective

$= [500/27] \ln[15/0.15] = 62.985137 \sim 63$ hours more testing to reach objective of 0.5/hr

Sandia National Laboratories

Software Logistics Identification

Identification Issues*

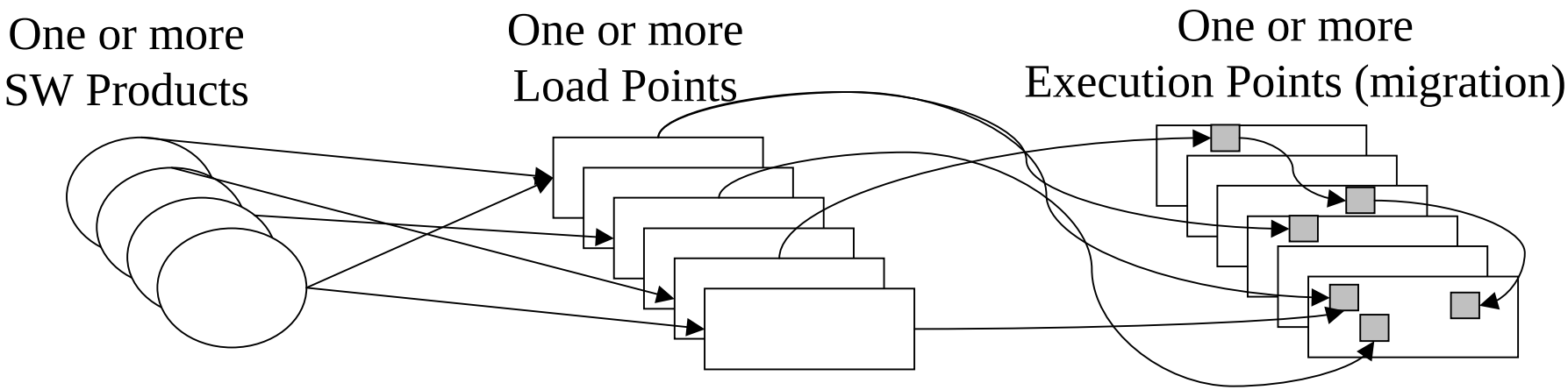
- Software Identification Infrastructure Issues
  - How is the software product identified - Part Number?
  - How is the link of software to its computational hardware processor identified - LSA Control Number (LCN)?
  - How is the link of software to where it is loaded into a computational equipment identified?
  - How are software supportability analyses results identified for purposes of logistics management?
  - What identification is required if the same software product is loaded at several different load points and operates in several different processors?
  - What complications arise if a software product is actually a collection of components that can be separately loaded into multiple processors over time in some arbitrarily random manner?
  - Software Supportability Plan should address all applicable issues

* [Hutch00] Hutchison, Peter B., "Using Def Stan 00-60 LSA Reports To Address Software Supportability," MSc Dissertation in Software Engineering, Oxford University, March 9, 2000.

Software LSAR
Some Minimum ID Requirements

- Part Number
  - Each support significant software product shall be identified by part number and developer/supporter identification (cage code).
- Multiple to Multiple Trace
  - Software to Equipment Trace: it shall be possible to trace each support significant software product to all load points
  - Software to Execution Point Trace: it shall be possible to trace each support significant software product to all points in the system where it is “executed”.
  - Equipment Trace to Software Trace: it shall be possible to trace each system equipment software load point to all support significant software products that are loaded at that load point.
  - Execution Point to Software Trace: it shall be possible to trace each equipment processor/execution point to all support significant software products that can be executed at that point.

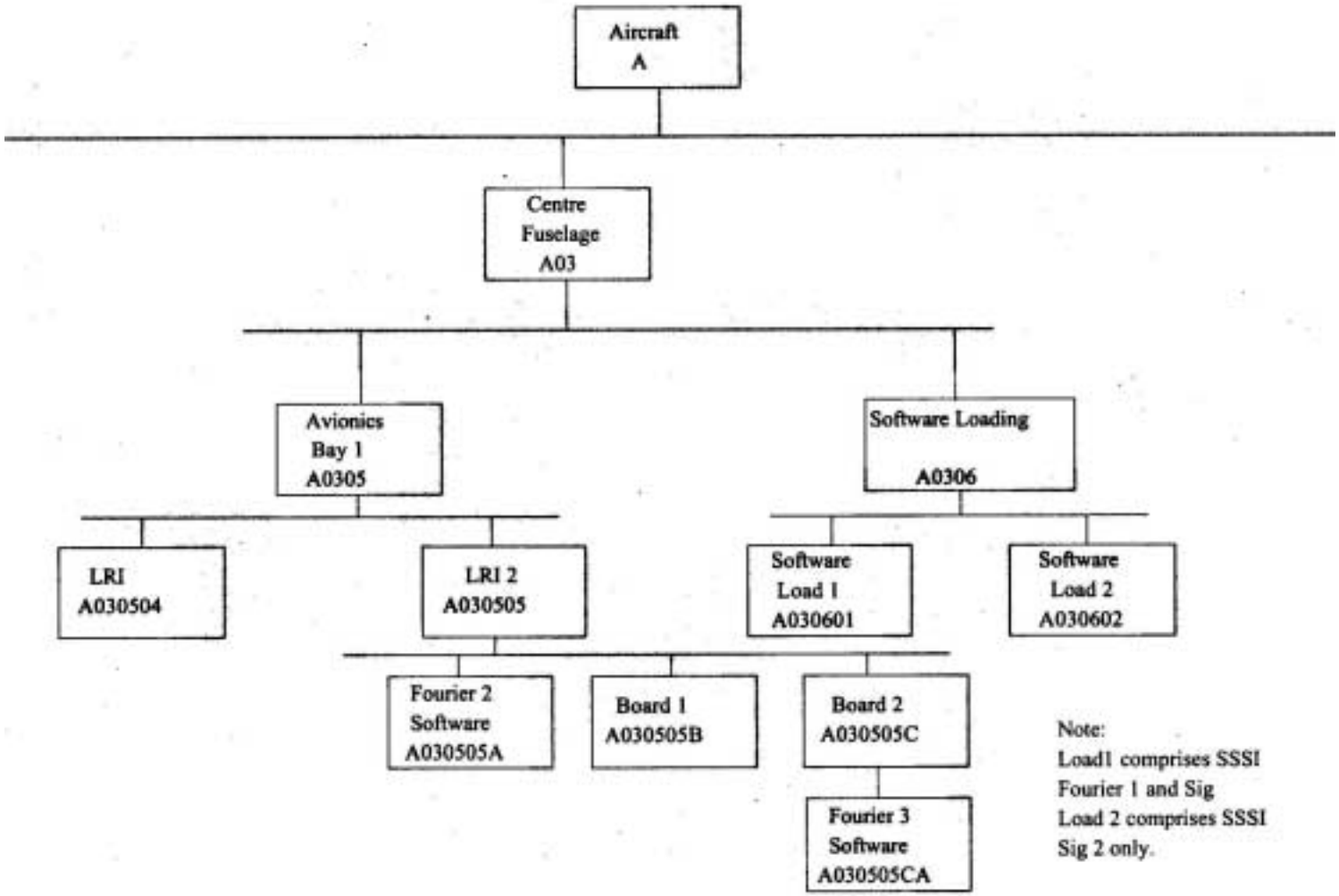
Software Logistic Identification Multiple to Multiple Issue



An Example Physical LCN - 0060

- Two Types of Software
  - Loadable Software
    - Two items of loadable software, Load 1 and Load 2 are delivered using a hand held loading device temporarily connected to a Software Loading Point
  - Resident Software
    - Software element is loaded onto a firmware device, embodied at 2-4th line
    - Two resident software items - Fourier 2 and Fourier 3 have been loaded
- Parent Child Relationship
  - In all cases the software item is a “child” of the hardware maintenance significant item where the physical loading activity takes place. See next slide for illustration
- Questions
  - What if the software is loaded from the load point into a PROM?
  - What if the software is loaded from the PROM into a processor’s RAM for execution that is located on another piece of equipment?

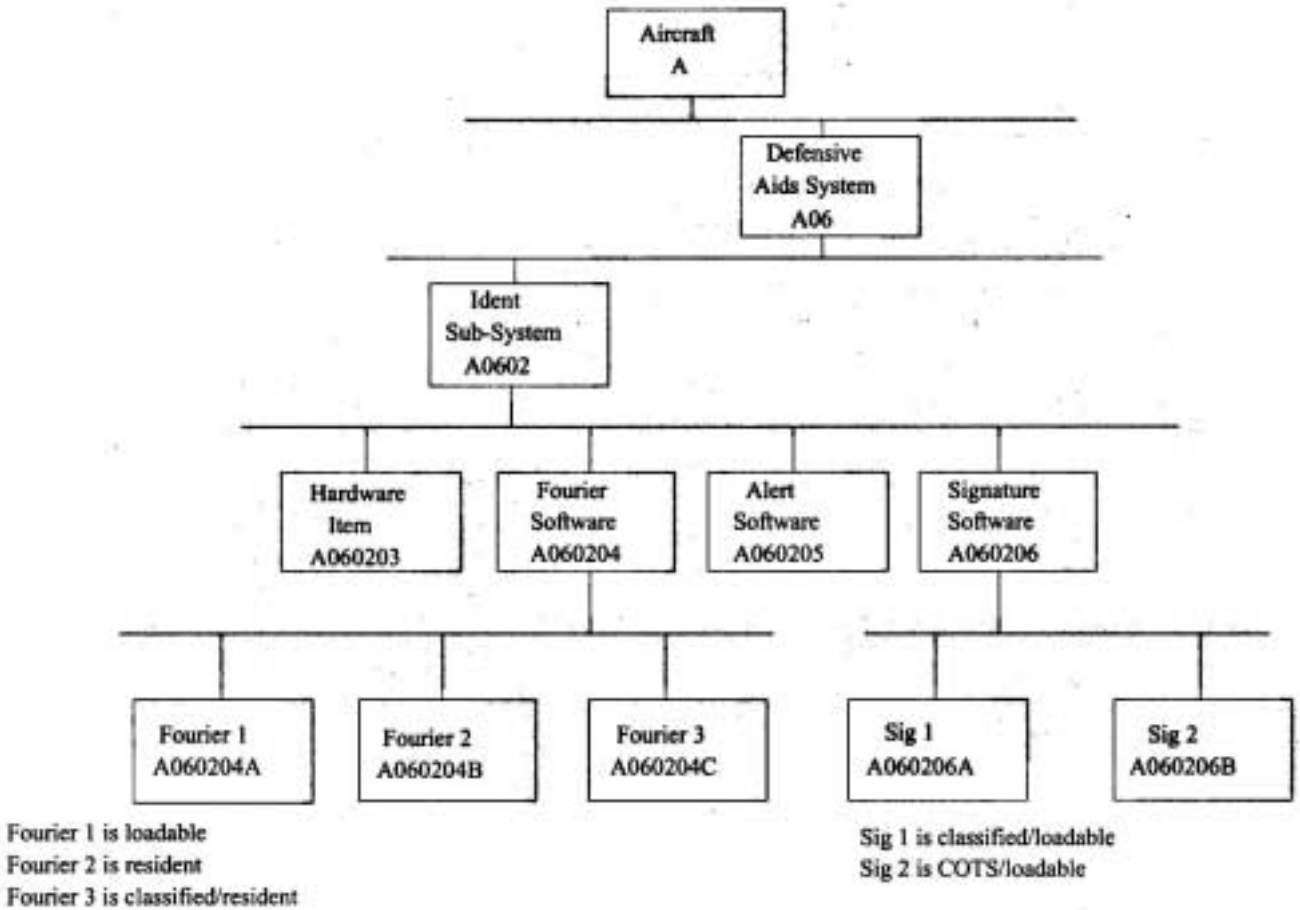
An Example
Physical LCN Structure - 0060



An Example Functional LCN-0060

- Support Significant Software
  - Assign Unique LCN
    - LCN should distinguish between software that is loadable and software that is resident; bespoke and OTS; security significant; ??
  - Previous example Continued
    - Software element is loaded onto a firmware device, embodied at 2-4th line
    - Two resident software items - Fourier 2 and Fourier 3 have been loaded
- Parent Child Relationship
  - In all cases the software item is a “child” of the subsystem within which they operate. See next slide for illustration
- Question
  - How are the Physical and Functional LCN information related?

An Example Functional LCN Structure-0060



Relating Functional & Physical LCN Structures - 0060

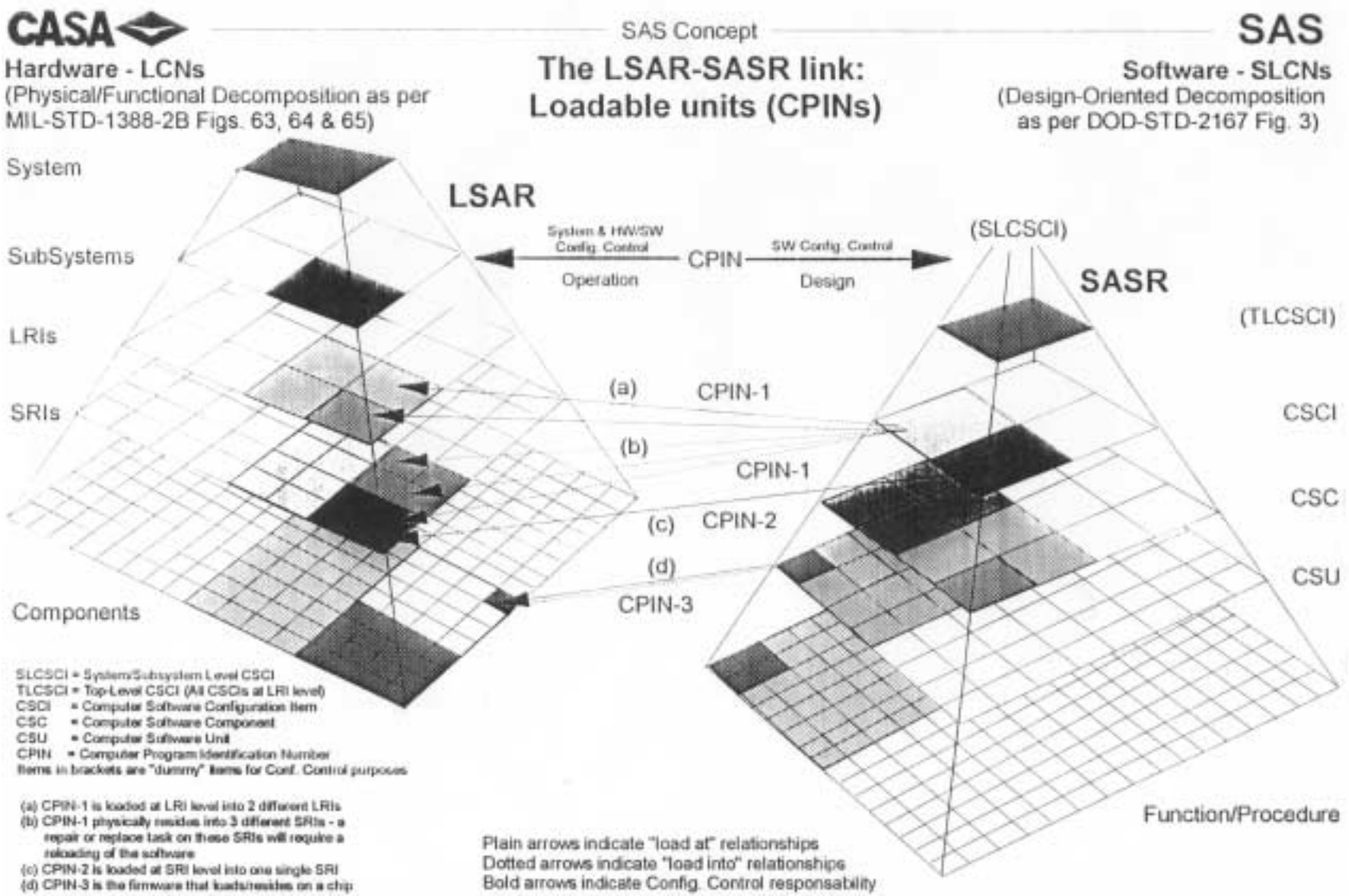
- LCN Group
  - The LCN Group consists of four key elements. The following are definitions of the LCN Group elements and will provide an overview of their functions within an LSAR
    - EIAC (DED 096). The EIAC is an alpha-numeric code, consisting of up to 10 characters, which uniquely identifies the system or end item.
    - LCN (DED 199). The LCN is an alpha-numeric code consisting of up to 18 characters which, when assigned with an associated LCN Type and ALC, identifies an item or function within the physical or functional breakdown of the system or end item.
    - LCN Type (DED 203). LCN Type is a one position alpha code ("P" or "F") indicating whether the LCN is part of a physical or functional structure breakdown.
    - ALC (DED 019). The ALC is a two position numeric code which indicates that an alternative item occupies the same position in the LCN breakdown and, therefore, the same position within the system, equipment or item, but has different logistic considerations.

Relating Functional & Physical LCN Structures - 0060

- XG Data Table
  - Establishes links between significant software support items in the functional domain and the associated executable code in the physical domain.
- Example Partial Table (not from previous example)

EIAC	LCN	ALC	LCN TYPE	LCN	ALC	LCN TYPE
MSL	1A01 (SENSOR UNIT)	00	P	AB01A01 (DETECT FUNCTION)	00	F
MSL	1A02 (SENSOR AMP)	00	P	AB01A02 (SIGNAL BOOST FUNCTION)	00	F
MSL	1B02A (ELECTRONICS UNIT)	00	P	AB01B (PROCESSING FUNCTION)	00	F
MSL	1B02C (FWD BATTERY PACK)	00	P	AB01C01 (DUMMY CROSS-MAPPING FUNCTION 1)	00	F
MSL	1D01A (FIN CONTROL UNIT)	00	P	AB02A (CONTROL FUNCTION)	00	F
MSL	1D01C (REAR BATTERY PACK)	00	P	AB01C02 (DUMMY CROSS-MAPPING FUNCTION 2)	00	F
MSL	1D02 (GUIDANCE FINS)	00	P	AB02C (STEER FUNCTION)	00	F

A Complex Example
Any Thoughts?



Sandia National Laboratories

Off-The-Shelf Technologies

Off-The-Shelf Technologies

- Off-The-Shelf Software
  - Terminology
  - TCCCS/Iris OTS Software Impact
  - Support Concerns
- Acquisition Policy Influence / Pressure
  - US Policy Memos and NATO Software Acquisition Guidance for COTS
  - Software Engineering Institute Component Based System Initiative
  - ISO OTS Standard
- OTS-Based Software Support
  - Through-Life Support Strategy
  - Software Logistics Support Concerns

Terminology

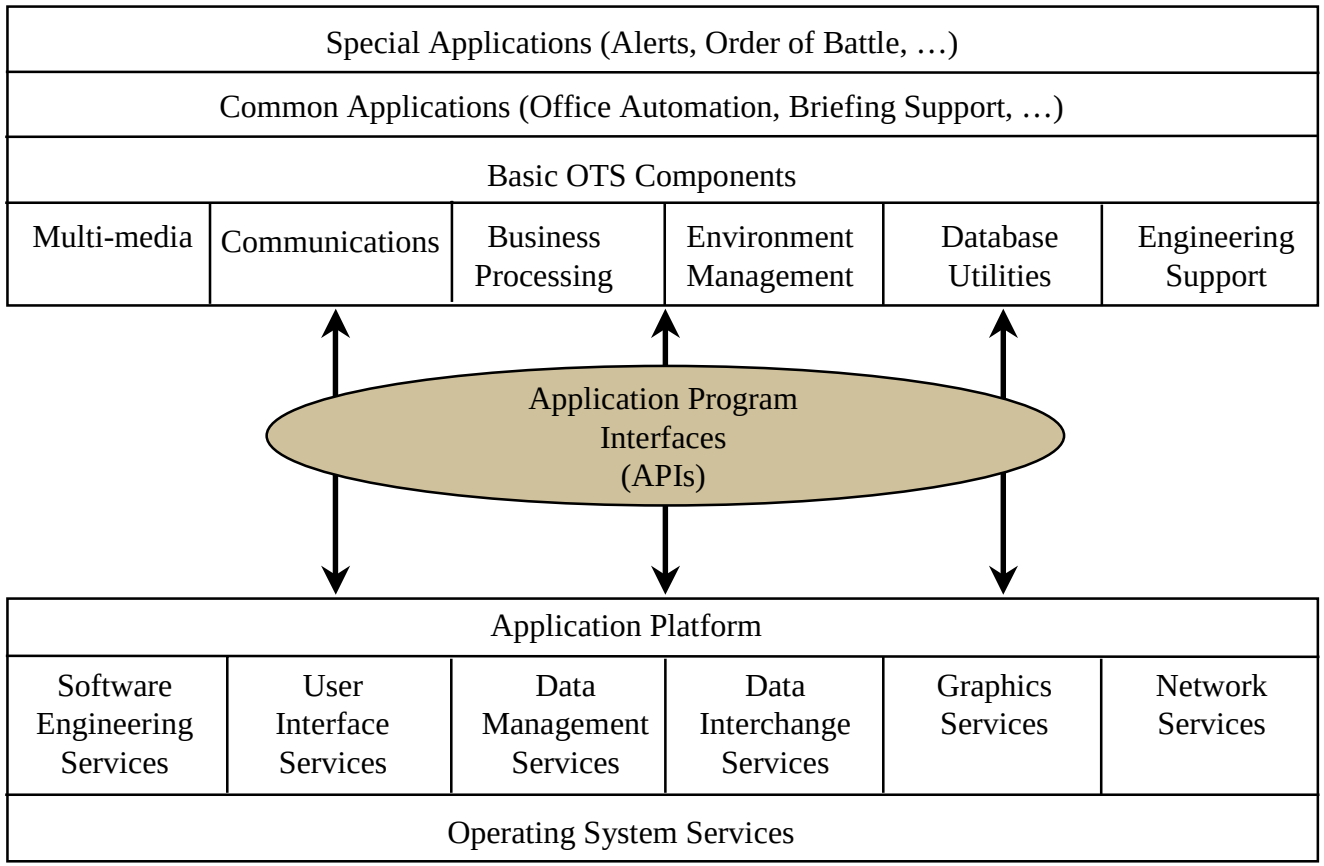
- Commercial-Off-The-Shelf (COTS)
  - pertains to a commercially marketed product which normally is used without modification
  - Note: COTS products are provided in large quantities and at relatively low cost to meet the demands of a wide range of user needs. For COTS software, source code is not provided and maintenance is provided by the vendor under license.
- Government-Off-The-Shelf (GOTS)
  - pertains to a product owned by the government and normally used by others without modification
  - Note: GOTS products are provided to meet specific user needs. The government exercises partial or full ownership and maintenance of the products and may make them available on mutually agreed terms. GOTS application software may be made available with its source code.
- Industry-Off-The-Shelf (IOTS)
  - pertains to a product owned by an industry company that is reused by the company within multiple applications and normally without modification

Sandia National Laboratories

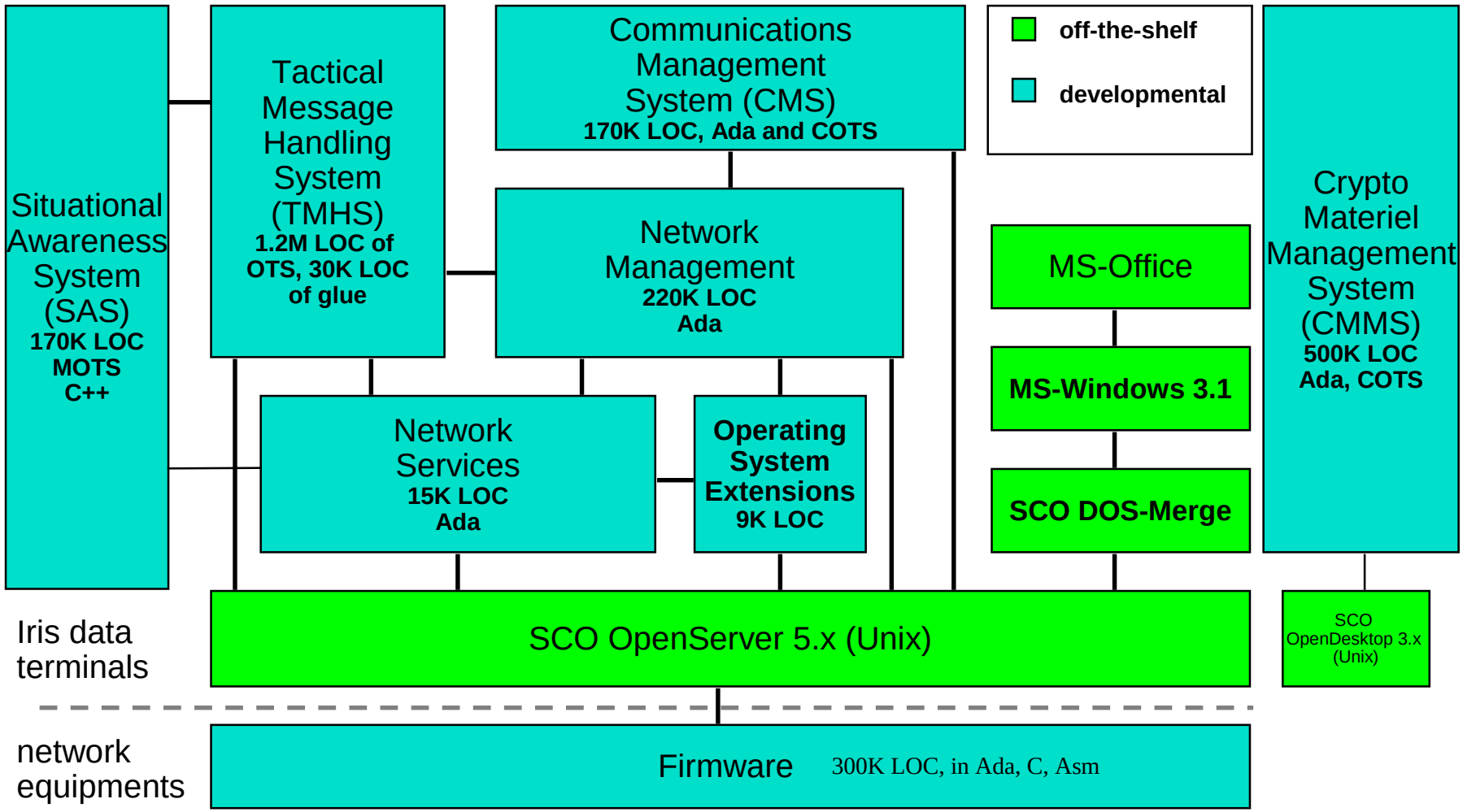
Terminology

COTS software is built by a commercial company to meet a wide range of user needs. As it is built for a large user market it is relatively cheap to procure. Maintenance is carried out by the vendor under license agreement. COTS software is, if properly selected, of general applicability, flexible, reliable and maintained within the commercial environment and represents state-of-the-art.

Basic COTS Architectural Framework



Software Architecture TCCCS/Iris



Software Architecture
TCCCS/Iris COTS Influence

Segment	Custom SW Prog Lang	Key COTS SW//Technology	COTS Influence
HIDS	Ada, C, Asm, Unix Nscripts	MS Office, SCO Unix	Medium
SAS	C++	Rockwell Firmware (PLGR)	Major
TMHS	Ada, C, D, Unix Nscripts	NEXOR MsgWare, CommPower ACPG	Major
CMS	Ada, SQL	Oracle, GemMap	Major
LRCS	Ada, D	MS Office, Time/View 2000, VOACAP, Powerlink, ITU, Coord Nw	Medium
Radios	C	Racal Firmware	Major
SCRA	C++	OS 68 (OSE)	Minor
CMMS	Ada, C, D, SQL	Oracle	Medium

OTS Benefits
What's being Shouted!

- Technology
  - acquisition risk is reduced
  - product availability and maturity is improved
  - product is state-of-the-art
  - external technical expertise and support is available
  - resource requirements may be reduced
  - fast prototyping facilities are typically available
- Cost
  - development and maintenance cost is reduced
  - development and maintenance schedule is shortened
  - resources are reduced for maintenance activities
  - licensing arrangements are centralized so overhead is reduced

OTS Concerns
What's Being Mumbled!

Technology

- functionality may not meet all requirements or may provide undesirable functionality
- integration with bespoke (custom) software and hardware may require special OTS software wrappers
- testing to validate integrated performance may be extensive
- specialty engineering activities to ensure reliability, security, safety requirements may be difficult, and may require external certification (e.g., trusted software certification)
- support activities by the bespoke software developer can be extensive due to embedded OTS updates/upgrades (frequency and/or complexity of upgrade)
  - software integration and retest, distribution of new releases
  - installation, acceptance and training for new releases
- response of the OTS vendor to correction of defects may not satisfy operational needs

Cost

- each OTS software upgrade will take a minimum of 5 person days of support staff time to implement; any operational software upgrade due to COTS will take much more effort to implement
- example: 200 software COTS packages, one upgrade per year, @ 5 pd per upgrade = 1000 pd per year, 4.4 person years per year, ~ $400K per year ==> ***MINIMUM!***

Acquisition Policy Influence US/NATO Policy

- Memorandum for US Military Departments / Agencies on Specifications and Standards, June 29, 1994, William Perry
  - "To meet future needs, the Department of Defense must increase access to commercial state-of-the-art technology and must facilitate the adoption by its suppliers of business processes characteristic of world class suppliers. "
- NATO COTS Software Acquisition Guidelines - 1st Revision, January 12, 1996
  - Software support is reasonably emphasized as an area of risk although the development processes do not seem to always reflect supportability analysis tasks

Acquisition Policy Influence SEI COTS-Based System

- Software Engineering Institute COTS-Based System (CBS) Initiative
  - "Shrinking budgets, the need to reduce time between releases (cycle-time), expanding system requirements, and the growth of the Commercial-Off-The-Shelf (COTS) software marketplace are all driving the shift from custom software to COTS-Based systems in both development and maintenance activities."
  - "In the future, high-quality, adaptable, mission-critical systems will routinely be assembled and evolved from software components."
  - "Program managers and system developers need to become smart consumers and practitioners, and understand the management, business, technical, and organizational implications of a COTS-based approach to system development."
  - "The SEI COTS-Based Systems (CBS) initiative is addressing the challenges of assembling, integrating, and evolving systems from commercially available and other pre-existing components, as well as modifying legacy systems to take advantage of a CBS approach."

Acquisition Policy Influence SEI COTS-Based System

- Why the SEI?
  - Expertise and Capabilities in:
    - acquisition management
    - open systems, CASE environments, and standards
    - product and technology evaluation of emerging technology
    - legacy system evolution
    - design, engineering, and component-integration approaches
    - heterogeneous, distributed, and net-centric computing
    - distributed object technologies such as Java, CORBA, etc
- Additional Information
  - Contact
    - Customer Relations (412-268-6900)
    - customer-relations@sei.cmu.edu for copy of SEI Technical Program, Spring 1998 which includes more information on the CBS initiative
  - Web Address
    - http://www.sei.cmu.edu/technology/dynamic_systems/cbs/

Acquisition Policy Influence ISO OTS Standard

- ISO/IEC JTC1/SC7/WG7 Life Cycle Management Group
  - Developing ISO/IEC 15288, System Life Cycle Processes
  - June 1996 - 2000/20001
  - Previously developed ISO 12207, Software Life Cycle Processes
  - ISO 12207 will be updated to be consistent with ISO/IEC 15288
- Purpose
  - Define and describe a generic, top-level architecture of the life cycle of a system which includes systems engineering. The life cycle spans from the conceptualization of a need or an idea through the retirement of the resulting system.
- Scope
  - Architecture of the life cycle is constructed with a set of processes, which are defined in terms of specific tasks or functions.
  - Applicable to any type of system: distributed, stand-alone, products, services
  - Covers all stakeholders in the life cycle - including maintenance/support
  - Covers project, enterprise, technical and management

Acquisition Policy Influence ISO OTS Standard

- ISO/IEC 15288 System Life Cycle Processes: Off-The-Shelf Items
  - Series of Workshops to Support Development of ISO/IEC 15288
  - OTS Focus Workshop: August 10-14, 1998, Monterey, CA
  - Collocated with 4th international Conference on Complex Computer Systems (ICECCS98)
- Contact Information
  - Dr. Jeffrey Voas (703-404-9293)
    - Reliable Software Technologies
    - Suite 250
    - 21515 Ridgetop Circle
    - Sterling, VA 20166 USA
    - Voice (703-404-9293); fax (703-404-9295); e-mail(jmvoas@RSTcorp.com)
  - Paper summary of information is available

Acquisition Policy Influence ISO OTS Standard

"The use of commercially available "off-the-shelf" items including software is accelerating as the variety of available products grows and the rapid evolution of software engineering technology reduces reliance on custom- coded software."

- "Off-the-shelf" items
  - stand-alone products
    - payroll, accounting software, 'shrink-wrapped software' (word-processing, spreadsheets, games)
  - integrated as components into a larger system
    - operating system, relational data base management system, graphical users interface (GUI)
  - embedded in hardware
    - communication data link, programmable array logic
  - embedded as part of a configurable software/hardware system
    - distributed control system
  - CASE tools used to support the software development and maintenance process
    - compilers, configuration management tools
  - OTS software libraries and component-based software engineering

Acquisition Policy Influence ISO OTS Standard

- Benefits Include
  - Reduced time-to-market
  - Decreased development costs
  - Potential for mega-reuse
- Problems Include
  - Trojan horses, inadequate testing, dead code, inferior reliability, and unknown design and specification assumptions
  - Proposed solutions such as wrappers are non-trivial to build and test
  - Intense focus on concomitant software product quality or self-contained software unit quality

The ISO/IEC 15288 OTS standardization effort does recognize support of OTS as a significant issue. This effort also recognizes that OTS is not a silver bullet and intends to address some of the problem issues as well.

Throughlife Support Strategy

- Support Considerations
  - Guidance: Make OTS an integral part of the integrated logistics support strategy and the software support plan/case concept. Ensure OTS licenses/agreements clearly define the legal extent of the vendor responsibilities regarding through life support including fault repair, on-site support, on-line advice, escrow for source.
- Technical Competence
  - Guidance: Ensure the OTS vendor has an established, stable support and maintenance organization. Assess any factors that might jeopardize future essential support requirements.
- Third Party Aspects
  - Guidance: Ensure clear responsibilities are addressed in separate licenses / agreements. Review the stability of customer support services.

Throughlife Support Strategy

- COTS Component Life Cycle
  - Guidance: Consider the stated / anticipated life span for which the OTS product(s) will be supported. Commercial practice is to support product through two major upgrades (2-7 years). Review effect OTS upgrades will have on the implemented system. Assess effects of product releases on the overall cost / effectiveness of the system. Recognize that OTS product will be non-viable without vendor support.
- Vendor Responsiveness
  - Guidance: Ensure the vendor mechanisms for support are adequate for the user's requirements such as: continuous on-site vendor field representatives, on-call fault repair services or workaround solutions, on-line advice. Identify the various critical vendor software support scenarios.

Throughlife Support Strategy

- Product Updating and Replacement
  - Guidance: Ensure there is an understanding of the vendor's policy on product upgrades and new releases. Assess what support the vendor would provide for the integration of new OTS packages into the operational system. Will the vendor participate if necessary in the establishment of an integration and test environment? What are the implications (configuration, availability, cost) of operational verification of new OTS products?

Throughlife Support Strategy

- Product Updating and Replacement (ISSUES)
  - Issue: Timing of OTS software releases
    - Guidance: Compatibility problems can exist with inter-dependent OTS products. Use a reference testbed to test and integrate different OTS and application products. Apply strict configuration management for timely release of update versions to the operational system.
  - Issue: OTS software changes very fast
    - Guidance: Plan for change through a well-defined COTS framework and a "Pre Planned Improvement" strategy. Make full use of current available OTS product catalogues.
  - Issue: Continual changes to OTS software
    - Guidance: Harmonize system software upgrade strategy with the vendor for release of new products onto the market. A policy of "no change" may work for awhile, but the OTS versions will quickly become out of date and support from the vendor will become increasingly difficult.
  - Issue: Support contracts
    - Guidance: Ensure that all through life support-related activities for the OTS components are considered in advance of the procurement to ensure that updates and replacements are an integral part of the overall acquisition strategy.

Throughlife Support Strategy

- Commercial and Government Available Product "Catalogues"
  - Guidance: Subscribe to the major component catalogues (commercial / government) that are available, updated regularly and accessible by multiple users through a network. On-line World Wide Web is also recommended.
- Support for Integration Software
  - Guidance: Integration software (integrates OTS software with its operational environment) will either be in-house developed or other OTS software that interfaces the system's major OTS packages. This software must be supported.

Throughlife Support Strategy

- Training
  - Guidance: Ensure required training for users and system integrators is timely and application-specific. Depending on the complexity of the OTS software and its level of integration with other software, user training should be acquired before the operational fielding of the system. Documentation and training material need to be available in advance of any proposed acquisition in order to ensure that all relevant user requirements can be adequately met. Consider training requirements as integral to the acquisition and support of OTS software components. A training license agreement may be the most cost-effective approach.

OTS Software Support Concerns

- Myth:
  - Using/buying OTS frees you from logistics concerns
- Fact:
  - Basic support needs for OTS are the same as the are for any other software. There still must be software support/maintenance plans; supply support and assets accountability; support equipment, training, and technical data to enable system restoral/recovery within a practical maintenance concept; and provision for module repair and/or upgraded versions. The differences are mainly in emphasis

OTS Software Support Concerns

- Scope of Support
  - OTS software products are now invading our systems and inventories on so much larger a scale that combat readiness economic realities and the demands of policy such as “Competition in Contracting Act” may dictate that we provide support on a worldwide scale through wholesale methods
  - Logistics decisions must recognize the limitations that come with adopting OTS products and document such limitations in an Integrated Logistics Support Plan (Software Support Plan).

OTS Software Support Concerns

- Design Interface
  - Software supportability analysis tasks should include OTS software on the premise that we need to know its impact on the supportability of the system.
  - Some suggested analysis tasks include:
    - Use Study to establish usage and environmental stress definitions
    - Mission Hardware, Software, and Support System Standardization to determine where standardization and interoperability is a requirement
    - Technological Opportunities to help identify those new technologies that can enhance the effectiveness and supportability of the system as well as the likely evolution of the OTS products themselves
    - Supportability and Supportability Related Design Factors with particular emphasis on identifying proprietary data (Intellectual Property Rights - IPR) constraints
    - Functional Requirements with particular focus on performing a Failure Modes and Effects Analysis
    - Evaluation of Alternatives and Tradeoff analysis provides a number of Design Interface considerations as affected by the use of OTS. Alternative OTS products as well as non-OTS products can be analyzed.

OTS Software Support Concerns

- Stability in Support Assumptions
  - Configuration Management and Data
  - Level of Support (e.g., two-level: Field and Depot/Vendor)
  - Packaging, Handling, Storage, and Transportation
  - Sustainability
- Interchangeability
  - Upgrading the system to enable a change to the new revision
  - Freezing the old revision as a Government “Unique” item
  - Myth: A little modification never hurt anybody.
- Modification
  - In-house vs vendor repair/upgrade
  - Modification to meet mission need: conduct suitability analysis, life cycle cost analysis, and reach coordinated agreement among the using, supporting, and acquiring agencies
  - Logistics management of new version release

OTS Software Support Concerns

Warranties

- Warranties may be available to protect the customer from defects in an OTS product, but rarely apply to OTS software. A warranty for software generally protects the supplier from any potential defect or misuse of the product. Supplier certification in software best practices (such as ISO 9000 or SEI CMM) may serve as useful evidence of good software practices, but are not typically backed by any software warranties of the product. If there are warranties, ensure that its provisions are known and understood. Make sure the warranty transfers through the primary supplier to the intended customer.

Escrow Agreements and Intellectual Property Rights

- Legal documentation and/or escrow data rights for Intellectual Property Rights to software may be necessary to minimize risk due to supplier business failure, failure to deliver required support services, or change in business products supported.
- The business failure of a COTS vendor can have very serious consequences for a customer, even if appropriate escrow data rights and service/product warranties have been attained through an Intellectual Property Rights agreement.
- The extent to which contingency measures are explored in the supportability analysis should be limited to instances where there is considered to be relatively high risk (e.g., software that is immature, has specialized functionality, or has high impact on system operation). In such cases, the analysis should consider to what extent there are suitable alternative products in the marketplace, and whether arrangements could be established for the acquisition of the necessary data rights by an alternative support agency.

Appendix A
Glossary of Terms

A. Glossary of Terms

A.1 Primary Acronyms

AFOTEC	Air Force Operational Test and Evaluation Center
AIR	Aerospace Information Report
CFSSU	Canadian Forces Supply System Upgrade
CMMI	Capability Maturity Model Integrated
COCOMO	COnstructive COst MOdel
COTS	Commercial Off-The-Shelf
CRISD	Computer Resource Integrated Support Document
CRLCMP	Computer Resources Life Cycle Management Plan
DoD	Department of Defense
DND	Department of National Defence (Canada)
DSI	Delivered Source Instructions
FMECA	Failure Modes, Effects and Criticality Analysis
FRACAS	Failure Reporting and Corrective Action System
FTA	Fault Tree Analysis
FTE	Full Time Equivalent
IEEE	Institute of Electrical and Electronic Engineers
ILS	Integrated Logistic Support
ILSP	Integrated Logistic Support Plan
ISO	International Standards Organization
JA	Two character code for SAE ground vehicle (J) and aerospace (A) standards and guidelines
KSLOC	Thousands (K) of Source Lines of Code
LSA	Logistic Support Analysis
MOD	Ministry Of Defence (United Kingdom)
NCSLOC	Non-Commented Source Lines of Code
NDI	Non-Developmental Item
OTS	Off-The-Shelf
PDSS	Post-Deployment Software Support
RAMSS	Risk Assessment Methodology for Software Supportability
RCM	Reliability Centered Maintenance
RDIT	Replication, Distribution, Installation, Training
REVIC	Ray's Enhanced Version of Intermediate COCOMO
RFC	Request For Change
RLA	Repair Level Analysis
RMSL	Reliability, Maintainability, Supportability, Logistics
SAE	Society of Automotive Engineers
SCR	Software Change Request
SLOC	Source Lines of Code
SOLE	International Society of Logistics (formerly Society of Logistics Engineers)
SSA	Software Supportability Analysis
SSC	Software Support Concept
TCCCS	Tactical Command, Control and Communications System
UK	United Kingdom

A.2 Primary Definitions

A.2.1. JA1002, JA1004, JA1006

Customer: an organization that procures a system, software product or software service from a supplier. NOTE-The acquirer could be one of the following: buyer, owner, user, purchaser. Equivalent with ISO/IEC 12207 definition of Acquirer.

Defect: see software fault.

Failure: the inability of a software component to perform its required functions within specified performance requirements.

Fault: an accidental condition that causes a software functional unit to fail to perform its required function.

Failure Effects, Modes and Criticality Analysis: a proactive approach used for determining the potential failure modes of a system/equipment (including software), all likely ways in which a component or equipment can fail, causes for each failure mode, and effects/criticality of each failure mode.

Failure Reporting and Corrective Action System: a set of processes, procedures, and tools for reporting, reviewing, analyzing, correcting, and storing information about system/software failures.

Fault Tree Analysis: An analysis technique where identified potential system failure modes are analyzed in terms of what potential software faults (single point of failure) or multiple faults (multiple points of failure) might result in the potential failure mode.

Integrated Logistic Support: a disciplined, unified, and iterative approach to the management and technical activities necessary to 1)integrate support considerations into system and equipment design, 2) develop support requirements that are related consistently to readiness objectives, to design, and to each other, 3) acquire the required support, and 4) provide the required support during the operational phase at minimum cost.

Logistics Management Support: the software support activities related to the 'bridge' between the software support services for operational use and for post-delivery modification support. Activities may include help desk management, problem reporting and corrective action coordination, distribution of releases to field sites, management of configuration information concerning software releases in the field, and network communications among field sites and modification support sites.

Logistic Support Analysis: the selective application of scientific and engineering efforts undertaken during the acquisition [and sustainment] process, to assist in 1) causing support considerations to influence design, 2) defining support requirements that are related optimally to design and to each other, 3) acquiring the required support, and 4) providing the required support during the operational phase at minimum cost.

Maintainability: the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment. Also, a set of attributes that bear on the effort needed to make specified modifications.

Maintenance: the process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment.[1]

Modification Support: the software support activities of change analysis, implementation, test and release of software products. Changes may be termed corrective, perfective and adaptive, and may also embrace modifications that are designed to prevent foreseeable future software operating problems.

Operational Support: the software support activities related to the day-to-day operation of software by the software user. Activities may include installation, configuration, data preparation and loading/unloading, backup, recovery, failure reporting and training.

Reliability Program: the organizational processes and practices that are intended to: (1) Ensure the delivery of a software product that has been adequately designed to achieve its performance specifications;

[1] Software maintenance as defined above is essentially the same as software modification support. Software maintenance and the maintenance process have not always been interpreted as defined in this appendix. Hence, the term Modification Support is the preference within this course to more clearly distinguish and contrast with the Operational and Logistics Management Support activities.

and (2) Ensure there is adequate evidence that the performance specification for the delivered software product has been achieved and continues to be met during operational use.

Reliability: (1) the probability of failure-free operation of a software program for a specified time under specified conditions; (2) a set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time.

Supplier: An organization that enters into a contract with the customer for the supply of a system, software product or software service under the terms of the contract. NOTES- 1) The term "supplier" is synonymous with contractor, producer, seller, or vendor. 2) The customer may designate a part of its organization as supplier. Equivalent with ISO/IEC 12207 definition of supplier.

Support: the set of activities necessary to ensure that an operational software system or component fulfills its original requirements and any subsequent modifications to those requirements. Activities include all processes, resources and infrastructure required in order to provide support throughout the software's operational life.

Support Concept: the derived set of information that describes the strategy, primary factors, and performance requirements for one or more software products across the three dimensions of support classes, support functions, and support profile.

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

Processes: the inherent quality characteristics of the support processes that affect the capability to accomplish the required support functions;

Product: the inherent quality characteristics that are designed into the software product that affect the capability to accomplish the required support functions; and,

Environment: the inherent quality characteristics of personnel resources, support systems and physical facilities that affect the capability to accomplish the required support functions.

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

Operational Support: the support activities required by users and operators of the software during application use;

Logistics Management Support: the support activities required to establish or restore the operational capability of the software or to provide distribution and control of an upgraded software product; and,

Modification Support: the support activities required to modify the software under controlled conditions to correct defects, upgrade performance capability, and adapt to environment changes.

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

Support Level: the different (generic) sites where support will be carried out, their capabilities, resources and overall role in the support process.

Support Agent: the generic roles taken by organizations and groups of people that are involved in software support, and the responsibilities and information they require.

Support Scenarios: the individual tasks required to carry out all support processes, their interrelationships and sequencing and the people, organizations, resources and infrastructure required to carry them out.

Supportability: a set of attributes of software design, the associated development tools and methods, and the support environment infrastructure that enable the software support activities to be accomplished.

Supportability Analysis: see Logistic Support Analysis

A.2.2. ISO/IEC 12207

Acquirer: An organization that acquires or procures a system, software product or software service from a supplier. NOTE-The acquirer could be one of the following: buyer, customer, owner, user, purchaser.

Acquisition: The process of obtaining a system, software product or software service.

Agreement: The definition of terms and conditions under which a working relationship will be conducted.

Audit: Conducted by an authorized person for the purpose of providing an independent assessment of software products and processes in order to assess compliance with requirements.

Baseline: A formally approved version of a configuration item, regardless of media, formally designated and fixed at a specific time during the configuration item's life cycle.

Configuration item: An entity within a configuration that satisfies an end use function and that can be uniquely identified at a given reference point.

Contract: A binding agreement between two parties, especially enforceable by law, or a similar internal agreement wholly within an organization, for the supply of software service or for the supply, development, production, operation, or maintenance of a software product.

Developer: An organization that performs development activities (including requirements analysis, design, testing through acceptance) during the software life cycle process.

Evaluation: A systematic determination of the extent to which an entity meets its specified criteria.

Firmware: The combination of a hardware device and computer instructions or computer data that reside as read-only software on the hardware device. The software cannot be readily modified under program control.

Life cycle model: A framework containing the processes, activities, and tasks involved in the development, operation, and maintenance of a software product, spanning the life of the system from the definition of its requirements to the termination of its use.

Maintainer: An organization that performs maintenance activities.

Monitoring: An examination of the status of the activities of a supplier and of their results by the acquirer or a third party.

Non-deliverable item: Hardware or software product that is not required to be delivered under the contract but may be employed in the development of a software product.

Off-the-shelf product: Product that is already developed and available, usable either "as is" or with modification.

Operator: An organization that operates the system.

Process: A set of interrelated activities, which transform inputs into outputs. NOTE-The term "activities" covers use of resources.

Qualification: The process of demonstrating whether an entity is capable of fulfilling specified requirements.

Qualification requirement: A set of criteria or conditions that have to be met in order to qualify a software product as complying with its specifications and being ready for use in its target environment.

Qualification testing: Testing, conducted by the developer and witnessed by the acquirer (as appropriate), to demonstrate that the software product meets its specifications and is ready for use in its target environment.

Quality assurance: All the planned and systematic activities implemented within the quality system, and demonstrated as needed, to provide adequate confidence that an entity will fulfill requirements for quality. NOTES – 1) There are both internal and external purposes for quality assurance: a) Internal quality assurance: within an organization, quality assurance provides confidence to management; b) External quality assurance: in contractual situations, quality assurance provides confidence to the customer or others. 2) Some quality control and quality assurance actions are interrelated. 3) Unless requirements for quality fully reflect the needs of the user, quality assurance may not provide adequate confidence.

Release: A particular version of a configuration item that is made available for a specific purpose (for example, test release).

Request for proposal [tender]: A document used by the acquirer as the means to announce its intention to potential bidders to acquire a specified system, software product or software service.

Retirement: Withdrawal of active support by the operation and maintenance organization, partial or total replacement by a new system, or installation of an upgraded system.

Security: The protection of information and data so that unauthorized persons or systems cannot read or modify them and authorized persons or systems are not denied access to them.

Software product: The set of computer programs, procedures, and possibly associated documentation and data.

Software service: Performance of activities, work, or duties connected with a software product, such as its development, maintenance, and operation.

Software unit: A separately compilable piece of code.

Statement of work: A document used by the acquirer as the means to describe and specify the tasks to be performed under the contract.

Supplier: An organization that enters into a contract with the acquirer for the supply of a system, software product or software service under the terms of the contract. NOTES- 1) The term "supplier" is synonymous with contractor, producer, seller, or vendor. 2) The acquirer may designate a part of its organization as supplier.

System: An integrated composite that consists of one or more of the processes, hardware, software, facilities and people, that provides a capability to satisfy a stated need or objective.

Test coverage: The extent to which the test cases test the requirements for the system or software product.

Testability: The extent to which an objective and feasible test can be designed to determine whether a requirement is met.

User: An individual or organization that uses the operational system to perform a specific function. NOTE- The user may perform other roles such as acquirer, developer, or maintainer.

Validation: Confirmation by examination and provision of objective evidence that the particular requirements for a specific intended use are fulfilled. NOTES – 1) In design and development, validation concerns the process of examining a product to determine conformity with user needs. 2) Validation is normally performed on the final product under defined operating conditions. It may be necessary in earlier stages. 3) "Validated" is used to designate the corresponding status. 4) Multiple validations may be carried out if there are different intended uses.

Verification: Confirmation by examination and provision of objective evidence that specified requirements have been fulfilled. NOTES – 1) In design and development, verification concerns the process of examining the result of a given activity to determine conformity with the stated requirement for that activity. 2) "Verified" is used to designate the corresponding status.

Version: An identified instance of an item. NOTE - Modification to a version of a software product, resulting in a new version, requires configuration management action.

Appendix B
References

B. References

B.1 Standards and Guidelines

[AIAAR013-93] ANSI/AIAA R-013-1992, “AIAA Recommended Practice for Software Reliability”, February 1993.

[AIR5121-97] AIR 5121, “Software Supportability - An Overview,” Society of Automotive Engineers, January 1997.

[AR127] AR-127, “Policies and Responsibilities for the Management of Acquisition Logistics,” US Department of the Army (DA).

[BS5760P8-97] BS 5760, “Reliability of Systems, Equipment and Components,” Part 8: “Guide to Assessment of Reliability of Systems Containing Software,” British Standards Institute, Draft for Approval for Publication, July 7, 1997.

[CECOM95] CECOM, Software Logistics Handbook, October 1995

[DEFSTAN0042-97] Defence Standard 00-42 (PART 2)/Issue 1, “Reliability And Maintainability Assurance Guides, Part 2: Software,” United Kingdom Ministry of Defence, 1997.

[DEFSTAN0060-98] Defence Standard 00-60, “Integrated Logistic Support”, Issue 2, “Logistic Support Analysis Application to Software Aspects of Systems”, Part 3, United Kingdom Ministry of Defence, March 1998.

[FAA_DOT_SWH-02] DOT/FAA/AR-01/116, "Software Service History Handbook," Federal Aviation Administration, Department of Transportation, Office of Aviation Research, Washington, DC, January 2002.

[FAA_DOT_SWR-02] DOT/FAA/AR-01/115, "Software Service History Report," Federal Aviation Administration, Department of Transportation, Office of Aviation Research, Washington, DC, January 2002.

[FAA_DOT_COTS-01] DOT/FAA/AR-01/41, "Review of Pending Guidance and Industry Findings on Commercial Off-The-Shelf (COTS) Electronics in Airborne Systems," Federal Aviation Administration, Department of Transportation, Office of Aviation Research, Washington, DC, August 2001.

[FAA-N8110_77] N 8110.77, "Guidelines for the Approval of Field-Loadable Software," Federal Aviation Administration, November 1998.

[FAA-N8110_79] N 8110.79, " Guidelines for the Approval of Field-Loadable Software by Finding Identicality Through the Parts Manufacturer Approval Process," Federal Aviation Administration, February 1999.

[FAA-N8110_84] N 8110.84, " Guidelines for Determining the Level of Federal Aviation Administration (FAA) Involvement in Software Projects," Federal Aviation Administration, August 2000.

[FAA-N8110_87] N 8110.87, "Guidelines for the Software Review Process," Federal Aviation Administration, January 2001.

[FAA-N8110_90] N 8110.90, "Guidelines for the Software Review Process," Federal Aviation Administration, January 2001.

[FAA-N8110_91] N 8110.91, " Guidelines for the Qualification of Software Tools Using RTCA/DO-178B," Federal Aviation Administration, August 2001.

[FAA-N8110_94] N 8110.94, " Guidelines for the Approval of Airborne Systems and Equipment Containing User-Modifiable Software," Federal Aviation Administration, August 2001.

[FAA-N8110_95] N 8110.95, "Guidelines for the Approval of Field-Loadable Software" Federal Aviation Administration, August 2001.

[FAA-N8110_97] N 8110.97, "Guidelines for Approving Reused Software Life Cycle Data," Federal Aviation Administration, February 2002.

[IEEE1219-98] IEEE Standard 1219-1998, “IEEE Standard for Software Maintenance,” IEEE Computer Society, June 1998.

[IEEE122070-98] IEEE/EIA Std 12207.0-1996, “Software life cycle processes,” IEEE Computer Society, March 1998.

[IEEE122071-98 IEEE/EIA Std 12207.1-1997, “Software life cycle processes - Life cycle data,” IEEE Computer Society, April 1998.

[IEEE122072-98 IEEE/EIA Std 12207.2-1997, “Software life cycle processes – Implementation considerations,” IEEE Computer Society, April 1998.

[ISO12207-95] ISO/IEC 12207, “Software Life Cycle Processes,” August 1, 1995.

[ISO14764-99] ISO/IEC 14764, "Information technology -- Software maintenance," 1999.

[ISO15288-00] ISO/IEC 15288 (Draft), “System Life Cycle Processes,” targeted for 2000/2001.

[JA1000-98] JA1000, “Reliability Program Standard,” Society of Automotive Engineers, 1998.

[JA1002-98] JA1002, “Software Reliability Program Standard,” ,” Society of Automotive Engineers, July 1998.

[JA1003-00] JA1003, “Software Reliability Implementation Guide,” ,” Society of Automotive Engineers, Draft 1999, expected publication 2000.

[JA1004-98] JA1004, “Software Supportability Program Standard,” Society of Automotive Engineers, July 1998.

[JA1005-01] JA 1005, “Software Supportability Implementation Guide,” Society of Automotive Engineers, May 2001.

[JA1006-99] JA1006, “Software Support Concept,” Society of Automotive Engineers, June 1999.

[MILH502-97] MIL-HDBK-502, “DoD Handbook Acquisition Logistics,” May 30, 1997.

[MILP49506-96] MIL-PRF-49506, “Performance Specification Logistics Management Information,” November 11, 1996.

[MILH467-97] MIL-HDBK-1467, “Acquisition of Software Environments and Support Software”, December 10, 1997.

[NATO96] NATO (Draft), “COTS Software Acquisition Guidelines and COTS Policy Issues – 1st Revision,” NATO Communications and Information Systems Agency, January 12, 1996.

[NATO97] NATO (Draft), “NATO Guidelines for the Integration of Off-The-Shelf Software,” Working Paper AC/322(SC/5)WP/4, NATO C3 Board Information Systems Sub-Committee, June 30, 1997.

[RCTADO178B92] RCTA/DO 178B, “Software Considerations in Airborne Systems and Equipment,” Federal Aviation Administration software standard, RTCA Inc., December 1992.

[RTCADO-248B01] RCTA/DO-128, Final Report for Clarification of DO-178B, " Software Considerations in Airborne Systems and Equipment,” Prepared by SC-190, October 12, 2001.

[CMMI2000] CMMI-SE/SW-Continuous, V1.02, CMMI for Systems Engineering/Software Engineering, Version 1.02, Continuous Representation, CMU/SEI-2000-TR-019, November 2000.

CMMI-SE/SW-Staged, V1.02, CMMI for Systems Engineering/Software Engineering, Version 1.02, Staged Representation, CMU/SEI-2000-TR-018, November 2000.

[SPICE98] ISO/IEC TR 15504:1998: “Software Process Improvement Capability Determination (SPICE) - Software Process Assessment,” ISO/IEC/JTC1/SC7/WG10/N111, ISO 1998.

B.2 Reliability Publications

[HERMANN99] Herrmann, Debra, Software Reliability and Safety, Computer Society Press, IEEE Inc, Piscataway, NJ, 1999.

[HERPER99] Herrmann, Debra and Peercy, David, "Software Reliability Cases: The Bridge Between Hardware, Software and System Safety and Reliability," RAMS International Symposium on Product Quality & Integrity, January 1999.

[LYU96] Lyu, Michael, Handbook of Software Reliability Engineering, McGraw Hill / IEEE CS Press, 1996.

[KEENE98] Keene, Sam, "Modeling Software R&M Characteristics," personal communication and notes, SAE G-11SW Software Committee, 1998.

[MUSA87] Musa,J., Iannino,A., and Okumoto,K., Software Reliability: Measurement, Prediction, Application, McGraw-Hill Series in Software Engineering and Technology, McGraw-Hill Book Company, NY, 1987.

[MUSA89] Musa,J., Ackerman,F., "Quantifying Software Validation: When to Stop Testing?," IEEE Software, May 1989, pp. 19-27.

[MUSA92] Musa, John D. "Operational Profiles in Software Reliability Engineering," IEEE Software, March 1993, pages 14-32.

[MUSA96] Musa, John D. "Software - Reliability - Engineering & Testing", IEEE Computing, November 1996.

[MUSA99] Musa, J., Software Reliability Engineering, McGraw-Hill Book Company, NY, 1999.

[ROME97] Lakey, Peter and Neufelder, Ann Marie, "System and Software Reliability Assurance," Rome Laboratory Report, 1997.

[VOAS97-1] Voas, Jeffrey, et al, "A 'Crystal Ball' for Software Liability," IEEE Computer, June 1997, pp 29-36]

[WELLER00] Weller, Edward, "Practical Applications of Statistical Process Control," IEEE Software, May/June 2000, pp 48-55.

B.3 Supportability and Maintenance Publications

[ADDY99] Addy, Edward, "Issues in the Adaptation of Software Support Activities to Reuse-Based Software Engineering," extended abstract submitted t0 IEEE CS Special Issue, 1999.

[AFOTECP2] AFOTEC Pub. 800-2 Vol.2, "Software Support Life Cycle Process Evaluation Guide", USAF, 1988

[AFOTECP3] AFOTEC Pub. 800-2 Vol.3, "Software Maintainability Evaluation Guide", USAF, 1989

[AFOTECP5] AFOTEC Pub. 800-2 Vol.5, "Software Support Resources Evaluation Guide", USAF, 1988

[AGRA98] Agrawal, Hira, et al., "Mining System Tests to Aid Software Maintenance," IEEE Computer, July 1998, pp 64-73.

[ARMYRD96] "Harnessing Information Technology," Army R&D, March-April 1996 (several software articles of interest).

[BASILI96] Basili, Victor, et al., "Understanding and Predicting the Process of Software Maintenance Releases," Proceedings of the 18th International Conference on Software Engineering (ICSE-18), March 25-29, 1996, Berlin, Germany.

[BOEHM81] Boehm, Barry, Software Engineering Economics, Prentice-Hall, Englewood Cliffs, NJ, 1981

[CHAR98] Charette, Robert, Adams, Kevin, and White, Mary, "Managing Risk in Software Maintenance," IEE Software, May/June 1997, pp 43-50.

[JOHNDRO94] Johndro, Anthony, "Ensuring Software Supportability During Acquisition: An Air Force Study," Air Force Journal of Logistics, Fall 1994, pp24-27,35.

[MILI99] Mili, Ali, Addy, Edward, and Mili, Hafedh, "Research Directions in Software Reuse," evolving paper based on panel at SSR'97, Boston, May 1997.

[MITRE96] Mitre, "Controlling the Post-Deployment Costs of Software," MTR 96B0000037, Mitre Centre for Air Force C3 Systems, July 1996.

[PEERCY81] Peercy, David, "A Software Maintainability Evaluation Methodology," IEEE Transactions on Software Engineering, 7 (1981), 4, pp 343-352.

[PEERCY85] Peercy, David, "Software Supportability Risk Assessment in OT&E: Historical Baselines for Risk Profiles, Volume 1&2," BDM/A-85-0510-TR, October 7, 1985.

[PEERCY87] Peercy, David and Tomlin, Eric "Assessing Software Supportability Risk: A Minitutorial," Proceedings of the Conference on Software Maintenance, 1987, September 21-24, 1987.

[PEERCY95] Peercy, Dave, "Task Description: Software Maintenance, Section 4.7.13, SAE R&M Guidebook, 1995.

[PEERCY96] Peercy, David, "Software Supportability – A Logistics Concern," Logistics Spectrum, August 1996, pp 8-10,17.

[SCHN98] Schneidewind, Norman, "How to Evaluate Legacy System Maintenance," IEEE Software, July/August 1998, pp 34-42.

[VERMA99] Verma, Dinesh, Plunkett, Galen, and McCaig, Robert, "Technology Refreshment for Commercial System Elements," Logistics Spectrum, July/September 1999, pp 12-15.

[VOAS98-1] Voas, Jeffrey, "Maintaining Component-Based Systems," IEEE Software, July/August 1998, pp 22-27.

[WEYUK98] Weyuker, Elaine, "Testing Component-Based Software: A Cautionary Tale," IEEE Software, September/October 1998, pp 54-59.

B.4 Integrated Logistics Support and LSA Publications

[HUTCH00] Hutchison, Peter B., "Using Def Stan 00-60 LSA Reports To Address Software Supportability," MSc Dissertation in Software Engineering, Oxford University, March 9, 2000.

[JONES87] Jones, James, Integrated Logistics Support Handbook, McGraw-Hill Book Company, NY, 1987.

[PEERCY94] Peercy, David, "Applying the Logistics Discipline to Software Support Concerns," Logistics Spectrum, Volume 28, Issue 4, Winter 1994, pp 44-48.

[ROGERS97] Rogers, Steve, "Performance Based Supportability," Logistics Spectrum, Volume 31, Issue 3, May/June 1997, pp 10-14.

[SOLE89] Peercy, David (editor), "National Workshop on Software Logistics," SOLE ad hoc Working Group, International Society of Logistics, August 1989.

B.5 Software Engineering and Quality Engineering Publications

[CAPERS95] Jones, Capers, "Measuring Global Software Quality", Software Productivity Research, Burlington, Mass., 1995.

[HARR98] Harrison, Rachel, Counseli, Steve, and Nithi, Reuben, "An Evaluation of the MOOD Set of Object-Oriented Software Metrics," IEEE Transactions on Software Engineering, Vol. 24, No. 6, June 1998, pp 491-496.

[PRESSMAN97] Pressman, Roger, Software Engineering, A Practitioner's Approach, McGraw-Hill Series in Computer Science, McGraw-Hill Book Company, NY, 1997.

[SED99] Grable, Ross, et all, "Metrics for Small Projects: Experience at the SED,"IEEE Software, March/April 1998, pp 16-19.

[WEID98] Weidenhaupt, Klaus, et al, "Scenarios in System Development: Current Practice," IEEE Software, March/April 1998, pp 34-45.

B.6 Off-The-Shelf Software Publications

[ABTS97] Abts, Christopher and Boehm, Barry, "COTS Software Integration Cost Modeling Study," University of Southern California Center for Software Engineering, Performed for the USAF Electronic Systems Center, Hanscom AFB, Massachusetts, Contract F30602-94-C-1095 June 29, 1997.

[VOAS97-2] Voas, Jeffrey, "Error Propagation Analysis for COTS Systems," Computing & Control Engineering Journal, December 1997, pp 269-272.

[VOAS98-2] Voas, Jeffrey, "COTS Software: The Economical Choice?," IEEE Software, March/April 1998, pp 16-19.

[VOAS98-3] Voas, Jeffrey, "An Approach to Certifying Off-the-Shelf Software Components," IEEE Software, May-June 1998.

[VOAS99] Voas, Jeffrey, "Mitigating the Potential for Damage Caused by COTS and Third-Party Software Failures," personal communication.

B.7 Interesting Web Sites

[G11SW] http://www.sae.org/TECHCMTE/g11soft.htm , G-11SW Committee

[IEEECS] http://www.ieee.computer.org/ , IEEE Computer Society

[ILS0060] http://www.dstan.mod.uk/dsmain.htm , UK MOD ILS

[ISO] http://www.iso.ch/ , International Standards Organization (ISO)

[LOGSA] http://www.logpars.army.mil/alc/logEngr.htm , Acquisition Logistics Center part of the US Army Materiel Command Logistic Support Activity (LOGSA), Mil-PRF & MIL-HDBK documents

[RAC] http://iitri.com/RAC/ , Reliability Analysis Center

[SEI] http://www.sei.cmu.edu/ , Software Engineering Institute

[SOLE] http://www.sole.org/ , International Society of Logistics (SOLE)

[SPMN] http://www.spmn.com/index.html , Software Program Manager's Network

[STSC] http://www.stsc.hill.af.mil/ , Software Technology Support Center

[USC] http://sunset.usc.edu/COCOMOII/cocomo.html , COCOMO Project

[MUSA] http://members.aol.com/JohnDMusa/ , John Musa

[CSR] http://www.csr.ncl.ac.uk:80/ , Centre for Software Reliability, Newcastle University, UK

[TRIVEDI] http://www.ee.duke.edu/~kst/ , Dr Kishor Trivedi, Duke University

[FAA-ACS] http://av-info.faa.gov/software/ , FAA Aircraft Certification Service Software Home Page