Defense Acquisition University

ISA 201
Intermediate Information Systems Acquisition
Lesson 9
Systems & Software Engineering
Learning Objectives

Today you will learn to:

Overall: Given a DoD IT/SW acquisition scenario, apply System Engineering Lessons Learned, Best Practices and Rules of Thumb (Heuristics) when dealing with software.

• Describe the role of the Software Engineer (SwE) in supporting the Systems Engineer (SE).
• Describe the DoD Systems Engineering Process in the context of an IT/SW Development.
• Describe the different activities that comprise the SE/SW Technical and Technical Management processes.
  - Recognize that Software Reuse must be considered when engineering a software design and development.
  - Describe the requirements management process for successful software acquisition.
  - Describe the configuration management process for successful software acquisition.
• Given a DoD IT/SW scenario, recognize aspects of these engineering management principles of Requirements Management, Risk Management, Configuration Management and Interface Management to a development effort (SE/SW Tower Exercise).
• Overview
  • Systems Engineering Process
  • Work Breakdown Structure (WBS)
  • WBS Exercise
  • Technical Reviews
  • Systems & Software Engineering Challenges
  • System/Software Development Requirements
  • Configuration Management
  • Tower Exercise
The Office of the Deputy Assistant Secretary of Defense for Systems Engineering (ODASD(SE)) is the focal point for all policy, practice, and procedural matters relating to Department of Defense Systems Engineering and its key elements to include technical risk management, software engineering, manufacturing and production, quality, standardization, and related disciplines.
DASD(SE)'s Priorities

• Support the current fight; manage risk with discipline
• Grow engineering capabilities to address emerging challenges
• Support realistic program formulation through the application of development planning and early systems engineering
• Increased focus on security, reliability, and affordability
• Champion systems engineering as a tool to improve acquisition quality
• Develop future technical leaders across the acquisition enterprise
Systems Engineering in IT Acquisition

IT Systems Engineering:

- Translates operational needs and capabilities into operationally suitable hardware and software increments of a system.
- Permeates design, coding/production, test & evaluation, and system support.
- Is implemented through multi-disciplinary teams (IPTs) of subject matter experts (SMEs).
- Takes a total life cycle, total systems approach to IT system planning, development, implementation and disposition/disposal.
Software Engineering:

- According to the Institute of Electrical and Electronics Engineers (IEEE), software engineering means applying the principles of engineering to the software development field.
- Software engineering differs from other branches of engineering in that professionals are building an intangible structure and not a tangible one.
- Since software is embedded in the machines used in various industries, though, malfunctioning software can actually have tangible effects.
- With software used in everything from medical equipment to airplanes, the end result of faulty software can indeed be loss of life.
- Software engineering often does involve writing code, but this is only one stage in the process. True software engineering has a well-articulated acquisition life cycle.
- Software engineering supports systems engineering.
• Overview

• **Systems Engineering Process**
  • Work Breakdown Structure (WBS)
  • WBS Exercise
  • Technical Reviews
  • Systems & Software Engineering Challenges & Best Practices
  • System/Software Development Requirements
  • Configuration Management
  • Tower Exercise
1. Introduction
2. Program Technical Requirements
   2.1. Architectures and Interface Control
   2.2. Technical Certifications
3. Engineering Resources and Management
   3.1. Technical Schedule and Schedule Risk Analysis
      3.1.1. Relationships with External Technical Organizations
      3.1.2. Schedule Management
      3.1.3. System of Systems Schedule
      3.1.4. Schedule Risk Analysis
   3.2. Technical Risk, Issue, and Opportunity Management
   3.3. Technical Structure and Organization
      3.3.1. Work Breakdown Structure
      3.3.2. Government Program Office Organization
      3.3.3. Program Office Technical Staffing Levels
      3.3.4. Engineering Team Organization and Staffing
   3.4. Technical Performance Measures and Metrics
4. Technical Activities and Products
   4.1. Planned SE Activities for the Next Phase
   4.2. Requirements Development and Change Process
   4.3. Configuration and Change Management
   4.4. Design Considerations
Appendix A – Acronyms
Appendix B – Item Unique Identification Implementation Plan
References

“Program Manager will prepare a SEP to manage the systems engineering activities in accordance with Department of Defense Instruction (DoDI) 5000.02, Operation of the Defense Acquisition System.” -- SEP Outline excerpt
Software Development Plan

• Describes a developer’s plans for conducting a software development effort in accordance with contractual software requirements.

• Provides the acquirer insight and a tool for monitoring the processes to be followed for software development.

• Details methods to be used and approach to be followed for each activity, organization, and resources.

• Documents all processes applicable to the system to be acquired, at a level of detail sufficient to allow the use of the SDP as the full guidance for the developers.

• References specific standards, methods, tools, actions, reuse strategy, and responsibility associated with the development and qualification of all requirements, including safety and security.

Source: USAF Weapon Systems Software Management Guidebook – Appendix I
IEEE 12207, "Systems and software engineering -- Software life cycle processes", is an international standard that establishes a common framework for software life cycle process, with well-defined terminology.

**Primary life cycle processes**
- Acquisition process
- Supply process
- Development process
- Operation process
- Maintenance process

**Organizational processes**
- Management process
- Organizational processes
- Improvement process
- Training process

**Supporting life cycle processes**
- Audit process
- Configuration Management
- Joint review process
- Documentation process
- Quality assurance process
- Problem solving process
- Verification process
- Validation process
Comparison of Software to Hardware

**Like Hardware**
- Can be broken down into manageable parts or pieces
- Has personal accountability by task
- Has reportable progress events
- Is traceable to requirements
- Relies on a defined set of operating principles and constraints

**Unlike Hardware**
- Lacks physical appearance and has no “manufacturing” phase
- Has greater logical complexity
- Has higher volatility
- Tends to propagate change effects
- Is data as well as logic
- Has limited standardization of design methods, designs, or components
DoD Systems Engineering Process

Systems Engineering

Operational Need → Delivered Capability

IOC/FOC

OT&E

Requirements → Validated Solution

Design → Product

Technical Processes
- Stakeholder
- Requirements
- Definition
- Requirements
- Analysis
- Architecture
- Design

Technical Processes
- Transition
- Validation
- Verification
- Integration
- Implementation

Technical Management Processes
- Decision Analysis
- Technical Planning
- Technical Assessment
- Requirements Management
- Risk Management
- Configuration Management
- Technical Data Management
- Interface Management

Enables a balanced approach for delivering capability to the warfighter
• Overview
• Systems Engineering Process

• Work Breakdown Structure (WBS)
• WBS Exercise
• Technical Reviews
• Systems and Software Engineering Challenges & Best Practices
• System/Software Development Requirements
• Configuration Management
• Tower Exercise
A product-oriented family tree composed of hardware, software, services, data, and facilities. Represents the entire program from the Government Program Manager’s responsibility.

Each WBS element provides logical summary levels for:
1) assessing technical accomplishments,
2) supporting the required event-based technical reviews, and
3) measuring cost and schedule performance.

Organizes risk management analysis and tracking.

Enables configuration and data management.

Develops work packages for work orders and material/part ordering.

Sources: PMBOK 5, MIL-STD-881D
As part of the Systems Engineering Process, complex systems are decomposed into smaller subsystems and discrete products.

The software requirements are broken down into Software Items (SI) and further into Software Units (SU)
**IT System Decomposition**

- **Typical Lifecycle Reviews:**
  - SFR (System Functional Review)
  - SSR (System Safety Review)
  - PDR (Preliminary Design Review)
  - CDR (Critical Design Review)

- **Typical Lifecycle Products:**
  - SSS (System Safety Specification)
  - SSDDs (System / Subsystem Design Description)
  - SRSs (System Requirements Specification)
  - IRS (Interface Requirements Specification)
  - IDD (Interface Design Description)

- **Software Components:**
  - SU (Software Unit)

- **Baselines:**
  - Functional
  - Allocated
  - Product

**Note:** "Components" is term for aggregation of SW units; not required on WBS.
Technical Baselines

**Functional Baseline**

- What the system must do - *functions*
- How well it must do it - *performance* - at the “system” level
- *Defines* the *interfaces*/dependencies among the functions, groups and the environment.
- Main artifacts = *System Performance Spec* and Subsystem/Segment Specs.

**Allocated Baseline**

- Describes how system level *functional and performance requirements are allocated to physical components* (hardware items, software items, and users)
- *Describes the interfaces* among system components and external systems/environment.
- Main artifacts = Hardware Configuration Item Performance Spec, Computer Software Configuration Item Requirements Spec, Interface Requirement Spec.

*This is the “Design to” Baseline.*

**Initial Product Baseline**

- *Describes in detail how to fabricate components and code software*, and how to manufacture, operate, and maintain the system and its components, and how to train the various users.
- Main artifacts: Detailed item specifications, material specifications, process specifications, various drawings and manuals.

*This is the “Build to” Baseline.*
• Overview
• Systems Engineering Process
• Work Breakdown Structure (WBS)

**WBS Exercise**

• Technical Reviews
• Systems and Software Engineering Challenges & Best Practices
• System/Software Development Requirements
• Configuration Management
• Tower Exercise
Training and Help System (THS):
THS provides operator and maintainer training to operate and maintain the components of the JTAMS systems to include Unmanned Aerial Vehicles and Unmanned Ground Vehicles. Training is automated and uses screen shots to walk the user through each lesson.

- **Training** includes how to use the Parts and Maintenance System (PAMS) to manage parts and supplies, how to load mission packages, how to maneuver their vehicles, how to respond to enemy attacks, how to execute mission packages, how to use the systems navigation systems (and backup systems GPS to Inertial Nav) how to bring the robot home (land or drive back to base), and how to use the Mission Rehearsal System (MRES).

- The **Help** system includes calling up the appropriate training modules when specific help is needed. The TAMS Help messages are updated based on the new features as appropriate JTAMS. Trouble-shooting modules are available to handle the unique situations with FUAV, JUGV and JCCS. THS interfaces with the Joint Command and Control System (JCCS) in a read-write mode.
Develop the Software Components in the WBS for the Training and Help System given the diagram below.

- Develop 2 Software Items
- Develop two Software Units per Software item
• Overview
• Systems Engineering Process
• Work Breakdown Structure (WBS)
• WBS Exercise

• Technical Reviews
• Systems and Software Engineering Challenges & Best Practices
• System/Software Development Requirements
• Configuration Management
• Tower Exercise
- Alternative Systems Review (ASR)
- Systems Requirements Review (SRR)
- Software Specification Review
- System Functional Review (SFR)
- Preliminary Design Review (PDR)
- Critical Design Review (CDR)
- Test Readiness Review (TRR)
- System Verification Review (SVR)
- Functional Configuration Audit (FCA)

- Production Readiness Review (PRR)
- Operational Test Readiness Review (OTRR)
- Physical Configuration Audit (PCA)
- Technology Readiness Assessment (TRA)
- In-Service Review (ISR)
- Developmental Testing (DT)
- Early Operational Assessment (EOA)

- Operational Assessment (OA)
- Initial Operational Test & Evaluation (IOT&E)
- Follow on Operational Test and Evaluation (FOT&E)
Sample System Reviews

- **Requirements Reviews**
  - Alternative System Review
  - System Requirements Review
  - System Functional Review

- **Design Reviews**
  - Preliminary Design Review (includes System Software Specification Review)
  - Critical Design Review
  - Test Readiness Review

- **Verification Reviews**
  - Production Readiness Review
  - Functional Configuration Audit
  - System Verification Review
  - Physical Configuration Audit
Technical Review Process

Before
- Identify participants
- Assign roles and tasks
- Establish guidelines and procedures
- Establish entry criteria
- Establish success criteria based on event-driven schedule

During
- Have overview meeting
- Examine artifacts
- Analyze artifacts
- Track and document analysis

After
- Assign responsibility

Follow-up
- Track action items and issues
- Track action item completion trends
- Document and distribute results of review and action item completions

DAG Chapter CH 3–3.3 Technical Reviews and Audits
• Overview
• Systems Engineering Process
• Work Breakdown Structure (WBS)
• WBS Exercise
• Technical Reviews

• **Systems & SW Engineering Challenges & Best Practices**
  • System/Software Development Requirements
  • Configuration Management
  • Tower Exercise
• General increase in system complexity in software reliant systems, due to over-allocation of functionality to software.
• Customer requirements are often not translated down to the appropriate sub-system, item, component, unit.
• New design approaches, such as modular open systems approach, are extending the lives of systems now in use.
• Software failures often occur at the external interfaces.

➢ Lesson Learned: Evidence shows that all successful DoD Program’s acquiring software implement the Best Practices of Risk Management, Configuration Management and Interface Controls on a continuous basis within their program.
Software Reuse must be considered when engineering a software architecture.

-SW reuse is indirectly mandated by DoDI 5000.02.

“Modular designs coupled with an appropriately open business model provide a valuable mechanism for continuing competition and incremental upgrades, and **to facilitate reuse across the joint force**.”

**10% Software Reuse Management Rule (Heuristic).** After roughly 10 percent of code changes to the reused code, **make** your software developers reassess their original cost savings due to reuse to ensure it is still valid.

**Reuse Consequences**

---

NOTE: At around 10% of the reused code changed, you have already spent about 40% of what it would have cost to build the code from scratch!
• Overview
• Systems Engineering Process
• Work Breakdown Structure (WBS)
• WBS Exercise
• Technical Reviews
• Systems and Software Engineering Challenges & Best Practices

• System/Software Development Requirements
  • Configuration Management
  • Tower Exercise
System performance attributes from the CDD (key performance parameters, key system attributes, and other attributes) are translated into system requirements.

These requirements are translated into engineering terminology in a system specification, subsequently flowed down to HW/SW subsystems and components.
• Every requirement has one interpretation
  - the interpretation of each requirement is clear.
• No unnecessary information is included in the requirement.
• Each requirement is traced to some document or statement of the stakeholders.
• Each derived requirement must be traceable to an originating requirement via some unique name or number.
• Each requirement is exclusive of a particular solution.
• A finite, cost-effective process has been defined to check that the requirement has been attained.
Qualities of Software Requirements Sets
(1 of 2)

- **Correct**
  - Is a true statement of something the system must do.
- **Complete**
  - Describes all significant requirements of concern to the user.
- **Consistent**
  - Does not conflict with other requirements.
- **Unambiguous**
  - Is subject to one and only one interpretation.

IEEE 830-1993, § 4.3.2, 1994
Qualities of Software Requirements Sets

(2 of 2)

- Verifiable
  - Can be tested cost effectively.
- Ranked for importance and stability
  - Can be sorted based on customer importance and stability of the requirement itself.
- Modifiable
  - Changes do not affect the structure and style of the set.
- Traceable
  - The origin of each requirement can be found.
- Understandable
  - Comprehended by users and developers.

• The most important requirements are stated as “shall,” to identify mandatory requirements.
• The second most important requirements are stated as “shall, where practical.”
• The third most important level of requirements are stated as “preferred” or “should.”
• The least important requirements are stated as “may.”
• The contractor shall allocate the X-Star Satellite System (XSS) requirements identified in the performance specification and shall identify the methods and procedures necessary for verifying compliance with these requirements.

• The XSS shall provide the capability to support a processing rate of at least 50% above the average aggregate throughput rate for non real-time data.

• The XSS ground system (XGS) software shall have a Software Reliability (Rsw) of 0.992 with a Mean Time Software Reboot (MTSWR) of 10 minutes.
Lesson Overview

Lesson Plan Status

- Overview
- Systems Engineering
- Systems Engineering Processes
- Work Breakdown Structure (WBS)
- WBS Exercise
- Technical Reviews
- Systems and Software Engineering Challenges & Best Practices
- System/Software Development Requirements
- Metrics and Technical Performance Measures
- Modeling and Simulation

Configuration Management

- Tower Exercise
Configuration Management is a process of applying administrative and technical procedures throughout the software life cycle to: identify and define software items in a system; control modifications and releases of the items; record and report the status of the items and modification requests; ensure the completeness, consistency, and correctness of the items; and control storage, handling, and delivery of the items.

A configuration management plan may also be part of acquisition, supply, development, operation, maintenance plan(s) or any other appropriate plan.

Source:
IEEE/EIA 12207.2-1997
Baseline
The point at which a document or other object becomes a configuration item.

Configuration Control
The process of managing change to a configuration item.

Configuration Item
A document or other object placed under configuration control.

Discrepancy
An error in software caused either by improperly implementing a correct requirement or failing to implement it.

Enhancement
A change to a product designed to improve or augment its performance.

Source: http://www.sei.cmu.edu/reports/87cm004.pdf
In software engineering, **software configuration management (SCM or S/W CM)** is the task of tracking and controlling changes in the software, part of the larger cross-disciplinary field of configuration management. SCM practices include **revision control** and the establishment of **baselines**. If something goes wrong, SCM can determine what was changed and who changed it. If a configuration is working well, SCM can determine how to replicate it across many hosts.

• **Work your plan**: Implement and conduct the configuration management activities according to the program's configuration management plan.

• **Use checklists**: A basic checklist can assist in capturing the necessary efforts.

• **Automate to manage complexity**: If the program is sufficiently complex, identify and install an automated tool to support the configuration management tasks. Consider the other stakeholders (engineers/programmers, users, contractors, interfacing systems, and sustainment organizations) in the selection of any automated configuration management tools.

Source: https://www.mitre.org/publications/systems-engineering-guide/acquisition-systems-engineering/configuration-management
SCM Checklist

- Have all items subject to configuration control been identified in the program plan?
- Has a closed loop change management system been established and implemented?
- Has a government configuration control board been established for both development and sustainment?
- Are impact reviews performed to ensure that the proposed changes have not comprised the performance, reliability, safety, or security of the system?
- Does the developer's CM create or release all baselines for internal use?
- Does the developer's CM create or release all baselines for delivery to the customer?
- Are records established and maintained describing configuration items?
- Are audits of CM activities performed to confirm that baselines and documents are accurate?
- Do sponsor, program office, primary developer team, and sustainment organizations have CM systems and expertise? Are developers and managers trained equivalently on CM?
- Are CM resources across development team interoperable and compatible (i.e., use CVS, CAD/CAM, Requirements Management may represent logistical issues if left unmanaged)?

(List excerpt only)

Source: https://www.mitre.org/publications/systems-engineering-guide/acquisition-systems-engineering/configuration-management
Lesson Overview
Lesson Plan Status

- Overview
- System Engineering Processes
- Work Breakdown Structure (WBS)
- WBS Exercise
- Technical Reviews
- Systems and Software Engineering Challenges & Best Practices
- System/Software Development Requirements
- Configuration Management

Tower Exercise
• Refer to the Exercise folder for this lesson to conduct the Tower Exercise.
Overall: Given a DoD IT/SW acquisition scenario, apply System Engineering Lessons Learned, Best Practices and Rules of Thumb (Heuristics) when dealing with software.

• Describe the role of the Software Engineer (SwE) in supporting the Systems Engineer (SE).
• Describe the DoD Systems Engineering Process in the context of an IT/SW Development.
• Describe the different activities that comprise the SE/SW Technical and Technical Management processes.
  • Recognize that Software Reuse must be considered when engineering a software design and development.
  • Describe the requirements management process for successful software acquisition.
  • Describe the configuration management process for successful software acquisition.
• Given a DoD IT/SW scenario, recognize aspects of these engineering management principles of Requirements Management, Risk Management, Configuration Management and Interface Management to a development effort (SE/SW Tower Exercise).
Backup slides
From MIL HDBK 61A: For software, the design evolves through a software engineering process, using a variety of integrated tools, often called the software engineering environment, e.g., Computer-aided software engineering (CASE). The process results in computer based versions of documentation [See Activity Guide: Table 3-9. Software Documentation], source, and executable code for every CSCI [computer software configuration item]. The process the contractor employs to manage the automated software documentation (e.g., software library management and archiving) is similar to the process used to manage automated hardware documentation, although different tools may be employed. Upon close examination, it is fundamentally the same process used to manage the files which contain software code.

Example:

Multi-User ECP Automated Review System (MEARS) for Engineering Change Proposal
(ECP) automation – see next slide
MEARS is a flexible, Software as a Service (SaaS), application that specializes in Configuration Management, Change Management, and Contract Data Requirements List (CDRL) Management.

MEARS manages the creation, change and archive of all information in a centralized or distributed data repository (securely or in a secure environment) and provides easy access to all users via visual collaboration capabilities.

Developing complex products, manufacturing and maintaining them through their life requires managing evolving product configurations embodying information developed by multiple groups inside the organization and in the supply chain. The challenge of managing complex configurations can lead to costly errors and delays. MEARS provides a comprehensive methodology for managing the configuration (hierarchical set of information) of a product or system throughout its life.

Benefits:
• Government off-the-shelf (GOTS) Product
• Authorization to Operate (ATO)
• APMS registered and IRB/DBC approved
• DoDAF Conformance
• 100% Reimbursable
• Joint Interest Program

Compliance:
MEARS is compliant with DoD standards and policies, such as MIL-HDBK-61A and MIL-STD-973 and its successors. We are certified on the Air Force network, the Navy/Marine Corps Intranet (NMCI), and Army networks.