ISA 201
Information System Acquisition
Lesson 13
Systems Design Considerations
Learning Objectives

Today we will learn to:

Overall: Given a DoD IT/SW Acquisition scenario, apply appropriate design considerations for software-reliant system’s design.

- Identify the impacts of interface design and management on your system.
- Recognize the design impacts when acquiring safety-critical systems.
- Identify design and development best practices to reduce software assurance vulnerabilities.
- Describe how Modular Open Systems Approach (MOSA) impacts your systems design and business outcomes.
- Recognize the benefits of using Open Source Software (OSS) in your systems design.
- Identify the impacts of designing COTS into your system.
- Recognize the importance of data design and management practices on systems acquisition.
- Recognize the accessibility standards of Section 508, Title 29 U.S. Code need to be designed in for DBS and DoD Infrastructure systems.
- Describe how spectrum design can impact your system throughout the lifecycle.
Lesson Overview

Lesson Plan

- Interface Management
- Software Safety
- Software Assurance (SwA)
- Modular Open Systems Approach (MOSA)
- Open Source Software (OSS)
- COTS
- Data Design and Management
- Spectrum Management
- Accessibility
- Design Practices Exercise

Defense Acquisition Guidebook (DAG) Chapter 3 Systems Engineering
Design Practices

How the customer described it.

How the requirement was understood.

How the contractor designed it.

How the programmer wrote it.

How the PEO described it.

How the project was documented.

What SYSCOM installed.

How the government was billed.

How the help desk supported it.

What the user REALLY needed.
• Interface Management
  • Software Safety
  • Software Assurance (SwA)
  • Modular Open Systems Approach (MOSA)
  • Open Source Software (OSS)
  • COTS
  • Data Design and Management
  • Accessibility
  • Spectrum Management
  • Design Practices Exercise
• Interface
  - A boundary across which two independent systems meet and act on or communicate with each other

• Interface Management
  - The management of communication, coordination and responsibility across a common boundary between two or more organizations, phases, or physical entities which are interdependent.
  - All of the interfaces between co-functioning items need to be identified and documented so that their integrity may be maintained through a disciplined configuration control process.
Interface Management Imperatives

• Design/define and verify interfaces early in the Product Life Cycle
• Control interfaces to ensure they are within designated boundaries
• Design interfaces into program hierarchy
• Develop an Interface Management Plan (IMP)
• Develop Interface Control Working Groups (ICWG)
• Document interfaces and make available to stakeholders through Interface Control Documents (ICDs)
An IMP is a part of a configuration management plan that:

- Documents a system’s internal and external interfaces and their requirement specifications
- Identifies preferred and discretionary interface standards and their profiles
- Provides justification for selection and procedure for upgrading interface standards
- Describes the certifications and tests applicable to each interface or standard
ICDs and ICWGs

- **Interface Control Documents (ICDs)**
  - Documents interfaces
  - Formal means of establishing, defining, and controlling interfaces and for documenting detailed interface design definition
  - For each of the interface types, the ICD typically exists between configuration items to establish and specify interface definition and design

- **Interface Control Working Group (ICWG)**
  - A specialized technical working group comprised of appropriate technical representatives from the interfacing activities and other interested participating organizations.
  - Serves as a forum to develop and provide interface requirements, as well as, focus on interface detail definition and issues
  - Detailed description and requirements are provided in the ICWG Charter.
Lesson Overview

Lesson Plan Status

- Interface Management
- **Software Safety**
  - Software Assurance (SwA)
  - Modular Open Systems Approach (MOSA)
  - Open Source Software (OSS)
  - COTS
  - Data Design and Management
  - Accessibility
  - Spectrum Management
  - Design Practices Exercise
Software Safety

- Both a subset of Software Development and Software System Safety
- The process of identifying Safety-Critical Software and assessing the criticality
- A series of requirements, design, and verification activities used to demonstrate that software will meet the customer’s safety requirements to an acceptable level of confidence.
**Software System Safety**—The optimization of system safety in the design, development, use, and maintenance of software systems and their integration with safety-critical hardware systems in an operational environment.

**Safety-Critical**—A term applied to a function, condition, event, operation, process, component, or other element of a system whose proper recognition, control, performance or tolerance is essential to safe system operation or use (e.g., safety-critical function, safety-critical path, safety critical component).

**Safety-Critical Software Function (SCSF)**—A function, whose proper execution is necessary to reduce the risk of a hazard OR if performed incorrectly, performed out-of-sequence, or not performed could contribute to a hazard, compromise a safety mitigation feature, or result in a diminished level of safety.

**Mishap**—An unplanned event or series of events resulting in death, injury, occupational illness, or damage to or loss of equipment or property, or damage to the environment.

**Hazard**—Any real or potential condition that can cause:
1. Injury, illness, or death to personnel;
2. Damage to or loss of a system, equipment or property; or
3. Damage to the environment.
**Software System Safety**

**System Safety and Risk Assess Activities**

- System Safety Analyses to include software & hazard reports
- Perform hazard risk assessments
- Identify System Safety requirements
  - Identification of safety critical software functions, safety-critical software requirements (SCRs), and associated criticality
- Identification of risk mitigation and perform risk assessment
- Produce Safety Assessment Report

**Software Safety**

**Software Safety Development Assurance Activities**

- Participate in Hazard Analysis to identify specific S/W causes and faults
- Assist in identification of safety-critical software functions, SCRs, & criticality
- Flag SCRs & criticality in requirements traceability database
- Trace requirements to design & code
- Perform development assurance analyses and tests to ensure software meets (Level of Rigor) LOR & (Safety Integrity Level) SIL
- Provide V&V evidence to System Safety for final risk assessment
Most Software Safety problems come from vague or incomplete requirements that are misunderstood or misinterpreted.

Legacy/Reused software must be included in Software System Safety Analyses

Perform Software System Safety Analyses early in program lifecycle

Safety-Critical Software requirements derived from Software System Safety Analyses must be included and identified as Safety-Critical in the Software Requirement Specification (SRS)
• Interface Management
• Software Safety

Software Assurance (SwA)
• Modular Open Systems Approach (MOSA)
• Open Source Software (OSS)
• COTS
• Data Design and Management
• Accessibility
• Spectrum Management
• Design Practices Exercise
As you know, we must design security in from the beginning of your program (Bake security in).

Software assurance is a lifecycle activity that looks at software design from a threat perspective before and during design.

Examples of system and software security design considerations:

- Access control design: authentication and password design; communications security;
- Inputs/Outputs: data validation design; encryption mechanisms on data transfer (I/O)
- Session design: error handling, data protection, file management, memory management
- Report design: cryptographic requirements, aggregation of data, sensitivity markings

Knowing which secure coding language will be used so that modular design leverages those benefits.

We cover more details on SwA in our Cybersecurity lessons.
What is software provenance?

According to NIST SP 800-161:

Software provenance is the recording of system and component origin along with the history of, the changes to, and the recording of who made the changes.

• Acquirers and their system integrators should maintain the provenance of systems and components under their control to understand where the systems and components originated, their change history while under government control, and who might have had an opportunity to change them.

• Provenance allows for changes from the baselines of systems and components to be reported to specific stakeholders.

• Creating and maintaining provenance within the Information and Communications Technology (ICT) supply chain helps government agencies to achieve greater traceability in case of an adverse event and is critical for understanding and mitigating risks."

NIST SP 800-53, Rev 4, lays out security and privacy controls for Federal and DoD Information Systems. Originally there were 18 ICT SCRM control families or categories. Supplemental guidance established Provenance as the 19th control family.
What is software provenance? (cont’d)

- COTS suppliers (e.g., OEMs or authorized distributors) and external service providers may use provenance to demonstrate that the source of goods (e.g., computer hardware or software) is genuine and not counterfeit.

- Provenance is a new control and is likely to require additional resources to implement. Although some suppliers may collect and preserve certain aspects of component provenance for their solutions, they may not be able to share such data due to varying sensitivities.

- Criteria for collecting and preserving component provenance should be determined based on component or other reasons for keeping provenance, such as intellectual property.

- Provenance requires careful consideration based on the level of rigor and implementation. Agencies should assess the need for better understanding the level of effort that may be required for the acquirers’ ICT supply chain to provide this data because the cost/resource may likely be reflected in the cost to the acquirer.

- Factors driving up cost include the collection, documentation, and storage for such data, which may require additional protection if there are intellectual or security properties to protect.
DoDI 5200.44 defines Software Assurance as:
“The level of confidence that software functions as intended and is free of vulnerabilities, either intentionally or unintentionally designed or inserted as part of the software throughout the lifecycle.”

- Software assurance, along with Supply Chain Risk Management, are key focus areas within program protection and cybersecurity.
- Software provenance contains aspects of Software Assurance and SCRM. It is going to be far more difficult to control and maintain, especially with the reuse of code.
- Provenance is an essential element for Software Assurance (SwA)
- Some examples of other SwA elements include
  - Use of secure coding standards
  - Static and dynamic code assessment for weaknesses and vulnerabilities
  - Software supply chain security
  - Use of authentication in the software supply chain

**Software Provenance**  *Tells me the origin of the software and records how it has been changed throughout its lifetime.*

**Software Assurance**  *Provides a level of confidence that the software will do only what it’s supposed to do and that it doesn’t have vulnerabilities.*
• Interface Management
• Software Safety
• Software Assurance (SwA)

• **Modular Open Systems Approach (MOSA)**
• Open Source Software (OSS)
• COTS
• Data Design and Management
• Accessibility
• Spectrum Management
• Design Practices Exercise
Open Systems Architecture (OSA)*—OSA is a DoD-centric business and technical strategy for developing or modernizing (technology/weapons) systems that relies on modular design of components and utilizes open standards and specifications.

*Previously called OSA: Open Systems Approach

**Benefits of a Modular Open Systems Approach (MOSA)**

- Increased Competition
- Reduced Lifecycle Cost
- Open System Benefits
- Increased Innovation
- Faster & Less Costly Repairs & Upgrades
- Reduced Schedule
- Enhanced Interoperability

*Source: GAO analysis of DoD and industry data*
• An integrated business and technical strategy that:
  - employs modular design,
  - uses widely supported and consensus-based standards for their key interfaces, and
  - has been subjected to successful validation and verification tests to ensure the openness of their key interfaces.

• The DoD preferred approach for implementation of open systems is Open Systems Architecture (OSA).

DoDI 5000.02, 7 Jan 2015 states: “Program managers are responsible for applying open systems approaches in product designs ...”
DoD's definition of open systems is a system that has these **5 key principals:**

1. Employs modular design
2. Enterprise investment strategies
3. Lower development risk through transparency
4. Transformation of the life cycle sustainment strategies — Technology Insertion
5. Strategic use of data rights

**MOSA is a strategic “Business and Technical” acquisition approach** that leverages the commercial market-place in a way to control and optimize design features to ensure that a level-field of competition provides the best valued product for our war-fighter.
Example Modular Open Systems Approach (MOSA)

http://acqnotes.com/acqnote/careerfields/modular-open-systems-approach
Modular Open Systems Approach (MOSA)

**Why**

- Interoperability
- Tech Refresh
- Competition
- Innovation
- Cost Savings

**How**

- Modular Design
- Defined Interfaces
- Standards Process
- Accessible Data
- Open Interfaces
- IP Rights

**What**

- Modular Technical Design Approaches
  - Design severable modules
  - Define interfaces between modules
  - Publish consensus-based standards
  - Define, standardize & describe data models

- Open System Business Approaches
  - Use open standards & specs for interfaces
  - Recognize the relevant technical community
  - Acquire necessary data & license rights

Supporting the goals for MOSA implementation are methods, processes and tools which underpin the approach.
MOSA is a strategy for developing a new system or modernizing an existing one.

MOSA assists acquisition and engineering communities to design for affordable change, employ evolutionary acquisition and spiral development, and develop an integrated roadmap for system design and development.

Basing design strategies on widely supported open standards increases the chance that future changes to the system will be integrated in a cost-effective manner.

http://www.acq.osd.mil/se/initiatives/init_osoa.html
• The guidebook provides recommendations for writing an OSA-based statement of work, guidance on special interest requirements, recommended contract line items, and guidance on obtaining intellectual property and data rights to support full life-cycle competition and recommended CDRLs.
• Interface Management
• Software Safety
• Software Assurance (SwA)
• Modular Open Systems Approach (MOSA)

Open Source Software (OSS)
• COTS
• Data Design and Management
• Accessibility
• Spectrum Management
• Design Practices Exercise
Free and Open Source Software (FOSS)

• Open source doesn't just mean access to the source code:
  - Free redistribution
  - Source code
  - Derived works
  - Integrity of the author's source code
  - No discrimination against persons or groups
  - No discrimination against fields of endeavor
  - Distribution of license
  - License must not be specific to a product
  - License must not restrict other software
  - License must be technology-neutral
• Well known:
  - Mozilla (Firefox, Thunderbird, Sunbird, Seamonkey)
  - Java
  - Open Office
  - Linux
  - Apache
  - Perl
  - Python
  - Ruby
Who's Using FOSS

- Every branch of US government
- Google
- Yahoo
- Ticketmaster
- Hackers
- IBM
- Sun
- Oracle
- Regular people—you!
FOSS Advantages

• Try Before You Buy
• Security (also a disadvantage)
• Quality
  - A software package created by a handful of developers, or a software package created by thousands of developers?
• Cost
• Support Options
• Interoperability
• Code Coverage Tools
FOSS Disadvantages

- Security (also an Advantage)
- Probably not optimized for the functions you need performed.
- May require sophisticated professional guidance on licensing issues
- May have little documentation
- “Hidden” costs
  - Hosting, training, etc.
- Project longevity/support
  - Developers could lose interest and leave
- No one is obligated to help you
Lesson Overview

Lesson Plan Status

- Interface Management
- Software Safety
- Software Assurance (SwA)
- Modular Open Systems Approach (MOSA)
- Open Source Software (OSS)
- COTS
- Data Design and Management
- Accessibility
- Spectrum Management
- Design Practices Exercise
COTS Defined

- A COTS product is a product sold, leased, or licensed to the general public
- Offered by a vendor trying to profit from it
- Supported and evolved by the vendor, who retains the intellectual property rights
- Available in multiple, identical copies
- Used without modification of the internals
- * Any given version of a COTS component will reach eventual obsolescence or end of life in which it will no longer be supported by the vendor
- Almost always requires **“glue code”** (middleware) to integrate into existing software architecture

**Glue Code**—code that does not contribute any functionality towards meeting the program's requirements, but instead serves solely to "glue together" different parts of code that would not otherwise be compatible.

*Source: Added Sources of Costs in Maintaining COTS-Intensive Systems, Clark, CrossTalk, June 2007*
Universal Truth #9: COTS are not necessarily the best solution. They bring risks + benefits…understand both!

Management Emphasis

• Investigate the pricing structure
• Select established products with large installed bases
• Budget for the complete cost of COTS integration and maintenance
• “Fly before you buy” + TEST a lot
• Adapt your requirements to COTS

Elephant Bungee Jumping #9: Avoiding Diseases that Are Fun to Catch
COTS—Sources of Added Costs (Compared to custom applications)

- Licensing
- Evaluation of New Releases
- Defect Hunting
- Vendor Support
- Upgrade Ripple Effect
- Hardware Upgrades
- Disabling New Features
- Early Maintenance
- Market Watch
- Continuous Funding

Source: Added Sources of Costs in Maintaining COTS-Intensive Systems, Clark, CrossTalk, June 2007
Different COTS System Compositions

- **COTS Solution Systems**
  - Typical business or standard IT systems
  - Major COTS component is essentially the system
  - Has its own architecture
  - Has internal business logic that must be followed to be used

- **COTS-Intensive Systems (CIS)**
  - Comprised of many COTS components
  - Often safety and performance critical systems
  - User interface and data transmission may be handled by many components
  - Interact with each other through custom-developed “glue code” using vendor provided application program interfaces and with custom-developed application code

*Source: Added Sources of Costs in Maintaining COTS-Intensive Systems, Clark, CrossTalk, June 2007*
• **System obsolescence, technology refresh, and upgrade planning**
  - Each COTS software product life cycle includes updates, refreshes, and obsolescence (i.e., unsupported releases).
  - Life cycle is not based on the users’ requests or budgetary cycles, but rather on marketplace demands and COTS software vendors’ business plans.

• **Source code escrow**
  - Source code may be owned by the COTS vendor or the third-party integrator.
  - Problems can arise when the COTS vendor goes out of business or no longer exists due to a business merger or acquisition.

• **Vendor license management**
  - During development, licenses may be managed by the system integrator.
  - The transition of license management tasks to the sustainment organization needs to be jointly planned by the program office and sustainment organization.

• **Architecture and COTS software interfaces**
  - During system development, third-party integrators/developers may capitalize on relationships with COTS software vendors to acquire system-specific capabilities.
  - These capabilities may not be in the official version of the product and there is no guarantee that these “extra” features will be maintained as the product evolves.
Universal Truth # 8: Never allow developers to modify COTS software products

Management Emphasis

• JUST SAY NO!
• An absolute prohibition
• Never pay a COTS vendor to specifically tailor or modify their product for you

Elephant Bungee Jumping # 8: A Poke In the Eye With A Sharp Stick
Lesson Overview

Lesson Plan Status

- Interface Management
- Software Safety
- Software Assurance (SwA)
- Modular Open Systems Approach (MOSA)
- Open Source Software (OSS)
- COTS

Data Design and Management

- Accessibility
- Spectrum Management
- Design Practices Exercise
Data Design means designing the data such that the Data Source and Data Consumer can exchange information seamlessly.

- DoD CIO memo (March 2013) declares intent to adopt NIEM
- Finds that NIEM is the best suited option for standards-based data exchange
- DoD organizations shall first consider NIEM for information sharing solutions
  - Exceptions to NIEM are expected
  - Will be approved when warranted

About NIEM
Data Design: A Standards-Based Approach using an IES Design

Data Interoperability

Information Exchange Specification (IES)
build-time description of the data to be exchanged

IES
defines

System / Application

Data Producers

Information Exchange Package (IEP)
the data exchanged at runtime

System / Application

Data Consumers

Developers

Understanding
• NIEM is emphatically not a single comprehensive data model for all data exchanges

• A system does not simply “implement NIEM” and thereby become completely interoperable with every other system “implementing NIEM”

• A system can implement a particular NIEM-conforming information exchange specification

• All systems implementing a particular IES are interoperable with each other, for that exchange

In NIEM, interoperability is defined at the IES level
“Consider NIEM First” Policy

• The key driver in DoD adoption is “Consider NIEM First”
• The “NIEM First” rule tells the program developers to either:
  - Provide a NIEM-based interface for every new exchange, or
  - Obtain an exception for their preferred alternative approach
• Exceptions are granted by the DoD CIO
  - Justified by the business case for the alternative
  - Through a process that is simple and easy as it can be
• You don’t have to retire existing non-NIEM exchanges
• You don’t have to replace an existing data exchange just to make it conform to NIEM
  - Even if you are modernizing the implementing system
  - So long as the exchange specification doesn’t change
• You can use any standard you like for a new exchange
  - Either by obtaining an exception
  - Or by providing a NIEM interface along with your favorite
Data Design (NIEM) Summary

• NIEM is a standards-based approach for defining machine-to-machine data exchanges
  - Repeatable process
  - Reusable components
  - Technical specifications and tool support

• The DoD is adopting NIEM in order to help with the data interoperability goals in its Net-Centric Data Strategy

• Expect increased DoD participation in all NIEM domains (not just MilOps)

• Expect a slow and steady increase in NIEM-based data exchanges, beginning in FY15
• **Definition**
  - The process of applying policies, procedures, and tools for the identification and control of data requirements, for assuring the adequacy of data and for facilitating the timely, economical acquisition and availability of data, including digital delivery or access. In simple terms, DM is the process for the acquisition of data (access or delivery) through contractual vehicles, so that data is available for use by authorized users. The type of data to which this applies includes research and development, acquisition, and logistics information.

• Critical to the process of system design and development (e.g., data design specifications (NIEM)).

• Data management and standardization reduces the cost, complexity, and overall level of resources expended on the development of software and computer system data components
Software Development and Data Rights

• Data rights refer to intellectual property regarding the use of the data developed, accessed, and/or delivered under a Government contract.
• Involve proprietary, restrictive, Government purpose, unlimited, limited, and may include patents, copyrights, and other data right provisions.
• Necessary in the determination of release, duplication, and disclosure of technical data.
• Data rights are generally determined by whose money is used in the development of the data.
  - If the data is developed with Government funding, then the Government has the right to access and receive the data with unlimited rights.
  - If data is developed with private sector funding, the Government, generally, will be allowed Government Purpose Rights.
• The FAR 52.227 and DFARS 252.227 and other related sections set forth the policies, procedures, and implementing instructions relating to the requirements for the acquisition of technical data and computer software.
• The Government should review the validity of any asserted restriction on technical data under a contract before acceptance of the data, but not later than three years after data delivery or final payment, whichever is later.

Source: Acquisition Data Management: Body of Knowledge (acc.dau.mil)
Data is an essential enabler of network-centric warfare (NCW) and shall be made **visible**, **accessible**, and **understandable** …

Data assets shall be made **visible** by creating and associating metadata (“tagging”) …

Data assets shall be made **accessible** by making data available in shared spaces.

Data assets shall be made **understandable** by publishing associated semantic and structural metadata in a federated DoD metadata registry.

To **enable trust**, data assets shall have associated information assurance and security metadata, and an authoritative source … shall be identified …

Data **interoperability** shall be supported by making data assets understandable and by enabling business and mission processes to be reused where possible.

Semantic and structural agreements for data sharing shall be promoted through communities (e.g., communities of interest (COIs)), …

Data sharing concepts and practices shall be incorporated into education and awareness training and appropriate DoD processes.
Lesson Overview

Lesson Plan Status

- Software Assurance
- Interface Management
- Software Safety
- Software Assurance (SwA)
- Modular Open Systems Approach (MOSA)
- Open Source Software (OSS)
- COTS
- Data Design and Management

Accessibility

- Spectrum Management
- Design Practices Exercise
What is the purpose of this law?

The purpose of Section 508, Title 29 U.S. Code paragraph 794.d, is to provide disabled employees and members of the public access to information that is comparable to access available to others (unless undue burden would be imposed on the agency).

Why is this important to acquisition professionals?

1) Addresses electronic and information technology (EIT) procurement, development, maintenance and use.
2) Section 508 standards apply to people with visual, auditory, physical, speech, cognitive and neurological impairments.
3) Section 508 lists the types of technologies that must be included in designing accessible systems.
4) Section 508 identifies the exceptions to 508 accessibility requirements.

(Accessibility guidance is also addressed in FAR Subpart 39).
When acquiring EIT, agencies must ensure that:

**Federal employees with disabilities** have access to and use of information and data that is comparable to the access and use by Federal employees who are not individuals with disabilities; and;

**Members of the public with disabilities** seeking information or services from an agency have access to and use of information and data that is comparable to the access to and use of information and data by members of the public who are not individuals with disabilities.

**EIT Section 508 Compliant list**

- Software applications and operating systems
- Video and multimedia products
- Self-contained, closed products (e.g., information Web-based information or applications, kiosks, calculators, and fax machines)
- Telecommunication products
- Desktop and portable computers
Lesson Overview

Lesson Plan Status

- Interface Management
- Software Safety
- Software Assurance (SwA)
- Modular Open Systems Approach (MOSA)
- Open Source Software (OSS)
- COTS
- Data Design and Management
- Accessibility

**Spectrum Management**

- Design Practices Exercise
Spectrum Supportability

Addresses the availability of sufficient electromagnetic spectrum for the development, training, and compatible operations of spectrum dependent systems in their intended operational environment. Spectrum supportability determination required by statute at MS B.

Vision: All systems fielded can obtain spectrum assignments and operate in such a way as to provide the capability (to the warfighter) needed when the requirement was generated.

Consideration of spectrum and control of electromagnetic environmental effects are essential and integral to successful program management, system development and fielding of global capabilities.
• Today, most military systems depend on the spectrum to interoperate.
• DoD does not own all of the spectrum in the United States.
• We don’t own the spectrum in foreign countries.
• Managing electromagnetic spectrum achieves electromagnetic compatibility for smooth operational capabilities.
• Continuous spectrum management is necessary due to continuous competition to use spectrum.
• Loss of spectrum means loss of your ability to communicate and interoperate.
• Spectrum management is critical to operational mission readiness.
Early in your program, use DoDI 4650.01, January 9, 2009, “Policy and Procedures for Management and Use of the Electromagnetic Spectrum,” to:

1. Identify spectrum-related risks as early as possible via spectrum supportability risk assessments.
2. Review these assessments at acquisition milestones.
3. Manage the risks throughout the system’s lifecycle.

Early in your program, use DoDI 5000.02, January 7, 2015, “Operation of the Defense Acquisition System,” to:

1. File the Frequency Allocation Application (DD Form 1494) IAW Enclosure 1, Table 2.
2. Create and manage the Spectrum Supportability Risk Assessment IAW Enclosure 1, Table 2.

Work with the Defense Spectrum Office (DSO) to help guide you through the complex spectrum development, management and approval processes for all possible deployment scenarios.
• Interface Management
• Software Safety
• Software Assurance (SwA)
• Modular Open Systems Approach (MOSA)
• Open Source Software (OSS)
• COTS
• Data Design and Management
• Accessibility
• Spectrum Management

• Design Practices Exercise
• Each team prepare a short (10 min) presentation for the class. Each team will propose a method to develop / implement DTS2 based on the following:

- Team 1—Legacy (existing)
- Team 2—Custom software
- Team 3—Free and Open Source (FOSS)
- Team 4—Act as MDA (see Instructor)
- Team 5—Commercial off the Shelf (COTS)

• Address how would your design choice impacts each of the following given the considerations listed: **system cost, schedule, performance and risk**?

• Include an assessment of you ability to **meet the KPP’s**

• Use the available lesson materials or other resources to develop your brief (DAG [Ch 3, 6], AKSS, Google, etc.)

• Read “Design Consideration Exercise DTS 2”— Word document

• Presentations start in 30 minutes.
Overall: Given a DoD IT/SW Acquisition scenario, apply appropriate design considerations for software-reliant system’s design.

- Identify the impacts of interface design and management on your system.
- Recognize the design impacts when acquiring safety-critical systems.
- Identify design and development best practices to reduce software assurance vulnerabilities.
- Describe how Modular Open Systems Approach (MOSA) impacts your systems design and business outcomes.
- Recognize the benefits of using Open Source Software (OSS) in your systems design.
- Identify the impacts of designing COTS into your system.
- Recognize the importance of data design and management practices on systems acquisition.
- Recognize the accessibility standards of Section 508, Title 29 U.S. Code need to be designed in for DBS and DoD Infrastructure systems.
- Describe how spectrum design can impact your system throughout the lifecycle.