

AIR Domain Software Reuse Repository and CM Strategy

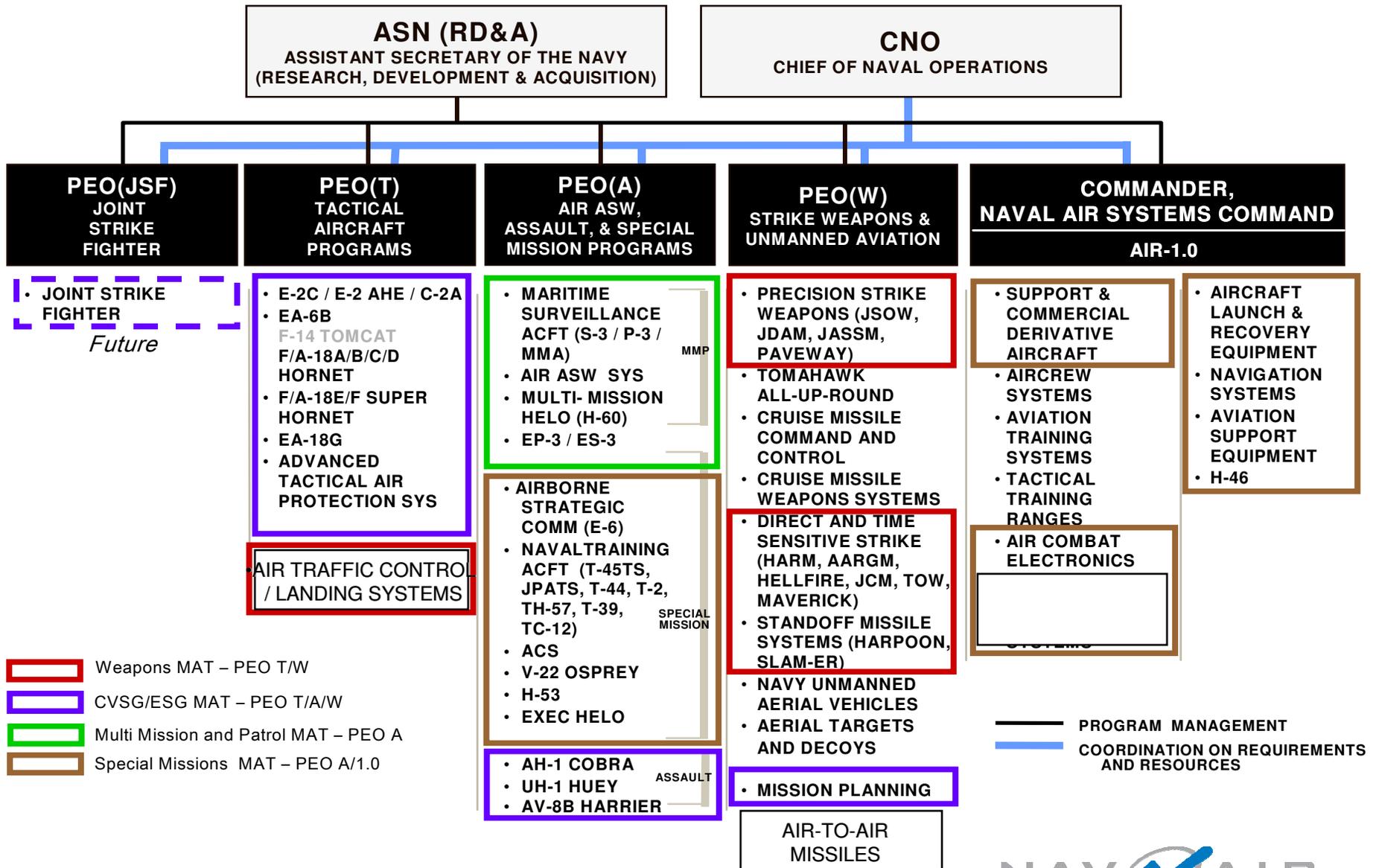
Karl C. Geist
AIR Domain Representative

21 February 2006

Agenda

- Mat Overview and Discussion
- Software Reuse Questions
- General Software Reuse Opportunities
- Specific NAVAIR Examples, Initiatives, or Philosophies
 - PMA-209 – Rex Coombs
 - PMA-281 – Mike Stine
 - PMA-205 – Ken Reams

NAVAIR Mission Area Teams (MAT)



MAT Vision

Customer Focus

- MAT vision and strategic plan are aligned with PEO, PMA, and NAVAIR Enterprise goals
- MAT vision and goals support and reinforce program decisions made within the PMA
- **MAT strives to constantly improve value of software acquisition, development, and sustainment throughout a product's lifecycle**
- MAT is accountable to PEOs / PMAs, and NAVAIR Enterprise for performance using objective, fleet driven metrics
- MAT is an agile organization better able to align resources with customer and fleet requirements in an environment of change
- MAT strives for joint solutions for integration of new capabilities across multiple platforms, yet at the same time maintains product-line specific domain expertise and touch points

MAT Goals

Process / Performance Improvement

- Continuous performance improvement is a key element of MMP MAT operations
- CMMI is used as a framework for systems /software process improvement
- **MAT strives for commonality where commonality makes sense**
- MAT performance is both predictable and competitive
- MAT processes are measured; metrics are used to manage the product

Work Culture and Environment

- Become an organization of collaboration and stewardship
- Migrate from a risk averse culture to a mindset of proactive risk identification and mitigation

MAT Configuration

- **A MAT is a group of IPTs/SSAs that acquire, develop, and maintain software products and services for a group of PMAs.**
- Each MAT's size varies greatly and is designed to be flexible in adapting to increases or decreases in product/platform tasking based on a common set of processes and tools. The primary responsibility of the MMP MAT is to work closely with the PMAs to develop an integrated product solution.
- Each MAT's scope of activities is based on product and PMA needs and may vary from non-recurring mission systems software development and contracts to acquisition engineering support expertise.
- Typical MAT responsibilities include development, maintenance, and integration of services and work products that are associated with the specified product(s).

MAT Configuration (cont.)

- Each MAT relies on core engineering disciplines such as systems/software engineering, software/system integration and test engineering to accomplish its mission.

- **A MAT has a single Leadership, Administration, Business Operations, and Process Team to support functions such as business operations; contracting, data reduction, program management, laboratory facilities, Quality Assurance (QA), Configuration Management (CM) and process improvement**

- **A “Fair Share” cost model is employed for shared resources within a MAT such as facilities, process improvement and tools.**

MAT Concept of Operation

- Each MAT utilizes a Leadership Team to plan, coordinate and administer overall MAT operations.
- The MAT Lead and Leadership Team work collaboratively to ensure and promote commonality of tools, processes, functions and sharing of resources within the MAT.

- **Initially, MATs will have very few common processes or tools within the MAT**

- **Reuse and consistent application will expand as each MAT matures and becomes a tight and cohesive organization.**

Common MAT Support

- **NAVAIR Software/System Support Center (NSSC) is a common support team for use by all the MATs**
- **The NSSC integrates several existing Enterprise Teams (ETs) into a single, collaborative entity that spans the NAVAIR organization in supporting innovation and improvement at all levels.**
- **The NSSC is the primary MAT resource in improving systems and software engineering and acquisition knowledge, process, product and organizational performance.**
- **MATs and projects/platforms acquire services and products provided by the NSSC as needed on a fee for service basis.**
- **The primary areas of NSSC expertise are:**
 - **Knowledge of acquisition, development, and maintenance of software intensive systems.**
 - **Systems and software performance improvement, data repository (measurement and process artifacts), measurement analysis and reporting.**
 - **Communication, collaboration and organizational change management**
 - **Program Related Engineering (PRE) funding management and coordination**
 - **Communication with Fleet representatives (e.g., TYCOMs) for issue identification, coordination and resolution via the chartered NAVAIR Fleet Software Working Group**

NSSC Resources

- **System Engineering Resource Center (SERC)**
 - **SERC has multiple areas of responsibility in systems engineering that cover more than software acquisition and engineering.**
 - **Several of these areas are of common interest to functions performed by the NSSC.**
 - **These common areas of interest will be the focus of SERC involvement with the NSSC.**
 - **Examples of common areas include:**
 - **close coordination of systems and software engineering processes**
 - **process and product measurement and analysis**
 - **implementation of the Software Engineering Institute (SEI) Capability Maturity Model Integration (CMMI) as our process improvement model for software and systems engineering**
 - **evaluation of tools for use on projects.**
 - **Although an external organization to the NSSC the SERC is available to support each MAT as needed.**
- **Software Resource Center (SRC)**
 - **The Software Resource Center (SRC) helps NAVAIR teams improve the acquisition, development and lifecycle maintenance of software intensive systems..**
 - **Key focus areas of the SRC to promote corporate performance improvement are:**
 - **Software Intensive Systems Acquisition**
 - **Software Intensive Systems Development and Maintenance**
 - **Fleet Support**

NSSC Resources (cont.)

- **Program Related Engineering (PRE) Technical Management**
 - The Program Related Engineering (PRE) Tactical Systems Software (TSS) team provides technical oversight, management, and coordination within NAVAIR of the Operations and Maintenance, Navy (O&M,N) budget that funds life-cycle support for Naval aviation tactical weapons systems software. This team has been in operation for over 10 years and has detailed processes in place that will continue to operate under the NSSC construct.
- **NAVAIR Fleet Software Working Group (NFSWG)**
 - The NAVAIR Fleet Software Working Group (NFSWG) is in place to foster communication and alignment between the Fleet and NAVAIR on key software acquisition, development and sustainment issues. The NFSWG focuses on tactical (1-2 year) software issues and liaisons with other groups on non-tactical issues. In addition to issue identification, the NFSWG uses available resources to:
 - Resolve problems of a tactical (vice strategic) nature
 - Reduce costs associated with software product development, deployment, and maintenance, and
 - Assist in determining tradeoffs between current and future readiness.
 - The NFSWG coordinates their efforts with the Software resource Center and the Software Leadership Team.
- **People, Process & Product Resource (P3R)**
 - The People, Process, and Product Resource (P3R) will continue its mission to enable adaptation, acceleration, and alignment of individual, team and organization improvement efforts. P3R defines their products and services to meet needs NAVAIR groups continually improving their performance. P3R team members are dedicated full time as internal change agents, facilitators, change and process consultants, and coaches working cooperatively with all NAVAIR improvement initiatives to improve performance.

What Questions Should We Address ?

- **What is Reuse?**
- **What is Software Reuse?**
- **Why Software Reuse?**
- **Why Not?**
- **How do We Reuse Software?**
- **How do We Get Started?**

What is Reuse ?

- **To use again especially after reclaiming or reprocessing**
- **Further or repeated use**
- **Note: This infers there is value remaining, thus not a through away item**

What is Software Reuse ?

- **Software reuse is the process utilizing existing software as the basis or starting point for your next “Software Product”.**
- **“Software Product” includes all things produced from a software development effort, e.g: requirements, proposals, specifications, software design, manuals, test suites, source code, software objects, applications, and related documentation**

Why Software Reuse ?

- **Provides a head start (Don't reinvent the wheel), thus saving development time for other efforts, or getting product faster**
- **Increases product quality and maturity for testing, usually yield fewer defects which results in higher quality**
- **Risk reduction by “working on more familiar ground”, and help provide repeatability**
- **Possibility of cost reduction as a result of the factors previously mentioned**

Why Not ?

- Initial investment “Cost of getting the ball rolling”. “Not my dime!”
- Data Rights issues
- Still a level of “known territory”
- High cost to stand up/implement reuse repository
- Thought “as more expensive” than new software
- Invasion of “rice bowl”
- Legacy equipment syndrome, how do we keep up with technology?
- Original design limitations

How Do We Reuse Software ?

- **Examples:**

- **Most common for AIR Domain**

- **Next Operational Baselines (Platform series – all PMAs)**
 - **Common integration (across platforms, e.g. weapons, sensors, etc. - PMA-201, PMA-209)**
 - **Common uses (Operational use related to training – PMA-205)**
 - **Joint Arena Requirements – Mission Planning – PMA-281)**

How Do We Get Started ?

- **Visionary planning (Think and look ahead, invest, and implement the sustainment strategy form the start)**
- **Wise investment (Spend the \$ today that saves the \$10 later – Note: \$10 you probably will not have later)**
- **Establish standards and mandate compliance**
- **Bite size chunks (“Don’t eat the whole elephant” at one seating – perform manageable tasking gradually increasing the challenges)**
- **Step on toes, force the issue, and break some rice bowls”**
- **Publicize successes (Low hanging fruit – make it attractive)**
- **Offer help, training, and advice**

Software Reuse

Rex Coombs
NAVAIR PMA-209

NAVAIR Software Reuse - DEFINITION

- **Software reuse is the process of implementing or updating software systems using existing “Software Products”.**
- **“Software Products” include all things produced from a software development effort.**
 - **Examples: requirements, proposals, specifications, design info, user manuals, test suites, source code, software objects, COTS or third-party applications within a larger system, and related documentation**

Software Reuse - BENEFITS

- ***Reuse Benefits – Don't reinvent the wheel***
 - **Reduced cost**
 - Shorter development time and fewer defects, mean lower cost.
 - **Shorter development schedule**
 - Finished product available earlier
 - **Lower risk of program failure**
 - Schedule, cost and quality
 - **High quality products and accelerated product maturity**
 - Repeated “Software Product” reuse results in the development of reliable, high quality products with fewer defects

Software Reuse – PAST WINS

“Pockets” of success

- **Multiple products, developed by the same supplier, hosted on the same or “similar” hardware system**
 - F/A-18 AYK-14 Operational Flight Program and AV8B AYK-14 Operational Flight Program use common subroutines
- **Common products used across multiple platform users**
 - In flight diagnostic software used in AYK-14 Operational Flight Programs (F-14D, F/A-18)
- **Algorithms as opposed to software**
 - Ground Proximity Warning System
 - Multiple host systems
 - Multiple platforms
 - Use of proven algorithm
- **Software products written in High Order Language**
 - Across multiple programs under one PMA umbrella, software CSCIs reusable

Software Reuse – PAST WINS (cont.)

“Pockets” of success (cont.)

- **SSAs that develop a library for reuse (Contractor and Government)**
 - Ground Proximity Warning System – V2.4 upgrade.
 - Goal – 90% reuse for upgrade of software capability across multiple platform users
 - » Achieved greater than 90% reuse for requirements, algorithms and documentation
 - » Achieved 76% reuse for source code
 - » Achieved 83% reuse for test case and test software development effort
- **Processor technology evolution with software backward compatibility as a requirement**
 - AN/AYK-14(V) – 30 years and 3 generations of processors with hundreds of hardware design changes – OFP subroutines are backward compatible
- **Mandated Programming Languages**
 - Ada
- **Libraries and tools**
- **Documentation**
- **Reuse seems to have worked better for small, experienced teams**
 - Better communication brings opportunity
- **A variety of software libraries and repositories exist today**

Software Reuse - BARRIERS

- **Reuse Barriers**

- **Data Rights issues can prevent reuse of “Software Products” between programs.**
- **High cost to stand up/implement reuse repository**
 - ROI will be worth it, if implemented and used properly
- **Perception that there is high cost to modify software CSCI reuse candidates**
 - Poor reuse candidate choices can give reuse a bad name. Carefully select reuse candidates.
- **Reuse deemed by some to be more expensive than new software - Myth or Reality?**
 - Takes 1.5 times the effort to develop reusable software
 - Takes 20% as much effort to reuse than to develop new
- **Legacy equipment used “machine” unique “Assembler Language” or “Machine Code”.**
 - Limited opportunity for reuse for “other” equipment

MAT CREATED TO HELP BREAKDOWN BARRIERS

Software Reuse – BARRIERS (cont.)

- ***Reuse Barriers (cont.)***

- **Software designs that are not conducive to reuse**

- Short term cost savings encourages teams to take the wrong approach
- Need modular designs
- Need software objects that perform a reusable function
- Need software designed and generated in a clearly defined, “open” way, with concise interface specifications, understandable documentation.

- **Lack of practitioner training and technical skills to develop software reuse candidates**

- **Lack of practitioner training and technical skills to perform analysis and find the right reuse candidates**

- **No organized repository with right level of identification and associated documentation**

- **Politics and “protecting our business” is a barrier to proactive development of reusable software by industry.**

MAT CREATED TO HELP BREAKDOWN BARRIERS

Software Reuse – BARRIERS (cont.)

- ***Reuse Barriers (cont.)***
 - No incentive to generate Software Products for reuse, or to reuse.
 - No management and process support for reuse concept.
 - Lack of formal software engineering staff training needed to successfully implement and execute software reuse.
 - Lack of project management and software management training for both technical and non-technical aspects of software reuse.
 - No organizational or team group that is responsible for the maintenance of the reuse infrastructure.
 - No project level responsibility for the acquisition and maintenance of reusable components for the project.
 - No clearly defined process with process team to assist organizations to identify reuse candidates and find items to reuse
- ***Adequate resources and funding must be provided to perform software reuse tasks***

MAT CREATED TO HELP BREAKDOWN BARRIERS

Software Reuse – BARRIERS (cont.)

- *Reuse Barrier related questions:*
 - Reuse has worked in “pockets”, and generally for “small” teams. How do we broaden reuse opportunity across teams and organizations?
 - How do we fund the standup of an organizational enterprise that organizes and manages software available for reuse?
 - How do we institutionalize rules for developing code so that it is reusable?
 - How do we motivate individual teams to develop reuse friendly software at a higher short term software development cost?
 - Note that the benefit realized might not even be for the developer’s own team
 - How would a developer find the piece of software he’d like to reuse when we broaden opportunity across teams or organizations
- *Overcoming barriers is critical to implementation of a successful software reuse program.*

MAT CREATED TO HELP BREAKDOWN BARRIERS

Software Reuse - RECOMMENDATIONS

- ***Software Reuse Recommendations***
 - **Ensure contractual Data Rights allow software reuse**
 - **Design software in functional units that are conducive to reuse**
 - Need modular designs
 - Need software objects that perform a reusable function
 - Need software designed and generated in a clearly defined, open way, with concise interface specifications, understandable documentation.
 - **Promote the use of widely used programming languages to maximize opportunity**
 - **Focus where biggest ROI can be achieved**
 - **Set tactical and strategic goals and use spiral design approach.**
 - Start small but with immediate benefit to the organization.
 - Get quick wins while working toward long term big wins.

Software Reuse – RECOMMENDATIONS (cont.)

- ***Software Reuse Recommendations***

- **Include design algorithms, applicability information, detailed design information and documentation along with source code in common repository.**
- **Don't reinvent the wheel. Team with other orgs with repositories.**
 - Provide links to other repositories
- **Repository should contain products/information that addresses all levels of reuse**
 - Links to info regarding COTS products that might be used in lieu of software development
- **Promote the use of common operating systems and host systems to allow best opportunity**
- **Restrict software developers to “standard” language instructions.**
 - Avoid language extensions unique to particular compilers

Software Reuse – RECOMMENDATIONS (cont.)

• *Software Reuse Recommendations*

- **Train practitioners to develop reusable software**
 - The training needs to be standardized and institutionalized.
- **Develop clear criteria for what should be designed for reuse and what shouldn't be**
 - Plan and training needed
- **Stand up consultant team that works with projects to ID reuse candidates and to help decide whether submittal into the repository is appropriate**
- **Quantify benefits**
 - Show that reuse results in reduced cost and reduced schedule risk to programs
- **Devise incentives for those who reuse available software**
 - Source selection process should give special attention to proposals that include reuse
 - Incentives for practitioners both organic teams and contractor organizations

Software Reuse – RECOMMENDATIONS (cont.)

- ***Software Reuse Recommendations***

- **Promote use of NDI/COTS products to reduce cost**

- Provide info and links to owners of these products

- **Advertise the wins and promote the benefits**

- **Identify existing repositories and make links readily available to developers and managers**

- Industry, AF, Army, Navy repositories

- Rex's - Informal survey of practitioners and SSA managers (2/06)

- 5 SSAs surveyed

- »1 sited repositories they've attempted to use outside of their organization (Army Reuse Center, etc.)

- »No common understanding of available repositories

Software Reuse – RECOMMENDATIONS (cont.)

- ***Software Reuse Recommendations***

- **Include the NAVAIR Mission Area Team (MAT) in this effort**

- New NAVAIR enterprise team with focus toward efficient effective use of limited NAVAIR software dollars
- Opportunity to organize and promote reuse across the NAVAIR software enterprise

- **NAVAIR System Software Center (NSSC)**

- Consultation support available to Navy teams

Software Reuse – RECOMMENDATIONS (cont.)

- **Software Reuse Recommendations**
 - *There is significant ROI potential for software reuse*
 - *A software repository is not a silver bullet to successful software reuse. There must be communication, management buy-in, training and ongoing product team consultation to implement an effective reuse program.*
 - *“Software Products”, including proposals, specifications, design documentation, user manuals, test suites, source code, software objects, links to COTS or third-party applications within a larger system, other documentation, etc. are all candidates for reuse.*
 - *Start small with quick wins. Evolve to big wins.*

Service-Oriented Architecture

Mike Stine

PMA-281

What are we trying to achieve?

- Today in the Navy we are surrounded by buzz words that seem to herald a new era in fleet network operations. Firm data seems to be lacking. So, what are we actually trying to achieve. Buzz words include:
 - Web Enabled
 - Web Services
 - Service Oriented Architecture (SOA)
 - NCEs
 - FORCEnet

Some Definitions:

- **Web Enabled:** An application that runs on a server and can be executed via a web browser, thus reducing the number of copies needed to be licensed and distributed.
- **Web Services:** Machine to machine (or application to application) communication via SOAP messages, not associated with browsers.
- **Service Oriented Architecture (SOA):** An architecture where information is exchanged between apps. at the application level (as compared to the component level) and the information that an application can make available to other users is “Published”- made public
- **Network Centric Enterprise Services (NCES):** The DoD implemented standards and services to enable a DoD wide Service Oriented Architecture, (the ES in GIG-ES). Primarily a tailored web services arch.
- **FORCEnet:** The Navy’s implementation of the Network Centric Operations and Warfare (NCOW)

We are trying to achieve a Service-Oriented Architecture

- What are the faces of a Service-Oriented Architecture:
- Software language implementations:
 - J2EE (<http://java.sun.com/j2ee/>)
 - .NET (<http://www.microsoft.com/net/>)
- Industry implementations:
 - BEA WebLogic Server (<http://www.bea.com/>)
 - IBM WebSphere
 - Sun' Java Web Services Development Program (JWSDP)
 - A myriad of others, some surprisingly good!

Standards

- Service-Oriented Architecture is unique in that it is achieved through the application of non-proprietary standards. (More importantly it appears both Sun and Microsoft have agreed to implement the same standards). Additionally, it abstracts (hides) functionality at the application level vice the component or subroutine level.
- These standards will allow one system to communicate with another regardless of hardware platform and operating system

Key Standards

- What standards
 - SOAP: Simple Object Access Protocol
 - Original name, SOAP now considered a specification name, not an acronym
 - WSDL: Web Services Description Language
 - UDDI: Universal Discovery Description Integration
 - XML: Extensible Markup Language
 - HTTP: Hypertext Transport Protocol
- SOAP, WSDL, XML and HTTP are managed by:
 - W3C: <http://www.w3.org/>
- UDDI is managed by:
 - OASIS: www.oasis-open.org/
 - More info: <http://www.uddi.org/specification.html>

What does this Alphabet Soup of Standards Really Mean?

- Objective – Request or transmit information over an interface
 - SOAP is the name of the message used for this communication.
 - It is an XML enabled message both for requests and responses. These functional calls and their data structures can be defined in a WSDL
 - XML based – read by an XML Parser
 - Sent/Received via http, usually via port 80
 - WSDL are essentially interface definitions
 - Akin to API documentation
 - Requests for data are placed in the format specified by the WSDL
 - Requests sent via a SOAP message
 - Data is provided in the format specified in the WSDL
 - Data provided via a SOAP message
 - Some great stuff:
<http://webservices.xml.com/pub/a/ws/2002/04/30/wSDL.html>
 - UDDI
 - Akin to White/Yellow Pages
 - Specifies where to find a given service and any potential WSDL
 - Some great stuff: <http://uddi.microsoft.com/default.aspx>

A Simple *Hypothetical* Example

The JMPS PTW Interface

- Lets say:
 - The NTW program develops a WSDL specification (lets call it *OpenNTW.wsdl*) through which other systems can request and obtain imagery. The means of obtaining *OpenNTW.wsdl* are placed in the DOD UDDI Registry
 - A second program, LEAP, finds out about NTW via the DOD UDDI Registry, requests a copy of *OpenNTW.wsdl* from NTW and develops an interface that allows users to input data specifying the imagery needed and creates a SOAP message template for requesting the information from NTW
 - In operation, the user specifies what imagery is needed. User input is placed into the SOAP template and the complete SOAP message sent to NTW. NTW receives the request and responds via a SOAP message, also formatted in accordance with the *OpenNTW.wsdl*, with the required imagery.

Other Examples to Consider

- Amazon.Com
 - Information is moved electronically between Amazon, its suppliers, warehouses and shippers as well as various credit card companies
- Web Based Travel Services
 - Information is moved between these services and hotels, airlines and rental car companies

These examples seem perfect for the application of the SOAP/WSDL/UDDI trinity

TRAINING SYSTEMS INITIATIVES & SOFTWARE REUSE EXAMPLES

Ken Reams
PMA-205

Training Systems Organization

- Training Device Software Baseline resides at Field Offices, in many cases hosted on standalone Software Support Systems (SSS)
- Software tool sets and languages vary with each Training System implementation

Software Reuse Initiatives

- TSD participation on the NAVAIR/Fleet Software Working Group
 - Reviewing Specifications for the NAVAIR Data Distribution System (NDDS) software repository
 - Collaborating to identify Training S/W hooks on Operational Flight Programs (OFP) for reuse purpose

Software Reuse Initiatives (cont.)

- TSD participation on the Special Mission MAT (Mission Area Team)
 - Adopt relevant processes and tools promulgated by the SM MAT
 - Support metrics collection
 - Implementing Key Process Areas from the CMMI framework

Examples of S/W Reuse

- Naval Aviation Systems Master Plan (NASMP)
 - Run-Time Interface (DMSO RTI version 1.3 or VTC RTI NGPRO 3.0)
 - Federation Object Model (FOM) Dataset
 - NASMP Portable Source Initiative (PSI) Visual Database

Examples of S/W Reuse (cont.)

- **Digital Voice Core Engine**
 - Use to develop voice communications for various training products
 - Provides a DIS interoperability interface
 - Encodes and decodes digital voice
- **Distributed Interactive Simulation – High Level Architecture Gateway**
 - Provides both DIS and HLA interoperability
 - Code reused for over a decade on multiple training systems
 - Distributed to other government agencies and contractors under a government licensing agreement

Examples of S/W Reuse (cont.)

- Aviation Training Systems Programs
 - VH60 trainers reused Host code from the MH60 trainer and Instructor Operation Station (IOS) code from the CH46 trainer
 - VH3 trainers reused Host code from the SH3 trainers and the IOS code from the CH46 trainer
 - CH46 Aircrew Procedure Trainer (APT) reused Host and IOS code from the CH46 Weapon Systems Trainers (WST)
 - CH53 APT reused Host and IOS code from the CH53 WST
 - F18 APT reused Host code from the F18 WST
 - E6B trainers reused code from P3 MAST
 - USCG HH60 and HH65 trainers reused IOS code from the CH46 WST
 - P3 Tactical Operational Readiness Trainer (TORT) reused the Acoustics Stimulator from the MH-60R Tactical Operational Flight Trainer (TOFT)

Examples of S/W Reuse (cont.)

- Integrated Learning Environment Cybrarian (under development)
 - Repository for learning assets
 - Functions:
 - Provides consolidation of all approved and accepted ILE media and content packages
 - configuration management of the ILE metadata schema
 - compliance testing and acceptance of ILE content packages
 - definition of media data package requirements necessary to ingest source files, rendered content, and manifests for ILE content packages authored externally to the LCMS
 - managing CORDRA registration of all accepted and approved content
 - building the new ILE architecture for the Integrated Metadata Repository (IMR) and its Authoritative Data Source and Operational Data Store environments

Questions?