

# Software Systems Architecture: a practical architecture method

David Emery, The MITRE Corporation  
(with contributions from Rich Hilliard and  
Timothy Rice, ConsentCache)

# Plan of Tutorial

- ◆ Goals of this tutorial
- ◆ Laying the foundations
- ◆ A case study
- ◆ A Practical Architecture Method
- ◆ Comparison with other approaches
- ◆ Wrap Up

# Goals of this Tutorial

- ◆ Why Architecture?
- ◆ What is Architecture?
- ◆ Who does Architecture? When?
- ◆ How does the Architect do the job?

# Laying the Foundations

- ◆ The Architectural Metaphor
- ◆ Some Definitions
- ◆ The Architect's Job
- ◆ Architectural Rendering

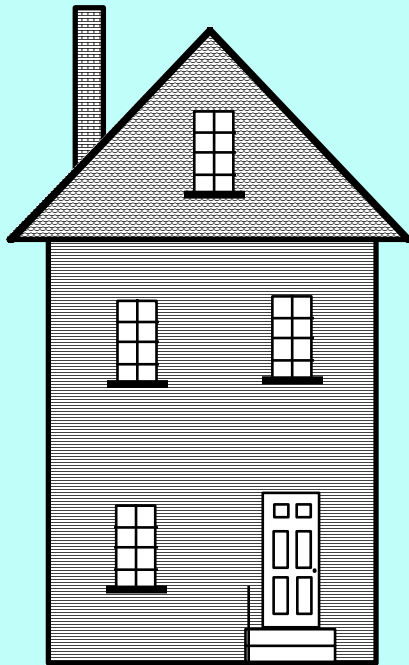
# The Architectural Metaphor

- ◆ Precedents from other fields (building, hardware)
- ◆ Roles and Stakeholders
- ◆ Multiple Views and Models

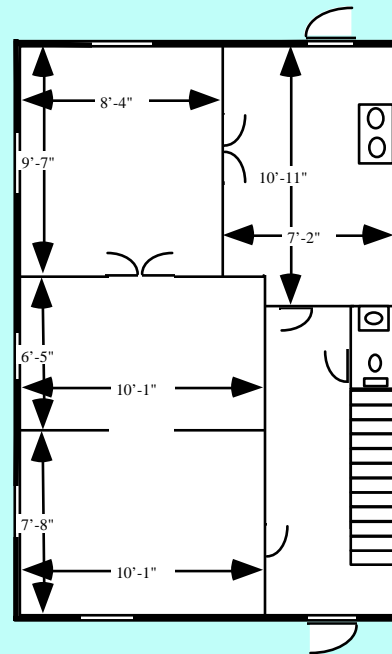
# Roles and Stakeholders

- ◆ Architect
- ◆ Engineer
  - Civil, Electrical, Mechanical
- ◆ General Contractor
  - Trades: Mason, Carpenter, Electrician, Plumber
- ◆ Building Inspector

# Multiple Views and Models



**Elevation**



**Floor Plan**

- ◆ Bill of Materials
- ◆ 2x4x8 250
- ◆ 2x6x8 150
- ◆ 4x4x10 10
- ◆ Siding 1500 sq ft
- ◆ Shingles 500 sq ft
- ◆ 8d nails 20 lb
- ◆ 6d nails 10 lb

**Materials List**

# Why Architecture?

- ◆ Explicitly “architected” systems seem to turn out faster, better and cheaper
- ◆ Separation of concerns:
  - Essential system characteristics
  - Multiple system stakeholders
  - Separate long-term goals, and evolution from immediate construction concerns
- ◆ “Architecture” as response to failure of specifying non-functional system characteristics as requirements

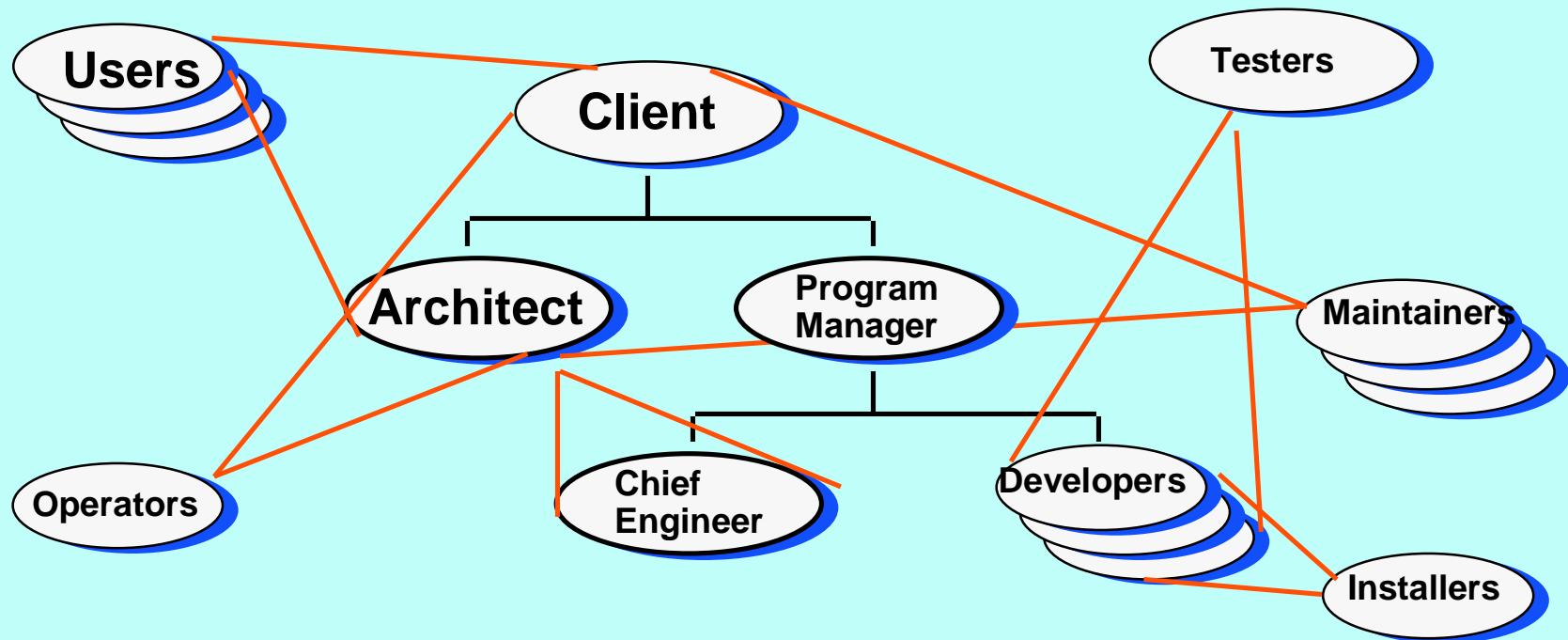
# What is “Architecture”?

- ◆ An *architecture* is the highest-level concept of a system in its environment
  - IEEE Architecture Working Group, 1999
- ◆ Where:
  - highest level = essential, unifying concepts and principles
  - system = application, system, platform, system-of-systems, enterprise, product line, ...
  - environment = developmental, operational, programmatic, ... context

## <Adjective> Architectures

- ◆ Application Architectures
- ◆ Data Architectures
- ◆ Enterprise Architectures
- ◆ Logical Architecture
- ◆ Makefile Architectures
- ◆ Operational Architectures
- ◆ Physical Architectures
- ◆ Security Architectures
- ◆ Systems Architectures
- ◆ Technical Architectures
- ◆ Occupant Architectures
- ◆ Heating and Lighting Architectures
- ◆ Building Code Architectures

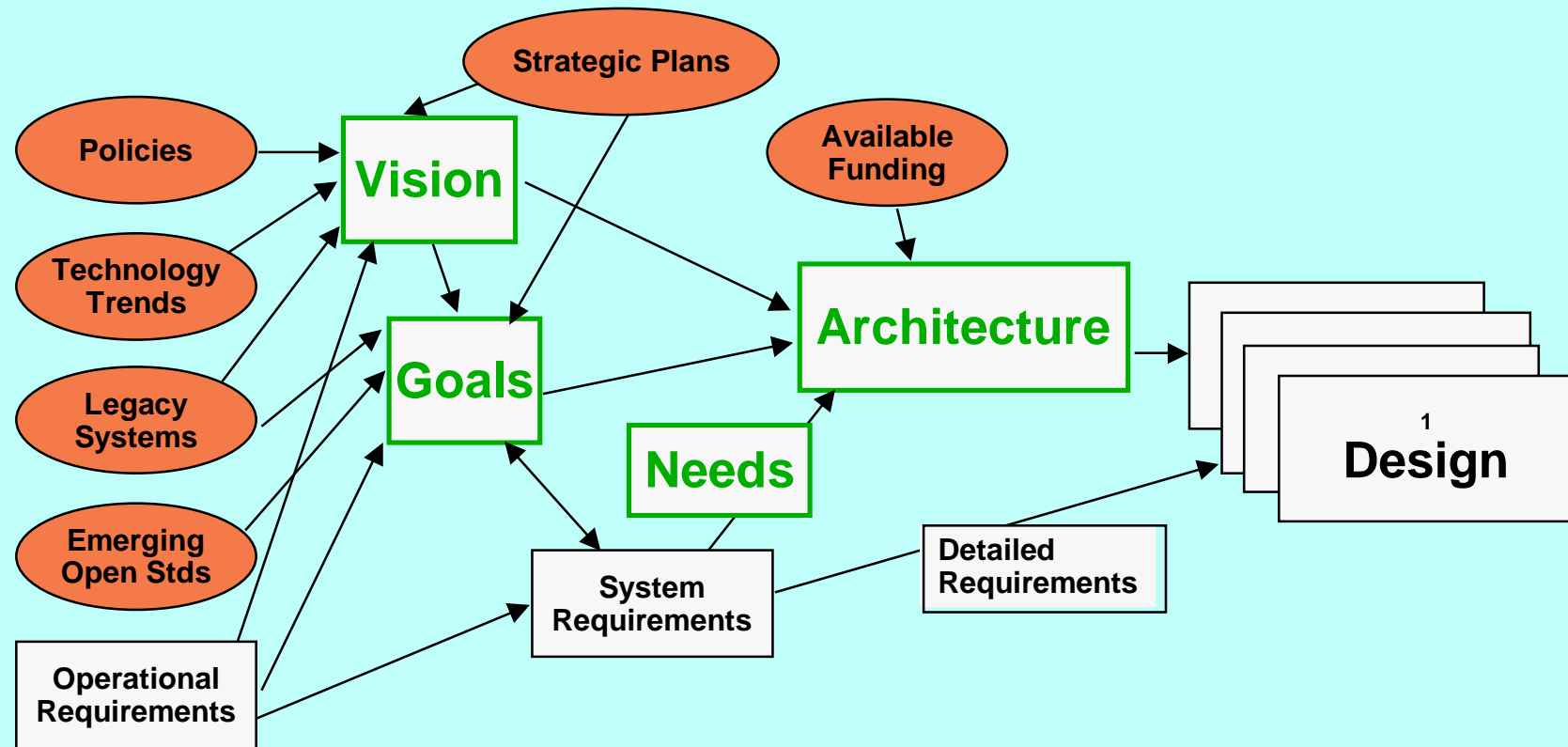
# The Architect's Job: Roles



The ideal architect should be a man of letters, a skillful draftsman, a mathematician, familiar with historical studies, a diligent student of philosophy, acquainted with music, not ignorant of medicine, learned in the responses of jurisconsults, familiar with astronomy and astronomical calculations.

— Vitruvius, *De Architectura* (25 BC)

# The Architect's Job: Products



*Life cycle Activities*

**Requirements & Concepts**

**Architecture**

**Design & Implementation •••**

# Characteristics of the Architect's Job

- ◆ Client-centered
  - Architect works for the client
- ◆ Systems orientation: bridging problem definition and solution conceptualization
  - Architect's job is to understand client's needs to produce one or more models (potential solutions)
  - Architect then works with Engineer
  - Engineer's job is to design and implement architect's model

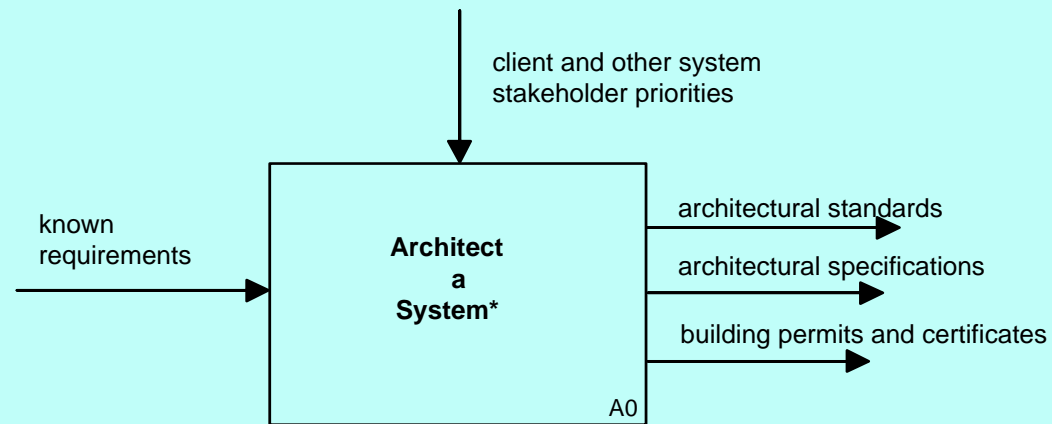
# Characteristics of the Architect's Job

No person who is not a great sculptor or painter can be an architect. If he is not a sculptor or painter, he can only be a builder.

— John Ruskin, *Lectures on Architecture and Painting* (1853)

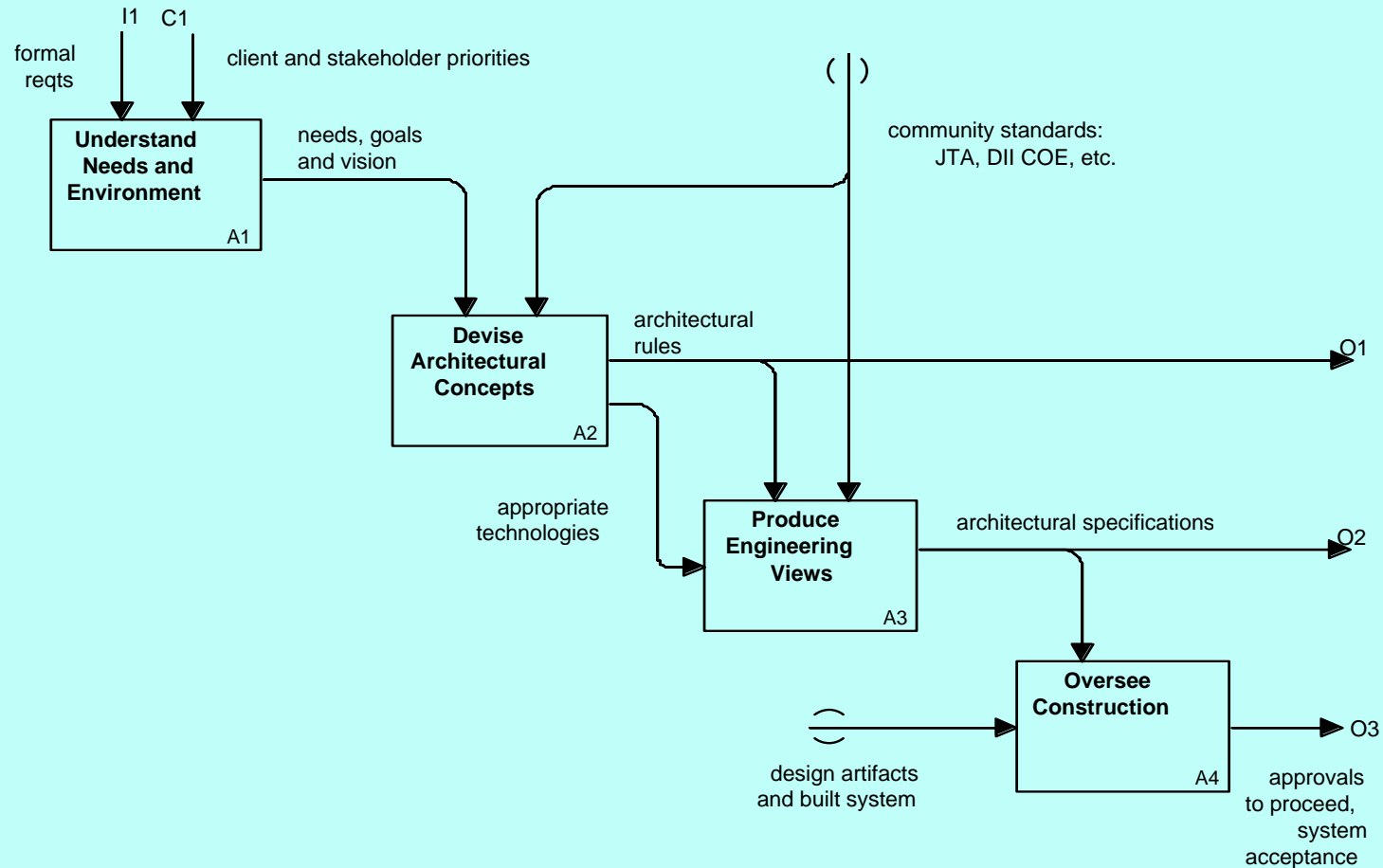
- ◆ Architect oversees construction, to ensure actual implementation meets design
- ◆ Responsible for acceptance of built system
- ◆ Multidisciplinary Synthesis
  - Technical, programmatic, managerial
  - Artistic, Heuristic

# The Architect's Job: Architect a System (context)

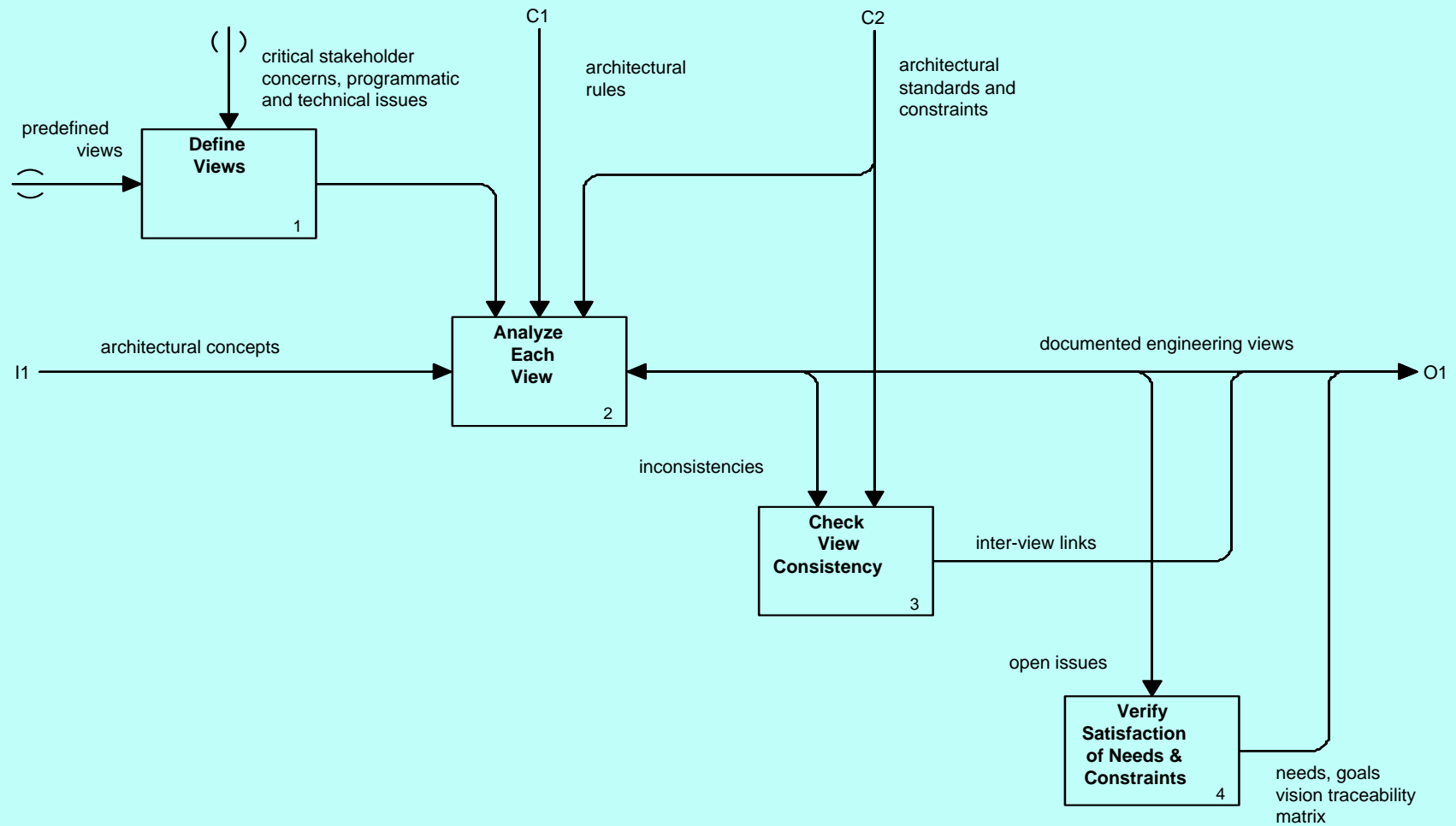


**\* Where “system” ranges over: individual applications, usual programs, product families, product lines, systems of systems or the whole enterprise.**

# The Architect's Job: Architect a System



# The Architect's Job: Produce Engineering Views



# Supporting Activities (Mechanisms)

- ◆ Operational modeling
- ◆ Market and strategic studies
- ◆ Financial planning and analysis
- ◆ Requirements analysis
- ◆ Ergonomics, time-motion studies
- ◆ Simulation and Prototyping
- ◆ Enabling technology studies:  
e.g., messaging, image processing, information retrieval, multimedia
- ◆ Formal Specification
- ◆ Design and implementation techniques and methods
- ◆ Collaboration
- ◆ Self-criticism and architectural assessment
- ◆ Project development and management
- ◆ Planning and scheduling
- ◆ “Process”
- ◆ Contracting
- ◆ Design reviews, inspections and audits
- ◆ Compliance, conformance testing and analysis
- ◆ Quality assurance

# Architectural Rendering

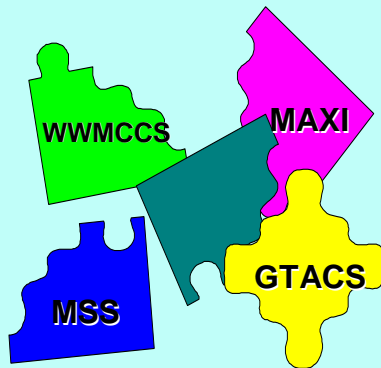
- ◆ An *architectural description* is a model of the structure and behavior of the whole system
  - It shows how the system fulfills the needs in the context of its environment
  - It identifies major system components, their interconnections and dependencies, and the limits within which they must operate
  - It should address the concerns of the system's stakeholders

# Architectural Description

- ◆ An architecture is documented as a model
- ◆ A model is comprised of one or more *views*
  - A view represents the whole system to focus on one or more critical concerns
  - Support multiple audiences each with their own concerns
  - Reduce perceived complexity through separation of concerns

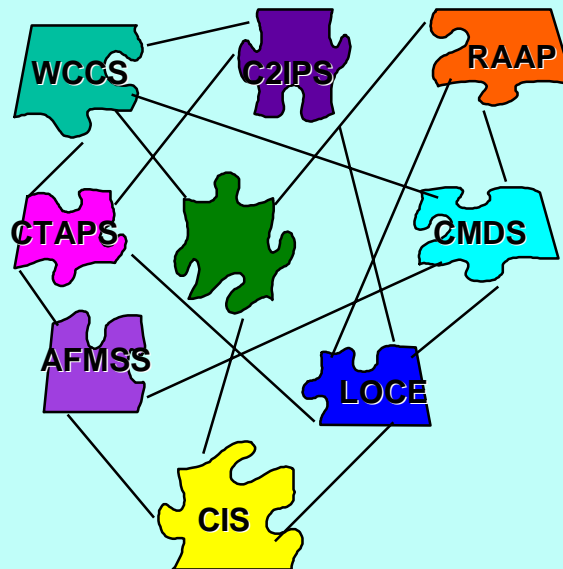
# Case Study: USAF Command and Control

## Yesterday's Efforts



Independent  
"stovepipe"  
C4I Systems

## Today's Attempts



Federated C4I  
Systems

## Tomorrow's Target

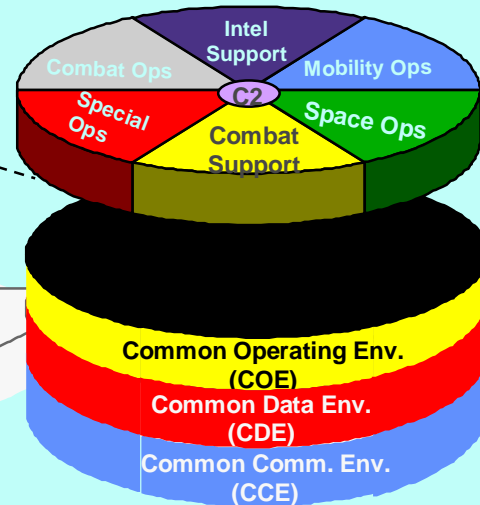


Integrated  
C2 System

# The Scope



INFRASTRUCTURE  
FRAMEWORK  
(DII JTA-based)



## Command and Control

“The exercise of authority and direction by a properly designated commander over assigned forces in the accomplishment of the mission (JCS Pub 1-02)”

## ◆ Ground-based Command and Control

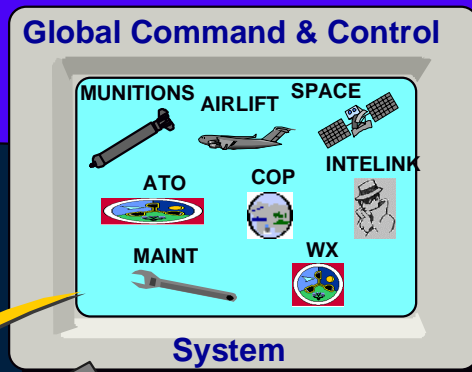
# The Problem

- ◆ Existing systems are not easy to use:
  - Do not exhibit full desired functionality
  - Non-intuitive, poorly designed user interfaces
  - Do not work seamlessly with each other
  - Multiple physical machines required to execute a job
  - Exhibit a significant footprint
- ◆ Redundant and duplicative efforts to plan/field infrastructure ongoing
- ◆ Inconsistent automation does not effectively aid the user in the execution of his job
- ◆ Users' needs are not being consistently addressed across mission domains

# Air Force Integrated C2 Vision

## C2 Vision

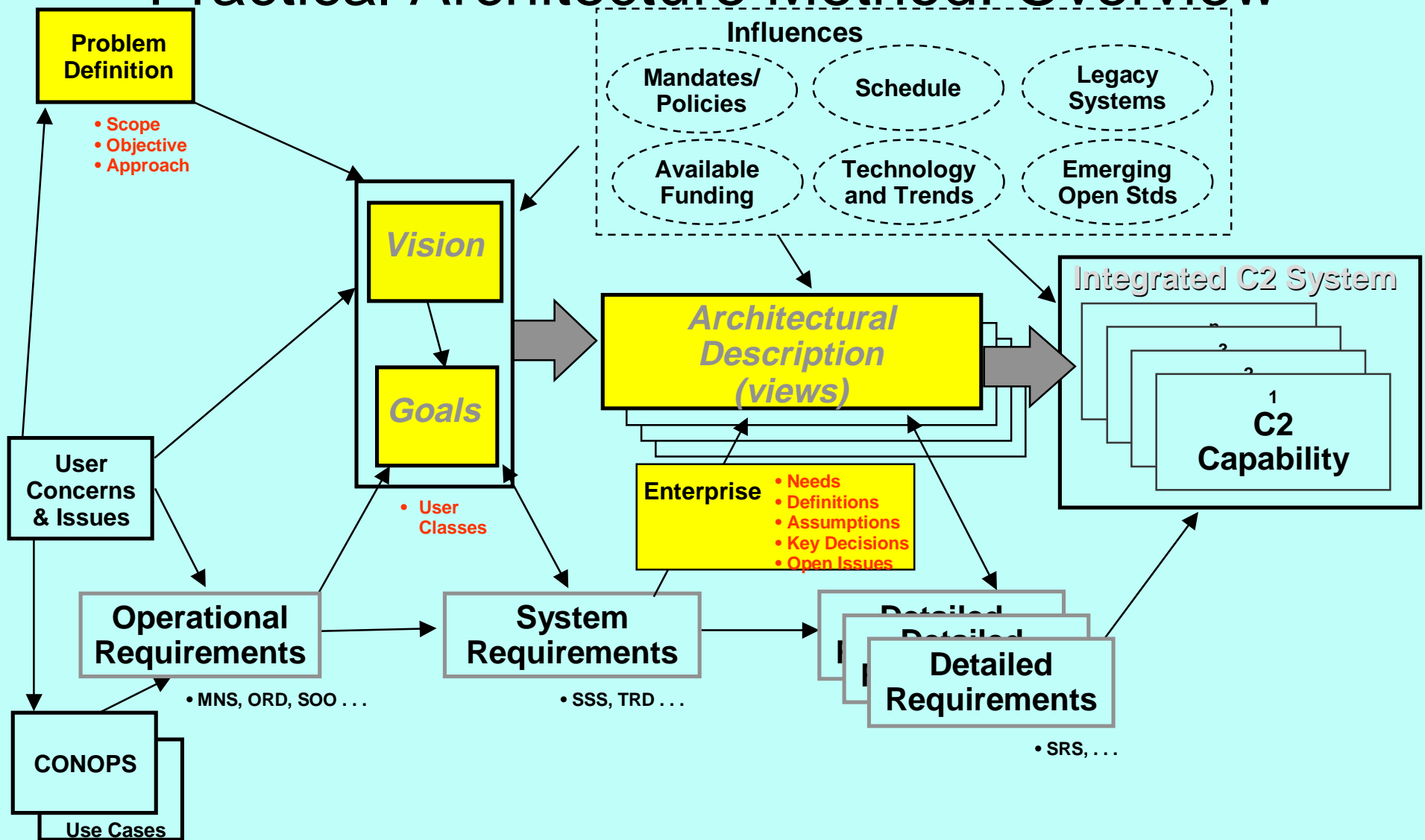
**Global,  
Integrated,  
Interoperable,  
Affordable**



**Common  
Terminals  
& Applications**



# Practical Architecture Method: Overview



# Practical Architecture Method

- ◆ The Steps
- ◆ Typical Products
  - Stakeholder Analysis
    - ◆ Vision, Goals and Needs
  - Viewpoint Analyses
    - ◆ Selection of Viewpoints
    - ◆ Construction of Views (the “blueprints”)
  - Architecture Integration
  - Overseeing System Construction
- ◆ Usage to date

# Practical Architecture Method

## The Steps

- ◆ Identify the problem
- ◆ Frame the Problem Space
  - Develop a problem statement and common vision
  - Stakeholders' goals and needs
  - Understand relevance of those needs to the architecture
  - Capture the 'defining' problems with use cases, scenarios
- ◆ Define the Solution Space
  - Model key concerns to frame the solution space for the developers/maintainers/other stakeholders
    - ◆ Establish the boundary conditions
    - ◆ Identify "absolutes" (commitments, obligations) and degrees of freedom
- ◆ Implement the solution
  - Develop implementation guidance, insertion strategy
  - Monitor implementation, maintain the architecture
    - ◆ External changes (e.g., requirements changes, new technology opportunities)
    - ◆ Internal changes (implementation problems that affect the entire system)

# Vision and Goals Defined

- ◆ Vision
  - Ranges from “nice-to-haves” to long range direction to unfunded requirements to “pie in the sky”
- ◆ Goals are the “measures of success” for the program at the architectural level
  - Should be measurable (qualitative or quantitative)
- ◆ Together, Goals and Vision represent the client’s direct input to the architectural process
- ◆ Goals and Vision provide the basis for architectural analysis
  - Identify issues that affect architectural decisions
  - Used to evaluate and select between alternatives

## AFIC2S Vision

“A system that will enable global command and control of joint and coalition aerospace forces throughout the spectrum of military operations by exploiting information to know, predict, and dominate the battlespace”

“A system that securely supports all levels of joint and coalition aerospace ops providing seamless connectivity—anywhere, anytime, ...”

# Vision Refinement

- ◆  $A^4$  = Anytime, Anywhere, Anyone, Anyway
  - Access to information is available within a *timely manner* at the *place of need* by the *individual* who needs the information using a *single information appliance* (one per user, PDAs to workstations)
- ◆ Single configurable system for planning, monitoring, executing and managing all phases of all kinds of mission operations
  - Distributed collaborative use environment
  - Expandable in both scope and scale by adding hardware
  - Expandable in function by adding new capabilities

# Vision Refinement

- ◆ The C2 system supports a range of users from “casual” to “power”
  - “Casual” user relies on the computer intermittently
    - ◆ Productivity is less dependent on system response time
    - ◆ Information access is primarily read-only
  - “Power” user requires the system to perform the mission
    - ◆ Rapid response time to maximize productivity
    - ◆ Significant training to take advantage of system
- ◆ C2 users typically fall somewhere within this continuum

# AFIC2S Goals: Better, Faster, Cheaper, Safer

- ◆ *BETTER* integrated C2 support to the warfighter
  - Provide appropriate information at varying levels of precision/abstraction
  - Take advantage of emerging technology as it matures
- ◆ *FASTER* C2 capability
  - Development/acquisition time
  - System response
  - Timeliness of information
  - Timely execution of information intensive tasks
    - ◆ Computer performs automation tasks quickly
    - ◆ Human, using computer, performs tasks quickly

# AFIC2S Goals: Better, Faster, Cheaper, Safer

- ◆ *CHEAPER* production, delivery, operation, and sustainment of C2 capability, for example:
  - Enterprise-wide software licenses
  - High productivity software development tools
  - System ease-of-use
  - Enhanced automation
- ◆ *SAFER* systems and acquisition
  - More reliable systems
  - Acquired with minimal technology risk; e.g.,
    - ◆ Use proven, low cost technology, typically COTS
    - ◆ Use existing infrastructure where possible

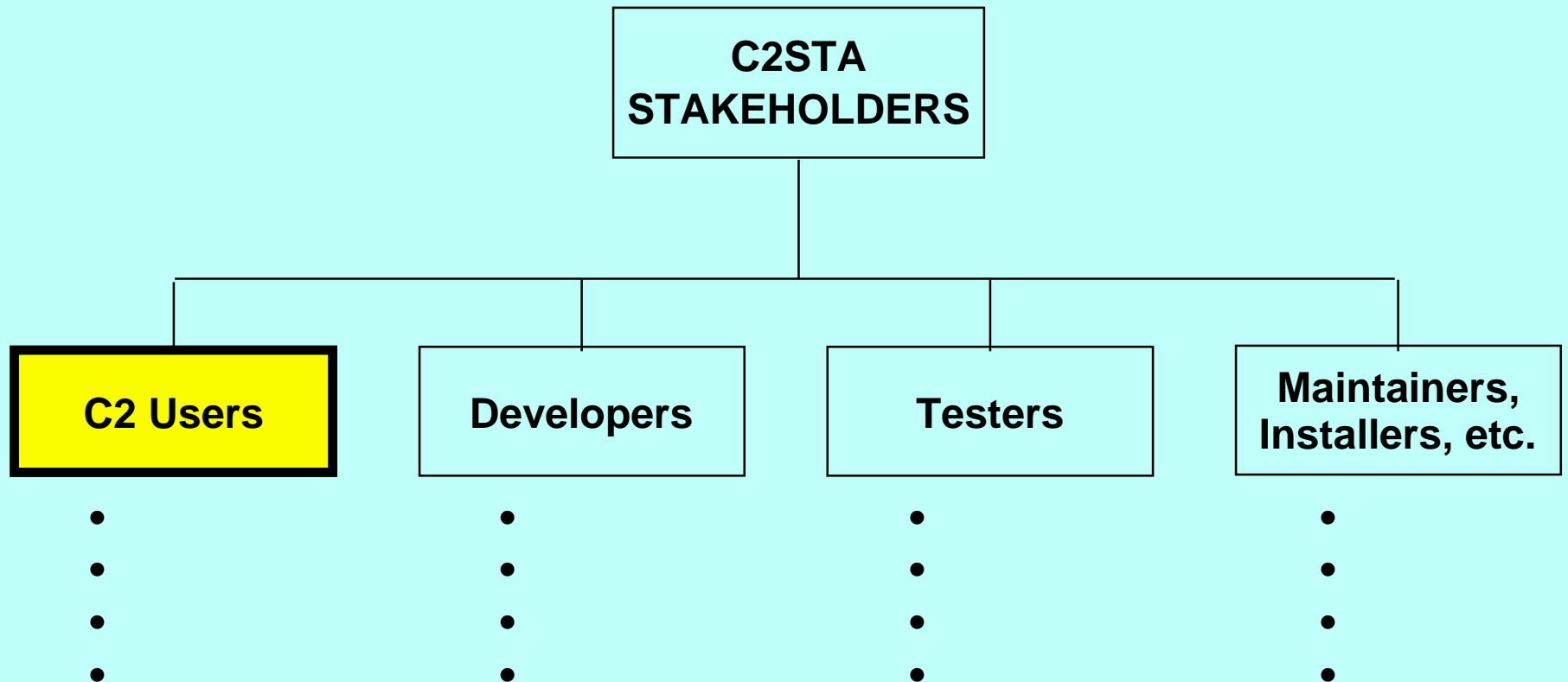
# Analysis of Goals

- ◆ High Usability
- ◆ “Come as you are” Interoperability
- ◆ Reliability, Maintainability, Availability
  - Fault tolerance
  - Graceful degradation in failure
- ◆ Scalability
- ◆ Flexibility/Adaptability
  - Support war-time and peace-time operations
    - ◆ “train as you fight”
- ◆ Minimal footprint (manpower and platform)
  - From single Information Appliance

# Analysis of Goals

- ◆ Seamless connectivity to all platforms
- ◆ Geographic and organizational transparency
- ◆ Infrastructure transparency
  - Location, distribution, computation, format, access
- ◆ Minimal training requirements
- ◆ Reduced development and deployment costs
- ◆ Dynamic bandwidth allocation
  - Supports bandwidth intensive nature of future C2 systems
  - Supports proper operation in austere environments

# C2STA Stakeholders



# Needs

- ◆ Needs are architecturally relevant requirements
  - usually associated with one or more stakeholders
  - typically more stable than requirements over the life cycle
  - not necessarily testable
- ◆ A need captures those concerns that will drive key decisions by the architect, such as decisions pertaining to performance, technology or cost drivers
- ◆ *Needs are to architecture as requirements are to a system.*

# AFIC2S Needs

- ◆ Provide interoperability with desktop, commodity Office Automation (OA)
- ◆ Support generation and dissemination of draft data
- ◆ Support austere operations
- ◆ Deliver flexible, customizable capability
- ◆ Provide intuitive, consistent, easy to learn user interfaces
- ◆ Provide hardware/software portability
- ◆ Support user collaboration
- ◆ Provide geographically transparent access to C2 data (anytime, anywhere, ...)
- ◆ Accommodate legacy infrastructure, data and applications

## AFIC2S Needs

- ◆ Provide timely access to data within varying real-time regimes
- ◆ Support multiple security policies, levels and environments
- ◆ Share data with external systems across arbitrary organizational boundaries
- ◆ Achieve continuity of operations
- ◆ Ensure integrity of information
- ◆ Provide accountability for user actions
- ◆ Support defensive information warfare
- ◆ Low-cost overall System Administration

# C2STA Assumptions

- ◆ Common Operating Environment (COE) is DoD mandate
- ◆ Multilevel Secure (MLS) DBMS and operating systems are infeasible in the general case for the near-to-medium future
- ◆ With pedigree, old information is better than no information

# Key Decisions

- ◆ Field the Integrated C2 system as a set of flexible, “pre-integrated” capabilities
  - Compose (assemble) unique system instances, from reusable components, using layered construction
- ◆ Treat C2 data as a whole
  - Manage at the data access level; not at the data model or data store levels
- ◆ Buy before build
  - Seek COTS-based technologies at key interfaces
- ◆ Adopt a core, shared infrastructure across C2
  - Utility metaphor

# Views ...

- ◆ A *view* addresses a specific set of stakeholder concerns
- ◆ Each view presents the (whole) system from a chosen viewpoint
  - Complete relative to that viewpoint
  - Consistent with respect to that viewpoint
- ◆ Each view should be constructed from a well-defined viewpoint ...

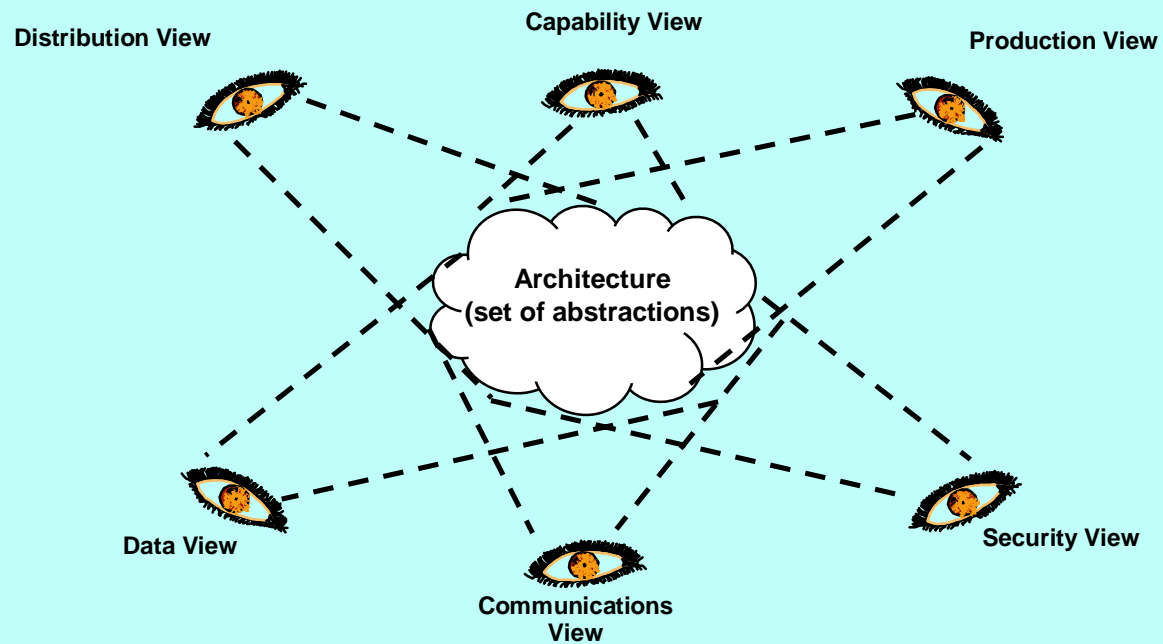
## ... and Viewpoints

- ◆ A *viewpoint* is a pattern for constructing views
- ◆ A viewpoint is formalized by:
  - Name
  - Purpose
    - ◆ What concerns does it address?
    - ◆ What questions does it answer?
  - Viewpoint elements
    - ◆ Viewpoint language/notation
- ◆ `view : viewpoint :: program : programming language`

# View “Template”

- ◆ Purpose
- ◆ Scope
- ◆ Selected Viewpoint
- ◆ Key needs
- ◆ Assumptions
- ◆ Key Decisions
  - Commitments
- ◆ Consequences
  - Obligations and Freedoms
- ◆ Open Issues

# Viewpoint Selection

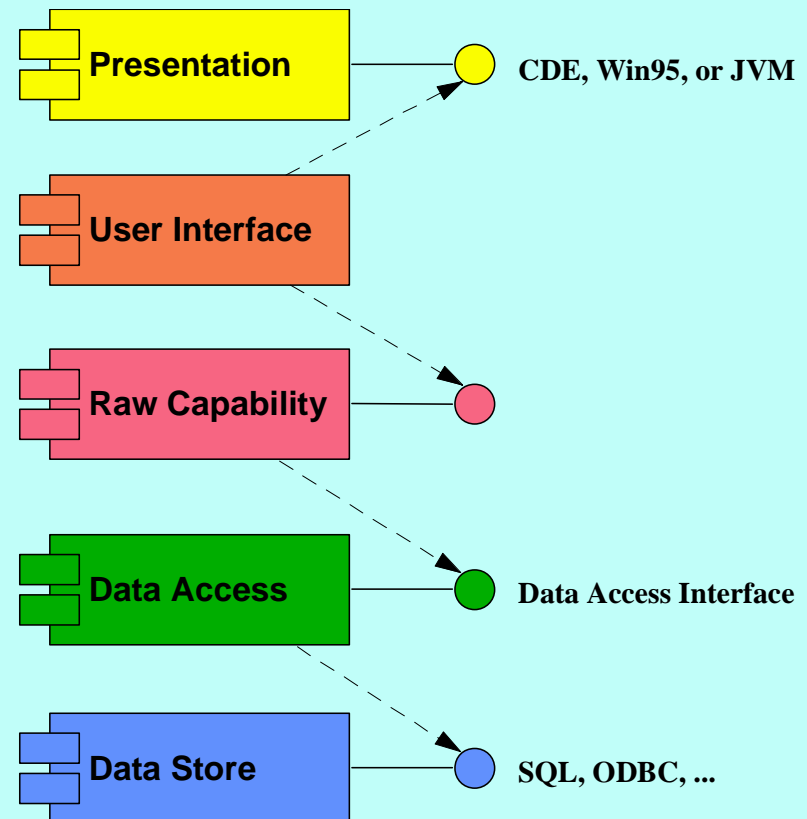


# Capability Viewpoint

- ◆ How is functionality packaged and fielded?
- ◆ Components: Units of structure, units of work
- ◆ Connections: provides, requires, client-server
- ◆ Constraints: resources, construction standards (SDE, TRM, APP), reuse models
- ◆ Viewpoint language: UML component diagram
- ◆ AKA: Static, Application, Structural Viewpoints

# C2STA Capability View

- ◆ The Capability View covers all C2 functionality for operating on data
- ◆ Capabilities are fielded using a 5-tier layered organization with interfaces between pairs of layers
  - Each layer is a capability
  - Entire stack is a *deployable capability*
- ◆ Capabilities can serve other capabilities

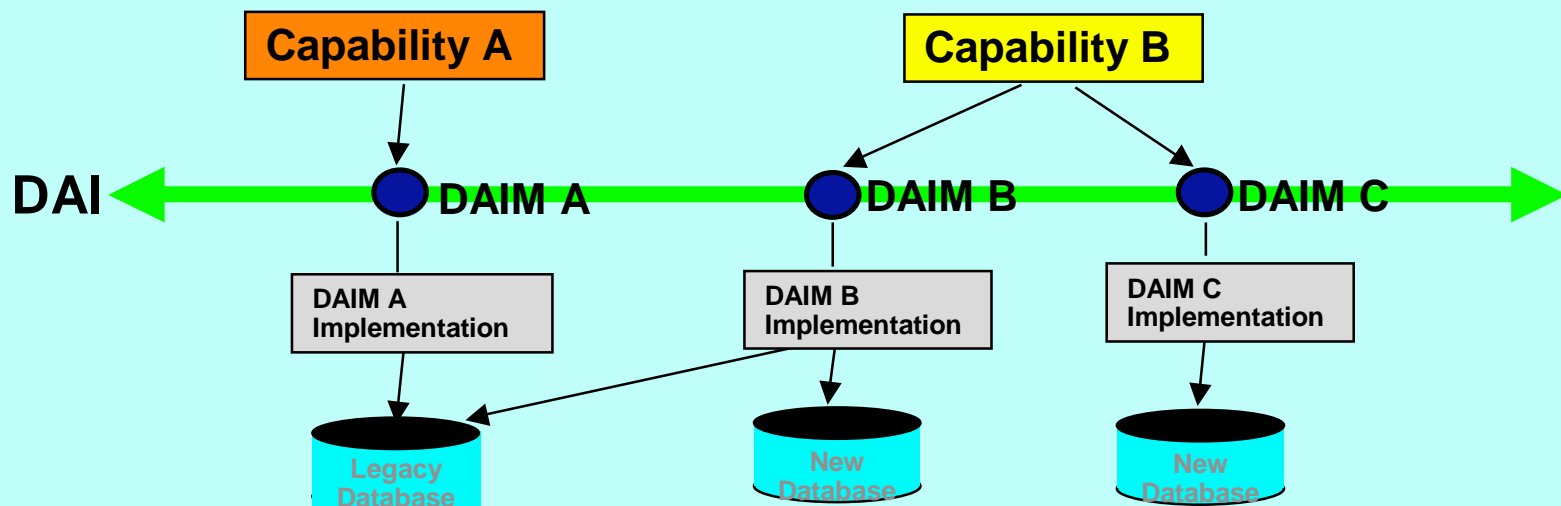


# Data Viewpoint

- ◆ Who reads data?
- ◆ Who writes data?
- ◆ Where is it?
- ◆ Who owns data?
- ◆ How is it managed?
  
- ◆ Components: Data models, formats, stores, clients (consumers, producers)
- ◆ Connections: Logical, implementation, and business rules
- ◆ Constraints: Tolerance for consistency, transaction properties

# C2STA Data View

- ◆ The Data View defines how C2 data is defined, organized, accessed, and maintained in the AFIC2S
- ◆ The Data Access Interface (DAI) belongs to the AFIC2S
  - The DAI is the primary means of accessing C2 data
  - The DAI is defined by a set of Data Access Interface Modules (DAIMs)
- ◆ A capability accesses C2 data through one or more DAIMs



# C2STA Data View

- ◆ When a database is established and its DAIMs published, other sub-systems may access
- ◆ DAIMs provide metadata for any C2 data
  - Metadata can be used by capabilities to determine pedigree of data, how data should be handled, etc.
- ◆ The builder of a database may not be the owner of the data
- ◆ New systems provide DAIMs as standard interfaces
- ◆ Legacy systems provide DAIMs as wrappers

# Distributed Viewpoint

- ◆ What runs where?
- ◆ Who talks to whom?
- ◆ How do they talk?
  
- ◆ Components: Processors, nodes, processes, networks
- ◆ Connections: Events
- ◆ Constraints: Availability, bandwidth, reliability, fault-tolerance, redundancy
  
- ◆ AKA: Physical viewpoint

# C2STA Distribution View

- ◆ The Distribution View defines how C2 data and capabilities are distributed
- ◆ Two key goals are location transparency and dynamic reconfigurability
  - Allows a sub-system to be configured for environment and mission
  - Facilitates failover when needed to achieve reliability
- ◆ Key decisions
  - Use the Web to publish data and for simple data entry
  - Move data forward to improve performance and enhance reliability
  - Install time-critical and/or more full-featured capabilities directly on the client
  - Distribute large, relatively static information via physical media where it is advantageous to do so
  - Support a broad range of client devices from palmtops to high-powered workstations

# Security Viewpoint

- ◆ Who may access what?
- ◆ How is access controlled?
- ◆ How are compromises detected and reacted to?
- ◆ Components: Threats, info domains, subjects, objects, mechanisms (e.g., guards, firewalls)
  - Subjects: things that need to access Objects
  - Objects: things that need to be protected
- ◆ Connections: Access, paths, gates
- ◆ Constraints: Privileges, rules, policies

# C2STA Security View

- ◆ The Security View provides mechanisms for confidentiality, integrity, availability, and accountability
  - Mechanisms are distributed to support distribution of capabilities and information
- ◆ Supports processing and interoperability among sub-systems and external systems with different security policies (e.g., different security levels, coalition operations)
- ◆ Addresses and reduces accreditation risks
- ◆ Security infrastructure is primary provider of security services
  - Security infrastructure will be as transparent as possible
  - Capabilities need not be cognizant of security; if they are, they will rely on the security infrastructure
  - Protects against Information Warfare (IW) attacks
  - Security Management Infrastructure (SMI) provides public key infrastructure (PKI), which is capable of cross-certification with other PKIs

# Construction/Sustainment Viewpoint

- ◆ Can we build this?
- ◆ Can we build this economically? Are there economies of scale, productivities of concern?
- ◆ Who does what?
  - Use when there is a strong product line, enterprise orientation, when there are multiple developers (or maintainers), when construction is unprecedented
- ◆ How will system be sustained?
- ◆ How do we field, maintain, and enhance capability?

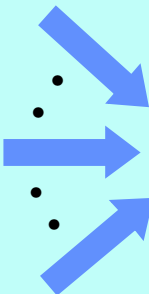
# C2STA Production View

- ◆ Provides guidance for capability development and sub-system composition
- ◆ Includes key principles on:
  - How components must present their interfaces (e.g., using COM or CORBA, and XML)
  - To use the least restrictive display option
    - ◆ HTML
    - ◆ JVM
    - ◆ Platform-native GUI
  - Guiding wisdom for development (e.g., buy, don't build, fight complexity with simplicity, design and document interfaces before implementing)

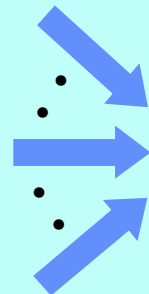
# C2STA Production View

## Construction Aggregation

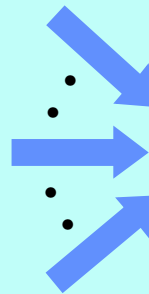
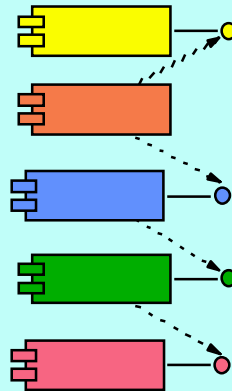
COMPONENT



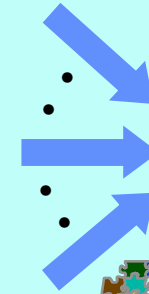
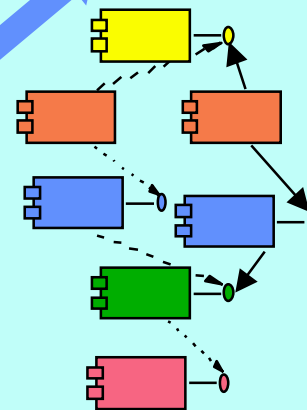
CAPABILITY



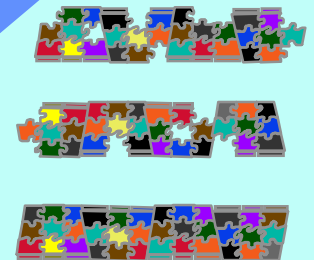
DEPLOYABLE CAPABILITY



SERVICE



PRODUCT LINE



## EXAMPLE:

**SORT, UI WIDGETS**

**TARGET DATA RETRIVAL,  
TARGET DATA SORTING,  
MAP RENDERING,  
MULT-LIST MERGE**

**TARGET PLOTTING,  
MULTI-TARGET LIST MERGE**

**AIR TARGETING**

**TARGETING &  
WEAPONERING**

## Functional Attributes

**COMPONENT**

**LOWEST LEVEL  
STRUCTURAL  
SOFTWARE ELEMENT**

**CAPABILITY**

**OPERATIONAL  
FUNCTION(S);  
C2STA COMPLIANT  
COMPONENT  
PACKAGE**

**DEPLOYABLE  
CAPABILITY**

**MULTI-LAYER  
CAPABILITY;  
"PRE-INTEGRATED"  
FUNCTION(S)**

**MISSION  
SERVICE**

**SET OF INTEGRATED  
DEPLOYABLE  
CAPABILITIES;  
MISSION SUPPORTING**

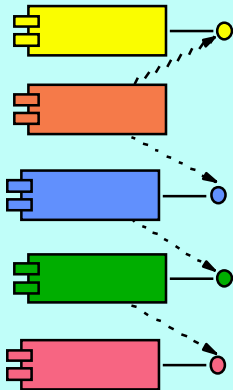
**PRODUCT  
LINE**

**SINGLE MISSION  
AREA  
FUNCTIONALLY  
RELATED  
CAPABILITY SET**

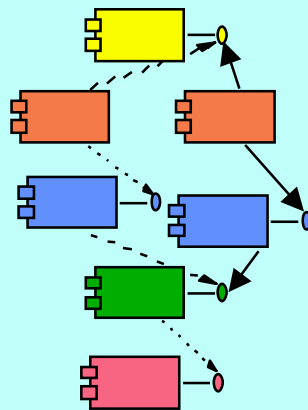
Deployment Aggregation

# Composition

DEPLOYABLE CAPABILITY



MISSION SERVICE

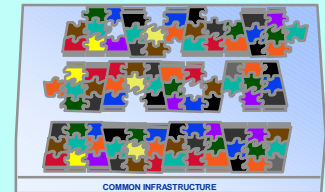


COMPOSABLE

FACILITY SUBSET



"THE AFIC2S"



COMMON INFRASTRUCTURE

EXAMPLE:

TARGET PLOTTING,  
MULTI-TARGET LIST MERGE

AIR TARGETING

JOINT AIR OPS ATO  
GENERATION

Functional Attributes

DEPLOYABLE CAPABILITY

MULTI-LAYER CAPABILITY;  
"PRE-INTEGRATED"  
FUNCTION(S)

MISSION SERVICE

SET OF INTEGRATED  
DEPLOYABLE CAPABILITIES;  
MISSION SUPPORTING

AFIC2S SUBSET

OPFAC SUPPORTING;  
COLLECTION OF  
FIELDED/SHARED C2  
MISSION SERVICES

AFIC2S

DISTRIBUTED,  
FULLY  
INTEGRATED,  
DEPLOYED  
C2 SYSTEM  
(multi-  
location/facilities)

## Another Example: Total Army Distance Learning Program

- ◆ MITRE asked to propose a 'reference architecture' for Total Army Distance Learning Program (TADLP)
- ◆ Process produced proposed Vision, Goals, Needs for TADLP
- ◆ Selected and produced views appropriate to TADLP
- ◆ MITRE work identified many areas that required further attention, such as
  - Importance of Enterprise Management at all levels
  - Impact of Courseware Development (outside scope of TADLP, but primary TADLP interface)
  - Alternate methods for courseware distribution throughout the Army

# Proposed Army Distance Learning Goals

## ◆ Improve Soldier Performance

- Develop Courses Relevant to Current Needs
- Deliver Courses Where They Are Needed
- Deliver Courses When They Are Needed
- Provide Training That Adapts to Individual Needs
- Facilitate Collective/Multi-Echelon Training
- Provide Expertise Not Otherwise Available
- Reduce Time Required to Achieve Task Proficiency
- Enhance Active Learning
- Reduce Training Attrition

# Army Distance Learning Goals (concluded)

- ◆ Reduce Training Costs
  - Transients, Trainees, Holdovers, and Students (TTHS)
  - Cost Associated with AMT (NET/DET)
  - Cost Associated with Special Teams
  - Courseware Development Time
- ◆ Keep Pace with Technology
  - Warfighting Technology
  - Training Technology
  - Infrastructure Technology

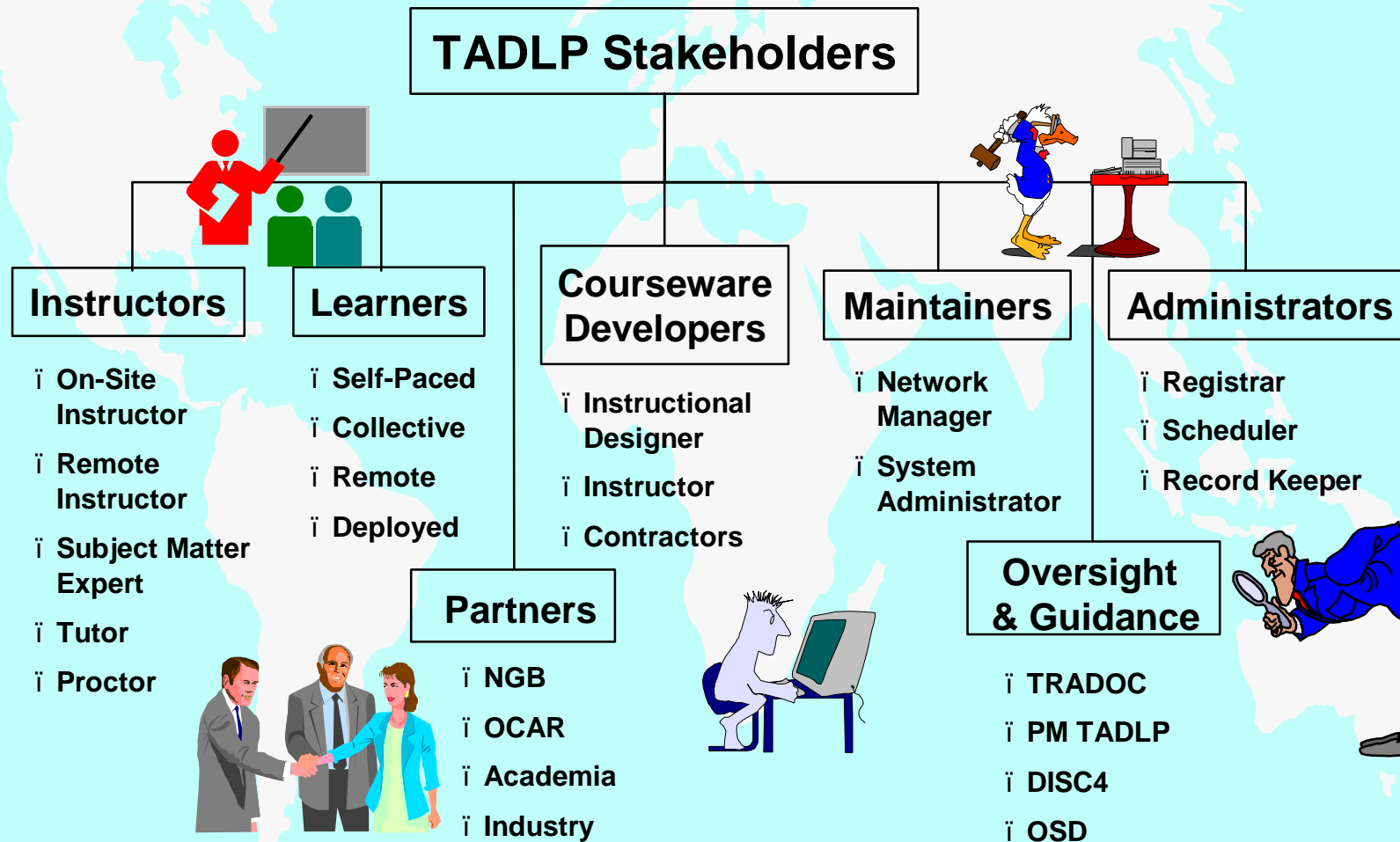
# Proposed Army Distance Learning Vision

- ◆ Access to Army, Other Service, Joint, & Non-DoD Courseware from Anywhere at Anytime
- ◆ Seamless Integration of Classroom and Unit Training (Digitized Training)
- ◆ Provide Classified Training Through Distance Learning Infrastructure without Compromise
- ◆ Reduce Number of Institutional/Resident Courses while Retaining Their Socialization Benefits
- ◆ Store Training Records on Smart Dog Tags or ID Cards
- ◆ Be Prepared for Technology Shifts
- ◆ Enable Participation in Simulation-Driven Exercises
- ◆ Provide Fee-For-Service Training to Non-Army Trainees
- ◆ Evolve Distance Learning Technology into a commodity-based Product Line

# Distance Learning Needs (extract)

Phase	Short Title	Technology	Applicable Views	Requirement	ORD 7 Oct 97	ATXXI CP 9/8/97
I	course management		Da, De	course management	4.a.	
	student registration		Da, De	student registration	4.a.	
	student administration		Da, De	student administration	4.a.	
	performance testing	mark sense test	Da, De, Se	performance testing	4.a.	
		mark sense test	Da, De, Se	technologies	4.a.(1)(a)	
	performance feedback		Da, Se	performance feedback	4.a.	
	distribute courseware	CD-ROM based courseware	Da, De	enable students to receive course materials via distance-learning media	4.a.(1)(a)	
	self-study	CD-ROM based courseware	Da, De	on distance learning course materials delivered for self-study	4.a.(1)(d)	5.b.(1)
	asynchronous training	CD-ROM based courseware	Da, De	Students must have access to training materials required for self-paced, non-instructor led, asynchronous training	4.a.(1)(e)	5.b.(1)
	synchronous training	VTT	Da, De, In	Instructors and students must have capabilities to hear, see, and communicate with each other at separate distance-learning locations	4.a.(1)(a)	5.b.(1)
	VTT	VTT	Da, De, In	Audio/video communications supporting synchronous VTT must transmit visual course presentations that are legible, readable, and audible	4.a.(1)(h)	
	Communicate with SME	telephone, fax	Da, De, In	Students enrolled in asynchronous courses must have the capability to communicate with SMEs at the proponent schools through electronic means	4.a.(1)(f)	
	SME response	telephone, fax	Da, De, In	Subject matter experts must have the capability to respond in kind to student questions, received electronically, within one hour of the SMEs' receipt of the questions	4.a.(1)(g)	
	keep pace with technology		Da, De, Dy, In, Se, Sy	investments in hardware/software will allow spiral development to preclude immediate obsolescence of infrastructure	goals	3.d., 6.e.

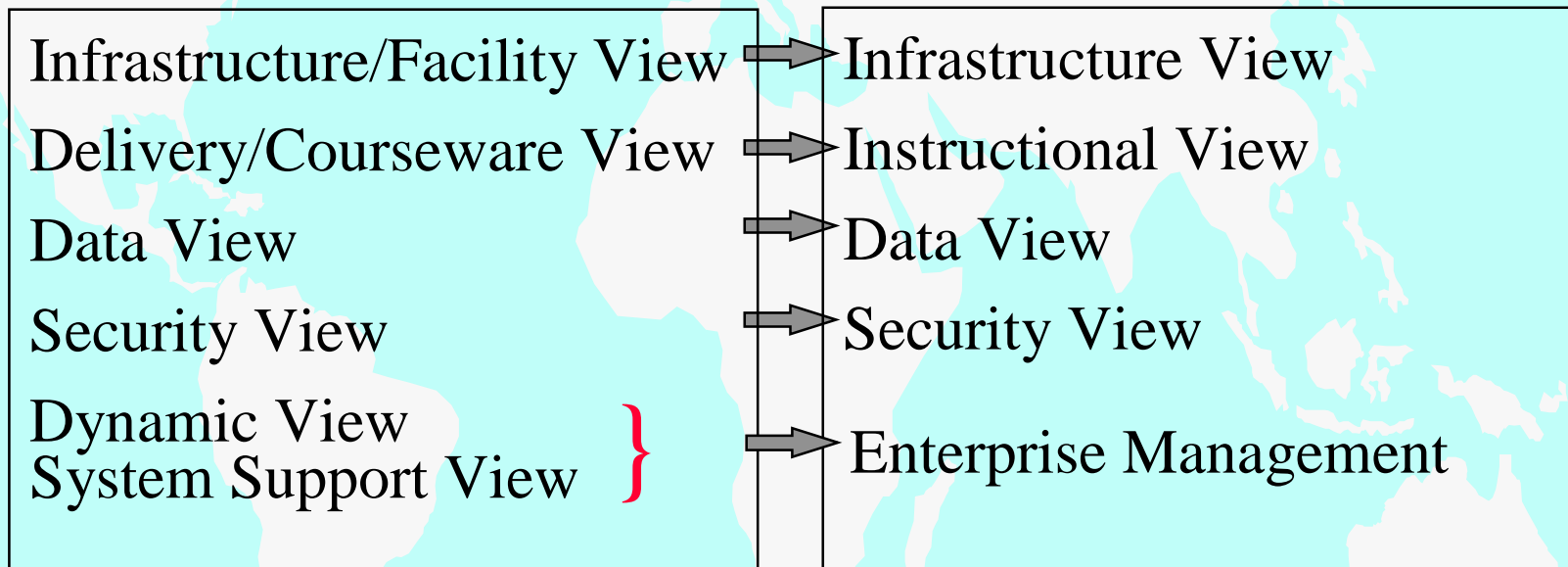
# Stakeholder Groups



# TADLP Architecture Views

6 Nov Delivery

Current



**Note: Infrastructure view includes Hardware, Software and Networking**

# TADLP Architecture Views & Stakeholders

Stakeholder

Views (primary view)

---

Instructors, Learners  
Courseware Developers  
Partners  
Maintainers  
Administrators  
Oversight and Guidance

Instructional, Data  
Instructional, Infrastructure,  
Data  
Infrastructure, Data,  
Enterprise Management  
Enterprise Management, Data,  
Infrastructure, Security  
Enterprise Management, Data,  
Security, Infrastructure  
Enterprise Management, Data,  
Security, Infrastructure

# Architecture Quality Assessment: Goals

- ◆ Repeatable method yielding objective results
  - Evaluation based on documentation, not “hearsay”
- ◆ Based on “open sources”
  - Architects will know the criteria on which architectures will be judged
  - Availability of the criteria may improve overall quality
- ◆ Independent from life cycle, documentation, methodology
  - Cannot assume traditional deliverables and milestones
  - No widely accepted architectural methods
  - Don’t assume a Contractor is the Architect

# Architecture Quality Assessment: Uses of an AQA

- ◆ Evaluate a candidate (proposed) architecture
- ◆ Review technical progress during ongoing architecture development
- ◆ Assess a complete, delivered architecture prior to acceptance/implementation
- ◆ Compare two or more architectures in a consistent fashion

# Comparison with other Approaches

- ◆ ISO RM-ODP (Reference Model - Open Distributed Computing)
- ◆ US DOD C4ISR Architecture Framework
- ◆ IEEE Std 1471 - Recommended Practice for Architectural Descriptions of Software Intensive Systems

# ISO/IEC DIS 10746: RM-ODP

- ◆ Reference Model for Open Distributed Applications
- ◆ DIS 10746-3 specifies “Architecture”, using 5 viewpoints:
  - Enterprise: purpose, scope and policies
  - Information: semantics of information and information processing
  - Computational: functional decomposition into objects and their interfaces
  - Engineering: mechanisms and functions for distributed interaction
  - Technology: choices of technology
- ◆ DIS 10746-3 also specifies consistency rules across viewpoints

# Using RM-ODP

- ◆ RM-ODP specifies a required set of viewpoints for use in our Method
  - But other viewpoints may also be necessary to completely describe an architecture
- ◆ RM-ODP does not address the ‘front-end’ activities, i.e. Goals, Vision, Needs
- ◆ Nor does it provide a process for populating the views specified by the required viewpoints

# The US DOD C4ISR *Architecture Framework*

- ◆ *The Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR) Architecture Framework*
  - “... provides the rules, guidance, and product descriptions for developing and presenting architecture descriptions that ensure a common denominator for understanding, comparing and integrating architectures.”
- ◆ “There are three major perspectives, i.e., views, that logically combine to describe an architecture ... the operational, systems and technical views.”
  - Operational: Functional description of how systems work together
  - Systems: Physical components and their relationships
  - Technical: Standards and conventions for interoperability and commonality

# Using the C4ISR Framework

- ◆ Framework concentrates on required deliverables, not architectural process
  - Deliverables presume an already fleshed-out architecture
  - Framework specifies 7 mandatory products and 19 optional products
- ◆ Thus the Method should be used to produce a complete architecture, with Framework documents as stakeholder requirements
  - C4ISR product requirements should be included as part of the viewpoints selected for the architecture
  - Then Framework becomes specification of how to present an architecture

# ISO/IEC 12207

- ◆ ISO 12207 specifies life-cycle activities for software development
  - Two activities use the term “architecture”:
    - ◆ System Architectural Design
    - ◆ Software Architectural Design
  - These activities are performed by software developers
- ◆ System Architectural Design must:
  - Show allocation of requirements to hardware, software and manual operations
  - Show traceability back to system requirements, consistency and feasibility

# Using ISO 12207

- ◆ Architectural Method oriented towards “System Architectural Design” activity
  - “Software Architectural Design” is too low-level
- ◆ This Method can be used to perform the System architecture activity
  - ISO 12207 requirements are included in the process specified by this method
  - ISO 12207 does not specify architectural viewpoints or document formats
- ◆ Note that Architecture work often performed with some of the ISO 12207 Acquirer activities
  - ISO 12207 does not require a ‘waterfall’ approach between the various activities and parties performing these activities

# IEEE Architecture Working Group: Goals and Objectives

- ◆ Take a “wide scope” interpretation of architecture as applicable to software-intensive systems
- ◆ Establish a conceptual framework and vocabulary for talking about architectural issues of systems
  - Provide a definition of "architecture"...
- ◆ Identify and promulgate sound architectural practices
- ◆ Allow for the evolution of those practices as relevant technologies mature

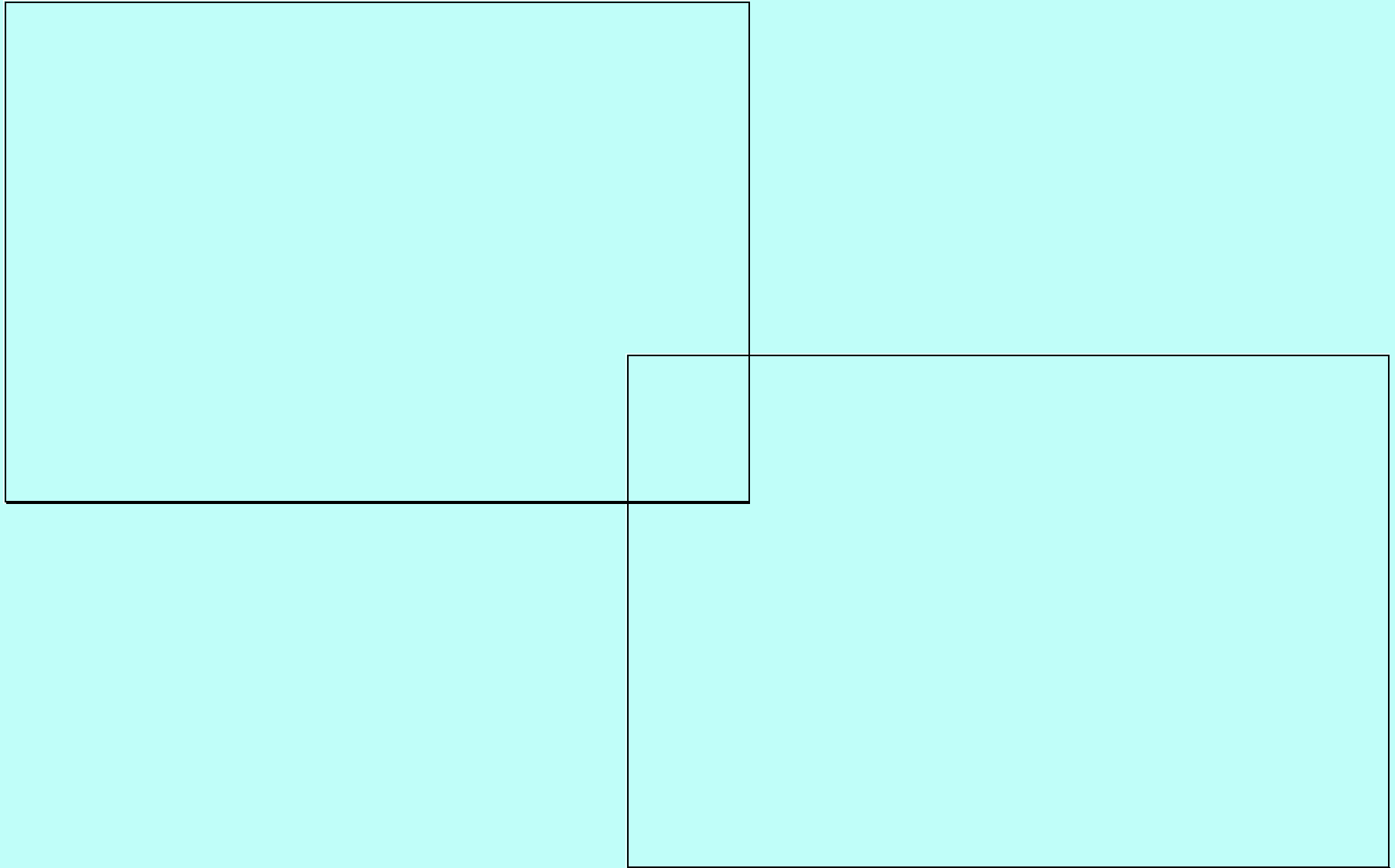
# IEEE Std 1471-2000: Recommended Practice for Architectural Description for Software Intensive Systems

- ◆ P1471 focuses on Architectural Description, not on "Architecture" per-se
  - More consensus on how to describe architectures than on the process or specific contents of architectures
- ◆ Primary contributions:
  - Architectures have to serve multiple stakeholders
  - Architectural descriptions consist of multiple views
    - ◆ No single view captures all salient aspects
  - Contents of views can be captured independent from the system being described
- ◆ Document entered ballot in 1999, completed in 2000
  - Publication in 2000

# IEEE 1471 Vocabulary

- ◆ *architecting*: the activities of defining, documenting, maintaining, improving and certifying proper implementation of an architecture.
- ◆ *architectural description*: a collection of products to document an architecture.
- ◆ *architecture*: the fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution.
- ◆ *view*: a representation of a whole system from the perspective of a related set of concerns.
- ◆ *viewpoint*: a specification of the conventions for constructing and using a view. A viewpoint acts as a pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis.

# Metamodel for Architectural Description



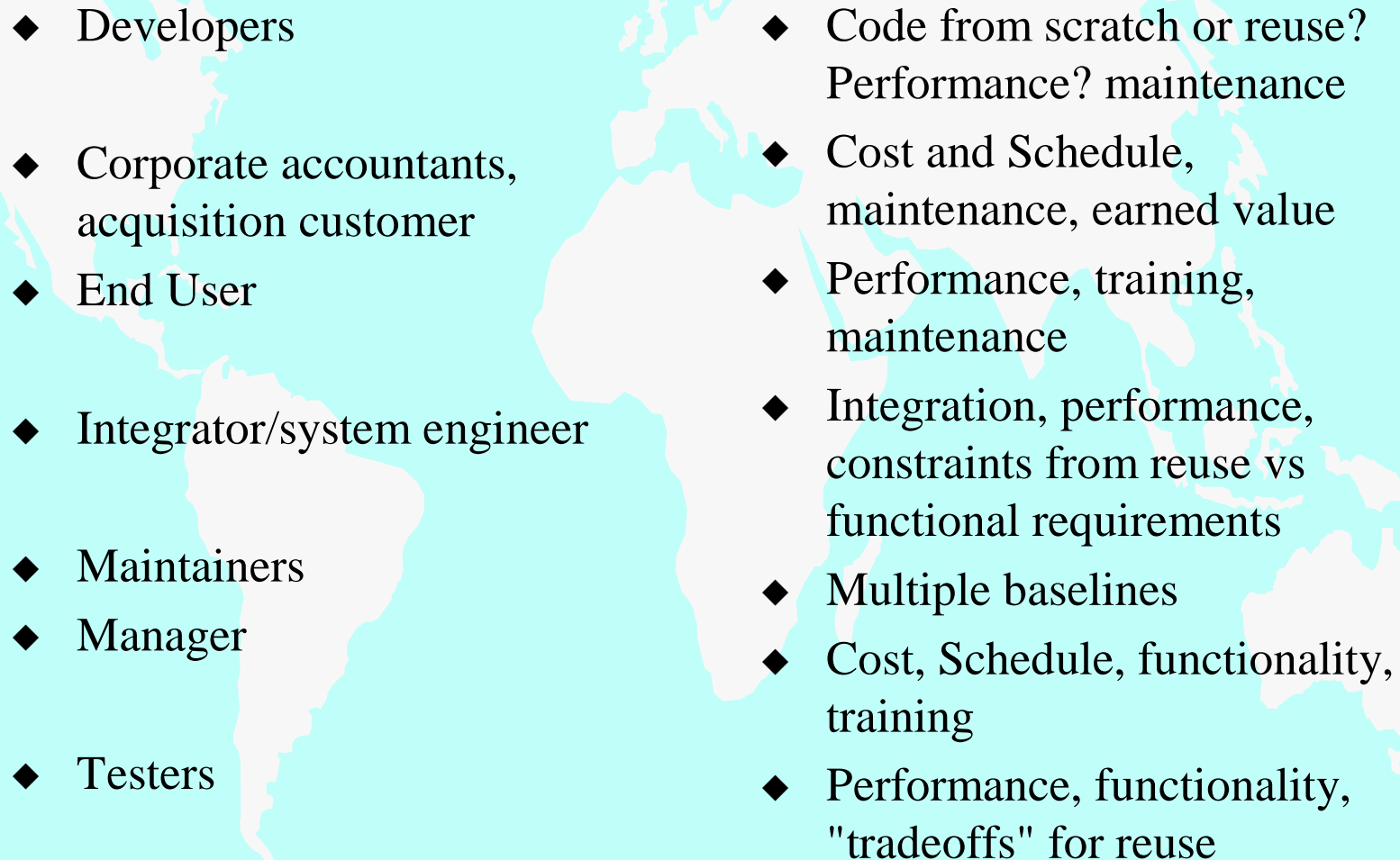
# Library of Viewpoints

- ◆ Kruchten's "4+1": Logical, Implementation, Process, Deployment, Use-Case
- ◆ RM-ODP: Enterprise, Information, Computational, Engineering, Technology
- ◆ C4ISR Framework: Operational, Systems, Technical
- ◆ MITRE-Army Distance Learning: Infrastructure, Instructional, Data, Security, Enterprise Management
- ◆ MITRE-Air Force C2STA : Distribution, Capability, Production, Data, Communication, Security

## Practical Exercise...

- ◆ Consider a radar system where your corporation has decided to bid extensive reuse from an existing product
  - Who are the stakeholders?
  - What are their concerns?
  - What viewpoints would be helpful?

# Stakeholders and concerns

- 
- ◆ Developers
  - ◆ Corporate accountants, acquisition customer
  - ◆ End User
  - ◆ Integrator/system engineer
  - ◆ Maintainers
  - ◆ Manager
  - ◆ Testers
  - ◆ Code from scratch or reuse? Performance? maintenance
  - ◆ Cost and Schedule, maintenance, earned value
  - ◆ Performance, training, maintenance
  - ◆ Integration, performance, constraints from reuse vs functional requirements
  - ◆ Multiple baselines
  - ◆ Cost, Schedule, functionality, training
  - ◆ Performance, functionality, "tradeoffs" for reuse

# Viewpoints



- ◆ Infrastructure : what runs where
- ◆ Static structure: what modules exist, how connected?  
Make vs Reuse?
- ◆ Behavior/Performance
- ◆ Development Sequence/Maintenance
- ◆ Data
- ◆ Security?
- ◆ Business model?

# Wrap Up

- ◆ Why Architecture?
  - To improve the properties of systems we develop and operate; because traditional design methods best address functional requirements; whereas “architecture” gives us a way to non-functional/ilities and trade-offs among requirements
- ◆ What is Architecture?
  - The highest level conception of a system in its environment

## Wrap Up (continued)

- ◆ Who does Architecture?
  - Architecture is an emerging discipline distinct from engineering, both in what is done and when and how it gets done.
  - The Architect works for the Client
- ◆ When?
  - Concurrent with “Requirements” Before design; Throughout life cycle (construction, maintenance and evolution)

## Wrap-Up (continued)

- ◆ How does the Architect do the job?
  - Frame and Understand problem
  - Vision, Goals and Needs: Customer buy-in
  - Identify Stakeholders
  - Select Viewpoints and Model Views
  - Integrate Views
  - Oversee Construction/Production
  - Maintain/Evolve Architecture
    - ◆ Bottom up (from construction), outside-in (from environment),
    - ◆ Variances, Interpretation, Modification consensus process

# Contacts and references

- ◆ David emery:  
emery@mitre.org
- ◆ Rich Hilliard:  
rh@consentcache.com
- ◆ Tim Rice:  
tbrice@consentcache.com
  
- ◆ ° IEEE Architecture WG  
Pages:
- ◆  
<http://www.pithecanthropus.com/~awg>
- ◆ <Http://www.architecting.org>