UNITED STATES
JOINT FORCES
COMMAND

USJFCOM

Service Oriented Architecture (SOA) Implications to End-to-End Assessment

Brian Eleazer
Brian Hall
Robert Kohout
Joint Systems Integration Center
U.S. Joint Forces Command
757-203-4421 / 4453 / 7598
John.eleazer@jsic.jfcom.mil

outline

- Definitional Grounding
- Elements of SOA Implementation
- Notional “V” Approach to SOA
- Lessons Learned and Considerations

Definitional

- SOA Services are autonomous, reusable components that provide specific business or mission capability.
- Two key conditions:
  - – Autonomous, reusable components
    - Exposed, accessible information
    - Availability, Quality of Service (QoS)
  - – Specific business/mission capability
    - Orchestrated usage
    - Measurable operational utility

Elements of SOA Implementation

Each layer presents development & testing challenges …

…characteristics and considerations

Operational Perspective

- End-to-end orchestrated usage of multiple services & systems to specific tasks
- Access of mission-critical data and information to allow commanders and users to adapt to changing mission needs

Systems / Services Perspective

- Loosely coupled components with minimum development assumptions about when, why, or under what environment conditions invoked
- Provide functionality abilities to allocated requirements and specifications

Infrastructure Perspective

- Multiple, independently constructed, ‘adopted’ products configured to interact via persistent network
- Means to connect
  - Registration, subscription, and discovery characteristics
- Cross-domain networks

Characteristics and Challenges

Operational Perspective

- End-to-end orchestrated usage of multiple services & systems to specific tasks
- Access of mission-critical data and information to allow commanders and users to adapt to changing mission needs

- Combination of humans, applications, web-services, networks/back-office, databases, business rules

- De-centralized ownership and control; variety of providers, infrastructure, and consumers
- Lack of design information; subtle engineering design limitations
- SLA and Web-Service Definition Language (WSDL) does not ensure behavior and desired expectation
- Complex end-to-end execution; variations in mission, configuration, and business activities
- Aggregated failures, difficultly in root-cause analysis & assigned correction

Systems / Services Perspective

- Loosely coupled components with minimum development assumptions about when, why, or under what environment conditions invoked
- Provide functionality abilities to allocated requirements and specifications

- Web-service implementation
- Message formats
- Service Level Agreements (SLA)
- Functionality, QoS, Conformance to open standards
- Multi-services constructs

- Unknown context, lack of usage understanding
- Unanticipated demand, impact to QoS & load
- Customized implementation of WSDL, Simple Object Access Protocol (SOAP), Extensible Markup Language (XML), Universal Descriptive Discovery and Integration (UDDI) …
- Lack of stimulation or modeling service behavior
- Misunderstanding data exchanges, lack of common data model
- Second-order and un-intended consequences
- Multiple provider schedules, testing, and increments
- Independent releases; un-defined regression testing parameters

Infrastructure Perspective

- Multiple, independently constructed, 'adopted' products configured to interact via persistent network
- Means to connect
    - Registration, subscription, and discovery characteristics
- Cross-domain networks

- Registration & Discovery
- Transform / Translation of meta-data
- Security authentication
- Message interaction
- Data bases & data stores

- Response time & latency
- Limited technical information (closed design or proprietary)
- Complexity in configuration, administration, and security protocols
- Dynamics in releases, patches, service packs
- Cross-domain variations, implementation of standard, meta-data attributes

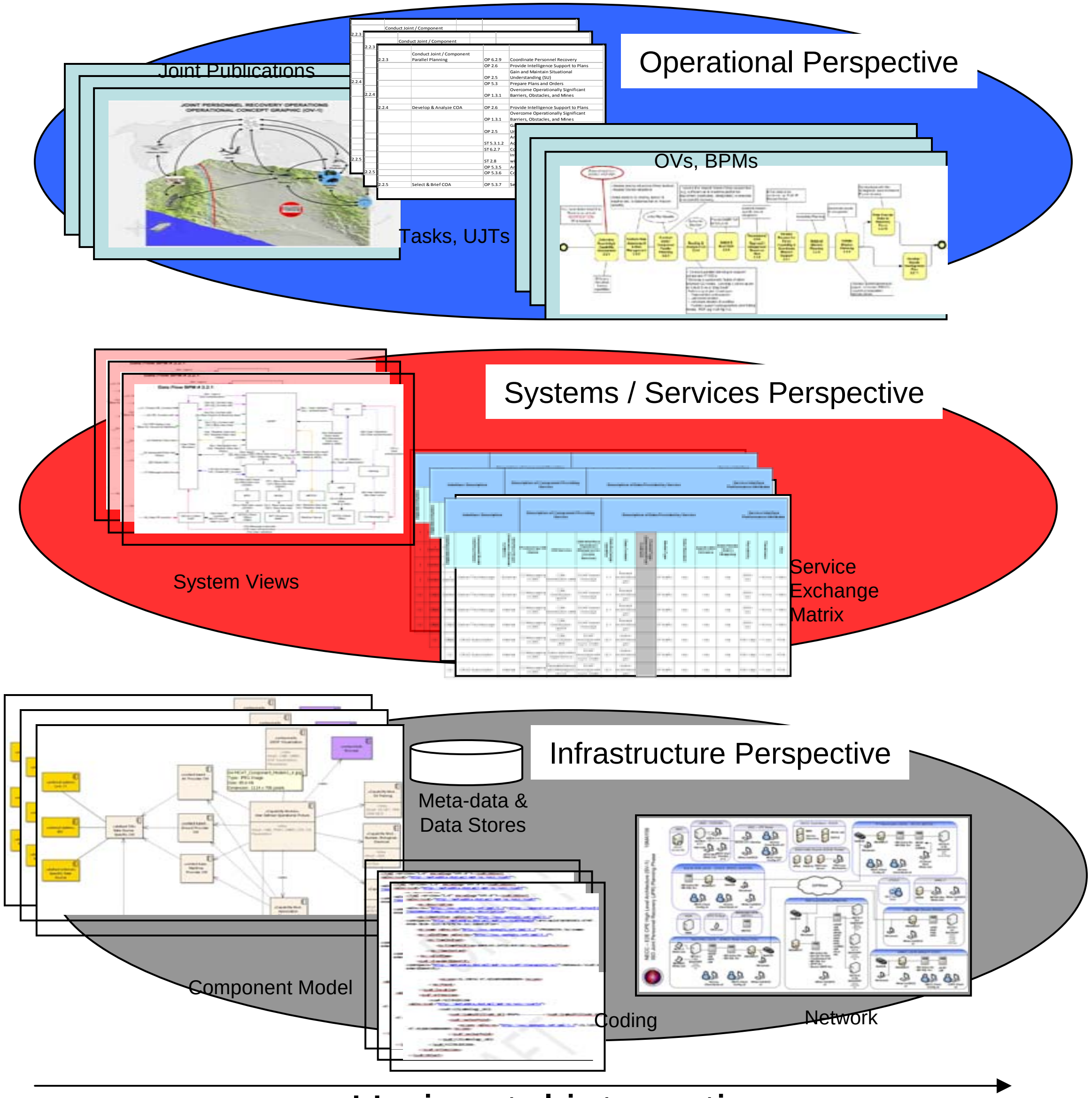
Horizontal Composition

Traditional view toward integration

Development of Operational-to-Tactical level of operations

Integration of systems, applications, technical exchanges, and components across development activities

Success dependent upon understanding and agreement on foundation



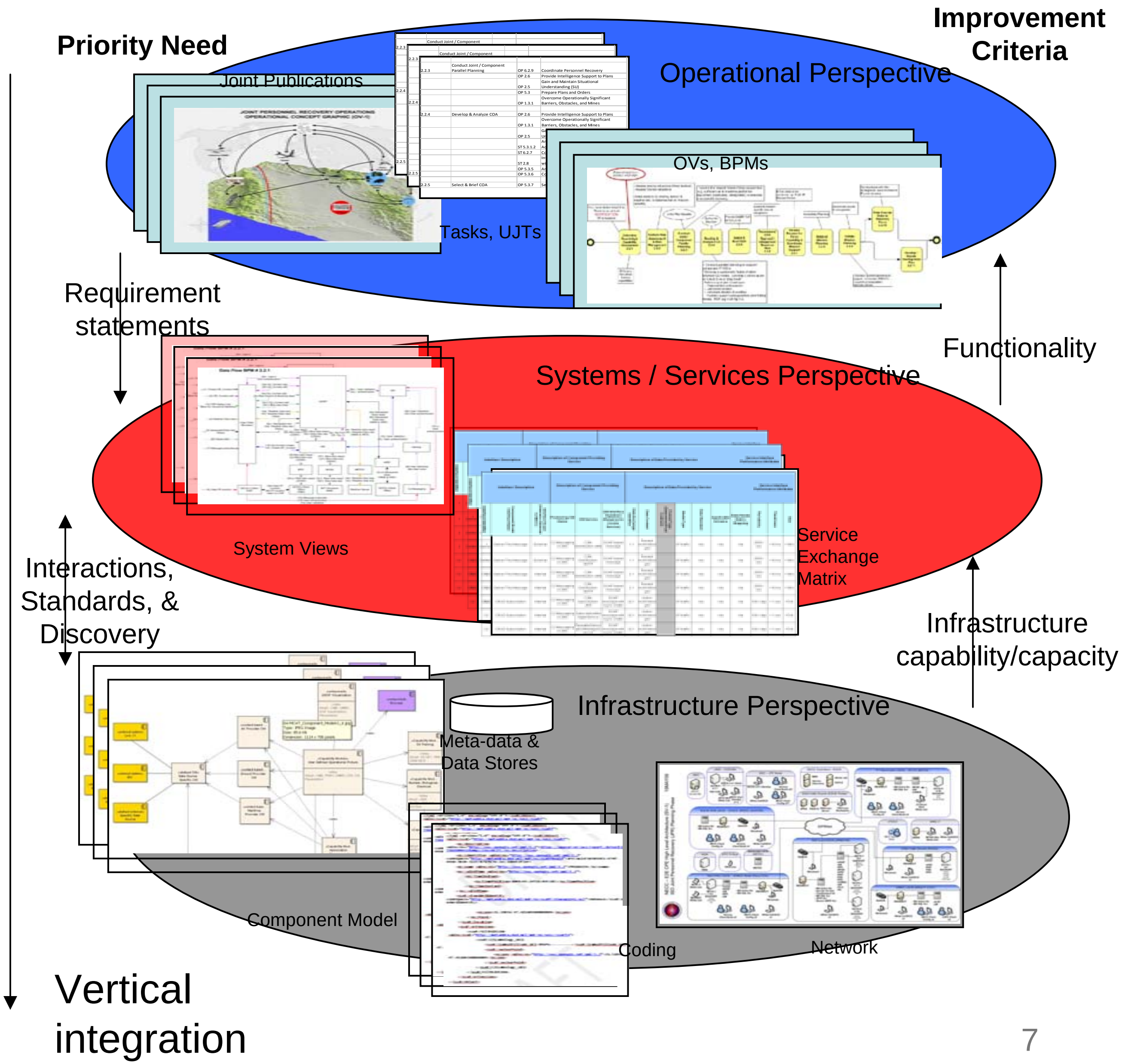
Vertical Decomposition

An Integrating views

Mission-to-task requirements driving prioritized functionality specifications

Functionality dependent upon SOA interactions, standards, discovery mechanisms

Success dependent on an iterative vertical communication, alternative analysis, trade-offs, and engagements



Priority Need

Improvement Criteria

Operational Perspective

Joint Publications

Tasks, UJTs

OVs, BPMs

Requirement statements

Functionality

Systems / Services Perspective

System Views

Service Exchange Matrix

Interactions, Standards, & Discovery

Infrastructure capability/capacity

Infrastructure Perspective

Meta-data & Data Stores

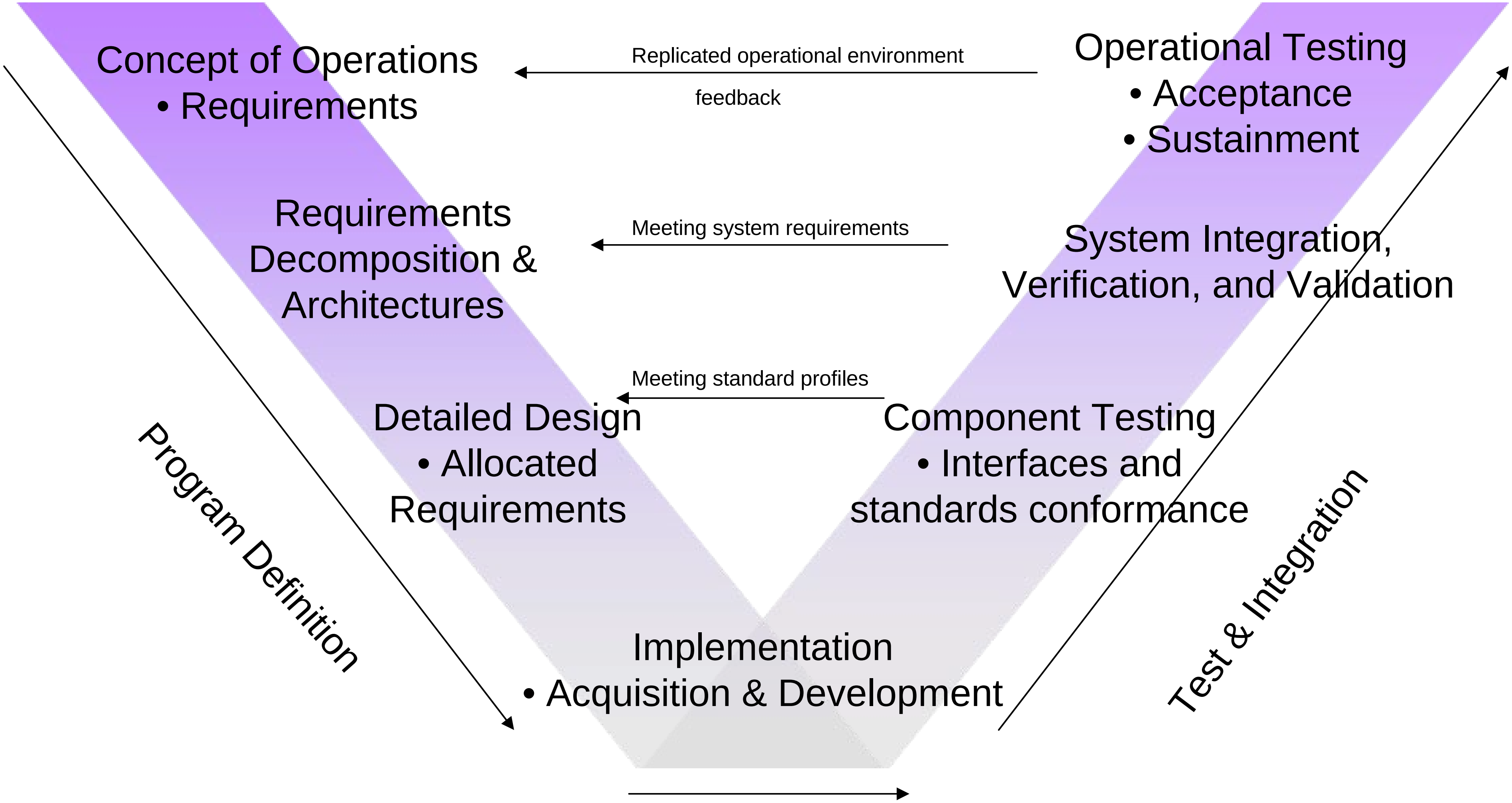
Component Model

Coding

Network

Vertical integration

Standard “V” System Engineering Development



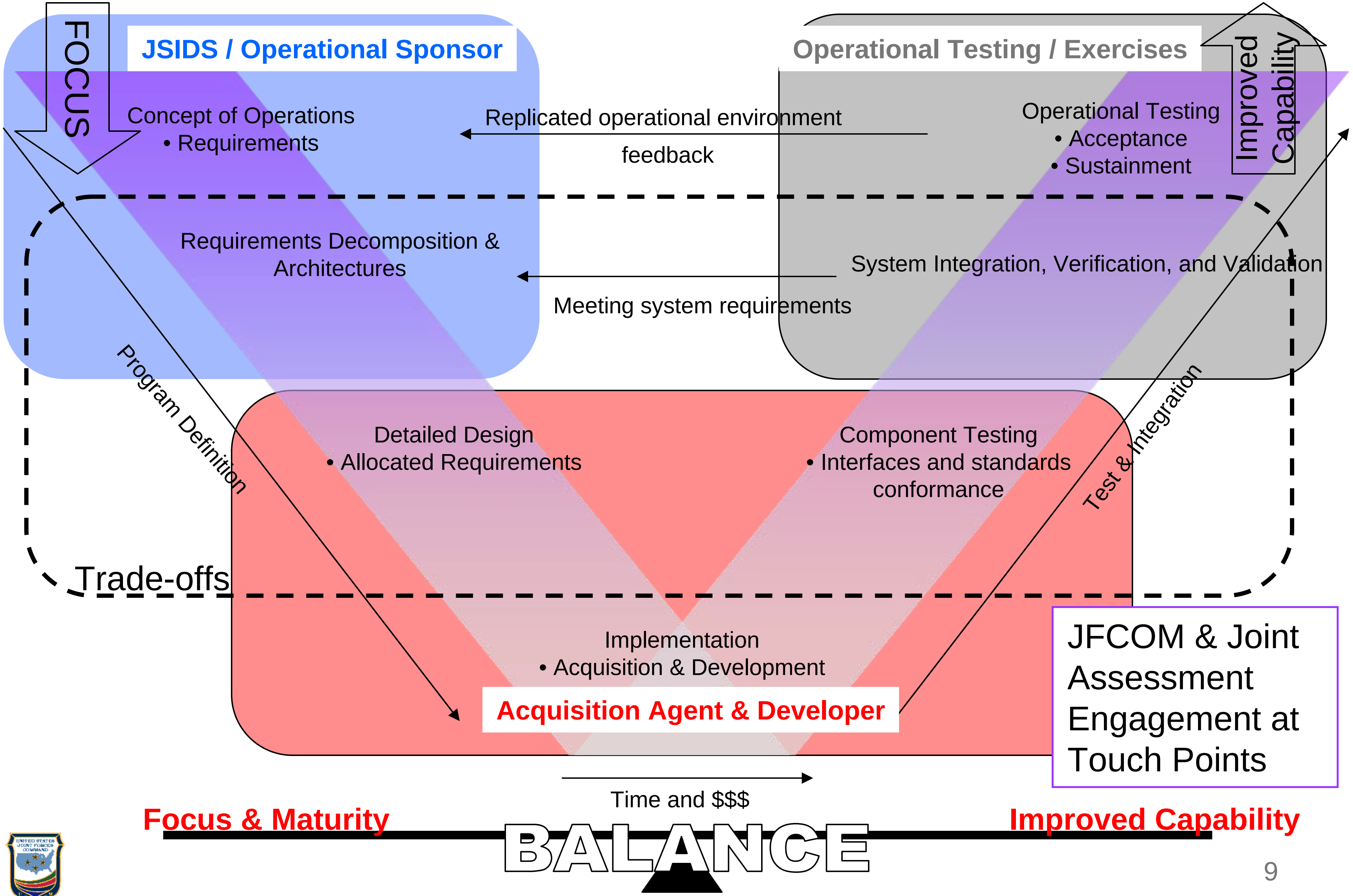
Concept of Operations
• Requirements
Replicated operational environment
feedback
Operational Testing
• Acceptance
• Sustainment
Requirements Decomposition & Architectures
Meeting system requirements
System Integration, Verification, and Validation
Meeting standard profiles
Detailed Design
• Allocated Requirements
Component Testing
• Interfaces and standards conformance
Implementation
• Acquisition & Development
Program Definition
Test & Integration

Time and $$$

UNITED STATES JOINT FORCES COMMAND

UNCLASSIFIED

Notional SOA Implementation



FOCUS

JSIDS / Operational Sponsor

Concept of Operations
- Requirements

Replicated operational environment

feedback

Operational Testing / Exercises

Operational Testing
- Acceptance
- Sustainment

Improved Capability

Requirements Decomposition & Architectures

System Integration, Verification, and Validation

Meeting system requirements

Program Definition

Test & Integration

Detailed Design
- Allocated Requirements

Component Testing
- Interfaces and standards conformance

Trade-offs

Implementation
- Acquisition & Development

Acquisition Agent & Developer

JFCOM & Joint Assessment Engagement at Touch Points

Time and $$$

Focus & Maturity

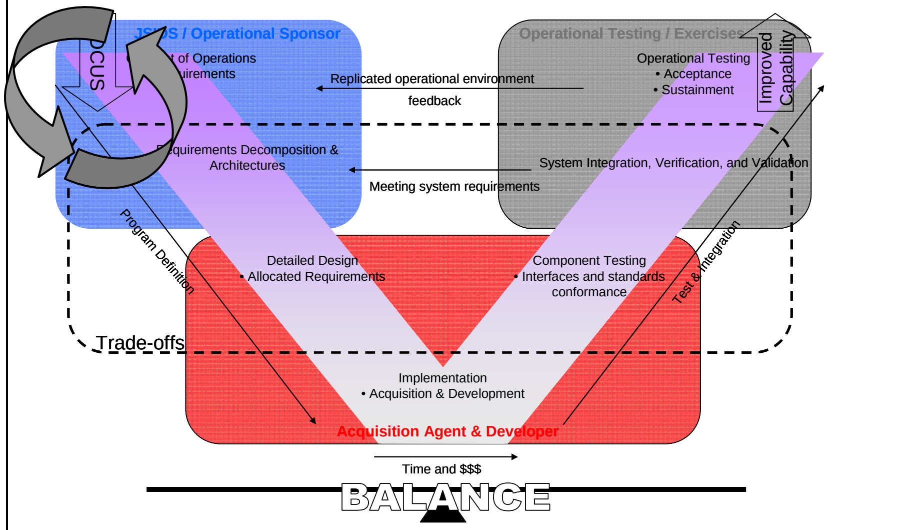
Improved Capability

BALANCE

Lessons Learned and Consideration

Lessons Learned:

- Need ***common theme*** to develop centralize plan
  - •"unwillingness of the services … to agree to a joint command and control modernization that is centrally managed" (Senate language)
  - •"committee concerned that [DoD] has been unable to develop a rational plan" (House language)
- NECC program did not establish an focus imperative linkage direct requirement issues* to test/assess to baseline
  - •Test constructs not based on warfighter utilization of GCCS FoS or specific shortfall mission objective
- The to-be migrated modules fell short of what the warfighter actually uses in the field today for mission accomplishment
- Doctrinal level operational architecture develop concurrent and post-selection of modules and functional to be enhanced.
- Late products and decision compressed time required to 'flesh-out' integrated architectures and cases
- Operational response times for mission executions were not used as testing benchmark due to SOA immaturity and integration with legacy baseline


JSIDS / Operational Sponsor
DCUS
t of Operations
uirements
Replicated operational environment
feedback
Operational Testing / Exercises
Operational Testing
• Acceptance
• Sustainment
Improved Capability
Requirements Decomposition & Architectures
System Integration, Verification, and Validation
Meeting system requirements
Program Definition
Test & Integration
Detailed Design
• Allocated Requirements
Component Testing
• Interfaces and standards conformance
Trade-offs
Implementation
• Acquisition & Development
Acquisition Agent & Developer
Time and $$$
BALANCE

Establish an operational Imperative

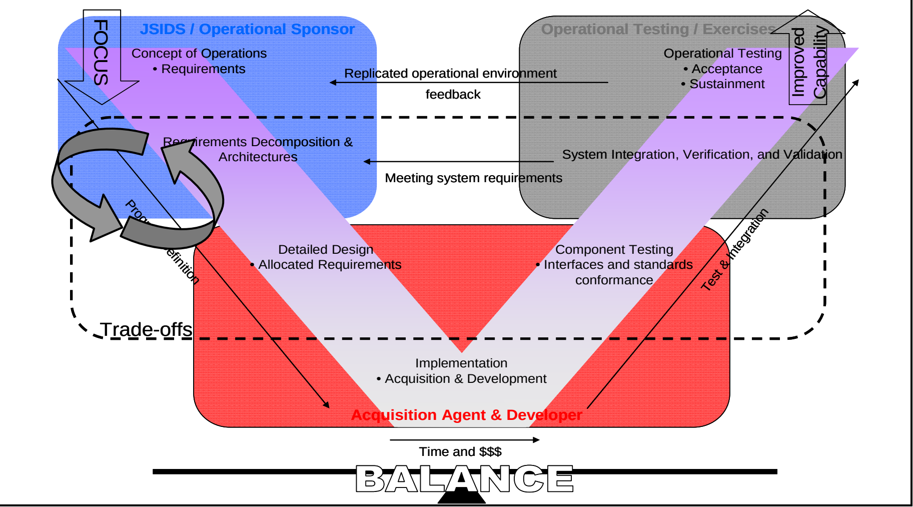
- Determine outcome-based focus
  - Measurable joint issue
- Associate with task / mission objectives
  - Pre-defined business processes
  - Characteristics of specific Joint Scenario
  - Identify 'the user' and 'intended environment'

* NECC Requirements Integration Document (NRID) or the Global Information Grid Requirements Integration Document (GRID) issues

Lessons Learned and Consideration

Lessons Learned:

- Maintain traceability of module development to “As-Is” Functional Transition Plan (FTP) to identify and integrate to current systems and functions.
- The decomposition of requirements and functional business processes (Mission Capability Area (MCA) Business Process Models (BPM)) must be reusable elements with trace-able to mission-task based driven requirements.
- Minimize parallel processes: developers and operational subject matter experts working with incomplete or not vetted information; i.e., interfaces, data sources, and component dependencies.
- Fluid baselines force fluid integration objectives and plausible test objectives
- Developer specifications and waiver based on ‘engineering mission threads’ devoid on operational usage; increase risk consequences in mission usage
- Lack of up-front integrated architecture (mission thread) assessment resulted in assortment of limited modules that are less enable to execute mission tasks (i.e., operational demonstration of end-to-end mission thread across enterprise and service infrastructure)
- ‘Tailoring’ of DoD Architecture Framework (DODAF) provided inadequate community understanding and details to support integration and testing.


FOCUS
JSIDS / Operational Sponsor
Concept of Operations
• Requirements
Operational Testing / Exercises
Operational Testing
• Acceptance
• Sustainment
Improved Capability
Replicated operational environment
feedback
Requirements Decomposition & Architectures
System Integration, Verification, and Validation
Meeting system requirements
Prog... efinition
Detailed Design
• Allocated Requirements
Component Testing
• Interfaces and standards conformance
Test & Integration
Trade-offs
Implementation
• Acquisition & Development
Acquisition Agent & Developer
Time and $$$
BALANCE

Provide an integration construct

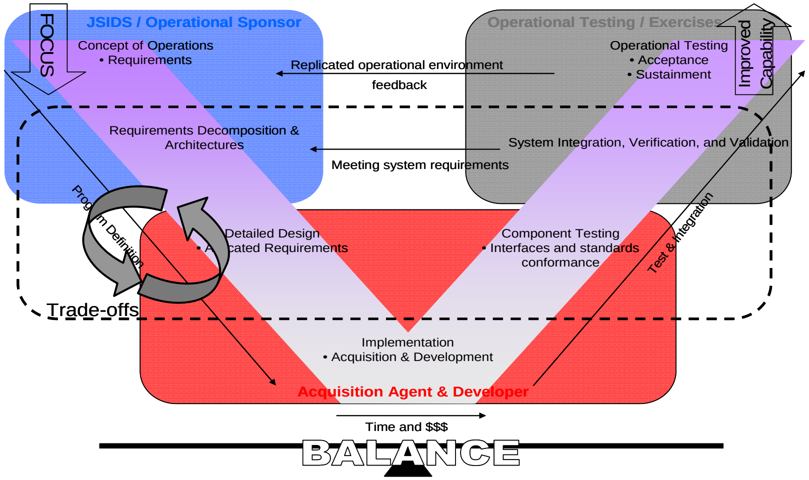
- Reflect critical business processes in architectures
- Develop an initial Operational Concept (Ops Con)
  - –Define mission profiles & conditions
  - –Identify intended environment (as-is) and benchmarks
- Collaboratively, define
  - –Dependencies and linkage to FTP
  - –Program recognition of mutual multi-program objectives (SoS requirements)

UNCLASSIFIED

Lessons Learned and Consideration

Lessons Learned:

- Interdependencies between program and core enterprise services, capabilities, and infrastructure created fundamental risks and coordinate challenges (i.e., findings, resolutions, and synch schedules)
- Evaluate services in terms of their maturity to support an engineering thread and/or operational thread
  - Include: understanding and evidence to meet enterprise interdependencies, linkage to FTP, and other development dependencies.
- Design and selection to high level and "button-logy" abstractions exasperate issues to spread over abstract Mission Capability Areas (MCAs) and designs required to support specific focused operational missions.
- Not grouping services for task-driven assessment decrease ability to evaluate trade-off
  - Resulted in difficulty executing any one of more than 800 Master Scenario Events List (MSELs)
- The key to ABC (Adopt-before-Buy, Buy-before-Create) approach requires 'assessment' of 'product' adaptation in commercial best practices, architectures, and standards (& past performance) for C2.
- Piloting and promoting services with priority #1 and priority #2 issues invites subsequent test headaches
- A collaborative ''sand-box; requires technical and usage maturity to successfully access critical information, collaborate, sharing knowledge products, and testing / validation infrastructure



JSIDS / Operational Sponsor
FOCUS
Concept of Operations
• Requirements
Replicated operational environment
feedback
Operational Testing / Exercises
Operational Testing
• Acceptance
• Sustainment
Improved Capability
Requirements Decomposition & Architectures
Meeting system requirements
System Integration, Verification, and Validation
Program Definition
Detailed Design
• Allocated Requirements
Component Testing
• Interfaces and standards conformance
Test & Integration
Trade-offs
Implementation
• Acquisition & Development
Acquisition Agent & Developer
Time and $$$
BALANCE

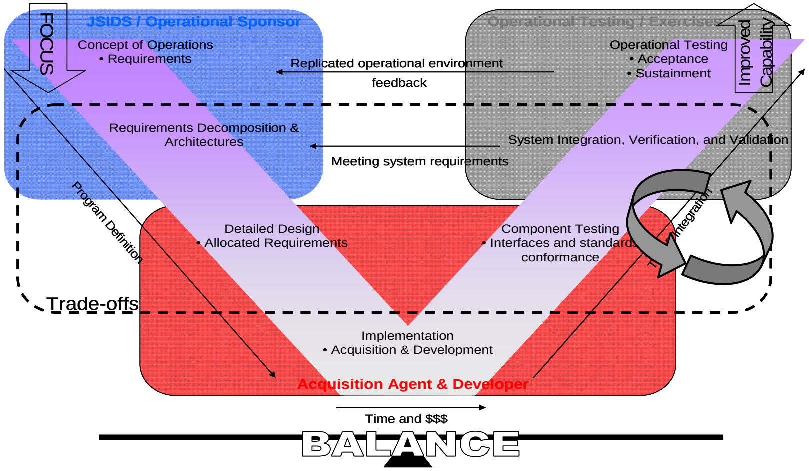
Assess Maturity & Trade-offs, iteratively

- Evidence-based: inclusive of technical maturity, limitations, and feedback
  - –Experimentation, demonstrations and Prototyping
- Associate candidate solutions to focus
  - –Use integrated architecture to identify gaps, shortfalls, outliers
  - –Review candidate solutions; acquisition efforts, schedules, funding, and risk
  - –Develop Mission-focus Analysis of Alterative to support acq/user decisions

Lessons Learned and Consideration

Lessons Learned:

- Joint Engineering / Analyst and Testers require timely access to program and technical information for pro-active engagement and mitigation
- Full visibility required in piloting and testing events for collaboration and assess issues, second order effects, and impacts
- Early opportunity for operational SMEs and developers discussions increase understanding on how technical operations and technical services will be operationally orchestrated and invoked.
- Without information exchange details, personal relationships and emails are not sufficient to communicate exchange paths required across JTF nodes and user-role actors necessary for testing
  - Impromptu Orchestration Team working groups can not guarantee on-the-fly table top analysis, feedback and thoroughness detailed to evaluate test cases
- Use operational mission thread (OMT) matrix approaches to pro-actively trace operational activities-to-functionality-to-service to identify 'holes' in information and gaps
- Lack of early assessment, table-top or limited pilot, prevents critical feedback and increases integration failures. Late issue discovery present root-cause determination challenges in larger end-to-end venues. Moreover, late issues are difficult to correct and integrate (likely across programs) late in development cycles


JSIDS / Operational Sponsor
FOCUS
Concept of Operations
• Requirements
Operational Testing / Exercises
Operational Testing
• Acceptance
• Sustainment
Improved Capability
Replicated operational environment
feedback
Requirements Decomposition & Architectures
System Integration, Verification, and Validation
Meeting system requirements
Program Definition
Integration
Detailed Design
• Allocated Requirements
Component Testing
• Interfaces and standards conformance
Trade-offs
Implementation
• Acquisition & Development
Acquisition Agent & Developer
Time and $$$
BALANCE

Evaluate, Plan, and Perform early integration and feedback

- Identify 'white-box', composite, and rapid feedback testing opportunities
  - –Synchronize program schedules
  - –Plan for within Test and Evaluation Master Plan (TEMP)
  - –Anticipate joint context & use cases
- Anticipate / Plan for regression testing criteria – ToR and responsibilities
- Use pre-defined business activity for simulation & interaction modeling

Summary

- Less-cost, faster cycle, and right objectives
  - Requires clear objectives, actively balance, and meaningful engagement
  - If you don’t have a specific focus – you won’t meet it
- Provide an integration construct
  - If you don’t plan for integration – it won’t integrate later
- Assess maturity and trade-off
  - Use evidence-based, informed decisions
  - It seems all programs share similarities: behind schedule, over cost
- Plan and conduct early integration mitigation and feedback
  - Anticipated in schedule and program test strategy
  - Mutual perspective leads to success