Veterans Health Administration
Common Services
October 27, 2005

Keith Cox
Acting Associate Deputy Director
Health Systems Design & Development
“A technical overview in non-technical terms…”
The Motivation

• Legislative and Departmental Mandates
  – The Clinger-Cohen Act
  – E-Government
  – One VA

• Improve level of interaction with private sector information systems.

• Enhance ability to utilize commercially produced software and IT products.

• Inadequacy of current technology and architecture
  – Incompatible technology
  – “Silo” application design
  – Difficult to maintain
Enterprise Architecture Implementation

- N-tier software design
- Modern technology
- Service oriented architecture
  - Decomposition of application capabilities
  - Creation of service components
  - Methodology for sharing services
- Centralized/distributed deployment
  - Virtual single record view
  - Synchronization across systems of interest
  - Support for continuity of operations
- Standards / Information Model based
N-tier Architecture
Shared Services Architecture

Application Services

1. Browser
2. Consuming application, service or external system
   Java, C++, dotNet, etc.

3. Shared Business Service
   Java

CAIP

Business Services

Consuming application, service or external system

Data Services

Consuming application, service or external system

Java, C++, dotNet, etc.

CAIP

Database

CAIP

Java

service sharing methodology

Client

Application

Client
What is CAIP?

- “Cross Application Interface Protocol” (CAIP)
- CAIP defines a framework for establishing a consistent foundation for integration across applications
- A layer of abstraction between shared business services and the applications that subscribe to them
- CAIP can serve as a technology adapter between business services created in Java and applications or systems based in other technologies
N-tier Shared Services Example
System / Service Deployment Assumptions

Central System
- Service “A”

Central System
- Service “B”
- Service “C”
- Service “D”

Central System
- Service “E”

Local System
- Service “B”
- Service “C”
- Service “D”
- Service “E”

Local System
- Service “B”
- Service “C”
- Service “D”
- Service “E”

Local System
- Service “B”
- Service “C”
- Service “D”
- Service “E”

Medical Center
- Legacy VistA

Medical Center
- Legacy VistA

Medical Center
- Legacy VistA

Web Interface
Publish & Subscribe

Scenario:
Site “A” publishes an event in Hub-and-spoke configuration

Central System

Shared Service
Publish / Process

Delivery Service
In
Out

Local System
Site “B”

Shared Service
Process / Publish

Delivery Service
In / Out

Local subscribers

Local System
Site “A”

App.

Shared Service
Process / Publish

Delivery Service
Out

Local subscribers

Local subscribers
Redundancy for Test / Failover?

Central HeV Systems

Local HeV Site “A”

App. Server (production)

Db Server (production)

Interface Engine (production)

Interface Engine (test/backup)

Local HeV Site “B”

App. Server (production)

Db Server (production)

Interface Engine (production)

Interface Engine (test/backup)

Redundancy for Test / Failover?
Data Migration Concepts

Business Service Interaction

Data & Metadata Cleanup

Source System

Extract Logic

Transport Method

Staging System

Import Logic

Target System

Sync. System

Synchronization Process

Data Migration

Data Migration Concepts
Common Services Focus

- Provision/coordination of core functionality
  - Business support functionality including Person Service and Standard Data Service
  - Middleware services that support the sharing of services and data between applications and systems
  - Infrastructure services such as security, auditing, archiving, etc.
  - Hardware deployment necessary to support application modernization.
- Maturation of organizational support structures
- Proof of concept for SOA implementation
Scope:
Common Services vs. common services

Application Services
- ... Enter order
- Make Appt.
- Patient Lookup
- Data Quality
- Provider Enter/edit
- SDS Maint.
- Registration
- ...

Shared Business Services
- ... Lab Service
- Appt. Service
- Delivery Service
- Person Service
- Standard Data Service
- Audit Service
- Eligibility/Enrollment Service
- ...

Data Services
- ... PATS
- VPFS
- ADR
- Local ADR
- SDS
- HDR
- Scheduling
- ...

Deploy Hardware
- ... Site Prep.
- Determine Capacity
- Coordinate Software Deployment
- Coordinate CS interests
- Monitor Timelines
- Site Prep.
- Procurement
- ...

Misc.
- ... Public Relation
- Educate Sites
- Coordinate Sequence
- Provide Organization Information
- Evaluate Process
- Train Staff
- Coordinate External Systems
- ...

HealtheVet
VistA
Common Services Efforts

• Person Service
  – Lookup
  – Demographics
  – Identity Management

• Standard Data Service
  – Table factory
  – Replication
  – Authoring
  – Maintenance

• Service Oriented Implementation
  – CAIP
  – Delivery Service
  – Subscription Management Service
  – Messaging Service
  – Dynamic Routing Service
  – Naming / Directory Service
  – HL7 Segment Service
Common Services Efforts (cont.)

- Application Preference Service
- Audit Service
- Archive Service
- Enterprise Exception Log
- Administrative Data Repository (ADR)
- ADR Data Migration
- Institution file re-design
- Data Standardization
- National Provider Identifier
- Metadata Service
- Cross Service and Extended Transaction Manager
- Non-Person Service
Note: This is a representation of concepts, not technical structure of files, etc.
Issues – A share of the wealth...

- Data
  - Standardization
  - Cleanup
  - Metadata uniformity
  - Synchronization
  - Migration
  - Ill formed data
  - History
  - Checksums
More Issues...

- Software Performance Engineering (SPE)
  - Invalid assumptions
  - Baseline
  - Usability vs. performance
  - Tolerance/threshold
  - Modeling/prediction
  - Benchmarking
  - Expectation management
  - Test environment
Yes, more issues...

- Versioning
  - Software
  - Metadata
  - Table content
- Overarching Requirements
- N-tier Requirements Management
- System interdependency
Questions?
Backup slides
Software Versioning Characteristics

- Co-existence of multiple component versions
- Abstraction of versioning consumption specifics
- Discoverable at development time and run time
- Centrally and locally manageable configuration
- Controlled consumption
  - Arbitrary – maintain FDA device relationships
  - Dynamic – upgrade per compatibility
  - Selected – honor consumer’s preference
  - Forced – upgrade by mandate
Software Versioning Motivations

- Limited exposure in production environments
- Reduced complexity in implementing software upgrades
- Maximize functional flexibility
- Coordinated software rollout and activation
- Effective software configuration management
- Shift impacts from hard coded application logic to application runtime configuration.
Factors Limiting Versioning Effectiveness

- Backwards and forwards compatibility
- Shared computing environment
- Software update types
- Deployment architecture
- N-tier development characteristics
Factors: Compatibility Considerations

- Backward compatible – new versions must be compatible with impact of older versions on the shared environment
- Forward compatible – old versions must be compatible with impact of newer versions on the shared environment
- Compatibility types
  - Syntactic – the structure of API signatures, message formats, etc.
  - Semantic – the meaning of data, business rules, GUI navigation logic, etc.
Factors: Shared Computing Environment

• Shared transactions – both semantic and syntactic concerns (e.g. message flow across systems)
• Shared data – both semantic and syntactic concerns
  – Shared data types
    • Primary data (patient records, etc.)
    • Reference data
  – Shared data semantic and syntactic elements
    • Conceptual content
    • Domain (e.g. elements in a pick list)
    • Metadata (field length, etc.)
Factors: Software Update Types

- Defect repair – largely transparent from a syntactic/semantic standpoint
- Enhancement – potentially syntactic or semantic (or both)
- Business Process Re-engineering – likely to have significant semantic changes and potentially syntactic changes as well
Factors: Deployment Architecture

- Central/Web based – Voluntary
- Central/Web/Integrated – PATS, VPFS, Blind Rehab
- Central Database/Distributed processing – Scheduling Replacement
- Centralized/Distributed Database & processing – Common Services
- Decentralized/Integrated – Imaging
Co-existent Version Deployment

<table>
<thead>
<tr>
<th>Opportunity</th>
<th>Limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>No data, local data</td>
<td>Shared, distributed data</td>
</tr>
<tr>
<td>Local transactions</td>
<td>Cross-system transactions</td>
</tr>
<tr>
<td>Forward/backward compatible</td>
<td>Forward/backward incompatible</td>
</tr>
<tr>
<td>Application Service updates</td>
<td>Database, Business Service updates</td>
</tr>
<tr>
<td>Decentralized data and processing</td>
<td>Centralized/distributed data or processing</td>
</tr>
<tr>
<td>Defect repairs</td>
<td>Enhancements, Business Process Re-engineering</td>
</tr>
</tbody>
</table>
Conclusions: Versioned vs. Incremental release

- Governed by factors of environment and software update characteristics
- Constrained by limiting factors that exist
- Forward/backward compatibility is not consistently achievable
- Changes to shared data and cross-system transactions cannot be effectively partitioned
- Versioning rigor is needed to support both co-existent versioned and incremental (big bang) release approaches
Conclusions: Reference Data

• Temporal model vs. versioned data
  – Temporal – provides longitudinal view of single version, preferable for frequent content changes
  – Versioned – necessary for multiple views of a single table to co-exist

• Recommendation – Versioned/temporal
Conclusions: Versioning Implementation

- Implement for software components, business rules, XML schemas, software distributions, API integrations and reference data content
- Implement as needed for database schemas and metadata
- Make versioned vs. incremental release decisions in Release and Configuration Mgt. (not development)
- Create tools to manage runtime version consumption
- Establish test process for both versioned and incremental release paths
Next Steps

- Establish organizational consensus
- Define versioning standard or policy development teams must follow
- Define software test/release process
- Create development cookbook for meeting versioning standard
- Develop runtime version consumption mapping tools
Scenarios

1. Update local system: update locally, add to local queue for central processing, publish to local sites
Scenarios
1. Update local system (VistA): update locally, add to local queue for central processing, publish to local sites
2. Update local system (RApp): update locally, add to local queue for central processing, publish to local sites
3. Update central system: Update central site then publish
Scenarios

1. Update local system: Check central site before updating locally, add to local queue for central processing
2. Update central system: Update central site then publish
3. Update external system: Update central site, then publish