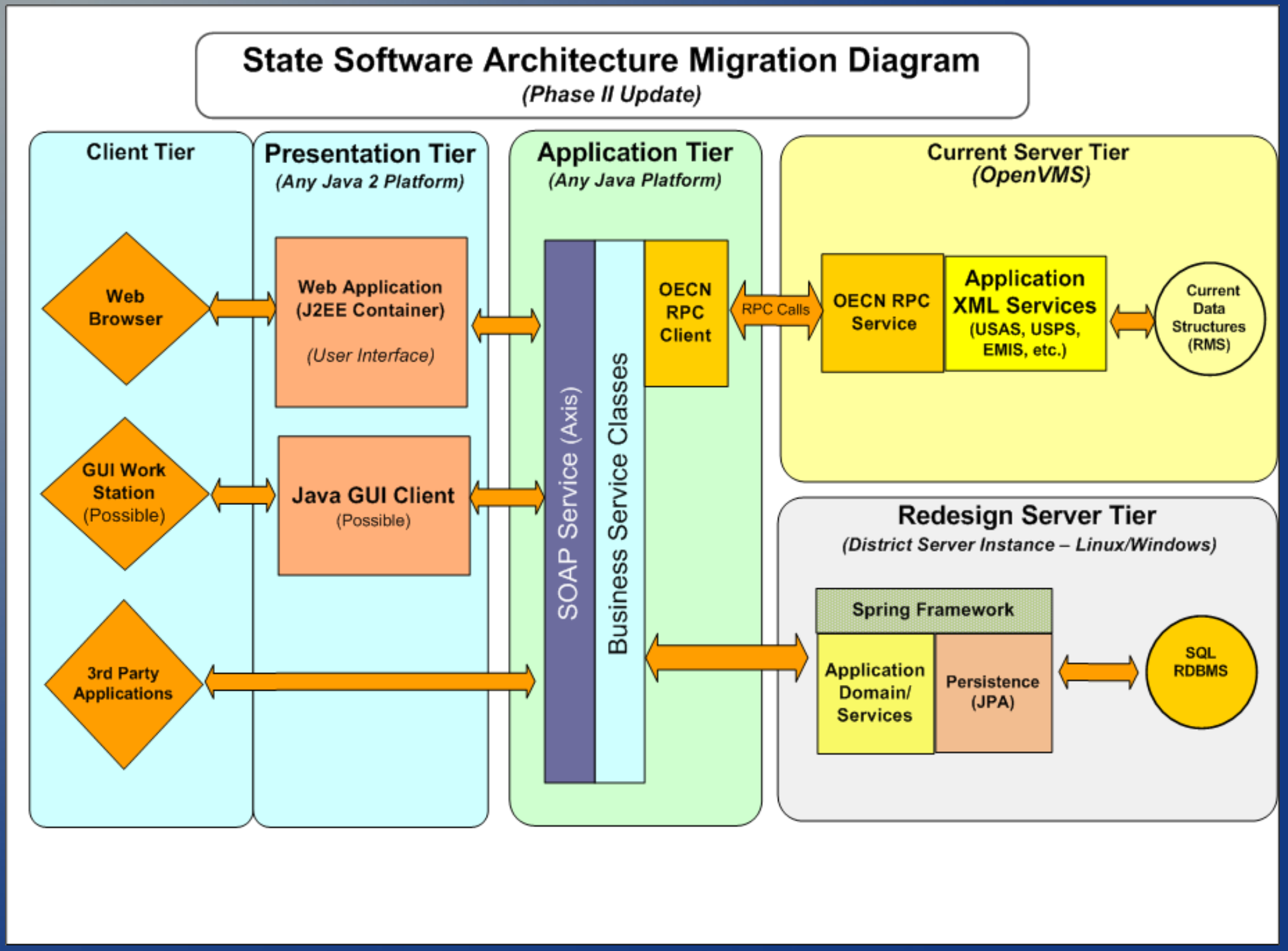
USxS-Redesign

Status, Current Design & Implications

MCOECN Directors

Dave Smith; SSDT


State Software Architecture Migration Diagram
(Phase II Update)
Client Tier
Presentation Tier
(Any Java 2 Platform)
Application Tier
(Any Java Platform)
Current Server Tier
(OpenVMS)
Web Browser
Web Application (J2EE Container)
(User Interface)
OECN RPC Client
RPC Calls
OECN RPC Service
Application XML Services
(USAS, USPS, EMIS, etc.)
Current Data Structures (RMS)
GUI Work Station
(Possible)
Java GUI Client
(Possible)
SOAP Service (Axis)
Business Service Classes
Redesign Server Tier
(District Server Instance – Linux/Windows)
3rd Party Applications
Spring Framework
Application Domain/ Services
Persistence (JPA)
SQL RDBMS

Why Redesign?

Wood Wright Shop

vs.

New Yankee Workshop

- Environment:
  - 1800's Techniques
  - Hand Tools
  - Makes Own Tools
- Skilled Craftsman
- Quality Product
- Labor Intensive

- Environment:
  - Modern Techniques
  - Power Tools
  - Off Shelf Tools
- Skilled Craftsman
- Quality Product
- Resource Intensive

Why Redesign?

OpenVMS/COBOL

vs.

Java / SQL Database

- Environment:
  - 1990's Techniques
  - Hand Tools
  - Makes Own Tools
- Skilled Craftsman
- Quality Product
- Labor Intensive

- Environment:
  - Modern Techniques
  - Power Tools
  - Off Shelf Tools
- Skilled Craftsman
- Quality Product
- Resource Intensive

Phase I - Review

- Developed:
  - SOAP Service (3rd Party Integration)
    - Employee Kiosk (MCOECN)
    - Form Share, etc
    - eProcurement
    - EMIS-R SIF Agents
  - Web Application:
    - Nearly all Data/Transaction
    - No Reporting Interfaces

Phase II Started

- SAC recommended starting of Phase II
  - Concentrate on Redesign
  - Classic systems in maintenance mode
- USAS-R
  - Project launched Nov-2009
  - Completely new model and architecture
- USPS-R
  - Project launched Aug-2010
  - Based on knowledge gained by USAS-R

Primary Redesign Goals

- Modernize Architecture/Platform
  - “Cloud” capable
  - Commodity Hardware/Software
  - Reduce Cost of Ownership/Support
  - Platform/Database Agnostic
- Redesign Data Model
  - Not “porting” existing data model
  - Simplify Application
  - Allow for Future Growth/Increase Flexibility
- Partial Compatibility with Classic

Secondary Design Goals

- Implement Features as “Plug-in” Modules
  - Only install features needed
  - Dynamically add features
  - Potential for Third-Party Modules
- “Incidental Enhancements”
  - New Tools provide features for “free”
  - For example:
    - “Event based” customization
    - Flexible filtering for query and reports
    - Output formats: PDF, Excel, Word, XML, JSON, etc

SSDT Redesign

- Team is also being Redesigned
- Agile Software Development Practices:
  - XP/Scrum-like practices
  - Two Week Iterations
  - Unit/Testing, Code Reviews, CI, etc
- Domain Driven Design
  - Iteratively designing/innovating Domain Model
  - Not locked into initial “Big Design Up Front”
- Work is in public view (Developer's Wiki)

Caveat

- All Software Improvement is done in Stages
- Early releases of Redesign may not support all new features
  - Data Model may support new feature
  - But won't be available in User Interface
  - User Interface will be updated in later stages
- Data Model must be compatible with Classic applications, until UI can be updated

Big Picture Design

- Single Tenant Database
  - One District = One Database
  - Similar to Classic systems
  - Easy backup/recovery
- Single Tenant Server
  - One District = One Server
  - Virtualized Server

- Advantages:
  - “Cloud” Capable
  - Flexible Hosting (location irrelevant)
  - Scaling & High Availability
  - Tailored to District

Yes, Virginia... Lots of Servers

- There will be 1000+ state-wide
- ITC's will need:
  - Virtual Infrastructure
  - Or, rent infrastructure from elsewhere
- Value of an ITC
  - Should not be in the data center
    - Increasingly expensive
    - Eventually too expensive relative to alternatives
  - People: Support, Customization, Integration
  - Bandwidth: To access services

Server Management Goals

- never login to a server -

- Easy Provisioning and Installation
  - Pre-configured VM templates provided by SSDT
  - Auto-install and configuration wizards
- Easy Software Updates
  - One-click updates from SSDT
  - Delegated to District personnel
- Centralized Management/Monitoring
- Leverage Cloud Tools:
  - Automated Provision
  - Puppet
  - Nagios, etc

Supportable Platforms

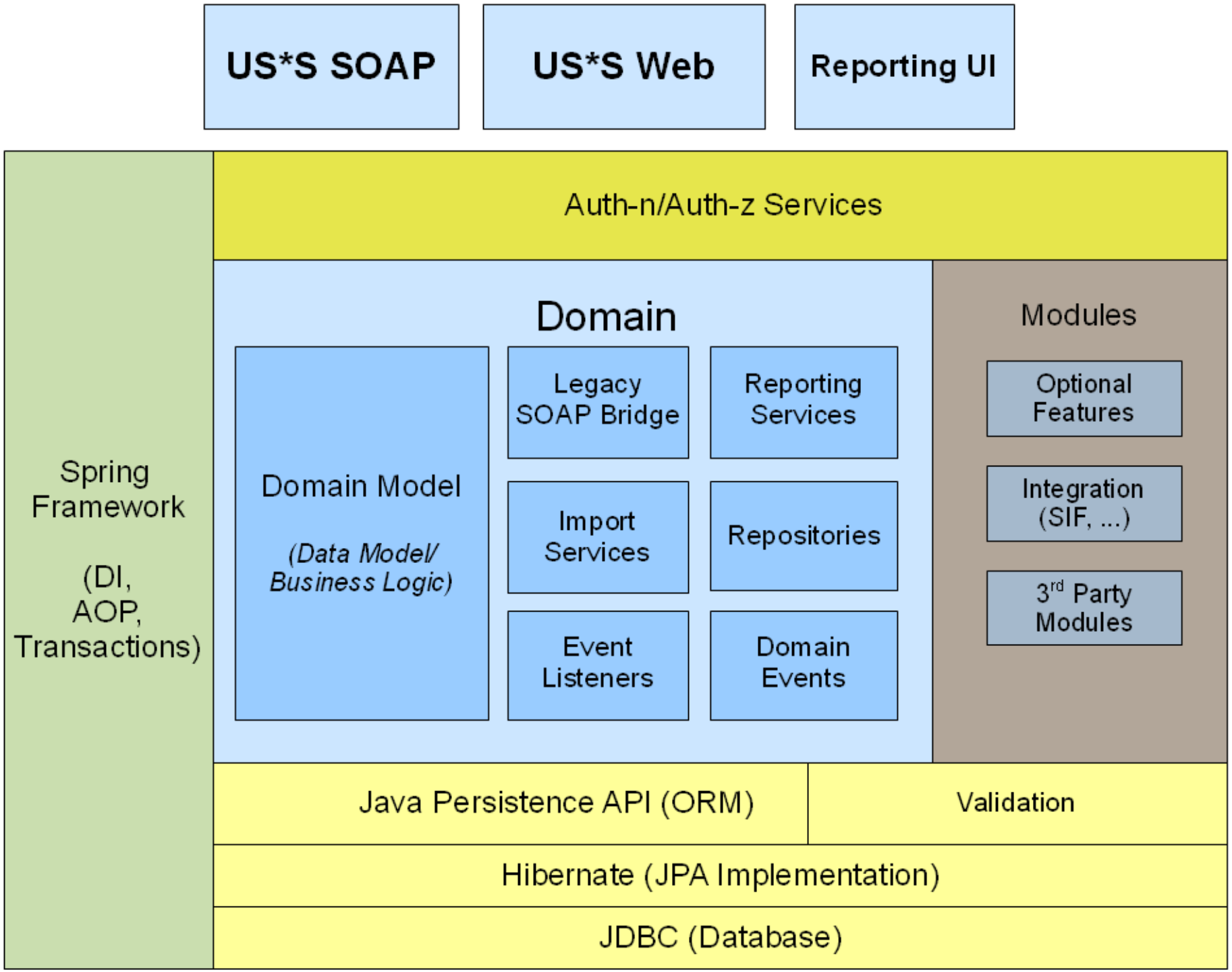
- Software Implemented to be Agnostic:
  - Operations Systems:
    - Windows
    - Linux
    - (Any O/S with Java VM)
  - Database:
    - Oracle
    - MS SQL
    - Open Source: PostgreSQL, MySQL, etc
    - (Any database supported by JPA)
  - Never locked-in to a Vendor again

Deployment Complexity

- Software that "*can run anywhere*" must be deployed "*somewhere*"
- Operating Systems/Databases:
  - Require specific deployment
  - Installation, Versioning, upgrade, backup scripts
  - Each combination increases cost and risk
- Operating System and DB are just Puzzles to Solve
  - They add no value to application
  - OS is container for Java VM
  - Database is dumb data store

Initial Deployment Target

- SSDT will initially support deployment to:
  - Linux (Ubuntu?)
  - PostgreSQL
- Packaging:
  - SSDT will ship “Template VM”
  - Minimal Linux install + Database
  - Bootstrap install of State Software
  - Auto-downloading of remaining software
- Additional deployment targets considered in future


US*S SOAP
US*S Web
Reporting UI
Auth-n/Auth-z Services
Domain
Domain Model
(Data Model/ Business Logic)
Legacy SOAP Bridge
Reporting Services
Import Services
Repositories
Event Listeners
Domain Events
Modules
Optional Features
Integration (SIF, ...)
3rd Party Modules
Spring Framework
(DI, AOP, Transactions)
Java Persistence API (ORM)
Validation
Hibernate (JPA Implementation)
JDBC (Database)

USAS-R Status

- Domain Model
  - 90% of Data Model designed
  - Import process
  - Authorization/Authentication Modules
  - SOAP Bridge (Legacy Compatibility)
  - USAS Web App connected to USAS-R
- Prototype Reporting Service
- Business Logic:
  - Pre-Encumbrance Tracking
  - Encumbrance Tracking
  - Budget Transactions
  - Invoicing

USPS-R Status

- Domain Model
  - ~ 60% of Data Model designed
  - Import process
  - Authorization/Authentication Modules
  - SOAP Bridge (Legacy Compatibility)
  - USPS Web App connected to USPS-R
- Prototype Reporting Service
- Beginning Implementation of Payroll Proces
  - Payroll initialization
  - Deduction Calculation

Import/Conversion

- Written in Parallel with Model Design
- Goal
  - One-step 100% import from Classic USxS
  - All relevant data imported accurately
- Process:
  - Full Extract on OpenVMS
  - USxS-R Import Utility:
    - FTP from VMS system (or local file)
    - Builds database
    - Imports all data

Not Your Father's USxS

- Domain Model will be radically different from Classic Systems
  - Data will be stored and related much differently
  - Far more flexible Data Model
- Modularized/Event Driven
  - Extensibility
  - Customization

Data Model Differences (Example)

- Classic USAS:
  - Expenditure & Budget on one Record
  - USAS Code on same record and every transaction
- USAS-R:
  - Separate Records:
    - Expenditure Account
    - Budget Account
    - USAS Code
  - Model will allow:
    - Multiple Expenditures per Budget
    - But not initially (and unless ODE/AOS change USAS)
  - Transactions will not store USAS Code

Data Model Differences (Example)

- Classic USPS:
  - YTD, FTD, QTD, MTD values stored on each record
  - Updated by CHKUP, QRTRPT, VOIDPY, etc
  - Makes error correction and reporting difficult
- USPS-R:
  - All transaction based
  - Amounts calculated on “On the Fly”
  - Easier error correction
  - Reporting over any time period (e.g. “a prior QTD”)

Data Model Differences (USAS Example)

- Classic USAS:
  - Purchase Order contains items
  - Each Item contains USAS Code
  - Multiple accounts per item is simulated in USAS Web App
- USAS-R:
  - Purchase Order contains:
    - Items
    - Charges (with reference to Expenditure Account)
  - Charges related to items
  - But a Charge could apply to multiple items, or PO
    - Allows possibility of charging entire PO
    - But not initially

USAS Example: Budgeting

- Classic USAS:
  - Initial/Current/Proposed budgets are static amounts
  - FTD & MTD Additions/Deductions
- USAS-R:
  - Budget Transactions (initial budget, adjustment)
  - By “Posting Period”
  - Import process will post transactions
  - WebApp will post delta transaction when “Addition” or “Deduction” is changed

Database ID

- UUID - Universally Unique Ids for primary keys
  - “06751225-8565-4a67-8e09-731882bebfc4”
  - Unique across database, ITC and state
  - Used for internal relationships between tables
  - User (should) never see actual ID
- Advantages:
  - “SIF-Ready”
  - Merges and replication (data warehousing)
  - Disconnected operations, REST-ful Idempotent services
  - Possible to identify data type just from ID
- Disadvantages:
  - SQL database performance (e.g. indexes can not be clustered)
  - May not survive performance testing

Permissions/Roles

- Permission System:
  - Software Defined Fine-Grained Permissions
  - Permissions can grant other permissions
  - Hierarchical, example:
    - USPS_EMPLOYEE grants:
      - USPS_EMPLOYEE_VIEW
      - USPS_EMPLOYEE_CREATE
      - USPS_EMPLOYEE_DELETE
      - USPS_EMPLOYEE_UPDATE
      - USPS_EMPLOYEE_REPORT
- Roles
  - Roles grant permissions
  - User are granted role(s)
  - Software or District Defined

Permissions/Roles

- Initial (Legacy) Roles:
  - USPS - Simulates “Standard” identifier
  - USPS_RO - Simulates “Read-Only” role
  - Others to simulate other Classic USPS Identifiers
- Future:
  - District will be able to define roles with permissions:
  - e.g. “SECRETARIES”, “SUPERVISORS”

Custom Fields

- Replaces “User Defined”
- True Custom Fields:
  - Types:
    - Code, Text, Money, Date
    - Possible: URL, Attachment, Calculated, email, image
  - Description
  - Validation (e.g. Code list of values)
- Defined by (UUID based):
  - District
  - SSDT
  - Third-party vendors

Custom Fields

- Predefined for current User Defined:
  - EMPLOYEE_MONEY1
  - EMPLOYEE_CODE1, etc
- Some Classic USxS Fields may move to CF:
  - “Contract Type”, “Vendor Category”, “Requisition Type”...
  - Maybe “EMIS reporting” fields
  - Allow district to disable if not using, so User will not have to see “User Money 1” on screen

Domain Events

- Allows:
  - Communication between modules w/ Loose Coupling
  - One-to-Many (Broadcast)
  - Module Listeners react to Events for business logic
  - New Plugins can join event processing
- Application will “publish” events:
  - Repository Events:
    - Create, Update, Delete
    - Query, Retrieve
  - Business Logic events:
    - Leave Balance Adjusted (Increase,Decrease)
    - Check Voided
  - Exception Events (Unexpected Errors)
  - Security Events (Login Failure, Role Granted)

Event Contents

- Events contain:
  - Date/Time
  - Elapsed Time
  - Authenticated User
  - Type of Event (Create,Update,...)
  - Source of the event (Repository, Security,...)
  - Target of the Event (Employee, Check, ...)

Event Listeners

- Domain Events do nothing unless something's is listening
- Event Listeners:
  - Are notified of events
  - Listener determines if event is of interest
  - Can Respond to event:
    - Cancel transaction
    - Process related business logic
    - Ignore

Listener Examples

- Listeners:
  - Perform Audit Logging
  - Implement business logic
  - Perform validation (budget check)
  - Send notification message
  - Send message to 3rd Party Application
  - Initiate SIF Event
- Events will be:
  - SSDT Defined
  - District Defined
  - 3rd Party Developers

Example: Pre-Encumbrance Module

- Implemented as:
  - Optional Module
  - Requisition Module is unaware
- Listener:
  - Listens for Requisition Create/Update/Delete Events
  - Responds by:
    - Posting Pre-Encumbrance Ledger against Budget
    - Issues Warnings if updated available budget < 0.00
- Service:
  - Provides Pre-Encumbrance calculation as needed
  - Could be used by 3rd Parties independent of Requisitions

Custom Event Listeners

- District Defined
- Customize USPS behavior:
  - Example #1 (Notification):
    - When a Employee is updated
    - Where “status” is “Terminated”
    - Send Email to Network Administrator
  - Example #2 (Custom Validation):
    - Employee is created or updated
    - Email address is blank and DD notice is “email'
    - Reject transaction and return error message

ODBC is dead, long live...

- Fair Warning:
  - ODBC access by end-users will be unlikely
  - Replaced by “Reporting Services”
- USxS-R design:
  - Database is organized for “Operational” needs
    - Highly normalized
    - Strictly a data store, no business logic
  - Security is only implemented in Domain Model
  - Calculated fields only exist in Model

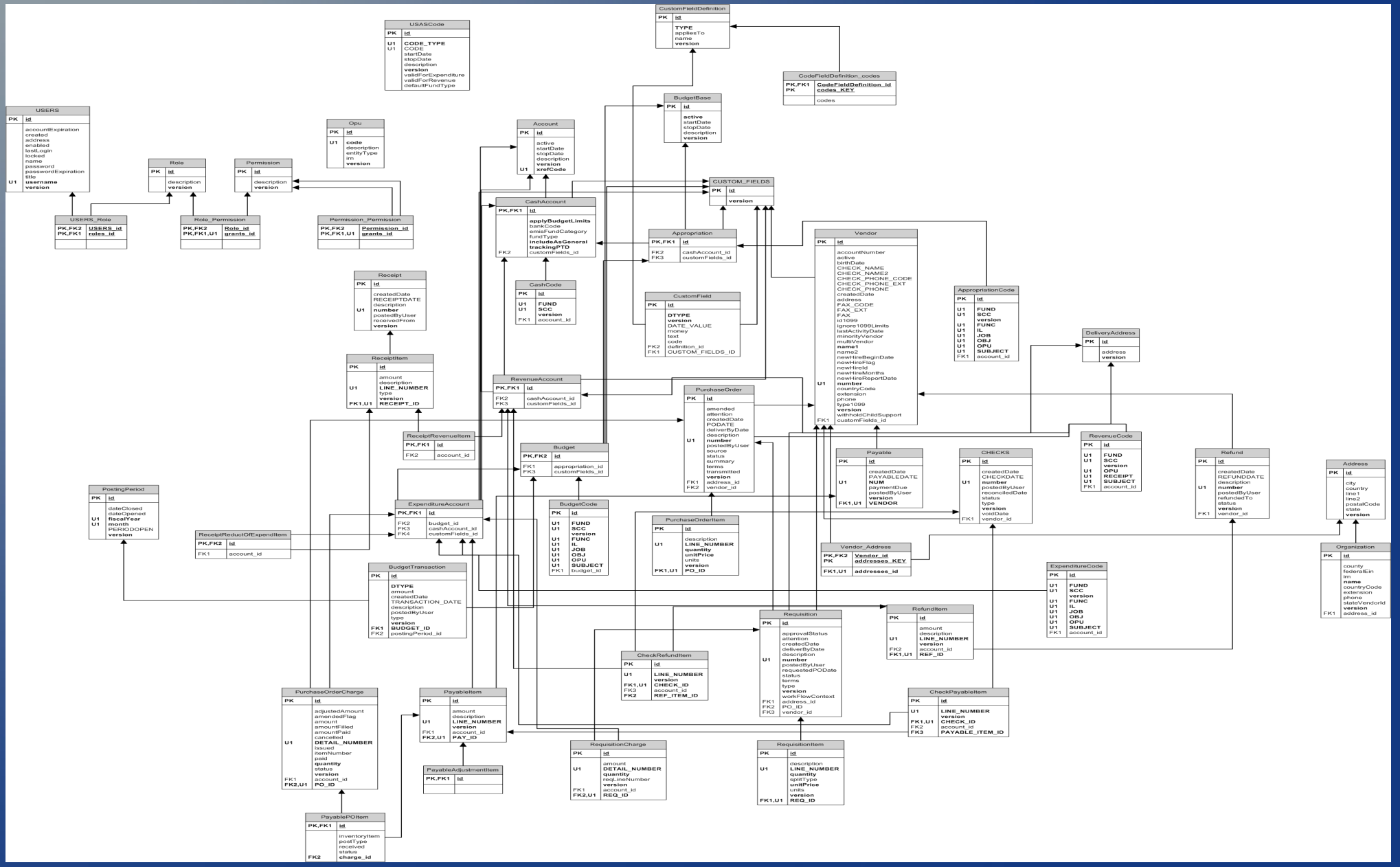
Don't Believe?

- Below is a “simple” query returning Vendors with Addresses and Custom Fields:

```
SELECT  *
from USAS.VENDOR V
   JOIN USAS.VENDOR_ADDRESS VA ON V.ID = VA.VENDOR_ID
   JOIN USAS.ADDRESS A ON A.ID = VA.ADDRESSES_ID
   JOIN USAS.VENDOR_CUSTOMFIELD VCF ON VCF.VENDOR_ID = V.ID
   JOIN USAS.CUSTOM_FIELDS CFS ON CFS.ID = VCF.CUSTOMFIELDS_ID
   JOIN USAS.CUSTOMFIELD CF ON CF.CUSTOM_FIELDS_ID = CFS.ID
   JOIN USAS.CUSTOMFIELDDEFINITION CFD ON CFD.ID = CF.DEFINITION_ID
```

- And is still largely useless:
  - Cartesian product between Address and Custom fields
  - “Correct Solution” would be involve sub-queries...

Still Don't Believe?



Reporting Services

- Exposes Data Model
  - “Flattens” model for reporting needs
  - Provides Calculated and reference fields:
    - “total” of Purchase Order
    - Expenditure Account code on PO Item
  - Query methods:
    - Form based
    - Simplified “Advanced” Query Language
- Export formats:
  - PDF, Excel, CVS, XML, JSON, etc
  - REST (URL) style request for application integration

The Big Question: When?

Sorry, we're out of time.

Enjoy the rest of the Meeting.

The problem of “Done”

- The Goal: “Reproduce Existing Functionality”:
  - Is imprecise
  - In user's head, not ours
  - Required Functionality varies by District
  - Parts of current USxS are no longer needed
- ITC Customizations:
  - New system must either replace, or support current ITC “add ons” (fiscweb, onBase, Kiosk, etc)
  - ITC will need time to re-integrate
  - SSDT is not aware of all integrations

The Plan

- “Milestone Releases” (USAS) to start in Fall (yes, 2011)
  - Installable at ITC
  - Import actual District Data
  - Experiment with and provide Feedback
  - Develop list of “*What's needed to use in Production*?”
  - Issue frequent (monthly) Milestones to shorten list
- “Release Candidates”:
  - When list gets sufficiently short
  - Will be proposed production releases
- “Production Releases”:
  - Districts start converting
  - Determine drop-dead date for OpenVMS support

Questions?

Presentation Available at:
Google: Dave Smith SSDT
Click {I'm Feeling Lucky}