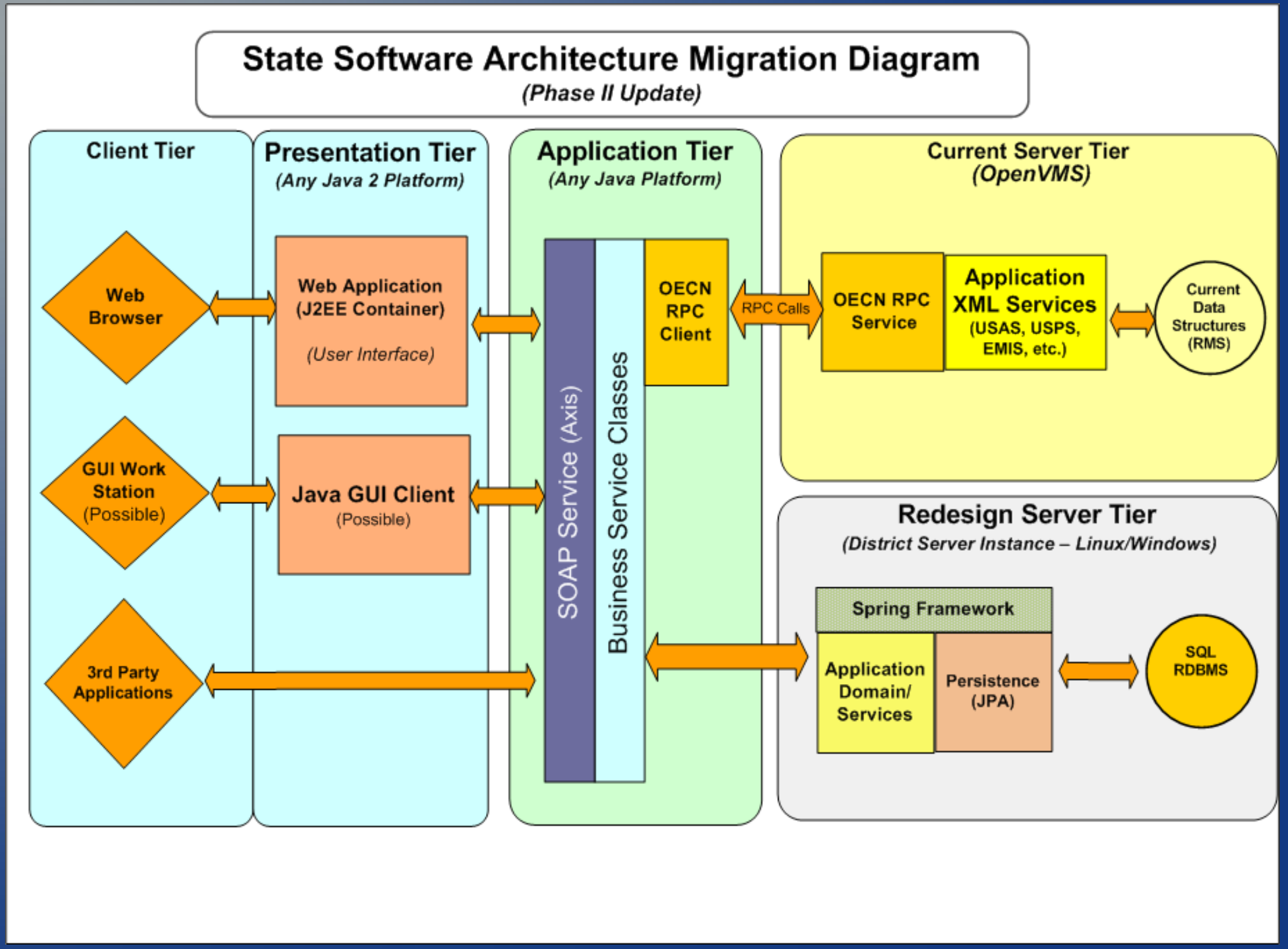
USAS-R

Status, Current Design & Implications

Fall OEDSA 2010

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
3rd Party Applications
Redesign Server Tier
(District Server Instance – Linux/Windows)
Spring Framework
Application Domain/ Services
Persistence (JPA)
SQL RDBMS

Phase II Started

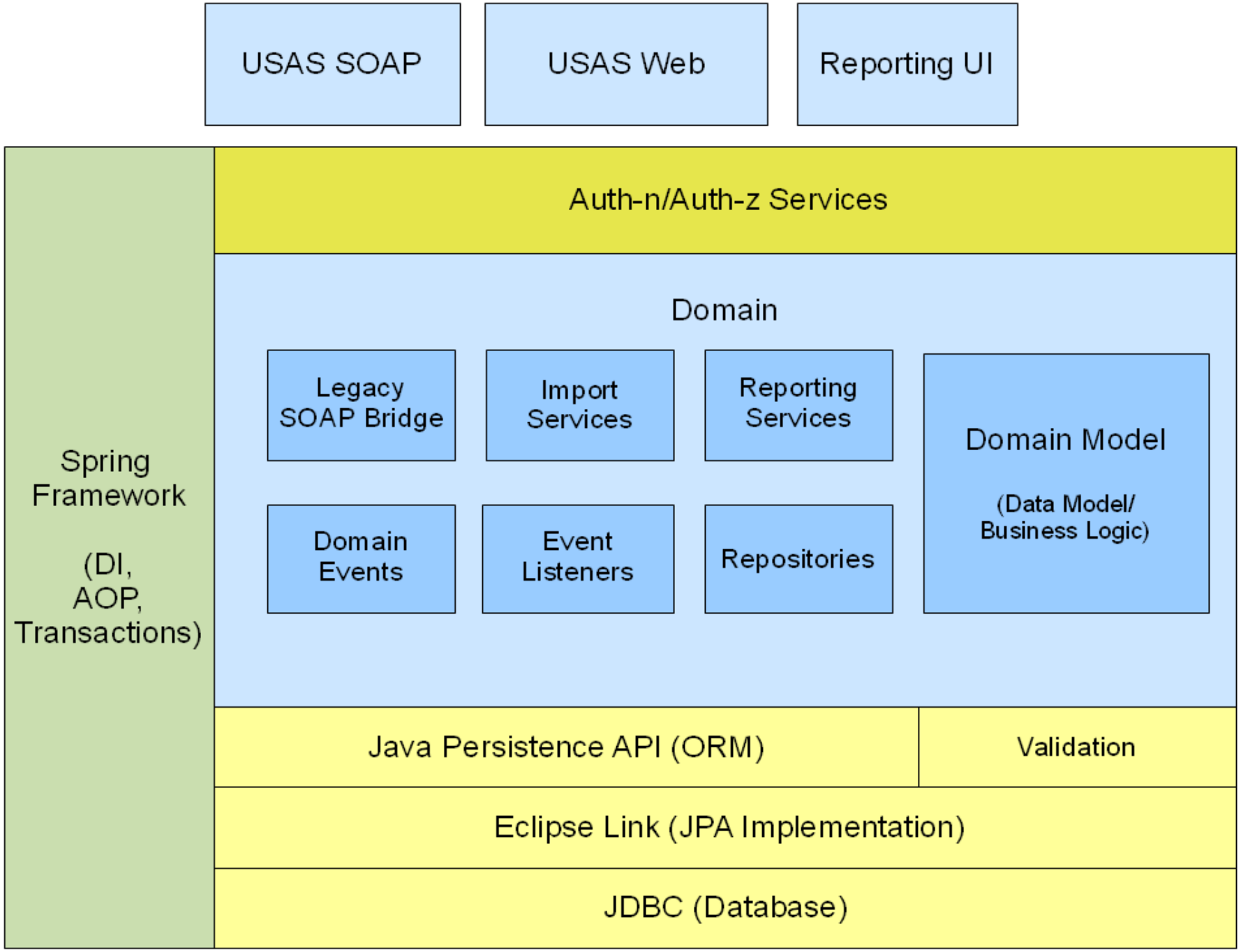
- SAC recommended starting of Phase II
  - Concentrate on Redesign
  - Classic systems in maintenance mode
- USAS-R
  - Project launched Nov-2009
  - Completely new model and architecture
- USPS-R
  - Project launched Aug-2010
  - Based on knowledge gained by USAS-R

Redesign Goals

- Primary:
  - Reproduce Existing Functionality
  - Redesign Data Model
    - Will not “port” existing data model
    - Simplify Application
    - Allow for Future Growth
  - Increase Flexibility
  - Partial Compatibility with Classic versions
- Secondary
  - “Incidental Enhancements”
    - Leverage new tools and frameworks
    - Improve usability, flexibility and integration

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


USAS SOAP
USAS Web
Reporting UI
Spring Framework
(DI, AOP, Transactions)
Auth-n/Auth-z Services
Domain
Legacy SOAP Bridge
Import Services
Reporting Services
Domain Model
(Data Model/ Business Logic)
Domain Events
Event Listeners
Repositories
Java Persistence API (ORM)
Validation
Eclipse Link (JPA Implementation)
JDBC (Database)

USAS-R Status

- Domain Model
  - Most Major Object types designed
  - Import process
  - Authorization/Authentication Modules
  - SOAP Bridge (Legacy Compatibility)
  - USAS Web App connected to USAS-R
- Prototype Reporting Service

Import/Conversion

- Goal
  - One-step 100% import from Classic USAS
  - All relevant data imported accurately
- Process:
  - Full SSWAT Extract on OpenVMS
  - USAS-R Import Utility:
    - FTP from VMS system (or local file)
    - Builds database
    - Imports all data

```
ftp: FTPAgent junit@nwoca.org[.sswat_ftp]
   Downloading: APPACCT_899999.txt
   Downloading: APPHIST_899999.txt
   Downloading: ARCUST_899999.txt
   Downloading: ARHEAD_899999.txt
   Downloading: ARITEM_899999.txt
   Downloading: BUDACCT_899999.txt
   Downloading: BUDHIST_899999.txt
   Downloading: CASHACCT_899999.txt
   Downloading: CASHHIST_899999.txt
   Downloading: CKHEAD_899999.txt
   Downloading: CKITEM_899999.txt
   Downloading: INVOICE_899999.txt
   Downloading: POAMNT_899999.txt
   Downloading: PODESC_899999.txt
   Downloading: POHIST_899999.txt
   Downloading: RECEIPT_899999.txt
   Downloading: REQHEAD_899999.txt
   Downloading: REQITEM_899999.txt
   Downloading: REVACCT_899999.txt
   Downloading: REVHIST_899999.txt
   Downloading: USASETUP_899999.txt
   Downloading: VENDOR_899999.txt
Initializing USAS Repositories
Defined standard custom field: Vendor Money 1
Defined standard custom field: Vendor Money 2
Defined standard custom field: Vendor Code 1
Defined standard custom field: Vendor Code 2
Defined standard custom field: Vendor Date
   .
   .
   .
```

```
Import routines to execute:
    org.ssdt_ohio.usas.imports.impl.OpuImportImpl
    org.ssdt_ohio.usas.imports.impl.VendorImportImpl
    org.ssdt_ohio.usas.imports.impl.CheckImportImpl
    org.ssdt_ohio.usas.imports.impl.CashAccountImportImpl
    org.ssdt_ohio.usas.imports.impl.ExpAccountImportImpl
    org.ssdt_ohio.usas.imports.impl.PayableImportImpl
    org.ssdt_ohio.usas.imports.impl.PurchaseOrderImportImpl
    org.ssdt_ohio.usas.imports.impl.ReceiptImportImpl
    org.ssdt_ohio.usas.imports.impl.RefundImportImpl
    org.ssdt_ohio.usas.imports.impl.RequisitionImportImpl
    org.ssdt_ohio.usas.imports.impl.RevenueAccountImportImpl
Import Summary:
  Import                                Records Loaded
  OpuImportImpl:opus                                0
  VendorImportImpl:vendors                         24
  CheckImportImpl:headers                           6
  CheckImportImpl:items                             7
  CashAccountImportImpl:cashAccts                  17
  ExpAccountImportImpl:expAccts                    98
  PayableImportImpl:invoice                        38
  PurchaseOrderImportImpl:headers                  30
  PurchaseOrderImportImpl:items                    71
  PurchaseOrderImportImpl:desc                    104
  ReceiptImportImpl:receipts                       22
  RefundImportImpl:headers                         22
  RequisitionImportImpl:headers                    15
  RequisitionImportImpl:items                      35
  RevenueAccountImportImpl:revAccts                 3
Totals                                            492
Elapsed: 31.985 secs
```

Not Your Father's USAS

- Domain Model will be radically different from Classic USAS
  - 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
  - USAS duplicated on every transaction
- USAS-R:
  - Separate Records:
    - Expenditure Account
    - Budget Account
    - USAS Code
  - Model will allow:
    - Multiple Expenditures per Budget
    - But not initially, tied together for compatibility
  - Transactions will not store USAS Code

Data Model Differences (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

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 and REST-ful 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:
    - USAS_VENDOR grants:
      - USAS_VENDOR_VIEW
      - USAS_VENDOR_CREATE
      - USAS_VENDOR_DELETE
      - USAS_VENDOR_UPDATE
      - USAS_VENDOR_REPORT
- Roles
  - Roles grant permissions
  - User are granted role(s)
  - Software or District Defined

Permissions/Roles

- Initial (Legacy) Roles:
  - USAS - Simulates “Standard” identifier
  - USAS_RO - Simulates “Read-Only” role
  - Others to simulate other Classic USAS 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
  - Description
  - Validation (e.g. Code list of values)
- Defined by:
  - District
  - SSDT
  - Third-party vendors

Custom Fields

- Predefined for current User Defined:
  - VENDOR_MONEY1
  - VENDOR_CODE1, etc
- Some Classic USAS Fields moved to CF:
  - Vendor Category
    - Will allow code values to be defined
  - Requisition Type
    - “Template” will be separate field
    - Types will be District Defined Custom Field
  - 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)
- Application will “publish” events:
  - Repository Events:
    - Create, Update, Delete
    - Query, Retrieve
  - Business Logic events:
    - Budget 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 (Vendor, PO, ...)

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 might:
  - Perform Audit Logging
  - Perform validation (budget check)
  - Send notification message
  - Send message to 3rd Party Application
- Events will be:
  - SSDT Defined
  - District Defined
  - 3rd Party Developer

Custom Event Listeners

- District Defined
- Customize USAS behavior:
  - Example #1 (Notification):
    - When a Purchase Order is posted
    - Where “total” amount is > $10,000
    - Send Email to Treasurer
  - Example #2 (Custom Validation):
    - Vendor is created or updated
    - Email address is blank
    - 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”
- USAS-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

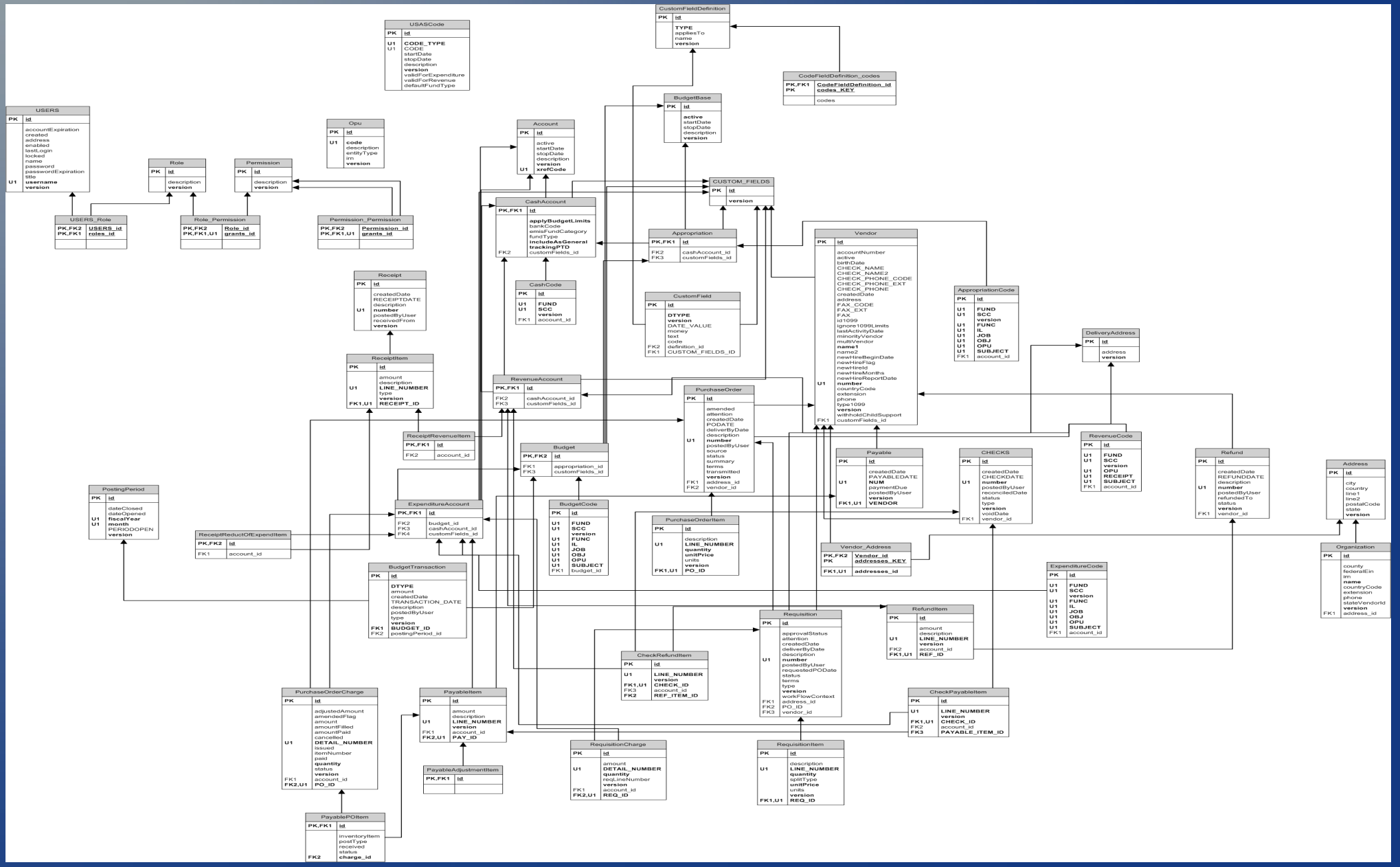
Don't Believe?

- 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