USAS-R

Status, Current Design & Implications
– Technical –

Fall OEDSA 2010

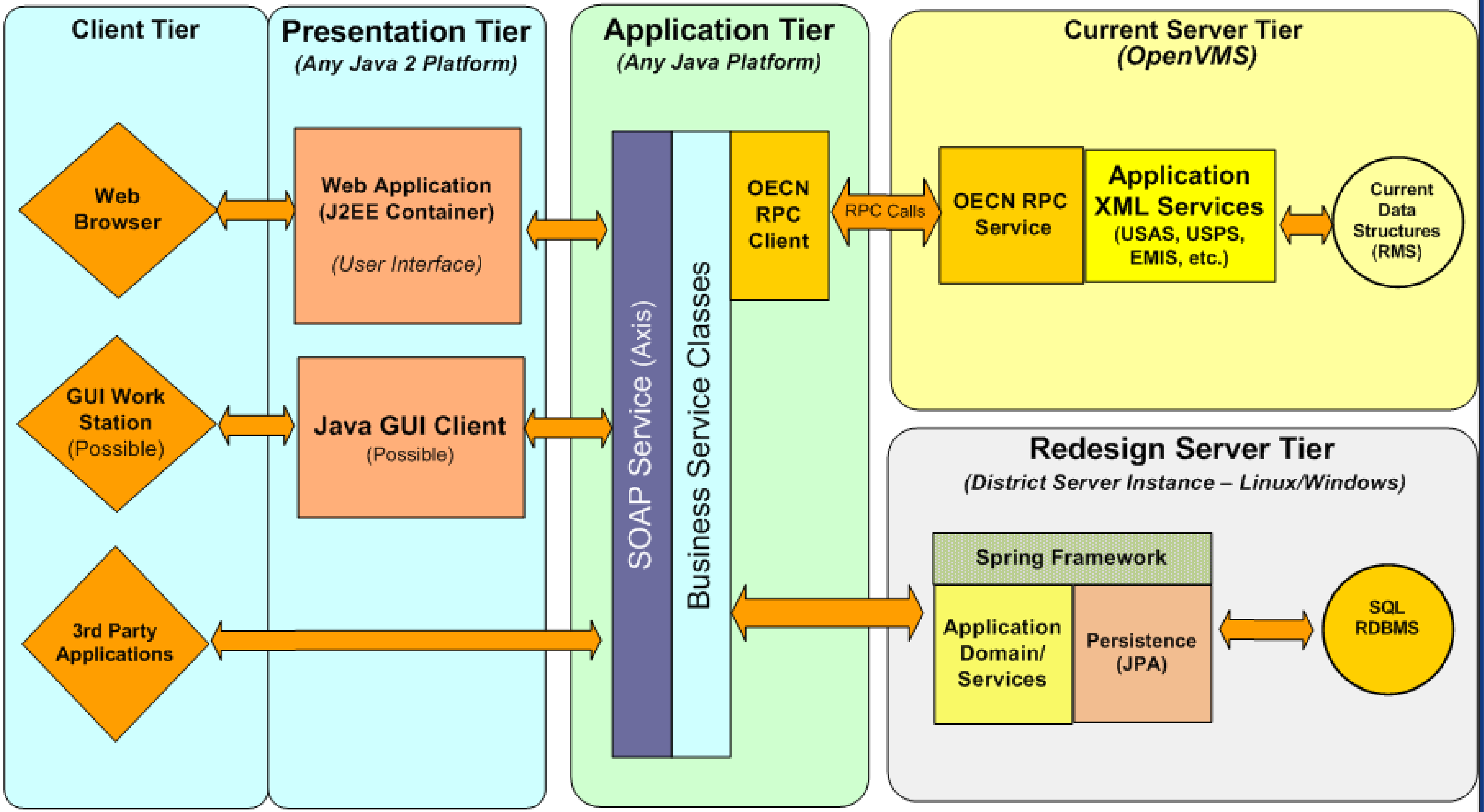
Dave Smith; SSDT

Disclaimer

- Still early in Development Cycle
- Not fully committed to some choices:
  - Database Platforms, likely:
    - Oracle
    - MS SQL Server
    - Open Source: MySQL or PostgreSQL
  - OS (Linux, Windows?)
  - Distribution Model:
    - Installation kit?
    - Virtual Applicance?

State Software Architecture Migration Diagram

(Phase II Update)



Client Tier

Web Browser

GUI Work Station (Possible)

3rd Party Applications

Presentation Tier

(Any Java 2 Platform)

Web Application (J2EE Container)

(User Interface)

Java GUI Client (Possible)

Application Tier

(Any Java Platform)

SOAP Service (Axis)

Business Service Classes

OECN RPC Client

RPC Calls

Current Server Tier

(OpenVMS)

OECN RPC Service

Application XML Services (USAS, USPS, EMIS, etc.)

Current Data Structures (RMS)

Redesign Server Tier

(District Server Instance – Linux/Windows)

Spring Framework

Application Domain/ Services

Persistence (JPA)

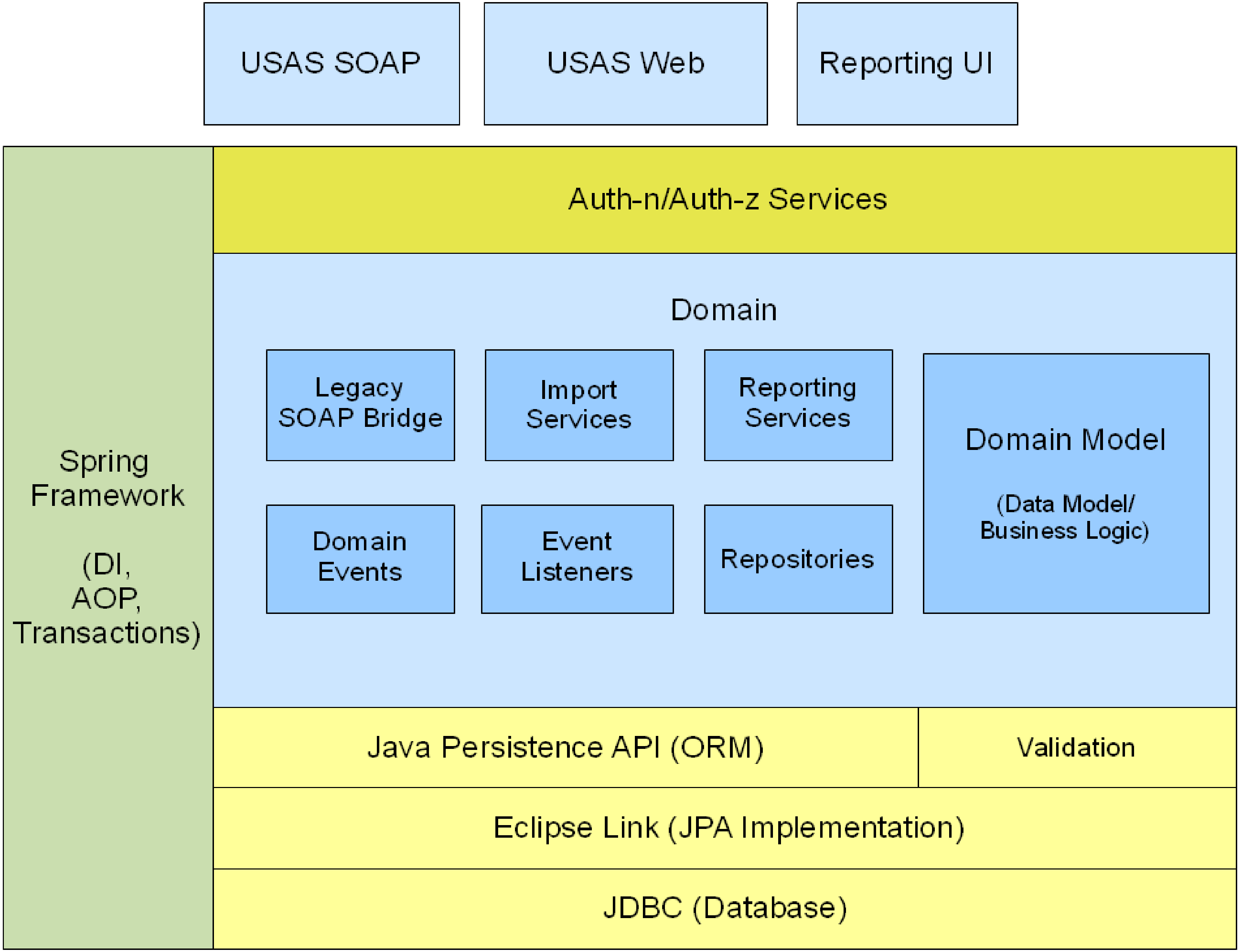
SQL RDBMS

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

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



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)

Architecture

- Modular based on Lightweight Container
- One District:
  - One Database (local or remote)
  - One Software Install
  - One Server (Virtual Machine)
- Each installation
  - Customizable per District
  - Modules loaded as needed
- Intended to be “Cloud Ready”

Likely Distribution Model

- SSDT will distribute “Virtual Appliance”
  - Linux-based (maybe Windows variation)
  - Pre-installed with OS and Container (OSGi?)
  - Appliance:
    - Prompt for Configuration
    - Download modules from SSDT
    - Install/create database (local or remote)
  - Check for updates, one-click install

Administrative Overhead

- Server per District creates Admin Overhead
- Intend to provide:
    - Administrative Console
        - List of servers
        - Status
        - Access to Application Console
        - Monitoring Events
    - Remote software updates

Auth-n/Auth-z

- Auth-n, multiple sources:
  - Local Authentication (database)
  - External Authentication (LDAP, ADS, OpenID?)
  - Plugin Auth-n modules via Spring
- Auth-z:
  - Authorized users mapped to USAS user profile
  - Roles in database (not in External source)
  - May provide Role mapping from external source

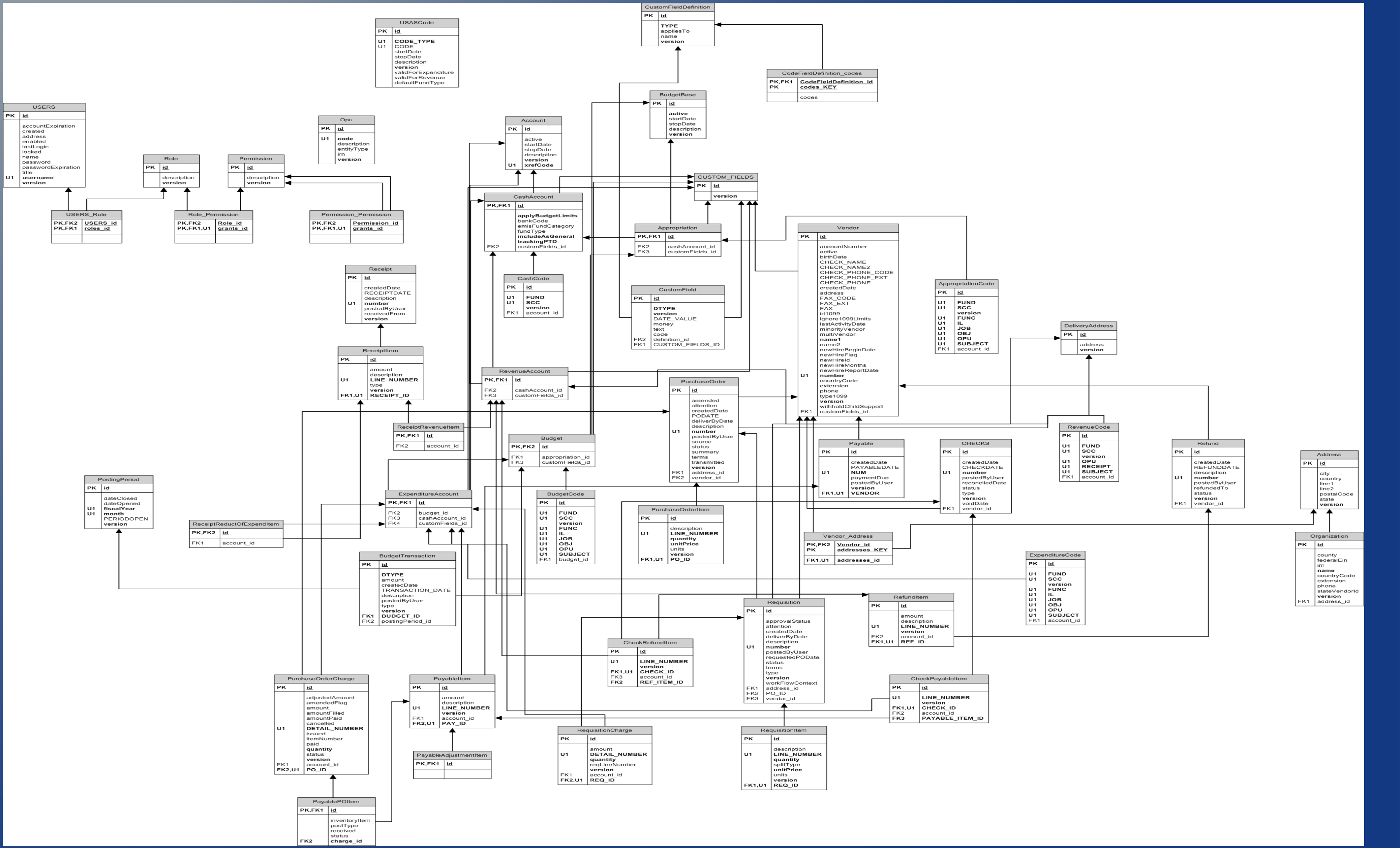
Development Process

- SSDT using “Agile-ish” process:
  - No “Big Design Up Front”
  - Two Week Iterations (Short plan cycles)
  - Design evolves iteratively
  - Decisions deferred until “Last Responsible Moment”
  - Continuous Integration/Automated Testing
- Object Oriented Design
  - Relational database is side-effect of Model
  - Persistence Layer Abstracts Away Database

Higher Levels of Abstraction

- SSDT writes Object Model and Business Logic:
  - Java and Groovy
  - Passes objects to Persistence layer (no SQL)
  - Persistence Layer writes database meta-data and SQL
- Database is a artifact of compiling Object model
- Programmers are
  - Aware of database
  - But don't (much) care about it
  - It's just a place where Objects go to until needed again
  - All business logic is in Domain

Database ER Diagram



USASCode	
PK	id
U1	CODE_TYPE
U1	CODE
	startDate
	stopDate
	description
	version
	validForExpenditure
	validForRevenue
	defaultFundType

CustomFieldDefinition	
PK	id
	TYPE
	appliesTo
	name
	version

CodeFieldDefinition_codes	
PK,FK1	CodeFieldDefinition_id
PK	codes_KEY
	codes

USERS	
PK	id
	accountExpiration
	created
	address
	enabled
	lastLogin
	locked
	name
	password
	passwordExpiration
	title
U1	username
	version

Opu	
PK	id
U1	code
	description
	entityType
	irn
	version

Account	
PK	id
	active
	startDate
	stopDate
	description
	version
U1	xrefCode

BudgetBase	
PK	id
	active
	startDate
	stopDate
	description
	version

Role	
PK	id
	description
	version

Permission	
PK	id
	description
	version

CUSTOM_FIELDS	
PK	id
	version

USERS_Role	
PK,FK2	USERS_id
PK,FK1	roles_id

Role_Permission	
PK,FK2	Role_id
PK,FK1,U1	grants_id

Permission_Permission	
PK,FK2	Permission_id
PK,FK1,U1	grants_id

CashAccount	
PK,FK1	id
	applyBudgetLimits
	bankCode
	emisFundCategory
	fundType
	includeAsGeneral
	trackingPTD
FK2	customFields_id

Appropriation	
PK,FK1	id
FK2	cashAccount_id
FK3	customFields_id

Vendor	
PK	id
	accountNumber
	active
	birthDate
	CHECK_NAME
	CHECK_NAME2
	CHECK_PHONE_CODE
	CHECK_PHONE_EXT
	CHECK_PHONE
	createdDate
	address
	FAX_CODE
	FAX_EXT
	FAX
	id1099
	ignore1099Limits
	lastActivityDate
	minorityVendor
	multiVendor
	name1
	name2
	newHireBeginDate
	newHireFlag
	newHireId
	newHireMonths
	newHireReportDate
U1	number
	countryCode
	extension
	phone
	type1099
	version
	withholdChildSupport
FK1	customFields_id

Receipt	
PK	id
	createdDate
	RECEIPTDATE
	description
U1	number
	postedByUser
	receivedFrom
	version

CashCode	
PK	id
U1	FUND
U1	SCC
	version
FK1	account_id

CustomField	
PK	id
	DTYPE
	version
	DATE_VALUE
	money
	text
	code
FK2	definition_id
FK1	CUSTOM_FIELDS_ID

AppropriationCode	
PK	id
U1	FUND
U1	SCC
	version
U1	FUNC
U1	IL
U1	JOB
U1	OBJ
U1	OPU
U1	SUBJECT
FK1	account_id

DeliveryAddress	
PK	id
	address
	version

ReceiptItem	
PK	id
	amount
	description
U1	LINE_NUMBER
	type
	version
FK1,U1	RECEIPT_ID

RevenueAccount	
PK,FK1	id
FK2	cashAccount_id
FK3	customFields_id

PurchaseOrder	
PK	id
	amended
	attention
	createdDate
	PODATE
	deliverByDate
	description
U1	number
	postedByUser
	source
	status
	summary
	terms
	transmitted
	version
FK1	address_id
FK2	vendor_id

ReceiptRevenueItem	
PK,FK1	id
FK2	account_id

RevenueCode	
PK	id
U1	FUND
U1	SCC
	version
U1	OPU
U1	RECEIPT
U1	SUBJECT
FK1	account_id

Budget	
PK,FK2	id
FK1	appropriation_id
FK3	customFields_id

Payable	
PK	id
	createdDate
	PAYABLEDATE
U1	NUM
	paymentDue
	postedByUser
	version
FK1,U1	VENDOR

CHECKS	
PK	id
	createdDate
	CHECKDATE
U1	number
	postedByUser
	reconciledDate
	status
	type
	version
	voidDate
FK1	vendor_id

Refund	
PK	id
	createdDate
	REFUNDDATE
	description
U1	number
	postedByUser
	refundedTo
	status
	version
FK1	vendor_id

Address	
PK	id
	city
	country
	line1
	line2
	postalCode
	state
	version

PostingPeriod	
PK	id
	dateClosed
	dateOpened
U1	fiscalYear
U1	month
	PERIODOPEN
	version

ExpenditureAccount	
PK,FK1	id
FK2	budget_id
FK3	cashAccount_id
FK4	customFields_id

BudgetCode	
PK	id
U1	FUND
U1	SCC
	version
U1	FUNC
U1	IL
U1	JOB
U1	OBJ
U1	OPU
U1	SUBJECT
FK1	budget_id

PurchaseOrderItem	
PK	id
	description
U1	LINE_NUMBER
	quantity
	unitPrice
	units
	version
FK1,U1	PO_ID

ReceiptReductOfExpendItem	
PK,FK2	id
FK1	account_id

Vendor_Address	
PK,FK2	Vendor_id
PK	addresses_KEY
FK1,U1	addresses_id

Organization	
PK	id
	county
	federalEin
	irn
	name
	countryCode
	extension
	phone
	stateVendorId
	version
FK1	address_id

BudgetTransaction	
PK	id
	DTYPE
	amount
	createdDate
	TRANSACTION_DATE
	description
	postedByUser
	type
	version
FK1	BUDGET_ID
FK2	postingPeriod_id

ExpenditureCode	
PK	id
U1	FUND
U1	SCC
	version
U1	FUNC
U1	IL
U1	JOB
U1	OBJ
U1	OPU
U1	SUBJECT
FK1	account_id

Requisition	
PK	id
	approvalStatus
	attention
	createdDate
	deliverByDate
	description
U1	number
	postedByUser
	requestedPODate
	status
	terms
	type
	version
	workFlowContext
FK1	address_id
FK2	PO_ID
FK3	vendor_id

RefundItem	
PK	id
	amount
	description
U1	LINE_NUMBER
	version
FK2	account_id
FK1,U1	REF_ID

CheckRefundItem	
PK	id
U1	LINE_NUMBER
	version
FK1,U1	CHECK_ID
FK3	account_id
FK2	REF_ITEM_ID

PurchaseOrderCharge	
PK	id
	adjustedAmount
	amendedFlag
	amount
	amountFilled
	amountPaid
	cancelled
U1	DETAIL_NUMBER
	issued
	itemNumber
	paid
	quantity
	status
	version
FK1	account_id
FK2,U1	PO_ID

PayableItem	
PK	id
	amount
	description
U1	LINE_NUMBER
	version
FK1	account_id
FK2,U1	PAY_ID

CheckPayableItem	
PK	id
U1	LINE_NUMBER
	version
FK1,U1	CHECK_ID
FK2	account_id
FK3	PAYABLE_ITEM_ID

RequisitionCharge	
PK	id
	amount
U1	DETAIL_NUMBER
	quantity
	reqLineNumber
	version
FK1	account_id
FK2,U1	REQ_ID

RequisitionItem	
PK	id
	description
U1	LINE_NUMBER
	quantity
	splitType
	unitPrice
	units
	version
FK1,U1	REQ_ID

PayableAdjustmentItem	
PK,FK1	id

PayablePOItem	
PK,FK1	id
	inventoryItem
	postType
	received
	status
FK2	charge_id

Frameworks/Languages

- Java VM based
  - Java
  - Groovy (Dynamic/Meta Language)
- JPA (Persistence)
- Aspects (Aspect Oriented Programming)
  - Cross cutting concerns:
    - Transactions
    - Security
    - Logging
- Spring Framework:
  - Lightweight Application Container
  - IoC/DI (Inversion of Control/Dependency Injection)
  - Auth-n/Auth-z

Groovy? Seriously?

- Dynamic/Meta-Programming Language
- Java's answer to Ruby

Java:

```
    List<Things> things = getListOfThings()
    for (int i = 0; i < things.size(); i++ ) {
         System.out.println(things[i]);
    }
```

Groovy:

```
    def things = getListOfThings()
    things.each {
      println it
    }
```

Aspect Oriented Programming

- Cross-cutting Concerns
  - Security
  - Transaction Handling
  - Exception Handing
- Avoid Boilerplate code
- Code is modified post-compile-time

Aspect Example (Without AOP)

```
Vendor update(vendor) {

   if (!user.isInRole('VENDOR_UPDATE') {
     Throw new SecurityException(...)
   }

   Transaction tx
   Try {
        tx = transManager.start()

        em.merge(vendor)

        tx.commit()
   } catch (Exception ex) {
        tx.rollback()
   } finally {
       tx.release()
   }
}
```

Aspect Example

- With Annotation-Based Aspects

```
@Transactional
@Secured('VENDOR_UPDATE')
Vendor update(vendor) {
    em.merge(vendor)
}
```

Example

- Query to get Vendor by ID (returns RowSet)

```
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
   WHERE V.ID = '06751225-8565-4a67-8e09-731882bebfc4'
```

- Equivalent using JPA:

```
Vendor vendor = em.get('06751225-8565-4a67-8e09-731882bebfc4',Vendor.class)
```

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

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 and every transaction
- USAS-R:
  - Separate Records:
    - Expenditure Account
    - Budget Account
    - USAS Code
  - Model will allow:
    - Multiple Expenditures per Budget
    - But not initially
  - 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 Types: URL, Attachment, Calculated, etc
  - Description
  - Help
  - 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

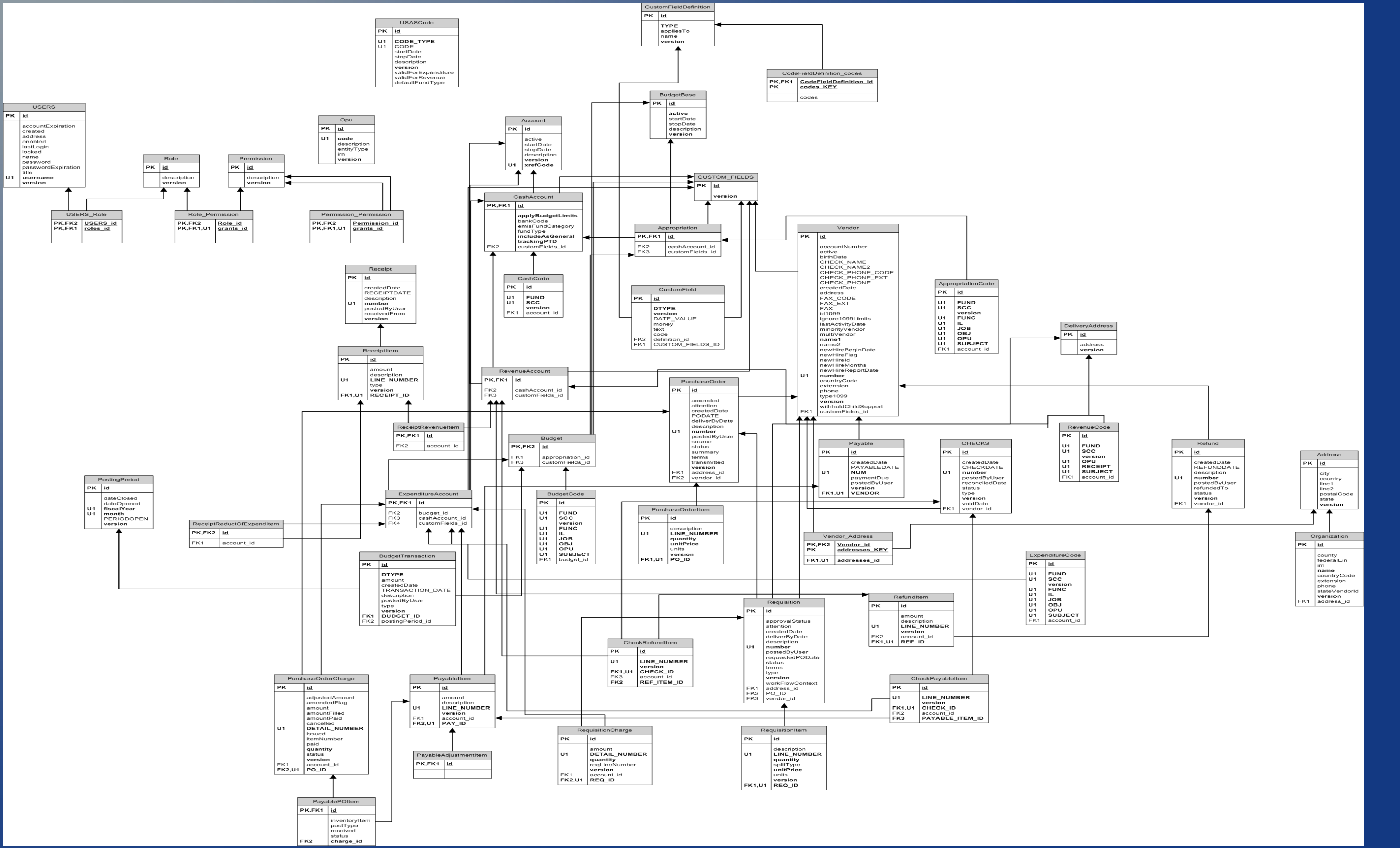
Don't Believe?

- 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?



CustomFieldDefinition

PK	id
	TYPE appliesTo name **version**

USASCode

PK	id
U1 U1	**CODE_TYPE** CODE startDate stopDate description **version** validForExpenditure validForRevenue defaultFundType

CodeFieldDefinition_codes

PK,FK1 PK	**CodeFieldDefinition_id** **codes_KEY**
	codes

BudgetBase

PK	id
	active startDate stopDate description **version**

USERS

PK	id
U1	accountExpiration created address enabled lastLogin locked name password passwordExpiration title **username** **version**

Opu

PK	id
U1	**code** description entityType irn **version**

Account

PK	id
U1	active startDate stopDate description **version** **xrefCode**

Role

PK	id
	description **version**

Permission

PK	id
	description **version**

CUSTOM_FIELDS

PK	id
	version

CashAccount

PK,FK1	id
FK2	**applyBudgetLimits** bankCode emisFundCategory fundType **includeAsGeneral** **trackingPTD** customFields_id

USERS_Role

PK,FK2 PK,FK1	**USERS_id** **roles_id**

Role_Permission

PK,FK2 PK,FK1,U1	**Role_id** **grants_id**

Permission_Permission

PK,FK2 PK,FK1,U1	**Permission_id** **grants_id**

Appropriation

PK,FK1	id
FK2 FK3	cashAccount_id customFields_id

Vendor

PK	id
U1 FK1	accountNumber active birthDate CHECK_NAME CHECK_NAME2 CHECK_PHONE_CODE CHECK_PHONE_EXT CHECK_PHONE createdDate address FAX_CODE FAX_EXT FAX id1099 ignore1099Limits lastActivityDate minorityVendor multiVendor **name1** name2 newHireBeginDate newHireFlag newHireId newHireMonths newHireReportDate **number** countryCode extension phone type1099 **version** withholdChildSupport customFields_id

Receipt

PK	id
U1	createdDate RECEIPTDATE description **number** postedByUser receivedFrom **version**

CashCode

PK	id
U1 U1 FK1	**FUND** **SCC** **version** account_id

CustomField

PK	id
FK2 FK1	**DTYPE** **version** DATE_VALUE money text code definition_id CUSTOM_FIELDS_ID

AppropriationCode

PK	id
U1 U1 U1 U1 U1 U1 U1 U1 FK1	**FUND** **SCC** **version** **FUNC** **IL** **JOB** **OBJ** **OPU** **SUBJECT** account_id

DeliveryAddress

PK	id
	address **version**

ReceiptItem

PK	id
U1 FK1,U1	amount description **LINE_NUMBER** type **version** **RECEIPT_ID**

RevenueAccount

PK,FK1	id
FK2 FK3	cashAccount_id customFields_id

PurchaseOrder

PK	id
U1 FK1 FK2	amended attention createdDate PODATE deliverByDate description **number** postedByUser source status summary terms transmitted **version** address_id vendor_id

ReceiptRevenueItem

PK,FK1	id
FK2	account_id

RevenueCode

PK	id
U1 U1 U1 U1 U1 FK1	**FUND** **SCC** **version** **OPU** **RECEIPT** **SUBJECT** account_id

Budget

PK,FK2	id
FK1 FK3	appropriation_id customFields_id

Payable

PK	id
U1 FK1,U1	createdDate PAYABLEDATE **NUM** paymentDue postedByUser **version** **VENDOR**

CHECKS

PK	id
U1 FK1	createdDate CHECKDATE **number** postedByUser reconciledDate status type **version** voidDate vendor_id

Refund

PK	id
U1 FK1	createdDate REFUNDDATE description **number** postedByUser refundedTo status **version** vendor_id

Address

PK	id
	city country line1 line2 postalCode state **version**

PostingPeriod

PK	id
U1 U1	dateClosed dateOpened **fiscalYear** **month** PERIODOPEN **version**

ExpenditureAccount

PK,FK1	id
FK2 FK3 FK4	budget_id cashAccount_id customFields_id

BudgetCode

PK	id
U1 U1 U1 U1 U1 U1 U1 U1 FK1	**FUND** **SCC** **version** **FUNC** **IL** **JOB** **OBJ** **OPU** **SUBJECT** budget_id

PurchaseOrderItem

PK	id
U1 FK1,U1	description **LINE_NUMBER** **quantity** **unitPrice** units **version** **PO_ID**

ReceiptReductOfExpendItem

PK,FK2	id
FK1	account_id

Vendor_Address

PK,FK2 PK	**Vendor_id** **addresses_KEY**
FK1,U1	**addresses_id**

Organization

PK	id
FK1	county federalEin irn **name** countryCode extension phone stateVendorId **version** address_id

BudgetTransaction

PK	id
FK1 FK2	**DTYPE** amount createdDate TRANSACTION_DATE description postedByUser type **version** **BUDGET_ID** postingPeriod_id

ExpenditureCode

PK	id
U1 U1 U1 U1 U1 U1 U1 U1 FK1	**FUND** **SCC** **version** **FUNC** **IL** **JOB** **OBJ** **OPU** **SUBJECT** account_id

RefundItem

PK	id
U1 FK2 FK1,U1	amount description **LINE_NUMBER** **version** account_id **REF_ID**

Requisition

PK	id
U1 FK1 FK2 FK3	approvalStatus attention createdDate deliverByDate description **number** postedByUser requestedPODate status terms type **version** workFlowContext address_id PO_ID vendor_id

CheckRefundItem

PK	id
U1 FK1,U1 FK3 FK2	**LINE_NUMBER** **version** **CHECK_ID** account_id **REF_ITEM_ID**

PurchaseOrderCharge

PK	id
U1 FK1 FK2,U1	adjustedAmount amendedFlag amount amountFilled amountPaid cancelled **DETAIL_NUMBER** issued itemNumber paid **quantity** status **version** account_id **PO_ID**

PayableItem

PK	id
U1 FK1 FK2,U1	amount description **LINE_NUMBER** **version** account_id **PAY_ID**

CheckPayableItem

PK	id
U1 FK1,U1 FK2 FK3	**LINE_NUMBER** **version** **CHECK_ID** account_id **PAYABLE_ITEM_ID**

RequisitionCharge

PK	id
U1 FK1 FK2,U1	amount **DETAIL_NUMBER** **quantity** reqLineNumber **version** account_id **REQ_ID**

RequisitionItem

PK	id
U1 FK1,U1	description **LINE_NUMBER** **quantity** splitType **unitPrice** units **version** **REQ_ID**

PayableAdjustmentItem

PK,FK1	id

PayablePOItem

PK,FK1	id
FK2	inventoryItem postType received status **charge_id**

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