Rendering/Visualizing Data from Multiple Entities

Herm Fischer
UBmatrix, Inc.
6 October 2009
Issues discussion
Technologies now available
  ◦ plain XBRL, XBRL processors, XSLT, iXBRL
Examples of multi-entity rendering
  ◦ XSLT processor-assisted rendering
  ◦ Web service multi-entity compositional rendering
What is coming up
  ◦ Multi-entity use of Formula, Versioning spec
  ◦ Multi-entity mapped relational databases
Aspects of topic by renderer
(Rendering/Visualizing Data from Multiple Entities)

- By producer of data at a single source
  - Constraints in preparing data
- By aggregator of, or authority collecting data
  - SEC – every submitter own extension, own linkbases
  - FDIC – every submitter exactly same stuff
  - EDI.net – submitter extensions but linkbases only tweaked, not replaced
- By consumer/integrator
  - Individual investor - fund/stock reports
Impact of multiple entities

- Instance document submissions
  - Multiple entities in one instance, one DTS
  - Separate instances, share same DTS
  - Separate instances, each own separate DTS

- Examples of multi-entity solutions
  - Simple rendering with ordinary XML tools
  - Web based multi-entity compositional rendering
What does rendering mean

- Show entity reports in different tabs/windows
- Align multiple entities to adjacent columns
- Align data of different periods
- Grab data from source prepared for merging
- Grab data on the fly from original sources
Producer issues (technological)

- XBRL
  - Model concepts
  - Presentation semantics
  - Definition/dimensional semantics (if not us-gaap)
  - Display formatting (inline-XBRL or XSLT)

- Web (+ Excel, Word docs)
  - Tabular rendered data may have clear semantics
  - Inferring semantics from formatting

- Relational database
  - Is data provided with schema & stored procedures, or just extractable post-rendering
XBRL vs Non-XBRL

- XBRL centric approach:
  - Map non-XBRL data into XBRL
  - Multi-entity rendering from XBRL model
  - XBRL has robust semantics and validation

- Database centric approach
  - Map XBRL into database table model
  - Render from table model
  - SQL has robust data joining mechanism

- Other
  - Excel centric views
Aggregator authority issues

- Submission integrity and contents
  - Name, tag, and label integrability
  - Semantic structure integrability
- Technology supportive of integration
  - XBRL
    - in XBRL native form
    - reduced into SQL or other neutral form
  - Non-XBRL data
    - promoted into XBRL
    - integrated in SQL
    - other
Consumer integrators

- Kind of consumer
  - Institutional or business entity with sophistication
  - Individuals with purpose and repetitive tasks (e.g. private investor)
  - Casual web surfer
Consumer integrator issues

- Multi-entity rendering integration
  - Tabular vs. entity-by-entity non-tabulated
  - Elision, reduction, and finding items
  - As-submitted (e.g., known-good XBRL) vs. as extracted (by software with humans)
  - Nomenclature reduction (unique names mapping for multi-entity tabulation)
Model concepts

- **Submitter constrained**
  - France: only one chart of accounts

- **Submitter freedom**
  - Prudential reporting in Europe
    - Pan-Europe with country and bank extensions
  - SEC
    - US-GAAP taxonomies are extended by submitter
    - Namespace and standard concepts different per year
  - Japan
    - EDIInet is extended by submitter
    - Entity concept model changes each period
Model concepts
multi-period merging

- Namespaces and names may need mapping
  - Versioning spec has mapping mechanism
  - XBRL Formula for semantic composition
  - Java etc for API based implementation
Presentational semantics

- Submitter extends core taxonomy linkbase
  - Europe prudential taxonomies (FINREP, COREP)
  - Japan EDIInet
- Submitter crafts own linkbase
  - SEC submittals
  - Extended tables of dimensions and line items
- Presentation coupling with dimensions
  - US- GAAP tightly coupled
  - Others (informally) decoupled
Model concepts multi-entity merging

- Core concepts dependable in France, Europe
- SEC submitter designs own presentation & dimension semantics
- EDInet submitter uses core presentation semantics
Rendering and visualizing issues

- Is it pre-rendered with iXBRL
- Is it on-line
- Is it dynamically generated
- Has merge mechanism been pre-prepared
- Is an XBRL processor available
iXBRL rendering

- HTML or XHTML embeds one XBRL instance
  - Producer’s source rendering
  - Instance extractable from HTML/ XHTML

- Multiple entities rendering could mean
  - All entities in one instance, share one DTS
    - Multiple entities joined and rendered by producer
  - Separate iXBRL per entity, share DTS
    - Up to aggregator or consumer to join and render
  - Separate iXBRL per entity, own separate DTS
    - Ditto
iXBRL renders as HTML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<head> ...
<link rel="stylesheet" type="text/css" href="iascf-style.css"/>
<title>IASCF 2007 ANNUAL REPORT</title>
</head> ...
<body class="body">
<table align="center" width="800px" border="0" cellpadding="0" cellspacing="0">
  <tr>
    <td>
      <p align="center" class="header">2007 ANNUAL REPORT</p>
      <p align="center">
        <a href="#statement1" class="nounderline">Statement of comprehensive income</a>
      </p>
    </td>
  </tr>
</table>
</body>
```
<table>
<thead>
<tr>
<th>iXBRL of a fact value</th>
</tr>
</thead>
</table>

| <td class="tableLightGrey" colspan="2" valign="top"> |
| <p title="iascf:Contributions id: id_footnote_elem_11688477 decimals: 0 ix:contextRef: FY07d ix:unitRef: GBP" style="text-align: right; color: black"> |
| <ix:nonFraction xmlns:ns0="http://www.xbrl.org/2008/inlineXBRL" id="id_footnote_elem_11688477" decimals="0" ix:contextRef="FY07d" ix:unitRef="GBP" ns0:format="comma" ns0:name="iascf:Contributions" ns0:scale="3">11,277</ix:nonFraction> |

</p> |
</td> |
iXBRL references, resources

<div style="display: none">
<iX:header>
<iX:references>
<link:schemaRef xlink:href="iascf_2008-02-28.xsd"/>
</ix:references>
<iX:resources>
<xbrli:context id="FY07d">
<xbrli:entity>
<xbrli:identifier
    scheme="http://www.iasb.org/AnnualReport/">IASCFI<br>
</xbrli:identifier>
</xbrli:entity>
<xbrli:period>
<xbrli:startDate>2007-01-01</xbrli:startDate>
<xbrli:endDate>2007-12-31</xbrli:endDate>
</xbrli:period>
</ix:resources>
</iX:header>
</div>
Formal definition of rendering in XBRL

- Depends on single DTS
- Static definition of formatting
- Dimensional rendering based on DTS
XSLT for XBRL

- XSLT is ubiquitous
- XBRL is XML
- XSLT utilizes XBRL processor functions
XSLT example, multi-entity XBRL

- Style sheet using XBRL processor functions
  - Vendor- provided functions (legacy)
  - Functions- registry provided functions (Formula WG)
- Dynamic rendering can be DTS independent
- Usually single- instance per rendering
  - Example here is multiple entity instance
Multi entity compilation by XSLT

- Dynamic composition based on
  - Entity model in dimensions axis
  - Share classes in dimension axis
  - Line items in presentation LB of table

- Example for XBRL- US RR instances
  - Each submission custom- extends taxonomy
  - Submission- provided presentation linkbase
    - Of dimensions (entities)
    - Of line items
Example of XSLT-based rendering

- Vertical axis represents line items tree
  - Dynamic rendering of instance DTS
- Horizontal represents dimensions axes
  - Hierarchy of multiple entities
  - Hierarchy of share classes
Single entity

DTS dimension axes members tree

DTS presentation of line items
Multiple entity (same stylesheet)

DTS dimensions, entity & share axes members trees

(sparse data rows/columns are elided)
XSLT model, view, rendering

- Two XSLT phases
  - Model and view extraction phase
    - Extract line items and dimension axes to xml
    - Extract instance data to xml
  - Rendering phase
    - Develop column headers
    - Develop row headers
    - Identify sparse rows/columns
    - Render populated rows/columns
- First phase uses XBRL processor functions
- Second phase uses XSLT keys and functions
Web service example

- Company tax report filings
  - Separate filing per period per company
  - Instance document converted into XBRL
  - Taxonomy matches company report structure
  - Single period data

- Viewers want
  - Side-by-side merged rendering of
    - Multiple periods
    - Multiple companies
Goal to convert public filings into XBRL
  ◦ Initially used Yuho (entity-based) taxonomy
  ◦ Transitioning to EDInet

Build multi-entity multi-period viewer
  ◦ Subscriber based web service
  ◦ Select up to 10 entities, multiple periods, merged view
Choosing items, periods & entities
User workflow

- A prior search engine screen narrowed down choice of reporting entities, choosing up to 10 to load to XBRL viewer.

After choice is narrowed down, loaded to viewer, then user selects entities in viewer.
Selecting periods and items

- A prior search screen allowed narrowing down choice of reporting entities, choosing up to 10 to load to XBRL viewer.

Selection of reporting period(s)

Selection of subtrees of merged concepts to view (Yuho taxonomy) and extended link role (EDInet).
Selecting 1 entity 2 yrs

entity items in merged-concepts tree grid
Japan has 3-columns per context
3 entities 2 yrs
Japan periods merged logically

Periods merged by FY date, e.g., 1st Qtr, 2nd Qtr, 3rd Qtr, semi-annual, annual

(does not consider context’s calendar date or reported-on date)
Each period a separate submission

- A period’s instance document is accompanied by the period’s taxonomy
  - Not the same from period to period
  - Different detailed line items
  - Different minority report line items

- Merging single entity multiple periods has same challenges as merging separate entities for same period
Items tree-merged by concept

(sparse leaf nodes elided)
Tree-merge is challenging

Concept trees can’t be merged on element name: inconsistent between periods and entity extensions

A unique label was added to be used for level-finding and tree-branch merging
Achieving server performance

- XBRL instance + taxonomy
  - Per period per entity
    - Entity taxonomy not common across periods
    - 50-150MB footprint
- A set of 10 entities * 3 yrs (for just 1 user!)
  - Nearly 1 GB footprint
  - ½ to 1 minute server time
- Caching strategy achieved performance
  - Few seconds to load and merge
  - About 20 MB footprint
Caching approach

- Pre-process each instance document
  - Home-made strings intern (for immediate GC)
  - Serialize hash-info for tree-merging
  - Serialize visualization object model
  - Focus on GC-able sessions

- Used binary serialization for speed
End users need data capture

- Web screen is nice to browse but
  - End users probably want the data, not the view
- One button captures CSV to browser
  - Renders in Excel or something equivalent
CSV to Local CSV (Excel)
Other technologies

- Formulas
  - Integrate multiple instances
- Versioning
  - Maps namespaces and local names
  - Update linkbase structures
- Relational databases
  - Efficient very- large fact bases
  - Join and query engines
XBRL Formula applicability

- Transform input instances(s) to output
- Proposed extensions for multiple input instances of separate DTSes
- Declarative means to specify of multi-entity merging
XBRL Versioning Specification

- Provides inter-DTS mappings of
  - Local Name
  - Namespace
  - Linkbase positional changes

- Could be used to:
  - Merge same entities from different periods
  - Merge different submissions to common line items
  - Support database processing (discussed later)
Versioning – now profile based

1) Base
   ◦ URI mapping: namespaces, ELR's
   ◦ actions, categories, assignments, documentation

2) Concept- basic
   ◦ name, namespace, add/delete/split/ merge

3) Concept- extended
   ◦ attributes, labels, references

4) Relationships
   ◦ add/delete... attributes

5) Dimensions

6) Resource- parts
   ◦ link:part, formulae, ...
Databases: multi-entity efficiency

- Experience with XBRL Gateway highlights
  - Issues of multi-instance processing in XBRL (DOM)
  - Need for fast tree-merge and concept mapping
- Versioning spec now provides
  - Profiles to support name/namespace mappings, label mappings, presentation differences
- Database stores instance data in neutral surrogate form
  - Efficient join logic to map to each submission DTS
Database engines

- Efficient parallel processing architecture
- Wide availability of XQuery interfaces
- Ability to search, join, and map

- Probably footprint of most databases about same as probably footprint of any XBRL processor with multiple instances active in XBRL- DOM form
Database key benefits

- As-filed document retention
  - instance and DTS
- Efficient access to fact base
  - Associative and small footprint processing
  - Versioning-based name and namespace mappings
- Metadata-based query
  - DTS fully supported
  - Efficient tree manipulation
  - Versioning supported
  - Formula with multi-instance support
Main points

- Multi-entity rendering means
  - Merging
    - Line item semantics
    - Dimensional semantics
    - Period versions of models
  - Rendering tooling issues
    - Online
    - Local
  - Technology
    - Data promoted into XBRL
    - Data processed by efficient databases
Questions

Herm Fischer
herman.fischer@ubmatrix.com
fischer@markv.com
+1- 818- 995- 7671
+1- 818- 404- 4708

THANK YOU!