

DAMA Mundial – April 25, 2006

Semantic Seeds:
Maintainable Configuration Dataload Using Ontologies

Stu Baurmann
Lead Instructor, LogicU
<http://www.logicu.com>

©2006 Stu Baurmann - www.LogicU.com

Overview

Semantic Seeds – *What and Why?*

Why?

- Repeatable config processes enable better testing & faster, cleaner deployment
- Config artifacts represent business intent, so they should have high & durable value
- The best config artifacts require wide participation, & deserve ongoing investment

What?

- ❖ Ontology artifacts capture your business concept space..
 - ❖ Expressive, declarative, queryable, based on standards
 - ❖ Created and maintained using low/no cost GUI and server tools
 - ❖ Numerous options for physical persistence, fully decoupled from logical model
 - ❖ Metadata, instance data, and mapping data may all be colocated (small pilot projects) or kept separate (to keep large projects maintainable)
- ❖ ..and are mapped to your application implementation space.
 - ❖ You can directly use ontologies for runtime config
 - ❖ Or, use a dataloader to generate web service calls, SQL code, and/or method calls

©2006 Stu Baurmann - www.LogicU.com

“Config Today” – gross generalizations

- ❖ Typical Enterprise App
 - ❖ J2EE or .NET
 - ❖ UML, SQL, XML
- ❖ Config is a combination of XML, SQL, .txt, (hardcodes?)
 - ❖ Artifacts are kept under version control and/or in live databases
 - ❖ Master and Reference data is maintained by DBAs
 - ❖ Boundary between Reference and Application data not obvious
- ❖ During dev and QA, the RDBMS is frequently reseeded

...while code and schemas may be changing!

©2006 Stu Baumann - www.LogicU.com

Ex: Dataload for a banking app

Load Externally Sourced Reference Data

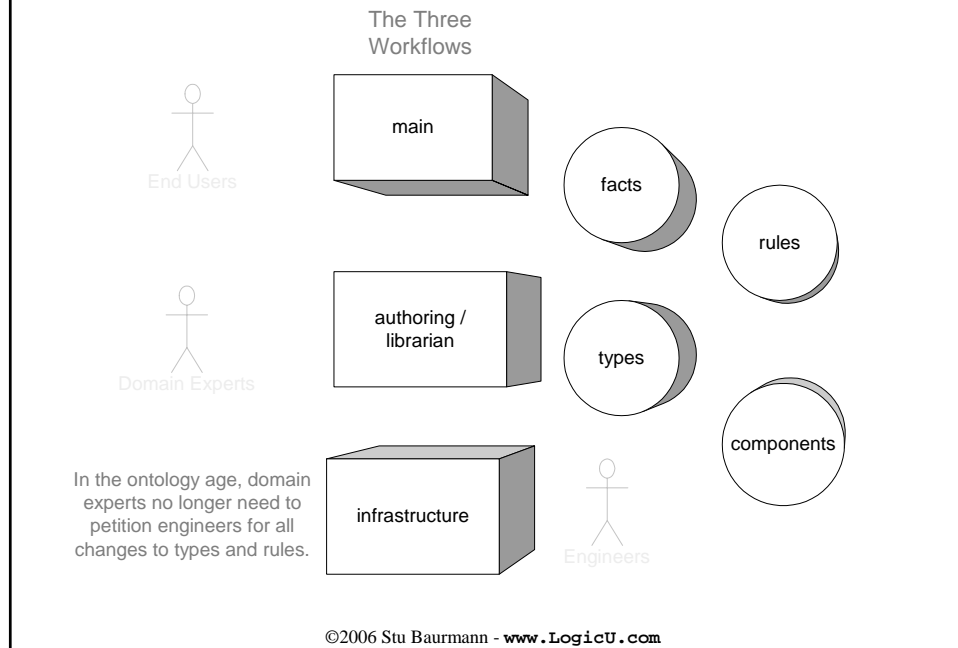
1. Load geographic and currency reference data
2. Load exchange rate and LIBOR history

Load Enterprise Master Data and Application Data

1. Load complex financial instrument model
2. Load partner bank links
3. Load accountholders
4. Load current positions and unsettled transactions
5. Load transaction history

©2006 Stu Baumann - www.LogicU.com

Archetypical App Definition Process



Where we still fall short

- ❖ Many requirements issues are related to questions of configuration
 - ❖ Example: "What rules govern financial transaction settlement?"
- ❖ At some level, most systems strive for business user authorability
 - ❖ Quality of end user experience is driven by "middle workflow" effectiveness
 - Example: "How much power do analysts have to author settlement terms?"
- ❖ Comprehensive test setup requires a maintainable library of data scenarios
 - Example: "We need to test system under all different settlement scenarios"

©2006 Stu Baurmann - www.LogicU.com

Dataload Challenges - Tech

Typical modern dataload processes involve a combo:

- SQL scripts
 - Imperative code (shell scripts, java classes, etc)
 - XML files sent to app over web services
 - Human or scripted manipulation of app using GUI
-
- What do we do when code / schema changes?
 - Mutate DB in place?
 - Reload DB from scratch?
 - Who does this? Who has the required knowledge?



(image from <http://www.webaroony.com/resume/dragon/images/anim.gif>)

©2006 Stu Baurmann - www.LogicU.com

Dataload Challenges - Process

- Who is responsible for configuration choices?
 - Applications are increasingly authorable, and may involve multiple conceptual models exposed to users
 - e.g. financial model of bonds and derivatives for traders, with complex settlement negotiations based on counterparty credit risk evaluations.
 - Building proper GUIs to do the authoring often winds up a low priority.hmmmm
 - With or without GUIs, some scripted dataload is almost always necessary.
- Distinction between requirements, configuration, and code is not always clear, and not always seen the same way by all parties concerned.
 - Infrastructure and Author workflows are often tangled.
- Resource drain and schedule risk
 - Time sink for developers, QA, DBA
 - Ineffective dataload blocks system testing, holding up the entire release cycle.



©2006 Stu Baurmann - www.LogicU.com

Enter The Ontology

Claim: The ontology paradigm enables maintainable, repeatable dataload!

How?

Conceptual model is captured declaratively

may include inheritance and complex metadata

Any relational or graph/OO structure may be represented and mapped to target DBs

Strong namespace support; clean merging and federation of models

Ontologies may be persisted in files, SQL DBs, or native triplestores

Ont. representation is orthogonal to your application implementation mechanisms

Sufficient GUIs and APIs available as open source or commercial components

At configuration design+maintain time:

Tools like Protégé, OntoStudio, TopBraid allow us to manipulate configuration models and author mappings graphically, without writing our own domain-specific tools

At dataload run time:

Selected RDF graphs are transformed by mappings into some combination of:

Coarse grained Web Service calls

SQL statements

Method calls

©2006 Stu Baumann - www.LogicU.com

Understanding Ontology Models

- Definitions

- Model is a physical unit of administration, containing a set of triples.
- Namespace is a globally unique domain of logical identity.
- QName (qualified name) uses prefix to refer to a namespace, and is equivalent to a URI, e.g.

QName

URI

lub:ExchangeRate == http://www.logicu.com/bond#ExchangeRate

- Some rough correlations between ontologies and the world of SQL:

• Ontology	<-?->	<i>No obvious analog</i>
• Class	<-?->	Table(s)
• Property	<-?->	Column(s)
• Axiom	<-?->	Constraint / Clause
• Individual	<-?->	Row(s)

- Key characteristics

- Logical metadata sets may be federated across namespaces and models
- Physical metadata sets (of triples) may be cleanly merged or divided
- Numerous inference mechanisms may be applied to derive implied triples
- Ubiquitous concerns: versioning, synchronization and workflow
 - Among semantic models
 - Across all IT artifacts, old and new

©2006 Stu Baumann - www.LogicU.com

Semantic Toolchest

Key Standards / Specs

- RDF (+RDFS), OWL-DL, SPARQL
- FSL (Fresnel Selector Language) – like XPath for RDF
- Common Logic, RuleML, SWRL, F-Logic
- OWL-S, WMSO, WSDL-S

Open Source Tools

Protégé + Pellet + Jena + IsaViz-FSL

... + Cocoon + Flora + Prova => www.peruser.net

Also see: Sesame, Kowari, eXist



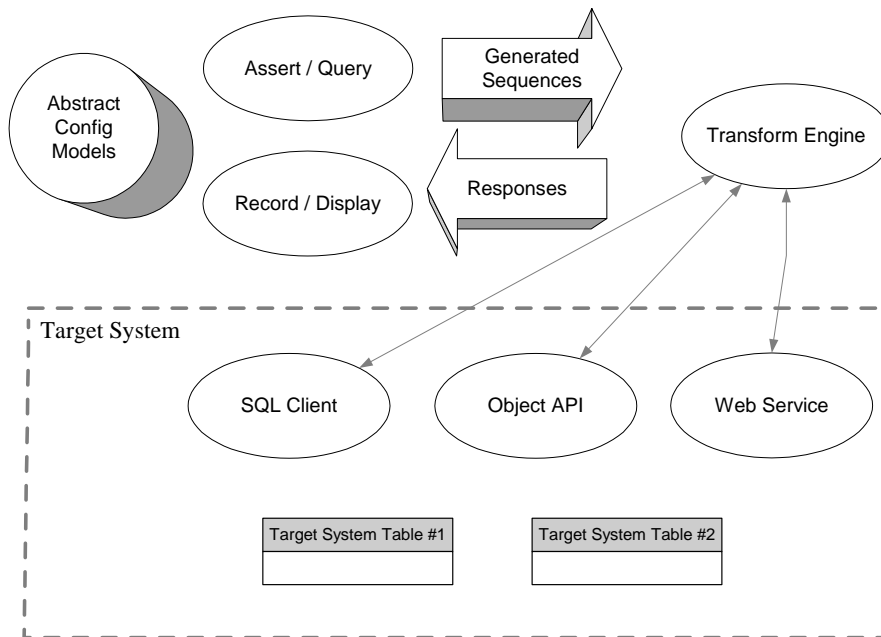
Pay Tools

US: Metatomix, TopQuadrant, Intellidimension, Cerebra, Oracle, IBM

Eur: Ontoprise, Software AG, Ontotext, Racer, Altova

©2006 Stu Baumann - www.LogicU.com

General Pattern for Dataload Runtime



©2006 Stu Baumann - www.LogicU.com

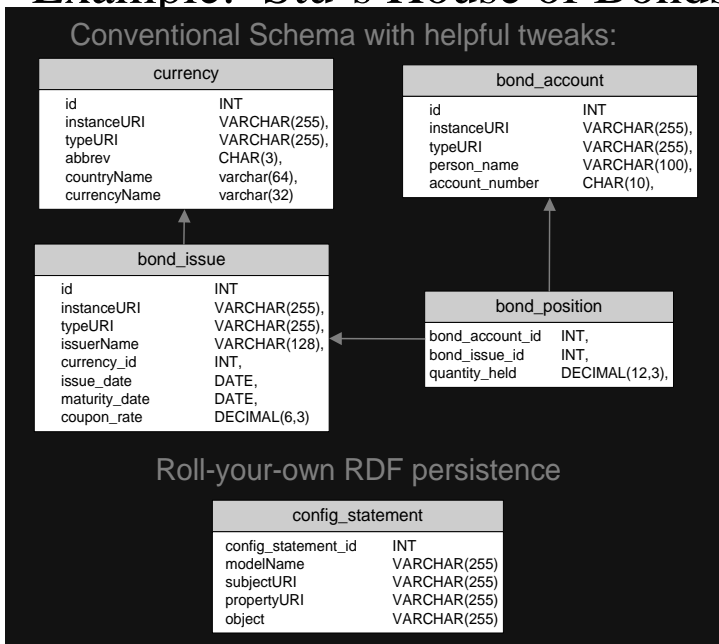
Steps to implementing onto-config

- Assign namespaces and prefixes – internal and external
- Assign well-known QNames to serve as entry points
- Create pilot models for metadata, instance data, and mappings
 - Initially you can choose to keep these models physically colocated, but as project grows, you will probably want to separate them.
- Make this abstract config data available to your apps
 1. Option: You may directly use RDF models for runtime config!
 - Store in files, use Jena+SQL, or roll your own RDF persistence.
 - Not much to do as far as actual “dataload”, in this case.
 2. Option: Pump data from models into your application
 - Easier Job: Reset from an empty database
 - Harder Job: Synchronize with existing data
- Getting Fancier
 - Consider using inference engine such as OWL-DL reasoner, etc.
 - Simple rules for transitive closure can provide surprising leverage

©2006 Stu Baumann - www.LogicU.com

Example: Stu’s House of Bonds

Conventional Schema with helpful tweaks:



©2006 Stu Baumann - www.LogicU.com

How Do We Pump Data?

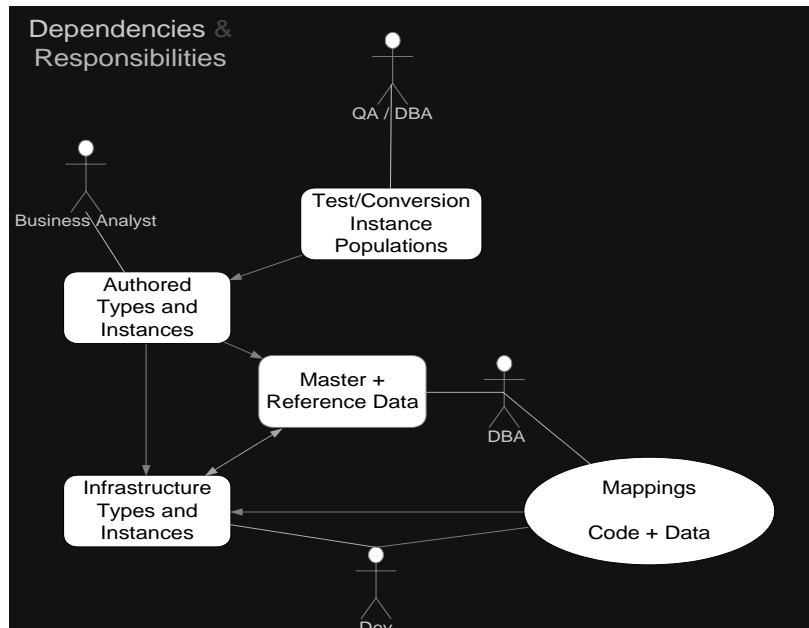
Defining an appropriate load processor for your context:

1. Pull information from models using SPARQL, FSL, Java,..
 - “What to Do?” is determined mainly by “Mappings” (which may involve code)
 - Models may be queried “raw” or after inference is applied
 - Use well known QNames as handholds
 - Jena / Protégé will generate a file of Java constants for you

 2. Push to your app, and stitch identifiers
 - Ideally, generate web service calls, with coarse-stitched IDs
 - And/Or generate SQL code, with fine-stitched IDs
 - Of course, stored procedure calls are a coarser grained option.
 - And/Or generate method calls to your system’s API
- Stitching of IDs is reasonably easy for complete reloads of modest size
- Keep an in-memory map from model-URI to tableName + rowID
- For incremental loads, or very large loads, things get trickier
- Need to be able to “find” rows in target DB that correspond to a URI
- Option A: Mark certain model properties as corresponding to known “business keys”
- Option B: Store mappings of model-URI to physical keys in a supplementary model
- Option C: Add columns for model-URI to your app schema (as on previous slide)

©2006 Stu Baurmann - www.LogicU.com

Maintenance Process Overview



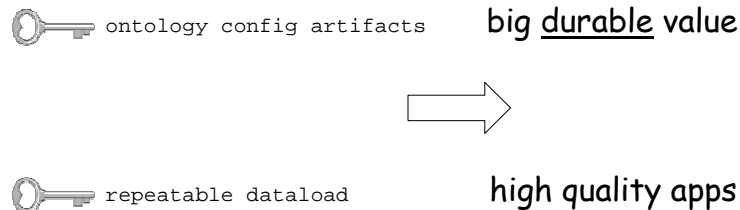
©2006 Stu Baurmann - www.LogicU.com

Guidelines for high value config authoring

- Focus on capturing “Articulate Artifacts”
 - Middle (business) workflow users should participate in artifact definition
 - Compare with configuration process grounded in Word and Excel!
 - Business concept space is idealistically modeled, but pragmatically mapped to legacy application.
 - Pursue iterative model improvement – start simple and partial
- Connect these artifacts to other work in your organization
 - Part of the draw of ontologies is their enterprise/global applicability
 - Configuration ontologies should overlap with ontologies built for EAI, etc.
 - Relating ontologies to other Enterprise models models:
 - OMG Ontology/Rules work: ODM, SBVR, PRR
 - XMI /UML import/export using Protégé, Eclipse
 - Other intersections: eBXML, BPML, WSDL, etc.
 - Next stop: OLAP!

©2006 Stu Baumann - www.LogicU.com

Sum



©2006 Stu Baumann - www.LogicU.com

References

Baumann S, 2005 -- "Promising Developments in Interoperable Semantics: *Ontologies, RDF, and OWL*"
Presentation to DAMA-PS in September, 2005
http://www.drma-seattle.org/stu_ontology_pres_dama_ps_2005_09_13_004.ppt

Rauschmayer A, 2005 -- "RDF as a Data Structure for Software Engineering"
<http://www.mel.nist.gov/msid/conferences/SWESE/repository/presentations/axel.pdf>

Rauschmayer A and Renner P, 2004 -- "Knowledge-Representation-Based Software Engineering"
<http://www.pst.ifi.lmu.de/~rauschma/publications.html>

The following are all part of the Proceedings of 2006 Wilshire Semantic Technology conference
<http://www.semantic-conference.com>

Kendall E, McGuinness D, 2006 --	"Ontology 101 Revisited"
Hay D, 2006 --	"Data Modeling, RDF and OWL: <i>About Ontologies</i> "
Kendall E, 2006 --	"The Ontology Definition Metamodel: <i>An MDA based framework for semantic interoperability</i> "
Zaremba M, 2006 --	"Web Services and Web Services Modeling Ontology"
Tabet S, 2006 --	"A framework for Rules Standardization on the (Semantic) Web"
Baumann S, 2006 --	"Semantic Testing: <i>Verification and the Knowledge System</i> "
Baden H, 2006 --	"Change Management and Versioning in Ontologies"
Alonso O, Lopez X, 2006 --	"Semantics in the [Oracle] Database"

Stu Baumann contributes to these websites:

logicu.com scrutable.com hotlaundry.com

peruser.net xmlexpertise.com suiteness.com

©2006 Stu Baumann - www.LogicU.com