Do You Need A “Data Development Methodology” (DDM) ?

John Singer
Senior Consultant – Data Architect
MasterCard Worldwide
jsinger0420@yahoo.com
John Singer – Data Architect

• 26 years experience
  – Information Center, End User Computing
  – DBA, DA, DW
  – Recovering IT Manager
  – Metagroup industry analyst

• Currently working on...
  – ITIL Configuration Management Database (CMDB)
  – Metadata management
  – Semantic data servers
Do You Need A DDM?

Do your models matter???
Feeling left behind or left out???

Presentation Outline

• Two IT trends challenge the DA status quo
  – Agile Methods De-emphasize Data
  – The “Disappearing Database”
• Crafting A Data Development Methodology (DDM)
  – Managing Modeling
  – Mapping Methodologies
• Two trends are challenging the traditional “Information Engineering” world view.
  – “New Age” Methodologies – how do DA groups adapt to methodologies that de-emphasize data?
  – “Disappearing Database” – application framework API’s hide the underlying SQL database making E/R models less relevant.
New Age Methodologies

From an InfoWorld Article – Jan 8, 2007

“Any shift towards scripting should be accompanied by a shift away from formal requirements”

Question: what are “informal” requirements?
New Age Methodologies

• New Age “Agile” or “Evolutionary/Iterative” methodologies
  – Rational Unified Process (RUP), Extreme Programming (XP), Agile, Scrum, .... others
  – Advocate “just good enough” data model to support “just good enough” project code...
  – Just “refactor” the database when it changes...

• Unfortunate consequence of this approach is typically – no data model.

• www.agiledata.org

We are entering an era of “throw-away” applications
New Age methodologies emphasize Project Management concepts over Methodology. They typically do not cover the entire life cycle.
Differentiating Methodologies

# of Reqt’s per “cycle” decreases as Agility increases
Adapt DDM To Differing Methodologies

- Recognize the 3 different OO – Relational Mismatches
  - UML models are not the same as E/R models
  - OO persistence frameworks (Hibernate etc.) force structure on the database
  - Agile project management vs. Info. Eng. Methods

- You must adapt – Agile methods and OO aren’t going away
  - Insist on up front entity only model (at least)
  - Focus on capturing business rules and definitions as the schema evolves – this blurs the line between “logical/physical” modeling
  - May require one person to be “Modeler/DBA”
The Disappearing Database

Application Frameworks increasingly hide the underlying data structure and familiar SQL API.

- Workflow and Business Rule Engines
- ERP packages
- Content Management / Semantic Servers
- Business Intelligence servers (OLAP etc.)
- MDM and CDI “hubs”

And the biggest buzz-word of the 21st Century....

- SOA - Service Oriented Architecture

E/R models no longer describe the data structures.
The Disappearing Database

• Application frameworks are metadata driven....the underlying data structure becomes less important.

• Visual tools create data mappings – the new data model
Adapting to SOA and Application Frameworks

- Extend modeling capabilities to include
  - SOA interfaces
  - Business process flows (BEPL)
- Leverage MDM and data hub projects as the new “Enterprise Model”
  - Mappings between hub and spoke
- Shift focus from enterprise E/R model (if you even have one) to data element definitions (IEEE 11179) and Business Rules efforts
Traditional Information Engineering

Three Schema Architecture
Published in 1978 by ANSI/SPARC

- External – how users see data
- Conceptual – single coherent definition
- Internal – Implementation dependant representation
- Physical – bits on media

ANSI/SPARC terminology

<table>
<thead>
<tr>
<th>External</th>
<th>Conceptual</th>
<th>Internal</th>
<th>Physical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intended Model terms</td>
<td></td>
<td>Conceptual</td>
<td>Logical</td>
</tr>
<tr>
<td>Commonly used model terms</td>
<td>Logical</td>
<td>Physical</td>
<td>n/a</td>
</tr>
</tbody>
</table>

"3 schema architecture" goal of Data independence and sharing largely un-met by today’s RDBMS application implementations
The goals are laudable yet -

- Modern Relational DBMS only partially support the “vision” of the integrated conceptual schema and data independence
- Confusion over terminology has never been resolved (logical vs. physical)
- Integrated Conceptual Schema is not be possible
  - consensus among “semantic” world is there is no single upper ontology
  - Ontologies “commit” to upper ontologies as a way of defining and limiting their “universe of discourse”
- www.dbdebunk.com

Data models can only “integrate” within limited application scope (i.e. universe of discourse)
The DDM Process Model

Enterprise Model

A Miracle Occurs Here

Project Requirements

People/Process/Technology

Implemented Database And Data Instances
Why do we have DA’s and DBA’s?

- Different tasks require different skills
- Different audiences require different analysis artifacts
- It makes sense to break down large efforts into intermediate steps
Define Modeling Deliverables

Models progress from requirements to delivered solution
- How many? Assessment Model, Schema Model, Technical Model
- Define specific modeling tools and methods, down to attribute level
- Is Assessment model maintained?
## Formalize DDM Tasks/ Roles/ Methods

<table>
<thead>
<tr>
<th>#</th>
<th>ADM Task</th>
<th>DDM Task</th>
<th>Description</th>
<th>Deliverables</th>
<th>Participant Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Review walk thru functional requirements.</td>
<td>Review walk thru Logical data model.</td>
<td>This review is performed with all necessary parties to insure accuracy and completeness</td>
<td>Meeting minutes Action items</td>
<td>B O B A D A D B A D V L P R</td>
</tr>
</tbody>
</table>

- Map DDM tasks/deliverables to ADM
- Define participant roles
- Include “doing, reviewing, signoff” type tasks
Present DDM as Separate Process Flow

- Analysis tasks/deliverables that map to your “enterprise” efforts (MDM, CDI, Business Rules, Common Data Elements, SOA interfaces)
- Separate sign-offs on important data deliverables (Models, physical database)
- Don’t forget DBA tasks – dev/test/prod setup
Crafting the DDM

• Inventory DA and DBA tasks
  – Define modeling artifacts
  – Define where “logical” ends and “physical” starts
• Formalize DA and DBA roles
  – Emphasize teamwork and dual accountability
  – Don’t forget important DBA implementation/tuning tasks
• Map DDM tasks/deliverables to DDM Methodology Phases
  – Provide multiple paths to support Agile and Traditional ADM’s
• Map DDM to ADM
• Publish “Methodology”
• Execution – SLA’s – SELL SELL SELL
Summary

• Get involved in BPM, SOA, MDM, CDI efforts – that’s where the External to Conceptual mappings really happen

• Focus on Data Dictionary and Model Design Patterns as the “Enterprise” model, not the single conceptual E/R model

• Focus modeling on translating requirements to delivered solutions

• Give Data Architecture meaning in the organization via a published methodology