Enterprise Architecture

The Zachman Framework: Intro to Sample Models

Observation

Enterprises are

COMPLEX

The US Pentagon

GM Plant

Agenda

I. Enterprise Models from Literature
II. Implementation Discussion
III. Architecture Discussion
IV. Column 1 Model Samples
V. Row 1 Model Samples
VI. Column 2 Model Samples
VII. Etc., Etc. 'till Time Is Up

Models on my Bookshelf

1. **"Requirements Analysis" by David C. Hay**
   - Activity Management
   - A Complete Sarson and Gane Data Flow Diagram
   - Physical Asset
   - Value Constraints
2. **"Information Modeling and Relational Databases" by Terry Halpin and Tony Morgan**
   - IT Company Schema and University Schema
3. **"Enterprise Architecture for Integration" by Clive Finkelstein**
   - Strategic Model for sample solution
   - Order entry data map with all attributes
   - 5BNF data map - ORG and ROLE STRUCTURES
4. **"Designing Quality Databases with IDEF1X Information Models" by Thomas A. Bruce**
   - Case Study Supplementary Material (Logical Data Model)
5. **"Business Process Management"by Roger T. Burlton**
   - The Scope for Global Software Human Resources

Models on my Bookshelf

6. **"Enterprise Architecture at Work" by Marc Langhorst**
   - Services provided by Handle Claims Process
   - Handle Claims and IT Support
7. **"Business Process Engineering" by August Scheer**
   - ERM for Human Resource Planning
   - Event-driven process chain for inbound logistics
8. **"Data Model Resource Book" by Len Silverston**
   - Work effort invoicing
   - Work order and work order effort model
9. **"Workflow Modeling" by Alec Sharp**
   - Hand-off level diagram for a to-be process
   - Milestone level diagram for a to-be process
10. **"The Practical Guide to Structured Systems Design" by Meiler Page-Jones**
    - Case study DFD (excruciating detail and high level of detail)
11. **"The Object Advantage" by Ivar Jacobsson**
    - The use cases and actors in our final model
    - The flow of events in a use case
12. **"Mainstream Objects" by Ed Yourdon et al**
    - Seminar registration system - object structure

Models on my Bookshelf

13."Business Process Improvement" by H.J. Harrington

- Relating external customer expectations to process

14."Industrial Dynamics" by Jay W. Forrester

- Subdivisions of the model, interconnections ...
- Continuous balance sheet at factory

15."Kaizen" by Masaaki Imai

- Quality Assurance Systems Diagram

16."Strategy Safari" by Henry Mintzberg

- Annual Planning Cycle at General Electric
- Stanford's proposed 'System of Plans'

17."The Fifth Discipline" by Peter M. Senge

- Codependency archetype

18."Competing Against Time" by George Stalk

- Order flow logistics - Location of Departments
- Order flow logistics - Departments

19."Competitive Advantage" by Michael E. Porter

- Value Chain

20."Building Enterprise Information Architectures" by Melissa A. Cook

- Ballpark view process classes
- Ballpark view global data classes

Models on my Bookshelf

21."Systems Analysis and Design Methods" by Jeffrey L. Whitten and Lonnie D. Bentley

- Member Services Fully Attributed Data Model
- A (Process) Systems Diagram
- Detailed Location Connectivity Diagram
- Member Services Use Case Model Diagram
- The End Product of Structured Design
- Activity Diagram for the Procurement Phase
- Star Networks and Hierarchy networks
- Sample Physical Data flow Diagram
- Sample System Flowchart
- Network Topology Data Flow Diagram
- Data Distribution Across the Network
- Physical DFD Design Unit for an Event
- SoundStage Logical Data Model - 3rd Normal Form
- Prototype for New Video Title Screen
- Typical External Turnaround Document
- Efferent Portion of Structure Chart
- SoundStage DFD Reflecting Central Transaction
- SoundStage Structure Chart from Transaction Analysis
- DFD with Transaction Center/Structure Chart
- SoundStage DFD Reflecting Transaction Center
- SoundStage Structure Chart from Transaction Analysis
- Interaction Diagram for Order use Case
- Partial Object Model for Use Case

Models on my Bookshelf

22.Enterprise Architecture Planning - Actual Example By Doug Erickson, ENTARCO

- Enterprise Motivation Model
- Logical Business Process Classes
- Business Entity Classes
- Logical Processes vs Data Classes matrix
- Logical Business Units vs Organization Units Matrix
- Logical Business Units vs Logical Business Processes Matrix
- Logical Business Units vs Logical Business Processes
- Logical Business Units vs Data Classes Matrix
- Logical Business Process Data Dependency Chart

23.Enterprise Architecture Methodology - Examples by Doug Erickson

- Enterprise Conceptual Data Model
- Enterprise Conceptual Data Model - Attributed
- Enterprise Conceptual Process Model

24.etc. etc. etc.

Composites

All of these example Enterprise models are good, relevant, useful, valuable, helpful for building automated and manual systems, analyzing problems, proposing solutions, etc., etc.,

Necessary for doing actual Enterprise work.

Composites

All of these example Enterprise models are good, relevant, useful, valuable, helpful for building automated and manual systems, analyzing problems, proposing solutions, etc., etc.,

Necessary for doing actual Enterprise work.

They are all:

different

multi-variable

"composite"

"views"

descriptive representations

models of an Enterprise

Composites

All of these example Enterprise models are good, relevant, useful, valuable, helpful for building automated and manual systems, analyzing problems, proposing solutions, etc., etc.,

Necessary for doing actual Enterprise work.

They are all:

different

multi-variable

"composite"

"views"

descriptive representations

models of an Enterprise

How many other good,

good,

relevant,

useful,

valuable,

helpful models

could you find?

Which is Best?

Which sample model was the best one?

I submit, the model needed will depend upon:

1. The work to be done
2. The skills, creativity, experience, culture of the worker
3. The state of the art, available descriptive technology
4. The Enterprise data available

etc.

In short, there is not one kind of relevant model … the kinds of relevant models are virtually infinite and not likely predictable until needed and created.

Enterprise Architecture

The idea of Enterprise Architecture is to discover the underlying, elementary, "primitive" components from which any relevant, unpredictable, "composite" model can be created:

for any given Enterprise
at any given point in time
for any given kind of work
by any given person.

This might not be as hard as you think ... because it is possible to identify the basic set of elementary Components that exist, that can be described and from which any "composite," "view" (Model) of an Enterprise (or for that matter, for any industrial object) can be created.

Humanity, for the last 7,000 years has proven this architectural idea and the older disciplines of:

Architecture and Construction, and
Engineering and Manufacturing

have tested the proof for the last 300 years.

The Zachman Framework for Enterprise Architecture™

The Zachman Framework for Enterprise Architecture™

The Enterprise Ontology ™

Version 3.0

Classification Names / Audience Perspectives	What	How	Where	Who	When	Why	Classification Names / Model Names
	Composite Integrations →		← Alignment →		← Composite Integrations		
Executive Perspective (Business Context Planners)	Inventory Identification e.g. List: Inventory Types	Process Identification e.g. List: Process Types	Distribution Identification e.g. List: Distribution Types	Responsibility Identification e.g. List: Responsibility Types	Timing Identification e.g. List: Timing Types	Motivation Identification e.g. List: Motivation Types	Scope Contexts (Scope Identification Lists)
Business Mgmt Perspective (Business Concept Owners)	Inventory Definition e.g. primitive model e.g. Business Entity Business Relationship	Process Definition e.g. composite model e.g. Business Transform Business Input/Output	Distribution Definition e.g. Business Location Business Connection	Responsibility Definition e.g. Business Role Business Work Product	Timing Definition e.g. Business Interval Business Moment	Motivation Definition e.g. Business End Business Means	Business Concepts (Business Definition Models)
Architect Perspective (Business Logic Designers)	Inventory Representation e.g. System Entity System Relationship	Process Representation e.g. System Transform System Input /Output	Distribution Representation e.g. System Location System Connection	Responsibility Representation e.g. System Role System Work Product	Timing Representation e.g. System Interval System Moment	Motivation Representation e.g. System End System Means	System Logic (System Representation Models)
Engineer Perspective (Business Physics Builders)	Inventory Specification e.g. Technology Entity Technology Relationship	Process Specification e.g. Technology Transform Technology Input /Output	Distribution Specification e.g. Technology Location Technology Connection	Responsibility Specification e.g. Technology Role Technology Work Product	Timing Specification e.g. Technology Interval Technology Moment	Motivation Specification e.g. Technology End Technology Means	Technology Physics (Technology Specification Models)
Technician Perspective (Business Component Implementers)	Inventory Configuration e.g. Tool Entity Tool Relationship	Process Configuration e.g. Tool Transform Tool Input /Output	Distribution Configuration e.g. Tool Location Tool Connection	Responsibility Configuration e.g. Tool Role Tool Work Product	Timing Configuration e.g. Tool Interval Tool Moment	Motivation Configuration e.g. Tool End Tool Means	Tool Components (Tool Configuration Models)
Enterprise Perspective (Users) The Enterprise	Inventory Instantiations Operations Entities Operations Relationships	Process Instantiations Operations Transforms Operations In/Outputs	Distribution Instantiations Operations Locations Operations Connections	Responsibility Instantiations Operations Roles Operations Work Products	Timing Instantiations Operations Intervals Operations Moments	Motivation Instantiations Operations Ends Operations Means	Operations Instances (Implementations) The Enterprise
Audience Perpectives / Enterprise Names	Composite Integrations →		← Alignment →		← Composite Integrations		
	Inventory Sets	Process Flows	Distribution Networks	Responsibility Assignments	Timing Cycles	Motivation Intentions	

*Horizontal integration lines are shown for example purposes only and are not a complete set. Composite, integrative relationships connecting every cell horizontally potentially exist.

This is Really Simple

The contents of any category, class (Framework Cell) of generic components ("Primitives") is composed of two and only two different kinds of things ... (1) the generic component itself and (2) its relationship with all other generic components of the same type. For example:

Col. 1: Inventories (Entities) and Counts (Relationships)
Col. 2: Transformations (Processes) and Inputs/Outputs (Flows)
Col. 3: Locations (Nodes) and Connections (Links)
Col. 4: People (Skills) and Assignments (Work Products)
Col. 5: Cycles (Durations) and Moments (Points in Time)
Col. 6: Ends (Objectives) and Means (Strategies)

The other dimension of the two dimensional classification system are the stages of Reification ... transformation of ideas into reality:

Row 1: Identification (Names, Boundaries Clarified)
Row 2: Definition (Concepts, Semantic Structures Defined)
Row 3: Representation (Logic, Concepts Systematized)
Row 4: Specification (Physics, Technology Constrained)
Row 5: Construction (Tooling Configured)
Row 6: Instantiation (Reality Created)

Elegance of Primitives

The elegance of a Primitive, single variable model is:

1. Enterprise-wide models - only one kind of thing in the Enterprise model reduces the complexity immeasurably and the model can be tested for completion.
2. "Net Set" - extraneous components can be eliminated producing a minimal, optimal, Enterprise model.
3. Correct - structure of the model supports the intent and the operation of the Enterprise. Composite models created through using (or re-using) Primitive components, by definition, will be "aligned".
4. Flexible - separation of independent variables allows Primitives to be changed independently from one another, changes to a Primitive becomes a "delta" and change impacts on other Primitives through 'integrations' (i.e. Composites) can be predicted.
5. Reduced "Total Cost of Ownership" - eliminates discontinuities, **Entropy**, through "normalization" of Primitives - Reduced General and Administrative costs of Enterprise operations.
6. Instances are traceable to every Reification Stage.

Enterprise Entropy

"**Entropy**: 1 a: a measure of the unavailable energy in a ... system that is also usually considered to be a measure of the system's disorder. 2 b: a process of degradation or running down or a trend to disorder. 3: CHAOS, DISORGANIZATION, RANDOMNESS."

Mirriam-Webster's Collegiate Dictionary

The Mythical Man-Month by Frederick P. Brooks

"All repairs tend to destroy the structure, to increase the **entropy** and disorder of the system. Less and less effort is spent on fixing the original design flaws; more and more is spent fixing flaws introduced by earlier fixes. ... Although in principle usable forever, the system has worn out as a base for progress. Furthermore, machines change, configurations change, and user requirements change so the system is not in fact usable forever. A brand new, from-the-ground- up redesign is necessary. ...

"Systems program building is an **entropy**-decreasing process hence inherently metastable. Program maintenance is an entropy-increasing process, and even its most skillful execution only delays the subsidence of the system into unfixable obsolescence."

Note: The Second Law of Thermodynamics

Redundancy

There are two problems with redundancy:

1. Spending money that doesn't have to be spent
2. Entropy. (If the same concept exists more than once but is not defined consistently ... or if the inventory record values don't agree: Disorder, Discontinuity.)

Disorder, discontinuity requires
reconciliation, compensation, cross-references, interfaces,
"work-arounds", MBA's with Excel Spreadsheets, etc., etc.
to continue operations ... or the urgent case,
to prevent failure.

Entropy

Entropy, General & Admin. Costs, Indirect Expenses costs (lots of) money and increases over time!

When the cost of Enterprise operations exceeds the Enterprise value contribution, the Enterprise becomes dysfunctional (extinct).

Total Cost

It is not adequate to value a system (any implementation) based on its development costs and implemented benefits alone.

You need to look at the "Total Cost of Ownership"
(maintenance and replacement) and
Total Cost of Ownership in the context of
THE ENTERPRISE.

If any one system implementation creates redundancies (discontinuities) with existing systems (legacy) or future systems (project slate) there WILL be
ENTERPRISE ENTROPY
to BEGIN WITH.

So: how do you reduce Entropy?
Reduce disorder.

So: How do you reduce disorder?
Reduce Redundancy.

Redundancy

So: How do you reduce redundancy (i.e. Make it LEAN)?
(If this was easy, it would already be done!!)
(I submit ... we can't even FIND the redundancies.)
(Because everything is instantiated as COMPOSITES!)

There are two possibilities:
1. Rebuild the systems, ground-up ... modernize. (Re: Fred Brooks)
2. Architect the Enterprise so redundancies and
inconsistencies can't be created in the first place. (Re: JAZ - Enterprise Architecture)

So ...How do you even FIND the redundancies?
A. Enterprise-WIDE models.
B. Single variable, **PRIMITIVE** Models

PRIMITIVE MODELS

We are not trying to do implementations with Primitive models. We are trying to identify and optimize the non- redundant, "net" set of Enterprise components from which any Enterprise implementation composite can be created dynamically, addressing the "Total Cost of Ownership"
... of THE ENTERPRISE.

Please Note

I never said, "Stop the music for 15 or 20 years and build out all the Primitive Models before you can do any more implementations or actual work!

I DID say,
"someday, SOMEDAY, you are going to wish ... "
In fact I said "Forget YOU, someday, THE ENTERPRISE is going to WISH they had all the Primitive models made explicit, all of them Enterprise-wide, all of them horizon- tally and vertically integrated and all of them at excruciating level of detail!

Why??: It would be LEAN AND MEAN ! However, and furthermore, I AM saying that any implementation (system) you build that is not derived by re-using components from Primitive Models may well be implemented ... but NOT Architected, certainly not
ENTERPRISE-Architected.
It is going to be ... more "legacy".

Think about the last 75 years or so. Have we ever built anything other than "legacy." What has changed??

Please Note Again!

I NEVER said, "Don't build any more systems!!!"

Notice:
The short term demand is NOT going to go away! There is substantive value to implementing a system. Systems are better, faster and cheaper than people. New technologies open up new opportunities.

Problems must be solved!

etc., etc.

However, I AM saying, "if all you are doing is building and running systems, what do you think you are going to end up with at the end of the day? SYSTEMS!

a "Legacy" by whatever name it is called !

And I don't care how fast your hardware is or how new your operating system is or how clever your programmers are. You are NOT going to end up with a coherent, optimal, flexible, integrated, aligned, lean & mean ENTERPRISE!

You are going to end up with more LEGACY!

Observations

So far, few people have made really serious efforts to discover the underlying Enterprise Primitives from which implementation Composites could be composed.

Maybe the one notable exception that most people would acknowledge as a worthy endeavor would be the Inventory Structures C1 Primitives (the Enterprise Entity R2 or "Data" R3 Models) which, if they were created, could be re-used in more than one implementation ...

which is the very concept of ALL of the Primitives.

It is interesting ... the great preponderance of books on systems or any kind of modeling on my bookshelf have a discussion on "Entity Models", or "Data Models".

But even at that ... we seem to have this uncontrollable urge to imbed other Primitive components (like Processes) into the Entity ("Data") Model because that's what we want for implementation purposes, making it into an implementation model, a "Composite".

Observations No. 2

Forget the Primitives for a moment ... and forget "Total Cost of Ownership".

Another reason why few people have made serious efforts to do Enterprise Architecture is because, by its very name, it implies, Enterprise-WIDE Architecture. And,

1. It would take too long and cost too much!
2. You don't need Enterprise-Wide models to get some one system built ... and deliver Enterprise BENEFIT!! (Immediate gratification.) And ... the smaller you make the systems, the faster you can deliver them.
3. Enterprise-wide would be soooo complex, who could understand it ... even if you could build it?
4. To simply build them and understand them, they would have to be so abstract they would have little value. They couldn't be correlated with implementations.

Here is the real problem ... the underlying assumption in all the above is: Enterprise-Wide COMPOSITE Models. And, the only practical solution under this assumption is to decompose, build smaller pieces (adding redundancy, adding discontinuity, DIS-integrating the Enterprise), building more LEGACY.

BUT PRIMITIVE MODELS!

IF you have only ONE kind of thing in each Model,
and
IF those one things are unitary concepts ...
NOT complex, composite constructs,
and
IF those unitary things are precisely defined by a two-dimensional schema, an Ontology (like the Periodic Table),
and
IF you manage the inventory of those things you create,
THEN
Enterprise-wide models become a feasibility.

Complexity is reduced immeasurably
and
You can build Primitive Models out "sliver" by "sliver" over some long period of time, controlling discontinuity (dis-integration) by controlling redundant creation of the unitary concepts, you are designing for change (separating independent variables)
AND
If you build more than one Primitive Model, you can
create Composite implementations as you go.

Why Enterprise Architecture?

- Reduce Enterprise Operating Costs, General and Administrative costs, make the Enterprise LEAN. Minimum possible cost of operations.
  - Enterprise Design Objective: INTEGRATION

- Reduce the time, disruption and cost of Enterprise Change.
  - Enterprise Design Objective: FLEXIBILITY

- Ensure Enterprise operations reflects the intentions of Management
  - Enterprise Design Objective: ALIGNMENT

- Make the Enterprise "MEAN" - Reduce response time to external demands.
  - Enterprise Design Objective: REUSE, MASS-CUSTOMIZATION

- Enable the Enterprise to "interoperate" with other Enterprises outside of its jurisdictional control.
  - Enterprise Design Objective: FEDERATED ARCHITECTURE

Complete Set of Samples

For the complete set of sample Primitive Models go to:
www.sybase.com/zachman

David Dichmann of Sybase PowerDesigner has mapped the metamodel of PowerDesigner against the metamodel for

ENTERPRISE ARCHITECTURE
(The Zachman Framework)

All of my sample models are resident in the tool and can be seen in the Zachman Plug-in for PowerDesigner demo.

Complete Set of Samples

David Dichmann demonstrated at the Enterprise Architecture Conference in London in May 2010 that Sybase PowerDesigner behaves as prescribed by the Framework schema including:

- Creating pure Primitive Models Creating Composite implementations from components of Primitives
- Maintaining horizontal integrative relationships across the Rows and vertical transformational relationships down the Columns
- Maintaining traceability relationships from instances in Row 6 to each Cell in each Column
- Plus various applications including impact analysis, configuration management, versioning, reverse engineering, etc.

I hope this is merely the first of ALL modeling tools to support Enterprise Architecture because until we, the information community, can agree, like the older disciplines of Architecture and Construction and Engineering and Manufacturing as to what constitutes standard descriptive representations of complex objects, we are relegated to building systems, more of the same IT "legacies" ... NOT ENTERPRISES.