Enterprise Architecture: The Issue of the Century
Architecture... what is it?
Some people think this is Architecture:

That is a common MISCONCEPTION
(Note: This same misconception about Enterprises is what leads people to misconstrue Enterprise Architecture as being big, monolithic, static, inflexible and unachievable and... it takes too long and costs too much.)
"Architecture" IS the set of descriptive representations relevant for describing a complex object (actually, any object) such that an instance of the object can be created and such that the descriptive representations serve as the baseline for changing an object instance (assuming that the descriptive representations are maintained consistent with the instantiation).
<table>
<thead>
<tr>
<th>What</th>
<th>How</th>
<th>Where</th>
<th>Who</th>
<th>When</th>
<th>Why</th>
</tr>
</thead>
<tbody>
<tr>
<td>Executive Perspective</td>
<td>Inventory Identification</td>
<td>Distribution Identification</td>
<td>Responsibilities Identification</td>
<td>Timing Identification</td>
<td>Motivation Identification</td>
</tr>
<tr>
<td>Architect Perspective</td>
<td>Inventory Orientation</td>
<td>Business Location</td>
<td>Business Connection</td>
<td>Business Process</td>
<td>Business Product</td>
</tr>
<tr>
<td>Engineer Perspective</td>
<td>Process Orientation</td>
<td>System System</td>
<td>System System</td>
<td>System System</td>
<td>System System</td>
</tr>
<tr>
<td>Technician Perspective</td>
<td>Distribution Configuration</td>
<td>Tool Location</td>
<td>Tool Connection</td>
<td>Tool Location</td>
<td>Tool Connection</td>
</tr>
<tr>
<td>The Enterprise</td>
<td>Inventory Sets</td>
<td>Process Flows</td>
<td>Distribution Networks</td>
<td>Responsibility Assignments</td>
<td>Timing Cycles</td>
</tr>
</tbody>
</table>

**Bills of Material**

**Functional Specs**

**Drawings**

**Operating Instructions**

**Timing Diagrams**

**Design Objectives**

**Scope Contexts**

**System Logic**

**Technology Physics**

**Tool Components**

© 1990-2011 John A. Zachman, Zachman International®
Engineering Versus Manufacturing

Engineering View
One Variable - Total Product

Bills of Material
Functional Specs
Drawings
Operating Instructions
Timing Diagrams
Design Objectives

© 1990-2011 John A. Zachman, Zachman International®
**Engineering vs Manufacturing**

Engineering work requires

single-variable, \textit{(Synthesis)}

ontologically-defined descriptions

of the \textit{whole} of the object.

(Primitive)

(This is RADICALLY different)

IN CONTRAST

Manufacturing work requires

multi-variable, \textit{(Analysis)}

holistic descriptions

of \textit{parts} of the object.

(Composite)

(This is STANDARD practice)
The Framework for **Anything** Architecture

<table>
<thead>
<tr>
<th>What</th>
<th>How</th>
<th>Where</th>
<th>Who</th>
<th>When</th>
<th>Why</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope (Boundaries)</td>
<td>Requirements (Concepts)</td>
<td>Design (Logic)</td>
<td>Plan (Physics)</td>
<td>Part (Configuration)</td>
<td>Design Objectives</td>
</tr>
</tbody>
</table>

- Bills of Material
- Functional Specs
- Drawings
- Operating Instructions
- Timing diagrams

© 2015 John A. Zachman, Zachman International®
The Framework for Enterprise Architecture

Scope Lists (Contexts)

Business Models (Concepts)

Systems Models (Logic)

Technology Models (Physics)

Tool Models (Configuration)
The Zachman Framework for Enterprise Architecture™

**The Enterprise Ontology™**

Version 3.0

© 1967-2011 John A. Zachman, all rights reserved. Zachman® and Zachman International® are registered trademarks of John A. Zachman

© 1990-2011 John A. Zachman, Zachman International®
Framework Graphic


(For a publication release of the Framework Graphic send requests to the Contact Us link on zachman.com)

You may be interested in several articles by John A. Zachman at Zachman.com

“Architecture Is Architecture Is Architecture”
“John Zachman’s Concise Definition of the Zachman Framework”

“The Zachman Framework Evolution” by John P. Zachman
I simply put Enterprise names on the same descriptive representations relevant for describing anything.

Why would anyone think that the descriptions of an Enterprise are going to be any different from the descriptions of anything else humanity has ever described?

I don't think Enterprise Architecture is arbitrary ... and it is not negotiable. My opinion is, we ought to accept the definitions of Architecture that the older disciplines of Architecture and Construction, Engineering and Manufacturing have established and focus our energy on learning how to use them to actually engineer Enterprises.
**The Zachman Framework for Enterprise Architecture**

**The Enterprise Ontology**

**Depth and Width**

**Enterprise Width**
- Sliver
- Half Ent
- Ent.-Wide

**Level of Detail**
- High
- Medium
- Low

**Classification Names**
- What
- How
- Where
- Who
- When
- Why

**Composite Integrations**
- Inventory Identification
- Process Identification
- Distribution Identification
- Responsibility Identification
- Timing Identification
- Motivation Identification

**Model Names**
- Excruciating
- Medium
- High

**Scope Contexts**
- Scope Identification (Why)

**System Logic**
- System Identification (Model)

**Technology Physics**
- Technology Identification (Model)

**Tool Components**
- Tool Identification (Model)

**Operations**
- Operations Identification
- Operations Configuration
- Operations Instanciation
- Operations Flow
- Operations Networks
- Operations Scheduling
- Operations Costs

**Instances**
- Business Concepts
- System Logic
- Technology Physics

**Responsibility**
- Responsibility Identification
- Responsibility Configuration
- Responsibility Instanciation
- Responsibility Flows
- Responsibility Networks
- Responsibility Alignments
- Responsibility Intention

**Timing**
- Timing Identification
- Timing Configuration
- Timing Instanciation
- Timing Intervals
- Timing Operations
- Timing Cycles

**Motivation**
- Motivation Identification
- Motivation Configuration
- Motivation Instanciation
- Motivation Operations
- Motivation Intention

**Inventory**
- Inventory Identification
- Inventory Configuration
- Inventory Instanciation
- Inventory Sets

**Process**
- Process Identification
- Process Configuration
- Process Instanciation
- Process Flows

**Distribution**
- Distribution Identification
- Distribution Configuration
- Distribution Instanciation
- Distribution Networks

**Responsibility**
- Responsibility Identification
- Responsibility Configuration
- Responsibility Instanciation
- Responsibility Assignments

**Timing**
- Timing Identification
- Timing Configuration
- Timing Instanciation
- Timing Intervals
- Timing Operations
- Timing Cycles

**Motivation**
- Motivation Identification
- Motivation Configuration
- Motivation Instanciation
- Motivation Operations
- Motivation Intention

**Enterprise**
- The Enterprise
- Enterprise Perspectives
- Enterprise Names

**Audience Perspectives**
- Executive Perspective
- Business Mgmt Perspective
- Architect Perspective
- Engineer Perspective
- Technician Perspective
- Enterprise Perspective

© 1990-2014 John A. Zachman, Zachman International®
Ontology

The Zachman Framework™ schema technically is an ontology - a theory of the existence of a structured set of essential components of an object for which explicit expression is necessary (is mandatory?) for designing, operating and changing the object (the object being an Enterprise, a department, a value chain, a "sliver," a solution, a project, an airplane, a building, a bathtub or whatever or whatever).

A Framework is a STRUCTURE.
(A Structure DEFINES something.)

Methodology

A Methodology is a PROCESS.
(A Process TRANSFORMS something.)

A Structure IS NOT A Process
A Process IS NOT a Structure.
An Ontology is the classification of the total set of "Primitive" (elemental) components that exist and that are relevant to the existence of an object.

A Methodology produces "Composite" (compound) implementations of the Primitives.

Primitives (elements) are timeless.

Composites (compounds) are temporal.
Ontology

Elements are Timeless

Until an ontology exists, nothing is repeatable, nothing is predictable. There is no DISCIPLINE.
A Process TRANSFORMS something.

This is a Process:

Add Bleach to an Alkali and it is transformed into Saltwater.

Compounds are Temporal

This is NOT an Ontology.
Process (Methodology)

Add Bleach to an Alkali and it is transformed into Saltwater.

HCl + NaOH $\rightarrow$ NaCl + H₂O

Compounds

- Salt: NaCl
- Aspirin: C₉H₈O₄
- Vicodin: C₁₈H₂₁NO₃
- Naproxen: C₁₄H₁₄O₃
- Ibuprophen: C₁₃H₁₈O₂
- Viagra: C₂₂H₃₀N₆O₄S
- Sulphuric Acid: H₂SO₄
- Water: H₂O

etc., etc., etc.

Compounds are Temporal
“Primitives” are Timeless.
Until an ontology exists, nothing is repeatable, nothing is predictable. There is no DISCIPLINE.
Process
(methodology)
Composites
(compounds)

COBOL Programs  COTS
Objects          Technology Architecture
BPMN Models      Big Data
Swimlanes        Missions/Visions
Business Architecture  Agile Code
Capabilities     Business Processes
Mobility         DoDAF Models
Applications     Balanced Scorecard
Data Models      Clouds
Security Architecture  I.B. Watson
Services         TOGAF Artifacts

Etc., etc., etc.

Compounds are Temporal

© 2012 John A. Zachman, Zachman International®
ALCHEMY - A Practice

This is a Methodology WITHOUT an Ontology

A Process with no ontological structure is ad hoc, fixed and dependent on practitioner skills.

This is NOT a science.

It is ALCHEMY, a "practice."
ONTOLOGY vs PROCESS
ENGINEERING vs MANUFACTURING
ARCHITECTURE vs IMPLEMENTATION

It is NOT
"EITHER/OR"

It is “AND”

The question is, “how did you get your
Composite, ‘Manufacturing’, Implementation?”

Did you reuse components of
Primitive, Ontological, Engineering constructs?

OR

Did you just build (Manufacture) the Composite
ad hoc to some problem or some “system requirement”?
Conclusions
1965 Systems Problems

1. Didn't meet Requirements. (not "aligned")
2. The data was no good:
   - Not consistent from system to system.
   - Not accurate.
   - Not accessible.
   - Too late.
3. Couldn't change the system. (Inflexible)
4. Couldn't change the technology. (Not adaptable)
5. Couldn't change the business. (Couldn't change the system or the technology so couldn't change business.)
6. Little new development (80% $ for maintenance)
7. Took too long.
8. Cost too much.
9. Always over budget.
10. Always missed schedules.
11. DP budget out of control.
12. Too complicated - can't understand it, can't manage it.

(Adapted from Doug Erickson)
2015 Systems Problems

1. Didn't meet Requirements. (not "aligned")
2. The data was no good:
   - Not consistent from system to system.
   - Not accurate.
   - Not accessible.
   - Too late.
3. Couldn't change the system. (Inflexible)
4. Couldn't change the technology. (Not adaptable)
5. Couldn't change the business. (Couldn't change the system or the technology so couldn't change business.)
6. Little new development (80% $ for maintenance)
7. Took too long.
8. Cost too much.
9. Always over budget.
10. Always missed schedules.
11. IT budget out of control.
12. Too complicated - can't understand it, can't manage it.

(Adapted from Doug Erickson)
It’s Funny...

COBOL didn't fix those problems!
MVS didn't fix those problems!
Virtual Memory didn't fix those problems!

IMS, DB2, Oracle, Sybase, Access, Fortran, PL/1, ADA, C++, Visual Basic, JAVA 2, 360's, 390's, MPP's, DEC VAX's, H200's, Crays, PC's, MAC's, Distributed Processing, didn't fix those problems!

Word, Excel, Powerpoint, Outlook Express, eMAIL, DOS, Windows 95, 98, 2000, NT, ME, XP, Unix, Linux, Object Oriented, COM, DCOM, CORBA, EDI, HTML, XML, UML, the Internet, B2B, B2C, Portals, Browsers didn't fix those problems!

IEF, IEW, ADW, ERWIN, POPKIN, Rational, Casewise, Rochade, Platinum, Design Bank, Data Warehouse, SAP, Baan, Peoplesoft, Oracle Financials, BSP, ISP, EAP, EAI didn't fix those problems!

And, I doubt that Web Services, .Net, Agile Programming, Service Oriented Architecture, Cloud Computing, BigData or I.B.Watson (whoever that is) is going to fix the problems.

IT MAKES ONE WONDER IF THERE ACTUALLY IS A TECHNICAL SOLUTION TO THE PROBLEMS!!!
I'm not saying that there is anything wrong with any of these technologies. In fact, any or all of them may well be very good ...

In fact, you may not be able to solve the Enterprise problem without employing some of these technologies.

However, The Enterprise problem is an ENGINEERING problem, NOT a technical problem.

My perception is that it is going to take actual work, ENGINEERING work, to solve the problems. My plan would be to start building out an inventory of models, PRIMITIVE MODELS, iteratively and incrementally, engineering them for alignment, integration, flexibility, reduced time-to-market, etc., etc.

What would be YOUR plan for solving the problems???
Enterprise Architecture

Bonus Material

John A. Zachman
Zachman International

© 1990-2011 John A. Zachman, Zachman International®
Professional Service Cycle

1. Diagnosis/Analysis of Need
2. Prescription/Recommendation
3. Application/Implementation
4. Evaluation

Manufacturing (Methodology)

Engineering (Ontology)

Roger Greer:
Dean
School of Library and Information Management
University of Southern California
(My notes from a 1991, IBM GUIDE Conference presentation)
Research Lessons

A. It is possible to solve General Management problems very quickly with a small subset of Primitive components (simply Lists and their inter-dependencies short of the complete Primitive Models)

B. Different complex, composite constructs can be created dynamically, virtually cost-free, from the inventory of Primitive Lists for addressing subsequent General Management problems.

C. Many scenarios can be evaluated to test strategy alternatives before making commitments.
Profound Significance

A. It alters the concept of Enterprise Architecture from one of building models to one of solving General Management problems.

B. Proves the validity of the Primitive Model concept: from a finite inventory of Primitive Concepts you can dynamically create a virtually infinite number of Enterprise implementation Composites.

C. Buys the time for “the experts” to build out the complete Enterprise Architecture (Thing-Relationship-Thing) Primitive Models iteratively and incrementally.

D. Builds significant credibility for the Information Technology community.

E. Establishes the basis for an Enterprise Architecture Profession.
Challenge to Enterprise Architects

Reframe the concept of Enterprise Architecture ...

It is not about building models!

It is about solving Enterprise problems while iteratively and incrementally building out the inventory of complete, reusable, Primitive Models that constitute:

Enterprise Architecture.