AGENDA

I. The Zachman Framework
II. A Zachman Framework Story
III. An Illustration of Primitive Modeling Concepts
IV. Methodology for Solving General Management Problems
V. Conclusions
"The next information revolution is well underway. But it is not happening where information scientists, information executives, and the information industry in general are looking for it. It is not a revolution in technology, machinery, techniques, software, or speed. It is a revolution in CONCEPTS."

Peter Drucker. Forbes ASAP, August 24, 1998

(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”
and
“The Zachman Framework Evolution” by John P. Zachman
**Engineering vs Manufacturing**

Engineering work requires

single-variable, (Synthesis)

ontologically-defined descriptions

of the *whole* of the object.

(Primitive)

(This is RADICALLY different)

IN CONTRAST

Manufacturing work requires

multi-variable,

holistic descriptions

of *parts* of the object.

(Analysis)

(Composite)

(This is STANDARD practice)
Engineering Versus Manufacturing

Engineering View
One Variable - Total Product

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

**Engineering View**
One Variable - Total Product

**Manufacturing View**
One Part - Multiple Variables
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.)

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

A Structure IS NOT A Process
A Process IS NOT a Structure.
Ontology vs Methodology

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.
Until an ontology exists, nothing is repeatable, nothing is predictable. There is no DISCIPLINE.
Process
(Methodology)

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

© 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."
"Primitives" are Timeless.
Until an ontology exists, nothing is repeatable, nothing is predictable.
There is no DISCIPLINE.
## Process (methodology) Composites

<table>
<thead>
<tr>
<th>Composite</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COBOL Programs</td>
<td>COTS</td>
</tr>
<tr>
<td>Objects</td>
<td>Technology Architecture</td>
</tr>
<tr>
<td>BPMN Models</td>
<td>Big Data</td>
</tr>
<tr>
<td>Swimlanes</td>
<td>Missions/Visions</td>
</tr>
<tr>
<td>Business Architecture</td>
<td>Agile Code</td>
</tr>
<tr>
<td>Capabilities</td>
<td>Business Processes</td>
</tr>
<tr>
<td>Mobility</td>
<td>DoDAF Models</td>
</tr>
<tr>
<td>Applications</td>
<td>Balanced Scorecard</td>
</tr>
<tr>
<td>Data Models</td>
<td>Clouds</td>
</tr>
<tr>
<td>Security Architecture</td>
<td>I.B. Watson</td>
</tr>
<tr>
<td>Services</td>
<td>TOGAF Artifacts</td>
</tr>
</tbody>
</table>

Etc., etc., etc.

Compounds are Temporal
Zachman Framework

Introduction to Enterprise Architecture

A Zachman Framework Story

John A. Zachman
Zachman International

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

Engineering (Ontology)

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

Manufacturing (Methodology)

Roger Greer:
Dean
School of Library and Information Management
University of Southern California
(My notes from a 1991, IBM GUIDE Conference presentation)
How does my enterprise look like? - get an enterprise xray
How does my enterprise look like?
- get an enterprise x-ray

Remember:
Generated, NOT Built!
The Key

1. Single-variable, precisely unique, relevant (not arbitrary), ontologically-based components.

2. Binary Relationships (only two components at a time).
Exercise - Deconstruct Business Process Model for multiple Targets

1. Which process needs to be automated?
   - Target-driven by change in activities or process flows
2. What roles need to be added or removed for improving the outcome?
   - Target-driven by change in "rules"

Nothing, may be? Not sure?

Only two interrogatives being used instead of six.

Zachman Certified – Enterprise Architect Program
Exercise – Pizza Delivery Process

This process consists of 14 Activities, 5 Data Elements, 5 Roles, 4 Rules, 3 Locations

Pizza Customer
- Hungry for pizza
- Vendor Contacts

Vendor Contacts

Pizza Order
- Order a pizza
- Address

Order a pizza

Order Received
- Place Order
- Calm Customer

Share Options

Share Options

Pay the pizza
- Is temperature > 80 degrees?
- Free Pizza Offer?

Delay in delivery?

Bake the pizza

Deliver the pizza

Receive payment

Pizza Delivered

Pizza Request

Check Customer
- Existing Customer
- Add Customer

Customer

Pizza Profile

Orders

Sales Person

Zachman Certified – Enterprise Architect Program
Instead of building ONE of these
Build “n” of these.

Pick the one you like.
The Key

1. Single-variable, precisely unique, relevant (not arbitrary), ontologically-based components.

2. Binary Relationships (only two components at a time).
Remember! This is a PRIMITIVE (single-variable) Model used for Engineering.

It cannot be used for implementations which require COMPOSITE (multi-variable) Models.

(Some possible COMPOSITE integration relationships may be shown at the periphery of the model. The COMPOSITE implementation "view" would be created by re-using components of other Enterprise-wide, "engineered" PRIMITIVES.

<table>
<thead>
<tr>
<th>What</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Column 1)</td>
</tr>
<tr>
<td>Inventory Identification</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Row 1</th>
<th>Executive Perspective</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Note:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Air Transportation Case</td>
</tr>
<tr>
<td>Inventories (Entities)</td>
</tr>
<tr>
<td>Countable Things (Nouns)</td>
</tr>
<tr>
<td>(Likely have serial numbers)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A List - Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level of Detail = High</td>
</tr>
<tr>
<td>Abstract (no instances)</td>
</tr>
<tr>
<td>As simple as Possible</td>
</tr>
<tr>
<td>No Recurring Concepts</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scope Contexts</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Composites</th>
</tr>
</thead>
<tbody>
<tr>
<td>There can be composite relationships with any or all other Row 1 Cells and with the Cell below and Instances in Row 6.</td>
</tr>
</tbody>
</table>

Row 6 Instances AS IS may or may not have anything to do with Owner’s, Designer’s, Builders perceptions until those are made explicit and transformed to Row 6. Inventory Sets

<table>
<thead>
<tr>
<th>Note:</th>
</tr>
</thead>
<tbody>
<tr>
<td>This sample model is meant to illustrate the form of the expected Primitive, not necessarily the content.</td>
</tr>
</tbody>
</table>

© 2010 John A. Zachman, Zachman International
Remember! This is a PRIMITIVE (single-variable) Model used for Engineering.
It cannot be used for implementations which require COMPOSITE (multi-variable) Models.
(Some possible COMPOSITE integration relationships may be shown at the periphery of the model. The COMPOSITE implementation “view” would be created by re-using components of other Enterprise-wide, "engineered" PRIMITIVES.

How
(Column 2)

Process Identification

Row 1
Executive
Perspective

Note:
Air Transportation Case
Processes (Transformations)
(Transitive Verb-Object)
A List - Scope
Level of Detail = High
Abstract (no instances)
As simple as possible
No Recurring Concepts

Scope
Contexts

There can be composite relationships with any or all other Row 1 Cells and with the Cell below and Instances in Row 6.

Row 6 Instances AS IS may or may not have anything to do with Owner’s, Designer’s, Builders perceptions until those are made explicit and transformed to Row 6.

Process Flows

© 2010 John A. Zachman, Zachman International
<table>
<thead>
<tr>
<th>Inventory Sets</th>
<th>Process Flows</th>
<th>Distr. Networks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airplanes</td>
<td>Acquire Routes</td>
<td>Airplane Network</td>
</tr>
<tr>
<td>Airplane Types</td>
<td>Schedule Flights</td>
<td>Parts Distr. Net.</td>
</tr>
<tr>
<td>Airports</td>
<td>Reserve Seats</td>
<td>Communications</td>
</tr>
<tr>
<td>Gates</td>
<td>Train Employees</td>
<td>Freight Net.</td>
</tr>
<tr>
<td>Passengers</td>
<td>Fly Airplanes</td>
<td>Airport Network</td>
</tr>
<tr>
<td>Shareholders</td>
<td>Schedule Crews</td>
<td>(Runways, etc.)</td>
</tr>
<tr>
<td>Local Carriers</td>
<td>Repair Facilities</td>
<td>Regulatory Net.</td>
</tr>
<tr>
<td>Seats</td>
<td>Develop Markets</td>
<td>Passenger Net.</td>
</tr>
<tr>
<td>Seats</td>
<td>Load Airplanes</td>
<td>Catering Net.</td>
</tr>
<tr>
<td>Routes</td>
<td>Release Flights</td>
<td>etc.</td>
</tr>
<tr>
<td>Employees</td>
<td>Develop Flt. Plans</td>
<td>etc.</td>
</tr>
<tr>
<td>Vehicles</td>
<td>Schedule Maint.</td>
<td>etc.</td>
</tr>
<tr>
<td>Flights</td>
<td>etc.</td>
<td>etc.</td>
</tr>
<tr>
<td>etc.</td>
<td>etc.</td>
<td>etc.</td>
</tr>
<tr>
<td>Respon Assmts</td>
<td>Timing Cycles</td>
<td>Motive Intent.</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>Pilots</td>
<td>Flight Cycle</td>
<td>Equip. Utilization</td>
</tr>
<tr>
<td>Co-pilots</td>
<td>Customer Cycle</td>
<td>New Markets</td>
</tr>
<tr>
<td>Engineers</td>
<td>Maintenance Cyc.</td>
<td>Revenue Growth</td>
</tr>
<tr>
<td>Flt. Attend.</td>
<td>Telephone Wait C.</td>
<td>Exp. Reduction</td>
</tr>
<tr>
<td>Reservations</td>
<td>Airplane Turnaround</td>
<td>Cust Convenience</td>
</tr>
<tr>
<td>Aircraft Maint.</td>
<td>De-Icing Cycle</td>
<td>Cust. Satisfaction</td>
</tr>
<tr>
<td>Flight Scheduling</td>
<td>Air Traffic Cntl. C.</td>
<td>Labor Contracts</td>
</tr>
<tr>
<td>Airport Ops. Mgt</td>
<td>Tarmac Cycle</td>
<td>Regulatory Comp</td>
</tr>
<tr>
<td>Customer Service</td>
<td>Airplane Cycle</td>
<td>New Capital</td>
</tr>
<tr>
<td>Marketing</td>
<td>Baggage Handling C.</td>
<td>Load Factor</td>
</tr>
<tr>
<td>Sales</td>
<td>Security (TSA)Cycle</td>
<td>Route Optimize</td>
</tr>
<tr>
<td>Flight Dispatch</td>
<td>Planning Cycle</td>
<td>Flight Expansion</td>
</tr>
<tr>
<td>Accounting</td>
<td>Budget Cycle</td>
<td>Acquisition</td>
</tr>
<tr>
<td>etc.</td>
<td>etc.</td>
<td>etc.</td>
</tr>
</tbody>
</table>
The Key

1. Single-variable, precisely unique, relevant (not arbitrary), ontologically-based components.

2. Binary Relationships (only two components at a time).
Introduction to Enterprise Architecture

Methodology for Solving General Management Problems

John A. Zachman
Zachman International

© 1990-2011 John A. Zachman, Zachman International®
1. Select General Management Problem.


3. Define “binary” (only two at a time) dependencies (horizontal and vertical) between Primitive (Single-variable) Components.

4. Create Composite (Multi-variable) “snapshot” of problem area and diagnose.

5. Pose new Composite (Multi-variable) scenarios (change Lists and/or change dependencies) and re-compose multiple targets.

6. Add time and cost to Primitive (Single-variable) Components and simulate alternatives, perform risk analysis, identify resource rqs, etc.

(CEO picks solution, assigns responsibilities for “quick-fix”, identifies subsequent CEO problem. Then re-iterate Steps 1 - 6.)
7. Pick several Cells in different Columns in some Row and assign modeling experts to build out complete (thing-relationship-thing) Primitive (**Single-variable**) Models, verifying horizontal alignment.

8. Create complete Composite (**Multi-variable**) integration for Row ensuring horizontal alignment. (Does each Cell have all components required for reuse in adjoining Cells for creating Composites.)

9. Have Columnar modeling experts transform Primitive (**Single-variable**) Models to next Cell below, ensuring vertical “alignment” and iterate until all Cells in the Column are transformed and vertically aligned.

10. Transform Row 5 Primitives (**Single-variables**) to Row 6 implementations (**Multi-variables**) using either machines (automated) or people (manual).

11. Add Primitive Components by Cell from next problem and reiterate Steps 7 - 11.
12. Institutionalize this process and govern Architecture as follows:
   a. prohibit redundancy except where explicitly controlled.
   b. maintain horizontal and vertical alignment
   c. use Primitive (**Single-variable**) Model inventory as base for managing **ENTERPRISE** changes.
   d. ensure EVERY new implementation Composite reuses components of Primitive models and migrate legacy to Architected Enterprise. (See Workshop “Migration Strategy”.)
13. Acquire subject matter expertise for building additional Primitive (**Single-variable**) Models to be added to the Enterprise Architecture capability inventory.

**Key:** Single-Variable, PRIMITIVE Models, and Binary Relationships

For details see Level 2 Zachman Certification at [www.Zachman.com](http://www.Zachman.com)

Note: This is the same process, somewhat abbreviated, and executed by students in the 4 day Zachman Level 1 Certification Workshop.
Conclusions

John A. Zachman
Zachman International
**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.

© 2012 John A. Zachman, Zachman International®
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.
"Although social systems are more complex than physical systems, they belong to the same class of high-order, non-linear, feedback systems as do physical systems.

People do not accept the idea that families, corporations, and governments belong to the same class of dynamic structures as do chemical refineries and autopilots for aircraft.

"Organizations built by committee and intuition perform no better than would an airplane built by the same methods ... As in a bad airplane design, which no pilot can fly successfully, such badly designed corporations lie beyond the ability of real-life managers.

"I anticipate future management schools devoted to 'enterprise design'. ... A fundamental difference exists between an enterprise operator and an enterprise designer. A manager runs an organization, just as a pilot runs an airplane. Success of a pilot depends on an aircraft designer who created a successful airplane. ...who designed the corporation that a manager runs?"

"Designing the Future" by Jay W. Forrester 12/15/98
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???