

Framework for Enterprise Architecture

Enterprise Physics 101



***John A. Zachman
Zachman International
2222 Foothill Blvd. Suite 337
La Canada, Ca. 91011
818-244-3763***

Preface

This seminar is NOT about increasing the stock price by the close of market, Friday afternoon.

It IS about the laws of nature that determine the success of an Enterprise ... particularly, continuing success in the turbulent times of the Information Age.

It is a presentation on Physics ...

Enterprise Physics.

The Information Age

"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

"Future Shock" (1970) - The rate of change.

"The Third Wave" (1980) - The structure of change.

"Powershift" (1990) - The culture of change.

Alvin Toffler

"We are living in an extraordinary moment in history. Historians will look back on our times, the 40-year time span between 1980 and 2020, and classify it among the handful of historic moments when humans reorganized their entire civilization around a new tool, a new idea."

Peter Leyden. Minneapolis Star Tribune. June 4, 1995

"On the Edge of the Digital Age: The Historic Moment"

The Challenge

What is your strategy for addressing:

Orders of magnitude increases in complexity,
and

Orders of magnitude increases in the rate of change?

Seven thousand years of history would suggest the only known strategy for addressing complexity and change is

ARCHITECTURE.

If it gets so complex you can't remember how it works,
you have to write it down ... Architecture.

If you want to change how it works, you start with what
you have written down ... Architecture.

The key to complexity and change: Architecture.

The question is: What is "Architecture,"

Enterprise Architecture?

Agenda

Half Day Agenda

- I. Introduction to Enterprise Architecture
 - A. The Framework for Enterprise Architecture
 - B. Basic Enterprise Physics
- II. Enterprise Engineering Design Objectives
 - A. Alignment, Integration, Flexibility, etc.
 - B. Reducing Time-to-Market
- III. Value Proposition
 - A. Industrial Age
 - B. Information Age
- IV. Frequently Asked Questions
 - A. Cheaper and Faster
- V. Conclusions

Introduction to Enterprise Architecture

The Framework for Enterprise Architecture



Different Perspectives

Buildings

Airplanes

Enterprise

OWNER

Architect's
Drawings

Wk. Bk. Dwn.
Structure

Model of
Business

DESIGNER

Architect's
Plans

Engineering
Design

Model of
Info. Sys.

BUILDER

Contractor's
Plans

Mfg. Eng.
Design

Technlgy
Model

Different Abstractions

WHAT	HOW	WHERE
Material	Function	Location
Bill of Materials	Functional Specs	Drawings
Data Models	Functional Models	Network Models

A Framework

	WHAT	HOW	WHERE
OWNER			
DESIGNER			
BUILDER			




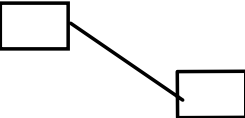
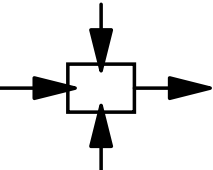
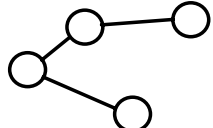
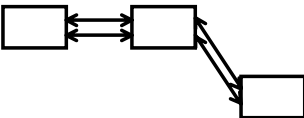
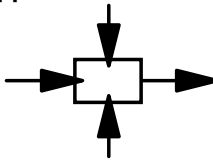
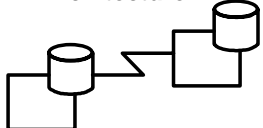
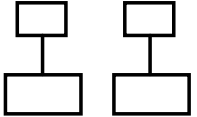
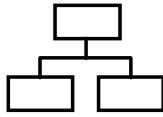
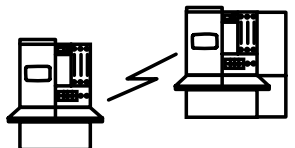



A Framework

	WHAT	HOW	WHERE
SCOPE			
OWNER			
DESIGNER			
BUILDER			
OUT OF CONTEXT			
PRODUCT			

A Framework

	DATA	FUNCTION	NETWORK
SCOPE			
BUSINESS MODEL			
SYSTEM MODEL			
TECH MODEL			
DETAIL RPSNTNS			
SYSTEM			


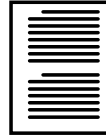

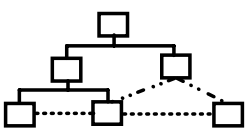
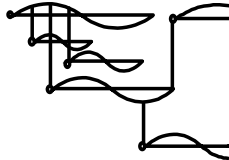
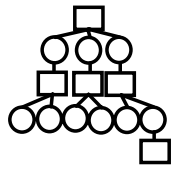
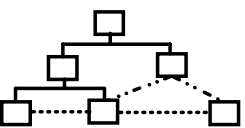
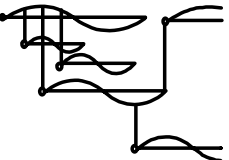
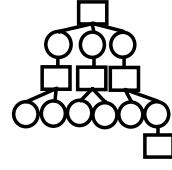
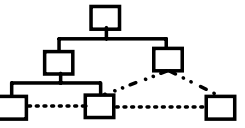
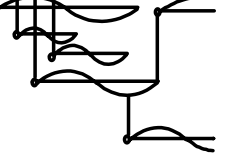
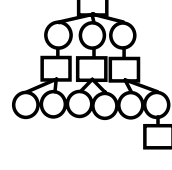



ENTERPRISE ARCHITECTURE - A FRAMEWORK

	DATA <i>What</i>	FUNCTION <i>How</i>	NETWORK <i>Where</i>
SCOPE (CONTEXTUAL) <i>Planner</i>	List of Things Important to the business  ENTITY = Class of Business Thing	List of Processes the Business Performs  Process = Class of Business Process	List of Locations in which the Business Operates  Node = Major Business Location
BUSINESS MODEL (CONCEPTUAL) <i>Owner</i>	e.g. Semantic Model  Ent = Business Entity ReIn = Business Relationship	e.g. Business Process Model  Proc = Bus Process I/O = Bus Resources	e.g. Business Logistics System  Node = Business Location Link = Business Linkage
SYSTEM MODEL (LOGICAL) <i>Designer</i>	e.g. Logical Data Model  Ent = Data Entity ReIn = Data Relationship	e.g. Application Architecture  Proc = Application Function I/O = User Views	e.g. Distributed System Architecture  Node = I/S Function (Processor, Storage, etc) Link = Line Characteristics
TECHNOLOGY MODEL (PHYSICAL) <i>Builder</i>	e.g. Physical Data Model  Ent = Segment/Table/etc. ReIn = Pointer/Key/etc.	e.g. System Design  Proc = Computer Function I/O = Data Elements/Sets	e.g. Technology Architecture  Node = Hardware/Systems Software Link = Line Specifications
DETAILED REPRESENTATIONS (OUT-OF-CONTEXT) <i>Sub-Contractor</i>	e.g. Data Definition  Ent = Field ReIn = Address	e.g. Program  Proc = Language Statement I/O = Control Block	e.g. Network Architecture  Node = Address Link = Protocol
FUNCTIONING ENTERPRISE	e.g. DATA	e.g. FUNCTION	e.g. NETWORK

A Framework

WHO	WHEN	WHY	
			SCOPE
			OWNER
			DESIGNER
			BUILDER
			OUT OF CONTEXT
			PRODUCT

ENTERPRISE ARCHITECTURE THE "OTHER THREE COLUMNS"

PEOPLE	TIME	MOTIVATION	
<p>List of Organizations Important to the Business</p>  <p>People = Major Organization Unit</p>	<p>List of Events/Cycles Significant to the Business</p>  <p>Time = Major Business Event/Cycle</p>	<p>List of Business Goals/Strategies</p>  <p>Ends/Means = Major Business Goal/Strategy</p>	<p>SCOPE (CONTEXTUAL)</p> <p style="text-align: right;"><i>Planner</i></p>
<p>e.g., Work Flow Model</p>  <p>People = Organization Unit Work = Work Product</p>	<p>e.g., Master Schedule</p>  <p>Time = Business Event Cycle = Business Cycle</p>	<p>e.g., Business Plan</p>  <p>End = Business Objective Means = Business Strategy</p>	<p>BUSINESS MODEL (CONCEPTUAL)</p> <p style="text-align: right;"><i>Owner</i></p>
<p>e.g., Human Interface Architecture"</p>  <p>People = Role Work = Deliverable</p>	<p>e.g., Processing Structure</p>  <p>Time = System Event Cycle = Processing Cycle</p>	<p>e.g., Business Rule Model</p>  <p>End = Structural Assertion Means = Action Assertion</p>	<p>SYSTEM MODEL (LOGICAL)</p> <p style="text-align: right;"><i>Designer</i></p>
<p>e.g., Presentation Architecture</p>  <p>People = User Work = Screen Formats</p>	<p>e.g., Control Structure</p>  <p>Time = Execute Cycle = Component Cycle</p>	<p>e.g., Rule Design</p>  <p>End = Condition Means = Action</p>	<p>TECHNOLOGY MODEL (PHYSICAL)</p> <p style="text-align: right;"><i>Builder</i></p>
<p>e.g., Security Architecture</p>  <p>People = Identity Work = Job</p>	<p>e.g., Timing Definition</p>  <p>Time = Interrupt Cycle = Machine Cycle</p>	<p>e.g., Rule Specification</p>  <p>End = Sub-condition Means = Step</p>	<p>DETAILED REPRESENTATIONS (OUT-OF-CONTEXT)</p> <p style="text-align: right;"><i>Sub-Contractor</i></p>
<p>e.g., ORGANIZATION</p>	<p>e.g., SCHEDULE</p>	<p>e.g., STRATEGY</p>	<p>FUNCTIONING ENTERPRISE</p>

The Framework Is a Schema

The Fmwrk is a two-dimensional classification system.

The classification scheme for each axis grew up quite independently from the Framework application.

The classification for each axis is:

- a. Comprehensive
- b. Primitive

Therefore, each cell of the Framework is:

- a. Unique
- b. "Primitive"

and the total set of cells is complete.

The Framework logic is universal, independent of its application.

**The Framework is a "normalized" schema ...
... NOT a matrix.**

That's what makes it a good analytical tool.

Do Not Add Rows or Columns

Do not add Rows or Columns to the Framework!
(You will DE-normalize it!)

Do not add Rows or Columns to the Framework!
(You will DE-normalize it!)

Do not add Rows or Columns to the Framework!
(You will DE-normalize it!)

Do not add Rows or Columns to the Framework!
(You will DE-normalize it!)

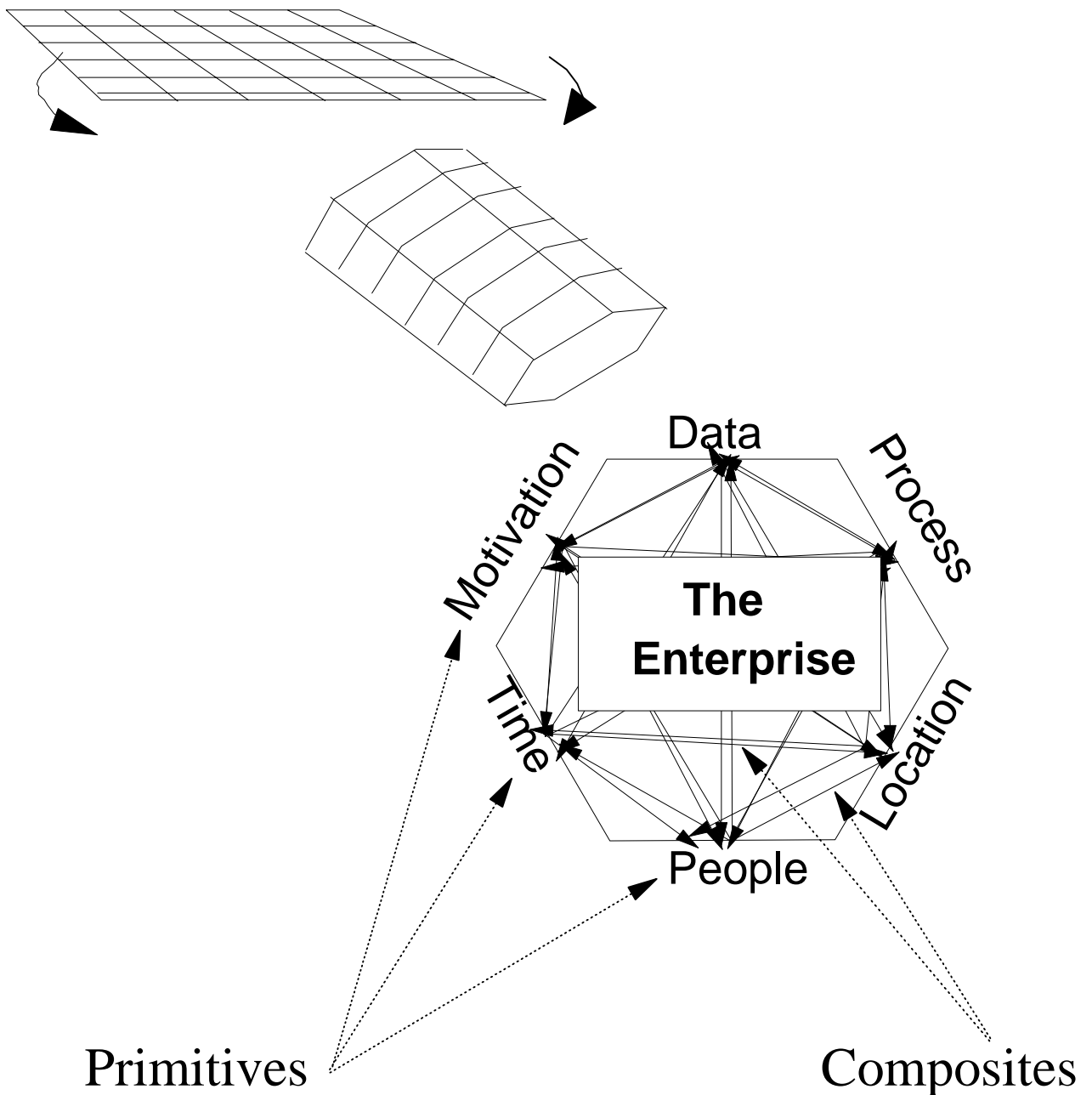
Do not add Rows or Columns to the Framework!
(You will DE-normalize it!)

Do not add Rows or Columns to the Framework!
(You will DE-normalize it!)

Do not add Rows or Columns to the Framework!
(You will DE-normalize it!)

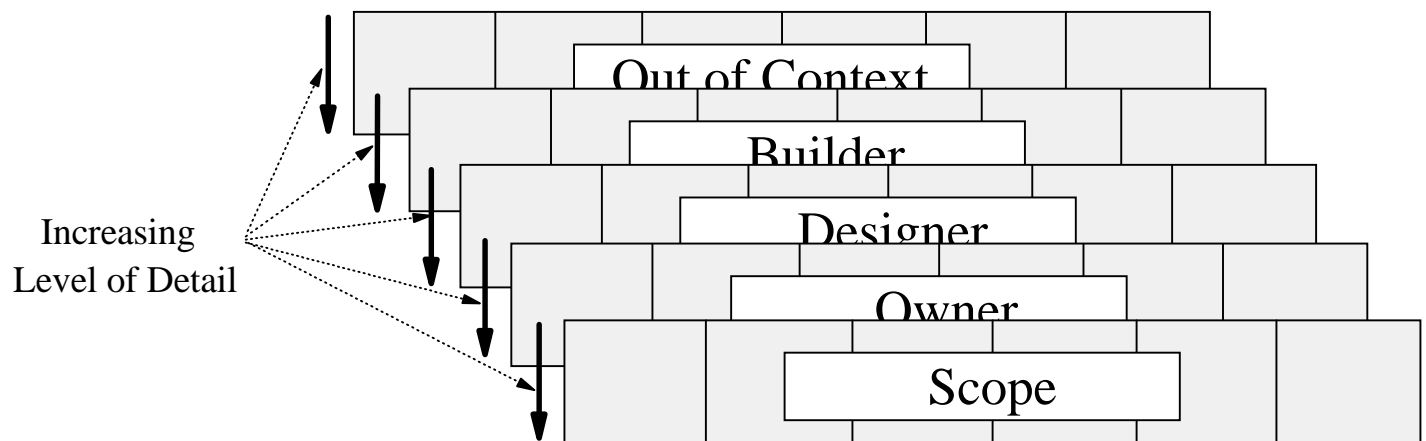
No Order To Columns

A better way to depict the Framework would be as a helix with any one cell related to all other cells in the same Row.



Excruciating Level of Detail

Level of detail is a function of a Cell, NOT a Column.



Less than Excruciating Detail

	DATA	FUNCTION	NETWORK
SCOPE <i>Planner</i>			
BUSINESS MODEL <i>Owner</i>			
SYSTEM MODEL <i>Designer</i>	Re: <i>Any Cell</i>		
TECHNOLOGY MODEL <i>Builder</i>			
COMPONENTS <i>Sub-Contractor</i>			
FUNCTIONING ENTERPRISE			

Less than Enterprise Scope

	DATA	FUNCTION	NETWORK
SCOPE <i>Planner</i>			
BUSINESS MODEL <i>Owner</i>			
SYSTEM MODEL <i>Designer</i>	Re: <i>Any Cell</i>		
TECHNOLOGY MODEL <i>Builder</i>			
COMPONENTS <i>Sub-Contractor</i>			
FUNCTIONING ENTERPRISE			

Basic Physics

1. If the Enterprise exists, ALL of the descriptive representations (models) exist ... by definition.

If they are not explicit, they are implicit (that is, you are making assumptions about them.)

2. The system IS the Enterprise

Manual systems employ pencils, paper, file cabinets. Automated systems employ stored programming devices and electronic media.

3. High level descriptions (models) are good for planning, scoping, bounding, segmenting. (High level descriptions are NO good for implementation.)

4. Narrow-in-scope descriptions are quick. (Narrow in scope descriptions result in "stove pipes.")

End State Vision

	DATA	FUNCTION	NETWORK
SCOPE <i>Planner</i>			
ENTERPRISE MODEL <i>Owner</i>			
SYSTEM MODEL <i>Designer</i>			
TECHNOLOGY MODEL <i>Builder</i>			
COMPONENTS <i>Sub-Contractor</i>			
FUNCTIONING ENTERPRISE			

**Enterprise - Wide
Horizontal
Vertical
Integrated
Architecture
at
Excruciating
Level of Detail**

The Future

- A. Build Models
- B. Store Models
- C. Manage (Enforce) Models
- D. Change Models

It is not adequate merely to produce running code.
(That is an Industrial Age idea.)

The long term Enterprise value
lies in Enterprise "Engineering,"
i.e. in the MODELS THEMSELVES!
(This is an Information Age idea!)

Enterprise Architecture

**Enterprise Engineering
Design Objectives**



Engineering Design Objectives

Alignment
Integration
Flexibility
Interoperability
Reduced Time-to-Market
Quality
Seamlessness
Adaptability
User-Friendliness
Usability
Reusability
...
etc.

Note: All of these desirable attributes of systems (i.e. of the Enterprise) PRESUME the existence of Architecture (i.e. models) and further, that the models were designed with those attributes specifically in mind.

If no Architecture (models) exist, these attributes are simply "platitudes."

Alignment

"Alignment" means ...

... you want the implemented systems (Row 6)
to align with the Enterprise intent (Row 1/2 Models).

Similar Manufacturing concept: "Quality"

"Alignment" would be the Enterprise equivalent of

"Total Quality Management."

Alignment

If you REALLY want the implemented systems (Row 6) (Enterprise) to be aligned with management's intent (Row 1/2), here is how you do it:

First, build Row 1 models.

Next, build Row 2 models.

Next, build Row 3 models.

Next, build Row 4 models.

Next, build Row 5 models,

ensuring the intent of each Row is successfully represented in the succeeding Row.

Next, compile and run.

Then, the implemented Enterprise will align with the Business Purpose.

It looks to me like any other alternative would require either a suspension of the laws of nature or, an inordinate amount of pure luck.

Alignment

Next question:

How do you intend to **KEEP** the implementation (Row 6) "aligned" with Management's intent (Rows 1/2) when everything is changing like gangbusters????:

Management's Intent (Rows 1/2)

The Design state of the art (Row 3)

The Technology constraints (Row 4) and

The Technology products (Row 5)

and

The Enterprise Data (Column 1)

The Processes (Column 2)

The Distribution Network (Column 3)

The Organization (Column 4)

The Dynamics (Column 5) and

The Business Strategy (Column 6)

Configuration Management

(You have to retain and maintain the models.)

End State Vision

	DATA	FUNCTION	NETWORK
SCOPE <i>Planner</i>			
BUSINESS MODEL <i>Owner</i>			
SYSTEM MODEL <i>Designer</i>			
TECHNOLOGY MODEL <i>Builder</i>			
COMPONENTS <i>Sub-Contractor</i>			
FUNCTIONING ENTERPRISE			

**Enterprise - Wide
Horizontal
Vertical
Integrated
Architecture
at
Excruciating
Level of Detail**

Integration

"Integration" means ...

... you want no discontinuity between the various related concepts within the Enterprise. For example:

- Scope Integration - no discontinuity within any one kind of model across the scope of the Enterprise. (Internal sharing, the antithesis of "stovepipes" ... efficiency.)
- Horizontal Integration - no discontinuity across the different kinds of related models from Column to Column. (Horizontal sharing - coordinated operations and change ... effectiveness.)
- Vertical Integration - no discontinuity between the various Rows, the Owner's, Designer's and Builder's constraints. (The end result is consistent with the requirements ... quality. This is "alignment.")

Three Definitions of "Integration"

	DATA	FUNCTION	NETWORK
SCOPE <i>Planner</i>			↑
BUSINESS MODEL <i>Owner</i>	←	Horizontal Integration (Any Row)	→
SYSTEM MODEL <i>Designer</i>			↓
TECHNOLOGY MODEL <i>Builder</i>	←	Scope Integration (Any Cell)	→
COMPONENTS <i>Sub-Contractor</i>			↓
FUNCTIONING ENTERPRISE			↓

Integration

"Integration" implies sharing, NORMALIZATION ...

You have any one concept defined only one time ...

Any time you need some one concept, you "reuse" what has already been defined (you DON'T re-create that same concept causing redundancies and potential inconsistencies, discontinuities, etc.)

Integration (normalization) implies "primitive" models. How would you evaluate a composite model for completion, wholeness, normalization? It is difficult enough analyzing single variable ("primitive") models.

Integration is the Enterprise equivalent of
Standard Interchangeable Parts.

Similar Concepts:

Seamlessness (Col. 2),
Interoperability (Col. 3),
Reusability (implied).

Reusability

Reusability is related to Integration ...
(Standard Interchangeable Parts)

Reusability is so easy to understand in physical objects. e.g. If you want to screw a carburetor on more than one (different kind of) engine, you have to engineer it to fit on whatever kinds of engines you want to screw it on **BEFORE** you get it (them) manufactured.

That is, if you want to engineer something to be reused, you have to know the total set of possibilities in order to engineer characteristics that will accommodate multiple uses.

The key to Reuse (and Integration) is Enterprise-wide, normalized, **PRIMITIVE** models.

If you don't know the total set and it turns out that whatever you manufactured can be reused, it is only by accident, a miracle ... or else whoever wanted to use whatever it is, changed their use to fit whatever you manufactured.

Post - Integration

Interfacing (Column 1 word.)

You can do some after-the-fact "transformations" as long as they are cosmetic, that is, relate to format, or name, or instances only. You can't add something that was not there to begin with nor change structure.

("Middleware": Column 3 word for the same idea.)

Interfacing (generic word)

Hard-binding two independent variables -
loss of flexibility.

Maintenance problem -

number of connections between "n" points is

$$\frac{n^2 - n}{2}$$

It is a "point-in-time" solution -

good as long as nothing changes.

Interoperability

Interoperability: If you REALLY want to run the same thing on different systems ...

- If the "systems" are all manufactured by the same manufacturer and are running the same operating system, interoperability is absolute.
- As soon as you introduce a different manufacturer and/or operating system, interoperability degrades.
- "Heterogeneous interoperability" is an oxymoron.
- Heterogeneous means you optimize the parts at the expense of the whole.
- Interoperability means you optimize the whole at the expense of the parts.
- Interfacing (gateways, middleware, etc.) mapping (many to many) independent variables increases maintenance liability to "m times n" and inhibits change.
- "Open Systems" is a great idea ...

but it is not magic!

(See "Open Systems.")

Open Systems??

"Open Systems" says:

Somebody (vendor or user) is going to write code multiple times such that a given design (set of row 5 models) will run:
under more than one operating system,
on more than one computer,
connected to more than one kind of line,
under more than one network operating system,
and using more than one data base management system.

Or else ... the user will have to build the logical models (row 3 models) and somebody (user or vendor) will write the code to dynamically (at run time) transform the row 3 models to row 4/5.

Or else ... everybody has to use a standard computer, operating system, line protocol, network operating system and data base management system.

Or else, the vendors will have to build their hardware, systems software, networks and data bases to standard specifications.

Or else ... somebody (vendor or user) will have to build and *maintain* a bunch of custom, gateway/conversion packages.

(In the last case, the question is going to become, "how many computers, operating systems, line protocols, network operating systems and database management systems can one organization (or vendor) afford to support in a changing environment?")

Integration In Summary

Integration: If you REALLY want "Integration", that is if you do not want any discontinuity in the meaning, in connectivity, in business rules ... if you want reusability, interoperability, seamlessness, standard interchangeable parts ...

... it has to be engineered into the product at whatever the desired scope of integration is at the outset, that is,
BEFORE
anything is implemented.

How do you achieve integration?

- ... not by accident,
 - ... not with hardware or software,
 - ... not by wishful thinking, or by announcement,
 - ... not after-the-fact,
- ONLY by ENGINEERING!!**

End State Vision

	DATA	FUNCTION	NETWORK
SCOPE <i>Planner</i>			
BUSINESS MODEL <i>Owner</i>			
SYSTEM MODEL <i>Designer</i>			
TECHNOLOGY MODEL <i>Builder</i>			
COMPONENTS <i>Sub-Contractor</i>			
FUNCTIONING ENTERPRISE			

**Enterprise - Wide
Horizontal
Vertical
Integrated
Architecture
at
Excruciating
Level of Detail**

Flexibility

Flexibility means ...

... you want things implemented such that they can be changed in minimum time with minimum disruption, at minimum cost.

(Similar Concept: Adaptability)

("Flexibility" is closely related
to "Change Management"
and somewhat related to "Integration.")

Flexibility No. 1

If you REALLY want flexibility, for starters it would be helpful to be working with "standard, interchangeable parts," that is, "integrated" architecture.

That is, it would be helpful to have "normalized" models ... any one concept exists only one time ... so if you need to change it ... you change it once and it is changed for every deployment. You don't re-create it and create redundancies with the inherent inconsistencies and discontinuities and insurmountable maintenance problems.

Next, it would be helpful to have an inventory of all the reusable concepts so you knew where they were and what they were and could find them when you wanted to re-use them. (That is, in a "database.")

I already talked about all of this under the heading:
"Integration."

Flexibility No. 2

Flexibility: If you REALLY want to change things in minimum time, with minimum disruption and cost ...

- Decouple (separate) the independent variables. (e.g. "Data Division" and "Procedure Division")
You potentially can change one thing without having to change everything.
- Every ("*primitive*") cell of the Framework is an independent variable.
- "3 - Schema" as a concept is the mapping of two independent variables to a common, "conceptual schema" reducing the change liability to "m plus n."
- If you want to change the Enterprise with minimum time, disruption and cost, you have to retain and maintain the architectural *primitives*.
(e.g. If you want to dynamically change the Row 4 technology constraints you have to manage the architecture at the Row 3 logical level.)

Managing Change

How do you change anything?

7,000 years of experience with older disciplines suggest that to change anything, you start with the drawings, the functional specs, the bills-of-material ...

How do you change buildings?

You start with the drawings, the functional ...

How do you change airplanes?

You start with the drawings, the functional ...

How do you change that PC on your desk?

You start with the drawings, the functional ...

If you DON'T HAVE the drawings, functional specs ...

You have 3 options:

1. Change by trial and error.

HIGH RISK

2. Re-create the drawings, functional specs ...

TAKES TIME AND COSTS MONEY

3. DON'T CHANGE IT!

Flexibility - Summary

In summary, if you REALLY want flexibility:

1. Normalized concepts - any one concept only exists one time in the Enterprise. Anytime it is changed, it is changed once.
2. "Primitive" models are KEY - separate out the independent variables. You can change one concept without having to scrap and rework the entire implementation. (Each Cell is an independent variable.) Also, you can create an "infinite" number of composites from a finite set of primitives.
3. Repository - retain and maintain the "primitive" models to serve as a baseline for managing change.

Note: If you have no architectural representations, you have no separation of independent variables, no baseline for managing change and therefore, *NO FLEXIBILITY* (all independent variables are implicit and imbedded in a unitary implementation ... the running system.)

End State Vision

	DATA	FUNCTION	NETWORK
SCOPE <i>Planner</i>			
BUSINESS MODEL <i>Owner</i>			
SYSTEM MODEL <i>Designer</i>			
TECHNOLOGY MODEL <i>Builder</i>			
COMPONENTS <i>Sub-Contractor</i>			
FUNCTIONING ENTERPRISE			

**Enterprise - Wide
Horizontal
Vertical
Integrated
Architecture
at
Excruciating
Level of Detail**

Enterprise Architecture

Reducing Time-To-Market



Reducing Time-To-Market

Reducing Time to Market MEANS:

1. If you are making-to-order,
reducing time-to-market means:
Reduce scope - simplify -
proliferate legacy problems.
2. If you are providing-from-stock,
reducing time-to-market means:
Packages - implement AS IS
and change the Enterprise to fit the package.
3. If you are assembling-to-order,
reducing time-to-market means:
an inventory of re-usable assets.
(whether you are making the "assets"
or buying them.)

What do you think has to be in inventory before you get the order?

End State Vision

	DATA	FUNCTION	NETWORK
SCOPE <i>Planner</i>			
BUSINESS MODEL <i>Owner</i>			
SYSTEM MODEL <i>Designer</i>			
TECHNOLOGY MODEL <i>Builder</i>			
COMPONENTS <i>Sub-Contractor</i>			
FUNCTIONING ENTERPRISE			

**Enterprise - Wide
Horizontal
Vertical
Integrated
Architecture
at
Excruciating
Level of Detail**

Enterprise Architecture

**Value Propositions:
Old and New**



Industrial Age (Old)

"Better, Faster, Cheaper"

Computers do it the same way every time

(People make mistakes)

Computers run at machine speeds

(People run at people speeds)

Computers are cheaper

(Labor is more expensive than machines)

"Better, Faster, Cheaper"

"Justify" the acquisition of computers based on

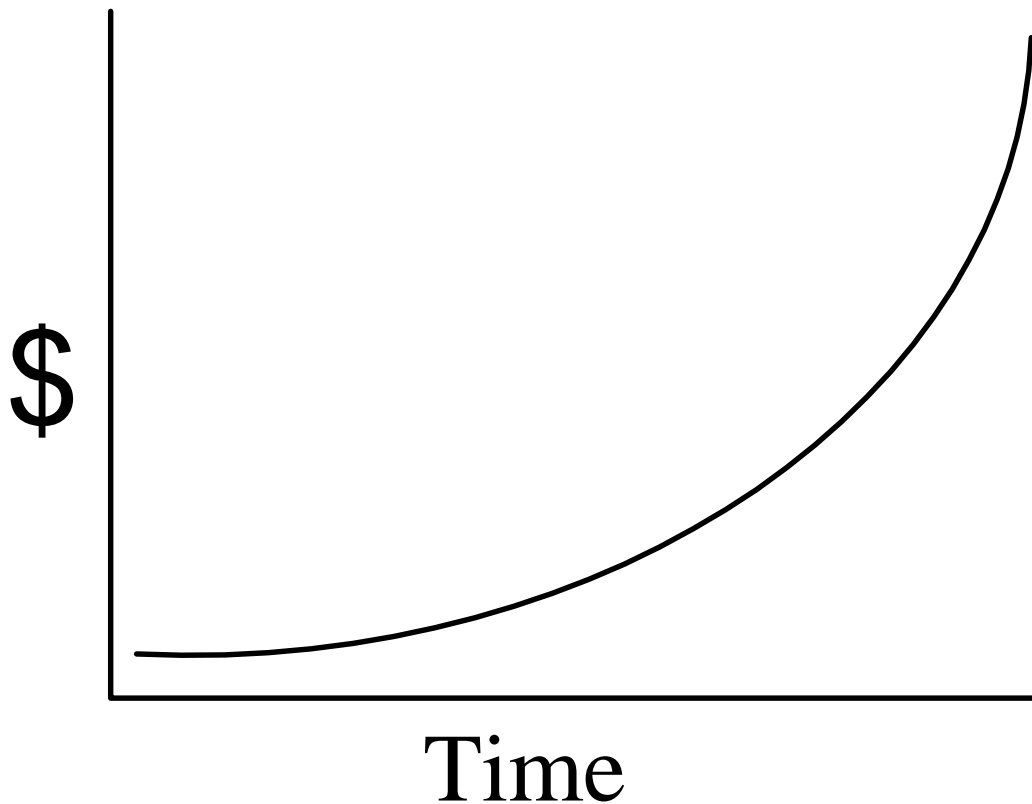
Cost-displacement -

how many people will it (they) replace?

Therefore, Value proposition:

"Cost-Justification"

Short Term Investment



Expense-based, short term oriented, implementation-dominant, "cost-justified," "you start writing the code ... I'll go find out what the users have in mind."

Start manufacturing before you do any engineering.

NO ARCHITECTURE

Quick implementations. Consumables.

Point-in-time solutions. One time use.

"Pay me now or pay me later" - Scrap and rework.

Information Age (New)

Value Proposition for ARCHITECTURE

(Note: You can't "cost-justify" Architecture because it doesn't displace any costs.)

Architecture is an INVESTMENT that enables you to do things you otherwise would be unable to do ... specifically:

Alignment

Integration

Change (Flexibility)

Reduced time-to-market

Value proposition: Asset Inventory

Value Proposition for Architecture

A. Alignment

The implemented enterprise reflects the intent of "the Owners."

(In manufacturing - this equates to "Quality" - "TQM")

B. Integration

The data means the same thing to everyone.

Messages are successfully (and consistently) transmitted from node to node.

Everyone understands the objectives/strategy.

(The enablers of "empowerment.")

(This is standard, interchangeable parts.)

C. Change Management (Flexibility)

Independent variables - baseline for managing change.

Retained, accumulated, Enterprise knowledge .

(Change with minimum time, disruption and cost.)

(This is "effectivity," change management.)

D. I/S Responsiveness

Architecture plus "assemble-to-order" processes.

(This is reduced "time to market" - "Mass Customization.")

Information Value

In addition to:

Alignment

Integration

Change

Reduced Time-to-Market,

the set of models that constitutes

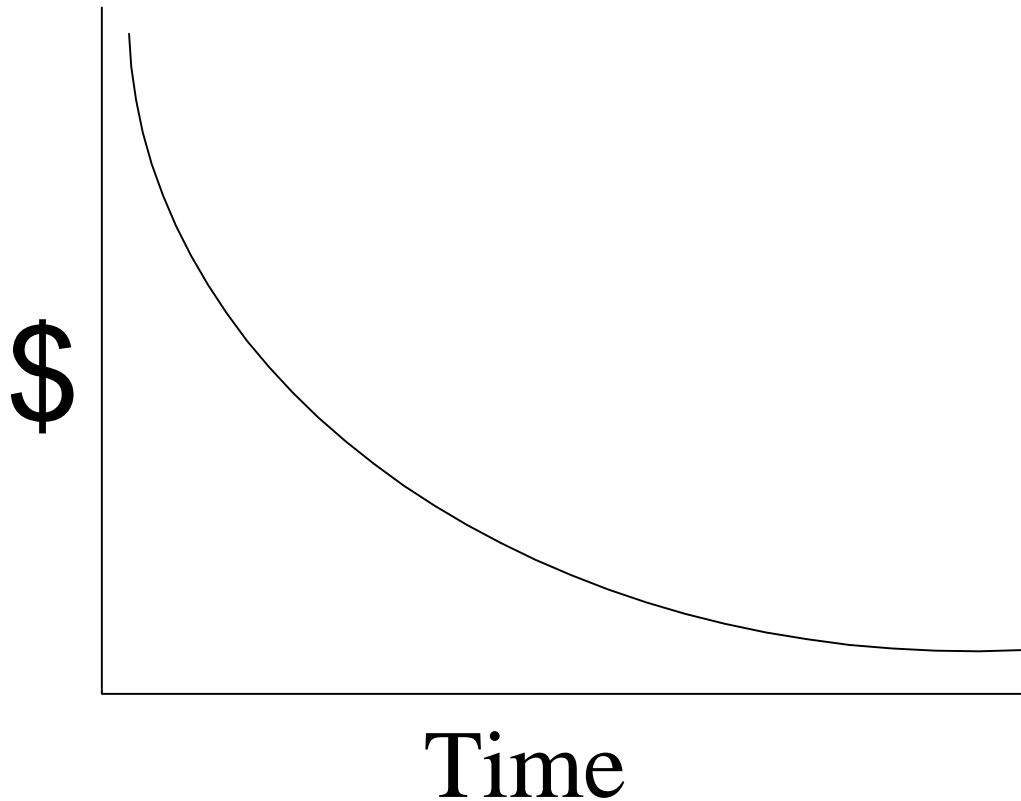
Enterprise Architecture

is the structured, explicit "Knowledge-Base" of the Enterprise against which the implicit, "tacit" knowledge can be mapped, classified, evaluated and/or transformed to "explicit" ...

... to give management the Information Age

Knowledge Advantage.

Long Term Investment



Long term, asset-based, integration and normalization dominant, investment-oriented, engineered for reusability.

Do the Engineering before you start manufacturing.

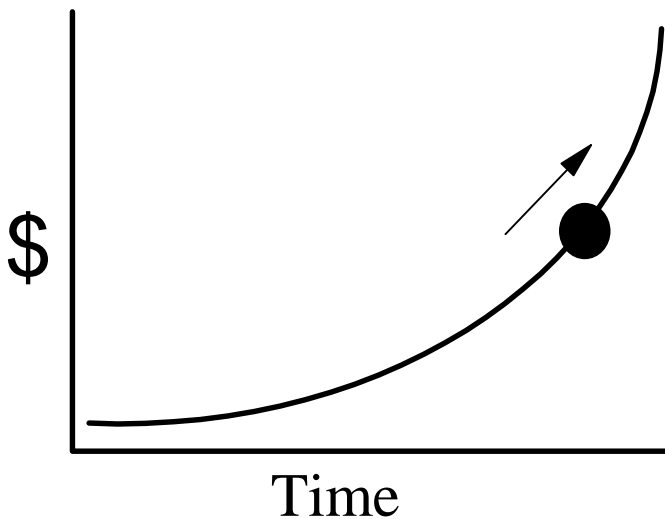
ARCHITECTURE

Create re-usable assets. Minimize scrap and rework.

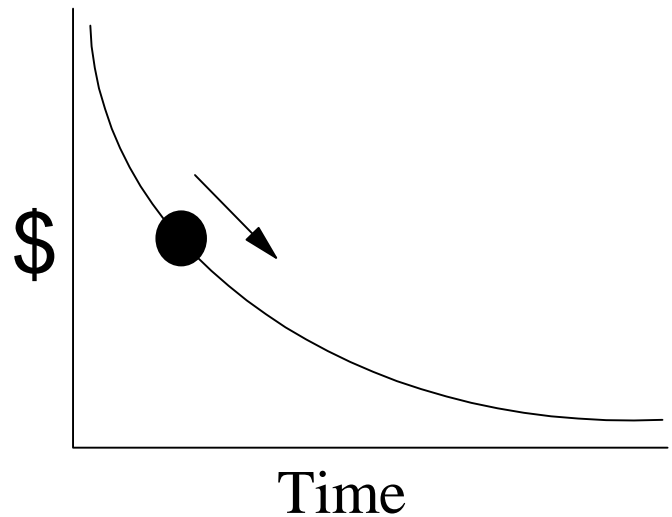
Takes up-front time and money.

Investment Curves

Cost Justification (Expense-based)



Architecture Investment (Asset-based)



Where are you?

Plot the last 50 (or however many) years of:

1. DP Budget

PLUS

2. Manual costs of reconciling data, compensating for network outages, and resolving business rule conflicts.

PLUS

3. Lost business opportunities and realized liabilities.

Observation

If you don't feel good about the current legacy of information systems ...

("We're just not getting the value ... ")

And if you don't feel good about I/S' ability to address current demand ...

("It takes too long and costs too much ... ")

Then, something has to change ...

One definition of insanity is to keep on doing what you have been doing and expect different results.

(Albert Einstein, I think.)

(You simply cannot continue to take the short term option.)

Revolution

You will need a "Revolution in CONCEPTS."

You will need to begin seeing:

the Enterprise as a contiguous system in its own right.

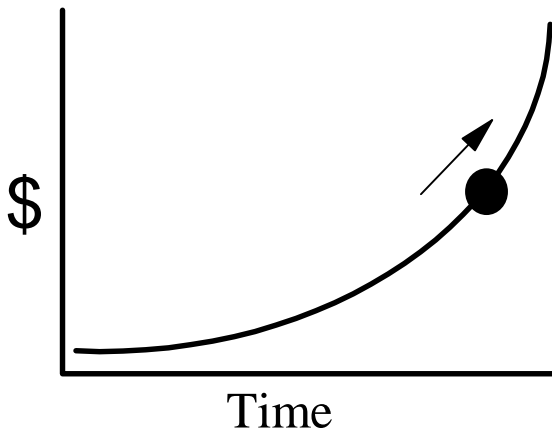
that the "System" IS the Enterprise.

that I/S is actually engineering and manufacturing your Enterprise for you.

that this is DIFFERENT from building and running systems.

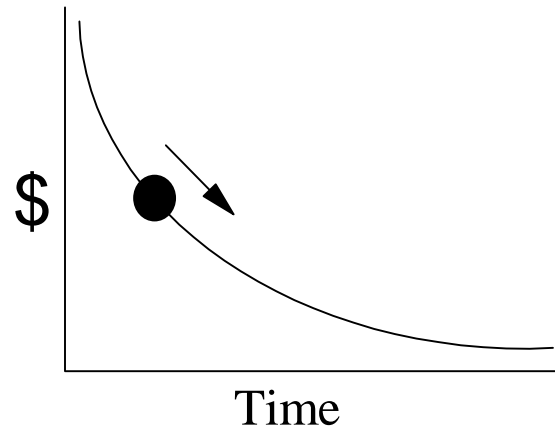
You will need a LONG TERM strategy.

Engineering Design Objectives



This is the short term
value:

Implementation



These are long term
values:

Alignment
Integration
Flexibility
Interoperability
Reduced Time-to-Market
Quality
Seamlessness
Adaptability
User-Friendliness
Usability
Reusability
etc., etc.

Some People Don't Get It!

One more time ...

For EVERY implementation,
from now on ...

Somebody should be working on

EVERY ONE OF THOSE 30 MODELS ...

And ... if any one of those models is not being worked on,

Somebody else should know why,

And also should know the implications
of omitting that model.

THIS IS THE INFORMATION AGE!!

THE NORM SHOULD BE THE MODELS!!

What I am Saying/NOT Saying

NOTE:

A. I am NOT saying: forget about satisfying short term demand ...

I am NOT saying: never take the short term option.

(In fact, you need a long term strategy
AND a short term strategy)

B. However, I AM saying we better start making the long term option the norm and the short term option the exception for a change or

WE ARE NEVER GOING TO SOLVE THE PROBLEM!
(Alignment, integration and quick response to change.)

C. The challenge is to figure out clever ways to carve this thing up into "slivers" satisfying short term demand but doing sufficient architectural work to build something coherent over time.

(See "Implementation Practicalities")

Compromise

If you are going to do things that compromise the long term, best interests of the Enterprise, you probably should:

- A. Know that you are doing it,
- B. Know why you are doing it,
- C. Make every effort to mitigate the downstream effects of the compromise, and
- D. Make sure that everybody who would have reason to be unhappy later understands all of the above ... and maybe even participates in the compromise decision process.

Conclusion

I would argue ... the question is NOT:

"Are you going to do
Enterprise Architecture?"

The only questions ARE:

"HOW are you going to do it?"

and

"Will you be able to do it before
someone else in your industry
DOES IT FIRST???"

Frequently Asked Questions

It's Cheaper and Faster!



Cheaper and Faster

Using a top-down, Enterprise Architecture approach, a rigorous, enhanced Information Engineering Method, Three-Schema Data Architecture and CASE technology:

Cost per new entity type/RDBMS table
reduced from >\$150,000 to <\$10,000.

Enterprise data handling labor reduced 50%.

Reduced development time of 25% through
improved communication and conflict resolution.

Development time and cost reductions for every
succeeding implementation of >50%,
compounded, through reuse of database and
application components with no modifications.

Reduced disk space for data (including history)
of 20% - 80% through elimination of redundancy.

Reference: Doug Erickson 614-751-5078

Cheaper and Faster

State of Ohio: Workers Compensation Board

Rates System 594 entity types

(2 years elapsed time)

Development costs \$2,952,364

Software costs (1 time) \$ 468,638

Cost per Entity Type Approx \$5,000

Recent Package implementation: \$30,000/Ent.Type

Recent Custom Apps: \$100,000- \$150,000/Ent. Type

REUSE

Payment System 690 E/T's (Reused 294)

Retro Billing 228 E/T's (Reused 218)

HCP Sys. 320 E/T's (Reused 179)

Total savings: 691 (reused) x \$5,000 = \$3,455,000

(Saves more than the original appln. cost!)

In the Rates System:

115 procedures (programs) 1177 modules (callable action blocks) used 3112 times. Reuse factor = 2.64 (attributable to granularity and precision of the data model, i.e. many processes use the same data.)

Reference: Doug Erickson 614-751-5078

Cheaper and Faster

State of Ohio

Different State

Workers Comp. *Same Application* Workers Comp.

IEF/CoolGen *Same CASE Tool* IEF/CoolGen

Architected *Different Methodology* Classic

594 *Entity Types* 300

2 Years *Elapsed Time* 12 Years

\$3.5 Mil *Development Costs* \$42 Mil.

\$5,000 *Cost per Entity Type* \$140,000

(2 prime contractors
and one local cntrtr.
Estimating 3 more years
to enhance/fix)

Reference: Doug Erickson 614-751-5078

Enterprise Architecture

Conclusions



As Bluntly As I Know How

You may think this is too much work ...

Or, it takes too long

And costs too much

Or is too theoretical

Or too high risk

Or too whatever.

However, if that's your assessment ...

You can't complain that

the systems aren't "aligned" with the enterprise,

or are inflexible, or cost too much,

or that vital information is not available,

or the data you get isn't any good, or too late,

or you can't change anything,

or that I/S is slow and unresponsive ...

and, I am here to tell you

Outsourcing isn't going to fix the problem.

Packages (in themselves) won't fix the problem.

Decentralization won't fix the problem.

And, the Internet isn't going to fix the problem.

As Bluntly As I Know How (cont)

No amount of money
or technology
is going to fix the problem!

It is NOT a technical problem,
it is an Enterprise ENGINEERING problem.

Only ACTUAL WORK is going to fix the problem

and

"Someday, you are going to wish you had
all those models,
Enterprise wide,
horizontally and vertically integrated
at excruciating level of detail."

You might as well start working on them ...
... anytime this afternoon is probably not too early!!