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

Physics 101

I. Introduction to Enterprise Architecture
   A. The Framework for Enterprise Architecture
   B. Basic Physics
II. The Trade-Off
   A. Long Term Options
   B. Short Term Options
III. A. Engineering design Objectives
IV. Cheaper and Faster
V. Conclusions
   A. Suggestions for Management
   B. 10 Year Technology Forecast

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			

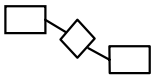
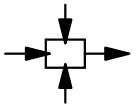
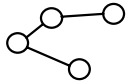
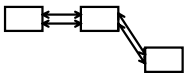
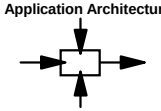
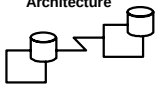
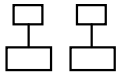
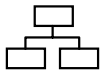
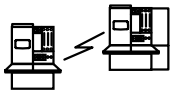
A Framework

	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			

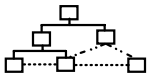
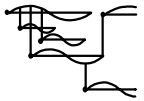
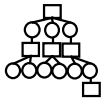
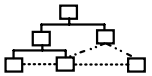
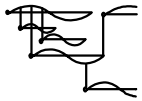
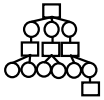
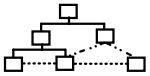
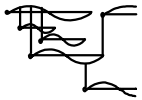
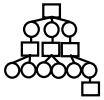
ENTERPRISE ARCHITECTURE - A FRAMEWORK

	DATA *What*	FUNCTION *How*	NETWORK *Where*
OBJECTIVES/ SCOPE (CONTEXTUAL) *Planner*	**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) *Owner*	**e.g. Semantic Model**  **Ent = Business Entity Reln = 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) *Designer*	**e.g. Logical Data Model**  **Ent = Data Entity Reln = 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) *Builder*	**e.g. Physical Data Model**  **Ent =Segment/Table/etc. Reln = Pointer/Key/etc.**	**e.g. System Design**  **Proc = Computer Function I/O = Data Elements/Set**	**e.g. Technology Architecture**  **Node = Hardware/Systems Software Link = Line Specifications**
DETAILED REPRESEN-TATIONS (OUT-OF-CONTEXT) *Sub-Contractor*	**e.g. Data Definition**  **Ent = Field Reln = 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	
List of Organizations/Agents Important to the Business  People = Class of Agent	List of Events Significant to the Business  Time = Major Business Event	List of Business Goals/Strat  Ends/Means = Major Bus. Goal/ Critical Success Factor	OBJECTIVES/ SCOPE (CONTEXTUAL) *Planner*
e.g., Work Flow Model  People = Organization Unit Work = Work Product	e.g., Master Schedule  Time = Business Event Cycle = Business Cycle	e.g., Business Plan  End = Business Objective Means = Business Strategy	BUSINESS MODEL (CONCEPTUAL) *Owner*
e.g., Human Interface Architecture"  People = Role Work = Deliverable	e.g., Processing Structure  Time = System Event Cycle = Processing Cycle	e.g., Business Rule Model  End = Structural Assertion Means = Action Assertion	SYSTEM MODEL (LOGICAL) *Designer*
e.g., Presentation Architecture  People = User Work = Screen Formats	e.g., Control Structure  Time = Execute Cycle = Component Cycle	e.g., Rule Design  End = Condition Means = Action	TECHNOLOGY MODEL (PHYSICAL) *Builder*
e.g., Security Architecture  People = Identity Work = Job	e.g., Timing Definition  Time = Interrupt Cycle = Machine Cycle	e.g., Rule Specification  End = Sub-condition Means = Step	DETAILED REPRESEN-TATIONS (OUT-OF-CONTEXT) *Sub-Contractor*
e.g., ORGANIZATION	e.g., SCHEDULE	e.g., STRATEGY	FUNCTIONING ENTERPRISE

Less than Enterprise Scope

	DATA	FUNCTION	NETWORK
SCOPE *Planner*			
BUSINESS MODEL *Owner*			
SYSTEM MODEL *Designer*			
TECHNOLOGY MODEL *Builder*			
COMPONENTS *Sub-Contractor*			
FUNCTIONING ENTERPRISE			

Re: *Any* Cell

Less than Excruciating Detail

	DATA	FUNCTION	NETWORK
SCOPE *Planner*			
BUSINESS MODEL *Owner*			
SYSTEM MODEL *Designer*			
TECHNOLOGY MODEL *Builder*			
COMPONENTS *Sub-Contractor*			
FUNCTIONING ENTERPRISE			

Re: *Any* Cell

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.)

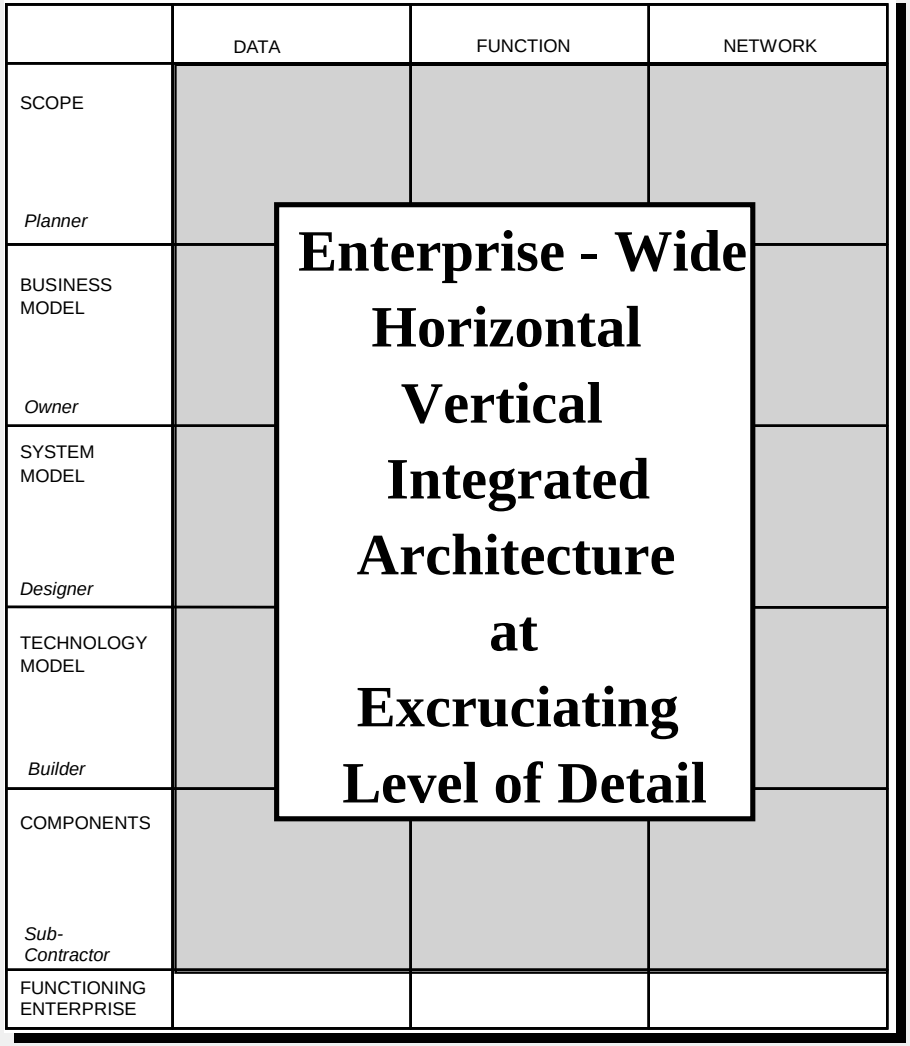
2. The system IS the Enterprise

   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



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!

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!)

The "Zachman Framework"

The Framework does not *prescribe* anything about which models you have to build before you can deliver an implementation. Only YOU (or your methodology) determines which models (or "slivers" of models) you are actually going to build (or NOT build) over the process of developing systems.

On the other hand, the Framework, by definition, identifies the total, comprehensive set of models relevant for describing the entire Enterprise. Therefore, the Framework is a convenient analytical tool to help you determine how you will feel if you (or your methodology) are NOT going to produce all the models ... or if you are going to do something that will INHIBIT integrated accumulation of the comprehensive set of models in the long term while you are satisfying current demand in the short term.

Enterprise Architecture

The Trade - Off

The Reality of It All

"In reality, life is a series of trade-offs."

John A. Zachman, 6/21/98

Immediate Gratification	**vs**	**What's Good for You**
short term options	vs	long term options
implementation	vs	integration
point in time solutions	vs	infrastructure solutions
expense based approaches	vs	asset based approaches
implementation optimization (at the expense of the Enterprise)	vs	Enterprise optimization (at the expense of the implementation)
quick and dirty	vs	right
pay me now or pay me later etc.	vs	takes too long and costs too much etc.

Long Term/Best Interests

	DATA	FUNCTION	NETWORK
SCOPE *Planner*			
BUSINESS MODEL *Owner*			
SYSTEM MODEL *Designer*			
TECHNOLOGY MODEL *Builder*			
COMPONENTS *Sub-Contractor*			
FUNCTIONING ENTERPRISE			

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

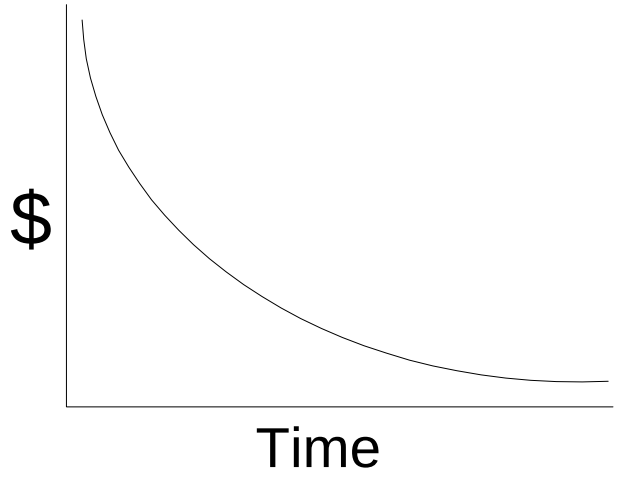
Long Term/Best Interests

"End State Vision" = Long Term Best Interests
This would constitute an Enterprise that was:

1. "Lean" - no redundancy.
2. "Integrated" - no discontinuity, "aligned"
3. Flexible - "de-coupled"
4. Changeable - baseline for managing change

(*Enterprise* optimized.)

Long Term Investment



Short Term/Do What Feels Good

	DATA	FUNCTION	NETWORK
SCOPE *Planner*			
BUSINESS MODEL *Owner*			
SYSTEM MODEL *Designer*			
TECHNOLOGY MODEL *Builder*			
COMPONENTS *Sub-Contractor*			
FUNCTIONING ENTERPRISE	Running Enterprise		

No Architecture (Nothing - Empty)

Short Term/Do What Feels Good

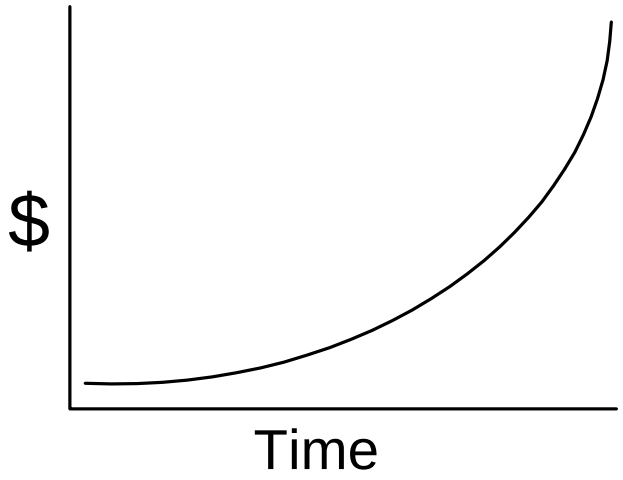
"You start writing the code and I'll go find out what the users have in mind."

(That is, NO architecture.)

(T hat is what we have now ... the legacy.)

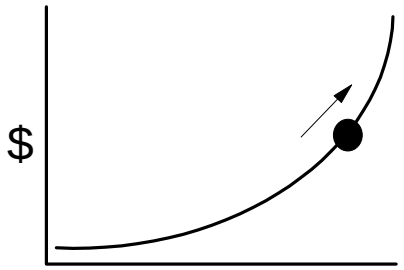
(*Implementation* optimized.)
(Enterprise *sub*-optimized.)

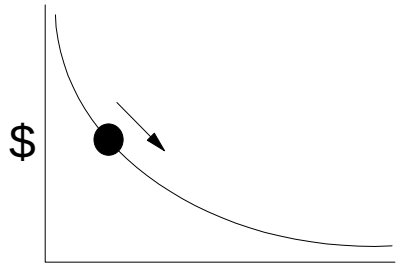
Short Term Investment



Where Are You??

Short Term **Long Term**





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.

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

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.

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.

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.

Implementation Practicalities

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 ... the LONG term option) 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 concept: Quality

"Alignment" would be the 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.

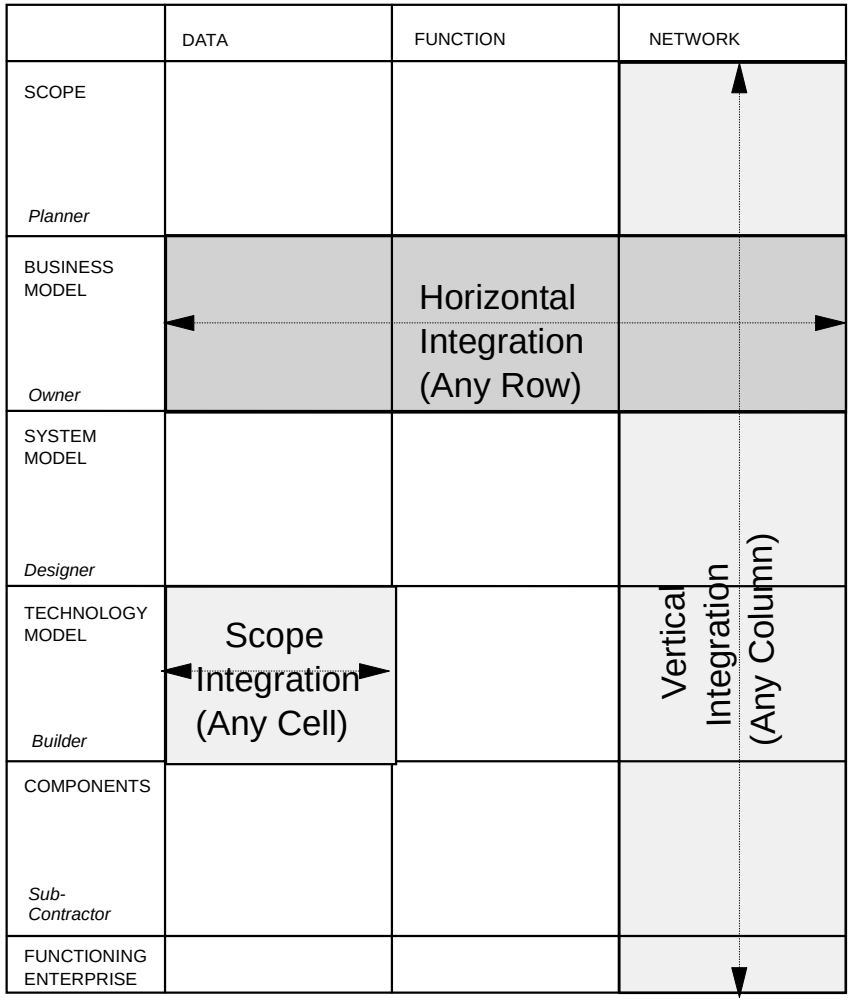
Integration

"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 - standard, interchangeable parts, 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.)

Similar Concepts: Seamlessness (Col. 2), Interoperability (Col. 3), Reusability
Integration is the equivalent of
Standard Interchangeable Parts.

Three Definitions of "Integration"



Integration

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.

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 content or change meaning.

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!!

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 a panacea!
  (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 4/5 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.

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?")

Flexibility

Flexibility means ...

... you want things designed such that they can be changed quickly with minimum disruption, at minimum cost.

(Similar Concept: Adaptability)

Flexibility

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.")
- "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."
- Every cell of the Framework is an independent variable.
- If you want to change the technology (i.e. Row 4 technology constraints) with minimum time, disruption and cost, you have to manage the architecture at the logical level (Row 3).
- If you have no architectural representations, you have no separation of independent variables and therefore, *NO FLEXIBILITY* (all independent variables are implicit and imbedded in a unitary implementation ... the running system.)

Reduced Time-to-Market

Reduced Time-to-Market: If you REALLY want to reduce the time it takes from when you discover you need a new "system" until you take delivery on it ... to virtually zero ...

A. Make to Order

If you wait until you get an order before you start to design and manufacture, you can only reduce time to market by reducing size/complexity of the product.

(Custom Products)

B. Provide from Stock

If you build finished goods to inventory before you get the order, you can reduce time to market to virtually zero, but the customer has to change the use of the product to fit the product.

(Standard Products)

C. Assemble to Order

If you pre-fabricate parts to inventory that are designed to be assembled onto more than one product, you can reduce the time to market to just the time it takes to assemble the parts to order ... virtually zero.

(Custom-Standard Products)

What do you think has to be in inventory before you get the order? "Someday, you are going to wish ... "

Enterprise Architecture

It's Cheaper and Faster

The Real Value of 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

Baseline for managing change.

Retained, accumulated, Enterprise knowledge .

(You are not always starting with a blank sheet of paper.)

(This is "effectivity," change management.)

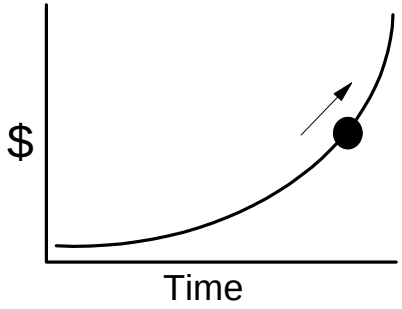
D. I/S Responsiveness

Architecture plus "assemble-to-order" processes.

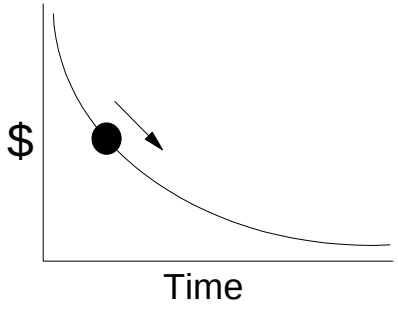
(This is reduced "time to market" - "Mass Customization.")

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.

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

Enterprise Management

My advice to the General Manager:

First, you don't have to know how to build all these models before you can manage the Enterprise (any more than you have to know how to engineer a building before you can occupy it) ... BUT ... I strongly suggest that:

A. **YOU** should make the long term/short term architecture trade-off decision yourself.

(Do you want to optimize the *Enterprise*?
Or, do you want to optimize the *implementation* - and *sub*-optimize the Enterprise?
Or, do you want some compromise?)
(I would NOT let a programmer make the decision by default!)

B. **YOU** decide who should be involved in defining what business you are in and in engineering your Enterprise for you. (That is, who should be involved in taking the assumptions out of the Rows 1 and 2 models for you.)

Enterprise Management

C. You ensure that your Enterprise acquires the skills, capabilities and tools to build the Rows 1, 2, 3 models and to maintain control of those models in house whether you use contractors to help build them or not, and whether you contract out the production and mgmt of some portions of the Rows 4 and 5 models or not.

D. You expect your Information Community to analyze your Enterprise Architecture strategy in the context of the Framework and communicate to you the alternatives and implications in the same context.

E. You establish an Executive Vice President in charge of Change Management to:

1. Define and manage the Enterprise Change Process.
2. Define and implement Plans and Controls for Architecture Management.
3. Develop the Architecture Strategy to achieve your long term vision.
4. Skew the Enterprise culture (including I/S) toward the long term end of the spectrum.

Enterprise Management

Then, I would capitalize on your ability

to manage complexity
and
to dynamically change the Enterprise
(with minimum time, disruption and cost)

by

destabilizing the market.

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!!

Conclusions

Ten Year Technology Forecast

Technology Forecast

Here is the Zachman 10 year technology forecast:

"BIGGER (smaller). FASTER. CHEAPER."

You will be able to store lots more data ... a lot cheaper,

process lots more transactions ... a lot cheaper,

send lots more messages ... a lot cheaper,

and, you will be able to generate lots more code ... a lot cheaper.

Therefore, there will be lots more technology in the hands of lots more people.

Absolute Certainty

There will still be only 24 hours in each day

... and the rate of change is going to get worse,
A LOT WORSE!
... and complexity is going to get worse,
A LOT WORSE!

... and THE ENTERPRISE MODELS ARE NOT GOING TO GO AWAY!!

And someday ...
... you are REALLY going to wish you had
all the models,
Enterprise-wide,
horizontally and vertically integrated,
at excruciating levels of detail.

The Enterprise challenge is NOT technology.
It IS
ENTERPRISE ARCHITECTURE
and
CULTURE CHANGE.

Conclusion

This is not mysticism ...
it is physics!

You don't need a magician ...
you need an architect.

... and a little patience to learn to communicate.

I suggest the "Framework."

(That's what I use to understand the issues.)