Zen and Enterprise Architecture
Beginner's Mind

Zen and Enterprise Architecture Beginner's Mind

The **TOGAF** and **Zachman** frameworks indicate a large number of views into an Enterprise Architecture .

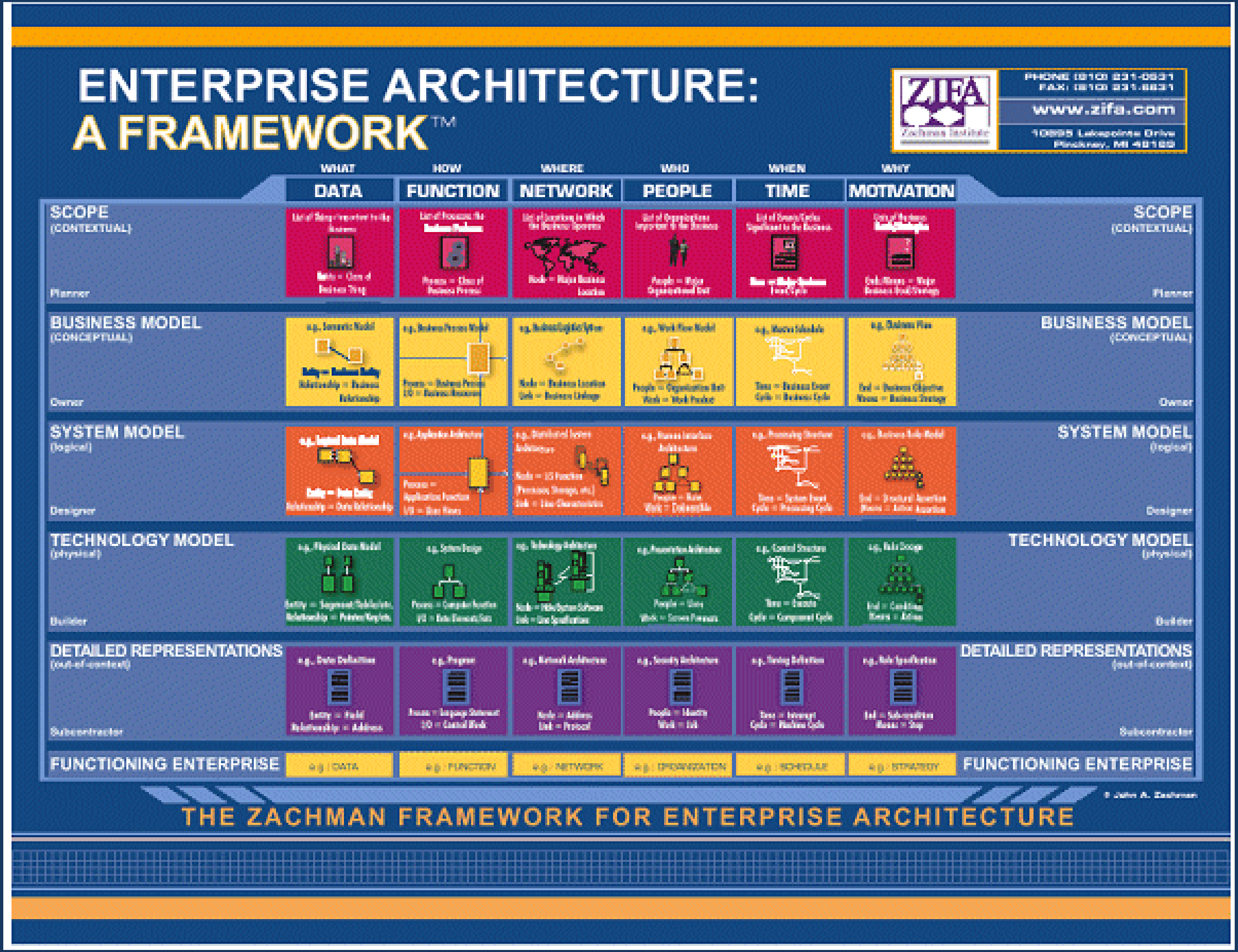
Beginner's may find this daunting.

This presentation provides a simplified overview with some hints about how to get started.

Architecture Frameworks

- Standardized Architecture Vocabulary
- Not a methodology
  - Not a sequence
  - Allows you to incorporate existing assets.
- Guidelines … How to:
  - Organize Enterprise Architecture artifacts.
  - Coordinate multiple teams.

Zachman Framework



ENTERPRISE ARCHITECTURE:
A FRAMEWORK™

ZIFA
Zachman Institute

PHONE (810) 231-0531
FAX: (810) 231-6631
www.zifa.com
10895 Lakepointe Drive
Pinckney, MI 48169

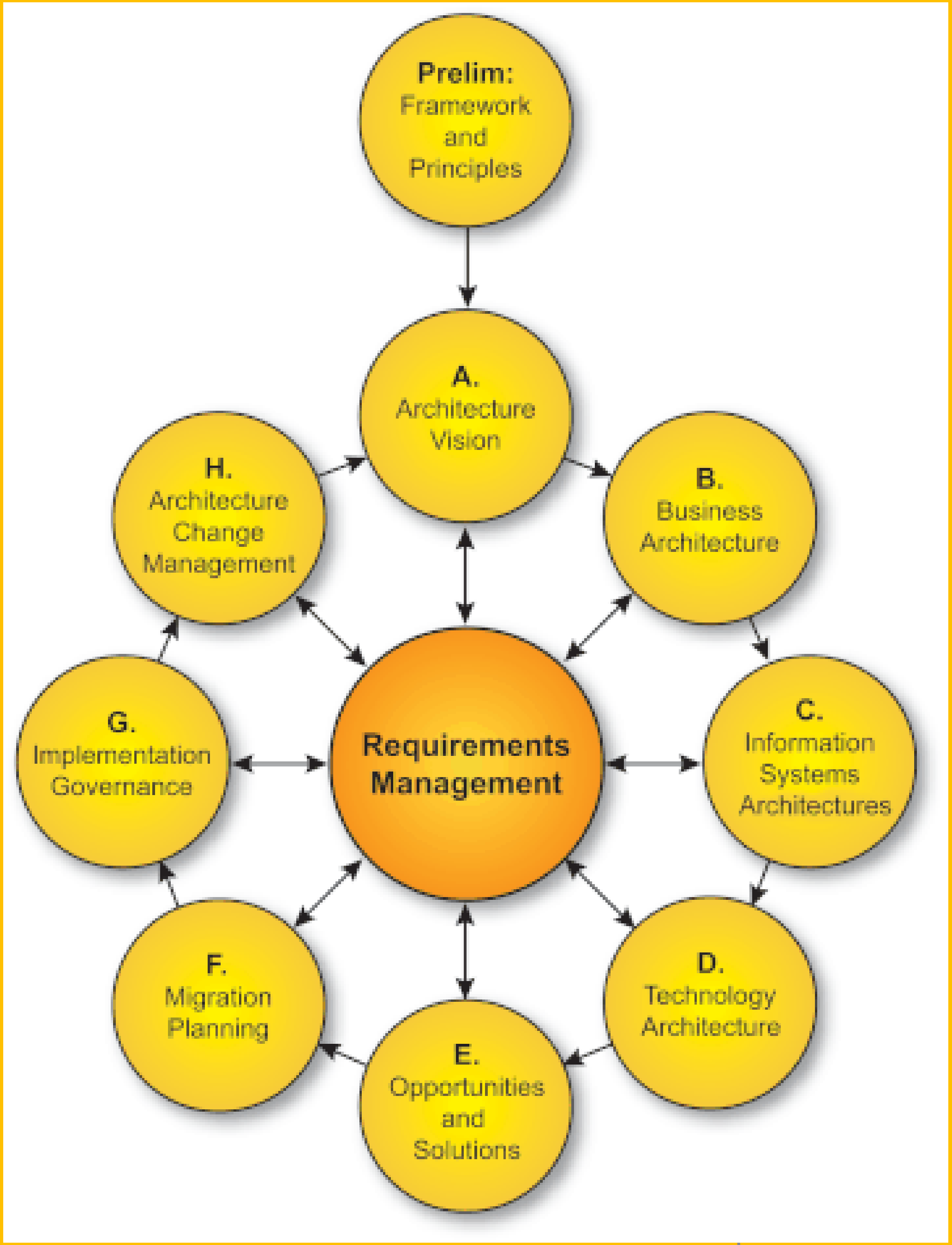
	WHAT DATA	HOW FUNCTION	WHERE NETWORK	WHO PEOPLE	WHEN TIME	WHY MOTIVATION	
SCOPE (CONTEXTUAL) Planner	List of Things Important to the Business Entity = Class of Business Thing	List of Processes the Business Performs Process = Class of Business Process	List of Locations in Which the Business Operates Node = Major Business Location	List of Organizations Important to the Business People = Major Organizational Unit	List of Events/Cycles Significant to the Business Time = Major Business Event/Cycle	List of Business Goals/Strategies Ends/Means = Major Business Goal/Strategy	SCOPE (CONTEXTUAL) Planner
BUSINESS MODEL (CONCEPTUAL) Owner	e.g. Semantic Model Entity = Business Entity Relationship = Business Relationship	e.g. Business Process Model Process = Business Process I/O = Business Resources	e.g. Business Logistics System Node = Business Location Link = Business Linkage	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
SYSTEM MODEL (logical) Designer	e.g. Logical Data Model Entity = Data Entity Relationship = Data Relationship	e.g. Application Architecture Process = Application Function I/O = User Views	e.g. Distributed System Architecture Node = I/S Function (Processor, Storage, etc.) Link = Line Characteristics	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
TECHNOLOGY MODEL (physical) Builder	e.g. Physical Data Model Entity = Segment/Table/etc. Relationship = Pointer/Key/etc.	e.g. System Design Process = Computer Function I/O = Data Elements/Sets	e.g. Technology Architecture Node = Hardware/System Software Link = Line Specifications	e.g. Presentation Architecture People = User Work = Screen Format	e.g. Control Structure Time = Execute Cycle = Component Cycle	e.g. Rule Design End = Condition Means = Action	TECHNOLOGY MODEL (physical) Builder
DETAILED REPRESENTATIONS (out-of-context) Subcontractor	e.g. Data Definition Entity = Field Relationship = Address	e.g. Program Process = Language Statement I/O = Control Block	e.g. Network Architecture Node = Address Link = Protocol	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 REPRESENTATIONS (out-of-context) Subcontractor
FUNCTIONING ENTERPRISE	e.g. DATA	e.g. FUNCTION	e.g. NETWORK	e.g. ORGANIZATION	e.g. SCHEDULE	e.g. STRATEGY	FUNCTIONING ENTERPRISE

© John A. Zachman

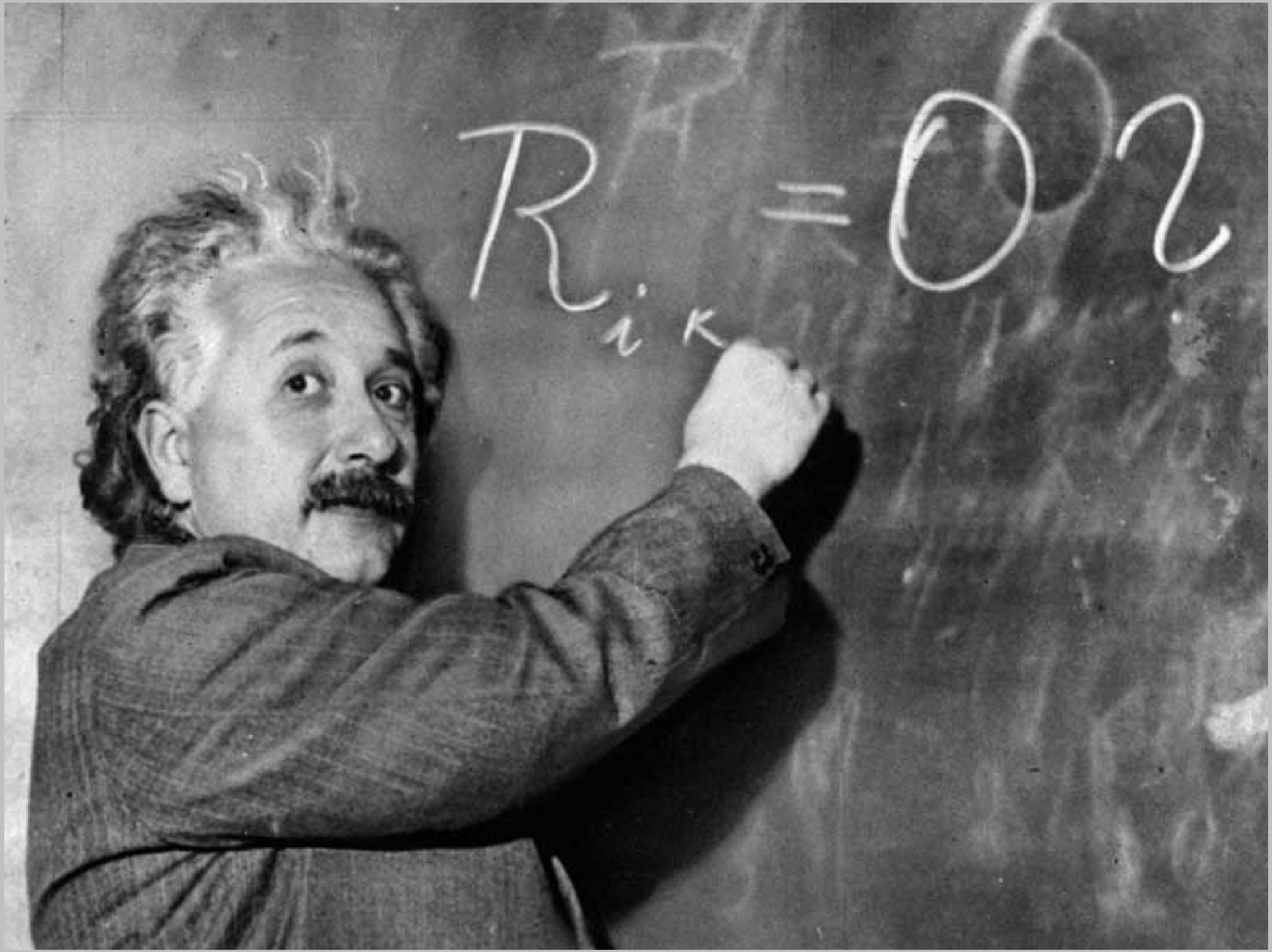
THE ZACHMAN FRAMEWORK FOR ENTERPRISE ARCHITECTURE

TOGAF

The **O**pen **G**roup **A**rchitecture **F**ramework


Prelim: Framework and Principles
A. Architecture Vision
B. Business Architecture
C. Information Systems Architectures
D. Technology Architecture
E. Opportunities and Solutions
F. Migration Planning
G. Implementation Governance
H. Architecture Change Management
Requirements Management

A (model) should be as simple as possible, but no simpler.

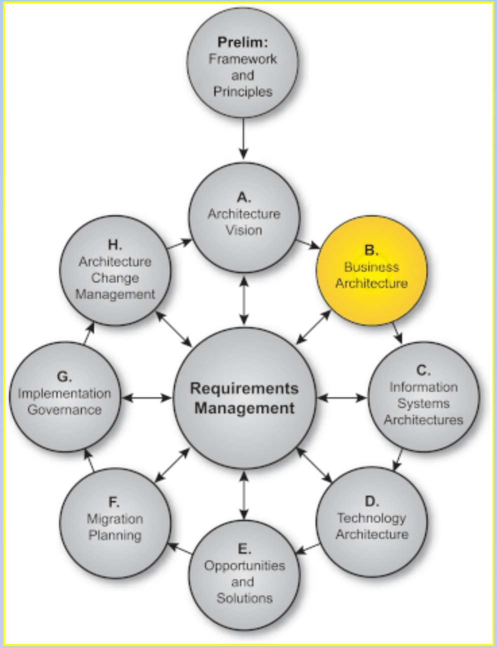


4 Dimensions

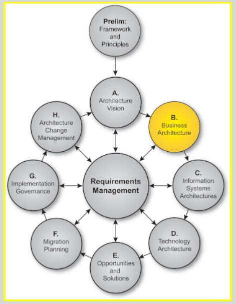
1. Process
   → Business Architecture
2. Information
   → Data Architecture
3. Infrastructure
   → Standard Operating Environment
4. Time
   → Roadmaps

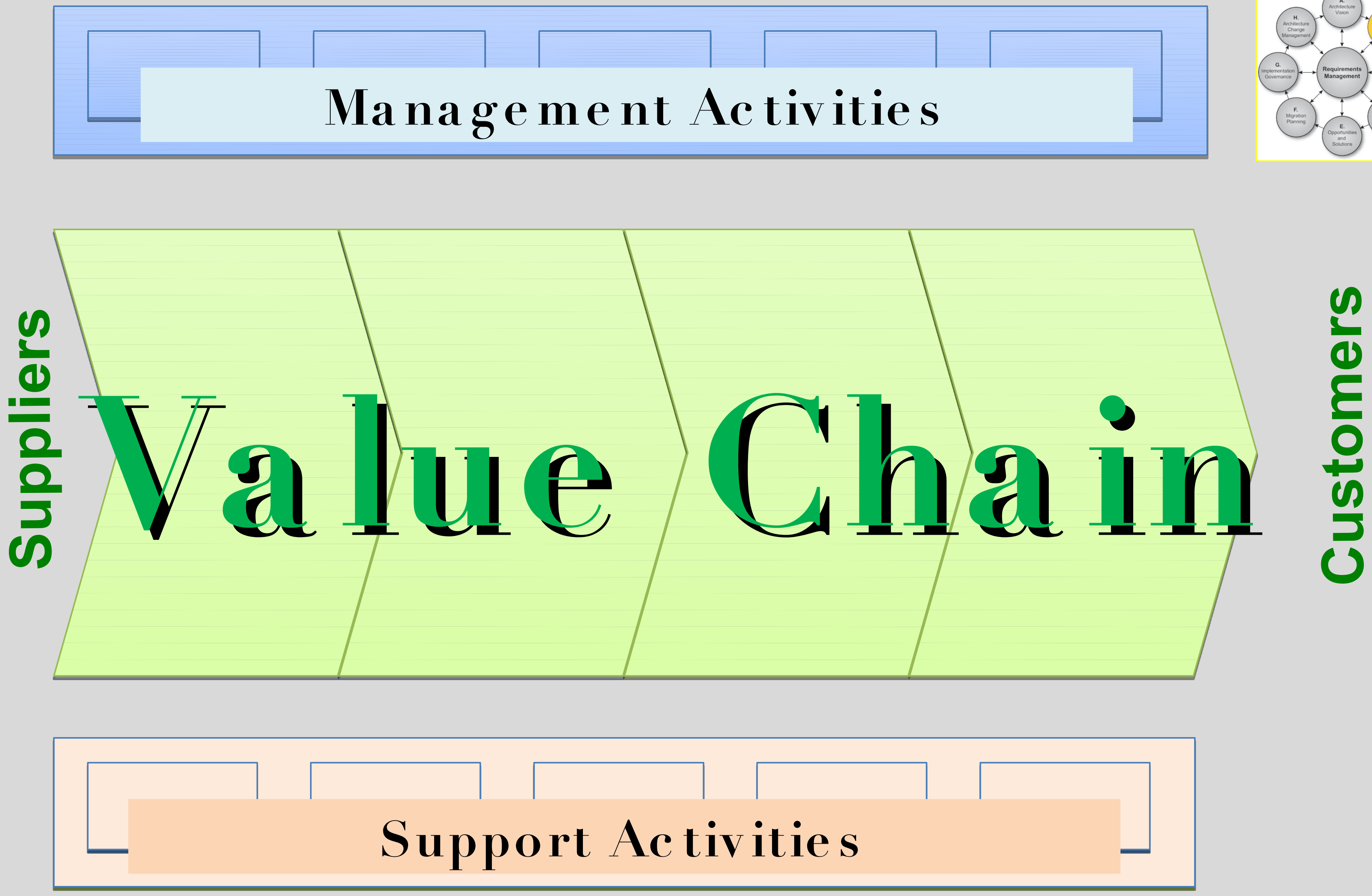
Where to start?

- **Advice:**
  Start with a **value chain model**.
- **Why? ... Synergy**
  - **Engages** the business.
  - Defines modeling **priorities**.
  - By-product: domain **entities**.
  - Provides the context for **Interface Driven Design.**
- **How:**
  - There are many best practices. Coming up: some suggestions

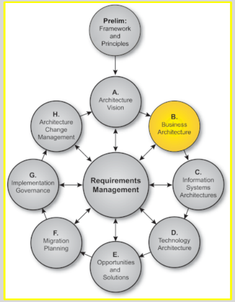


Prelim: Framework and Principles
A. Architecture Vision
B. Business Architecture
C. Information Systems Architectures
D. Technology Architecture
E. Opportunities and Solutions
F. Migration Planning
G. Implementation Governance
H. Architecture Change Management
Requirements Management


Prelim: Framework and Principles
A. Architecture Vision
B. Business Architecture
C. Information Systems Architectures
D. Technology Architecture
E. Opportunities and Solutions
F. Migration Planning
G. Implementation Governance
H. Architecture Change Management
Requirements Management


Management Activities
Suppliers
Value Chain
Customers
Support Activities

Enterprise-Level
Business Process Model

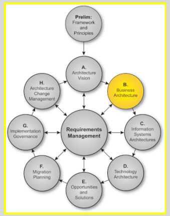

Prelim: Framework and Principles
A. Architecture Vision
B. Business Architecture
C. Information Systems Architectures
D. Technology Architecture
E. Opportunities and Solutions
F. Migration Planning
G. Implementation Governance
H. Architecture Change Management
Requirements Management

- Focus on the Value Chain.
- Follow the product, not the data.
- Remember that accounting, reporting, and ITS are all support activities, they are not the main-line.
- *Write an **imperative sentence** to describe each activity.*

Write an **imperative sentence** to describe each activity.

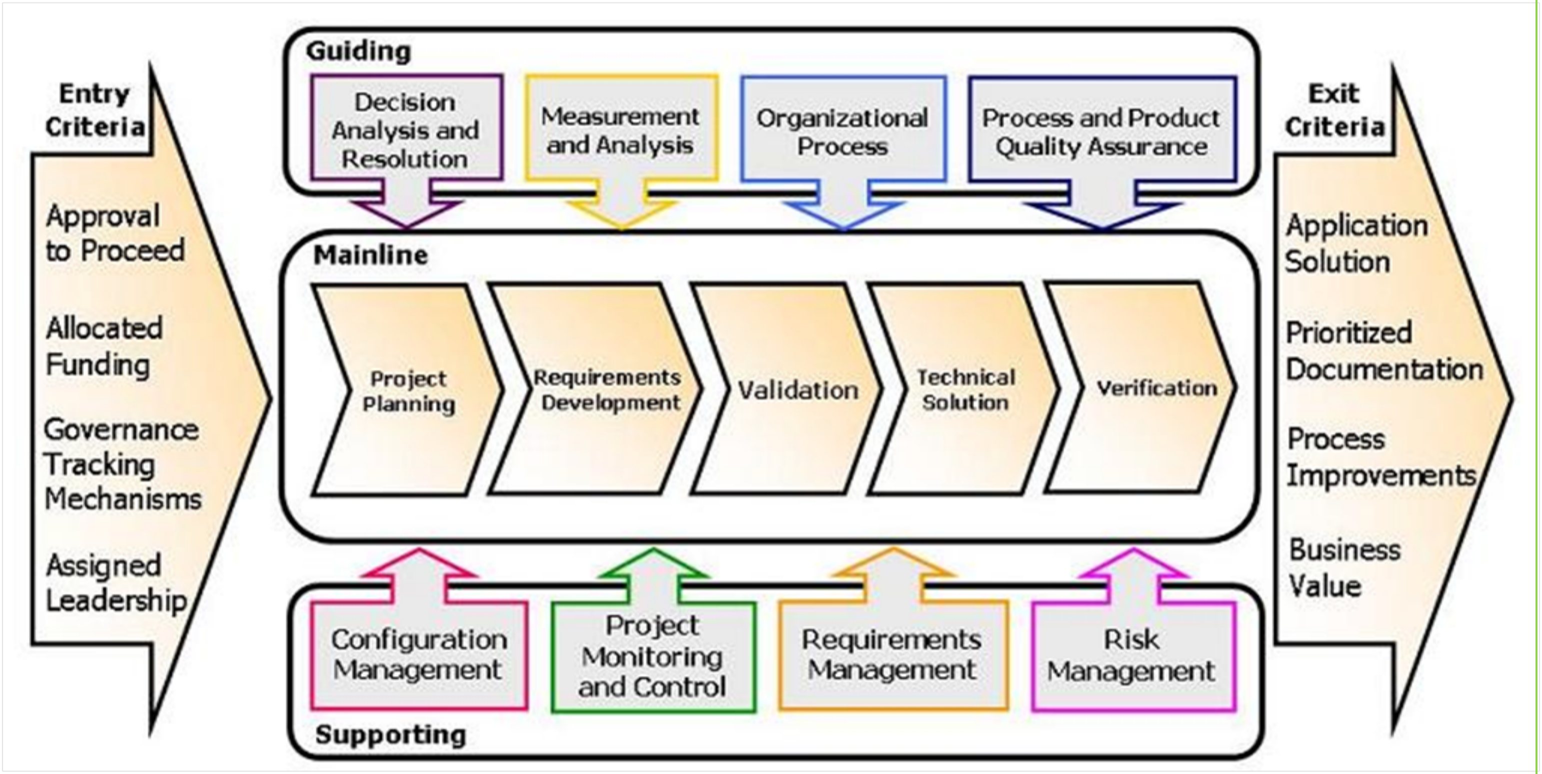
- An imperative sentence starts with a verb.
- A **weak verb** indicates **weak analysis**.
  - A sentence that begins with verbs like: process, manage, handle, …
    simply beg the question.
- Avoid **local jargon**, especially IT jargon.
  - **Industry jargon** might be okay.
  - Industry acronyms are marginal.

Agile Business Process Modeling


Prelim: Framework and Principles
A. Architecture Vision
B. Business Architecture
C. Information Systems Architectures
D. Technology Architecture
E. Opportunities and Solutions
F. Migration Planning
G. Implementation Governance
H. Architecture Change Management
Requirements Management

- Remember that the model is not the goal.
  It is the vehicle.
- Just Enough, Just In Time:
  - You only need enough information to plan the next step.
- Focus on **hand-offs** between business units.
  - At the enterprise-level, the interface is more important than the internal process.

Example: ITS as an enterprise



You cannot model in a vacuum.

Different organizations will have different terms for similar activities.

There will be differences in emphasis.

Boxes will be ordered and sized according to the local point of view.

Agile Software Development Value Chain

Plan future features.

Plan products / features.
- Identify future products.
- Request new features.
- Prioritize new features.
-

Plan software releases.
- Allocate resources.
- Publish roadmap.
- Forecast releases.

Design features.

Identify requirements.
- Business requirements.
- Derived requirements.
- Write test plan.
-

Design software.
- Design the UI.
- Decide architecture.
- Code the API.

Build current release.

Construct Software
- Use cases ➔ tests.
- Pass each test .
- Integrate continuously.
-

Final-Test Software
- Exploratory Testing
- Acceptance Testing
- Load and Stress Testing

Instant Enterprise Data Model

- Review the **nouns**
  found in the process model statements.
- Many of those nouns are **data entities**.
- Create a **glossary**
  to standardize the vocabulary.

- You don't need **attributes** for planning.
  They only serve to clarify entities.
- Nouns that refer to forms or other data processing
  artifacts indicate misplaced analysis.

Subject Areas – Data Entities

Release Plan Subject Area

- Product
- Feature
- Release
- Resource
- Roadmap
- Forecast

Design Subject Area

- Business Requirements
- Derived Requirements
- User Interface (UI)
- Program Interface (API)
- Architecture
- Test Plan

Test and Code Subject Area

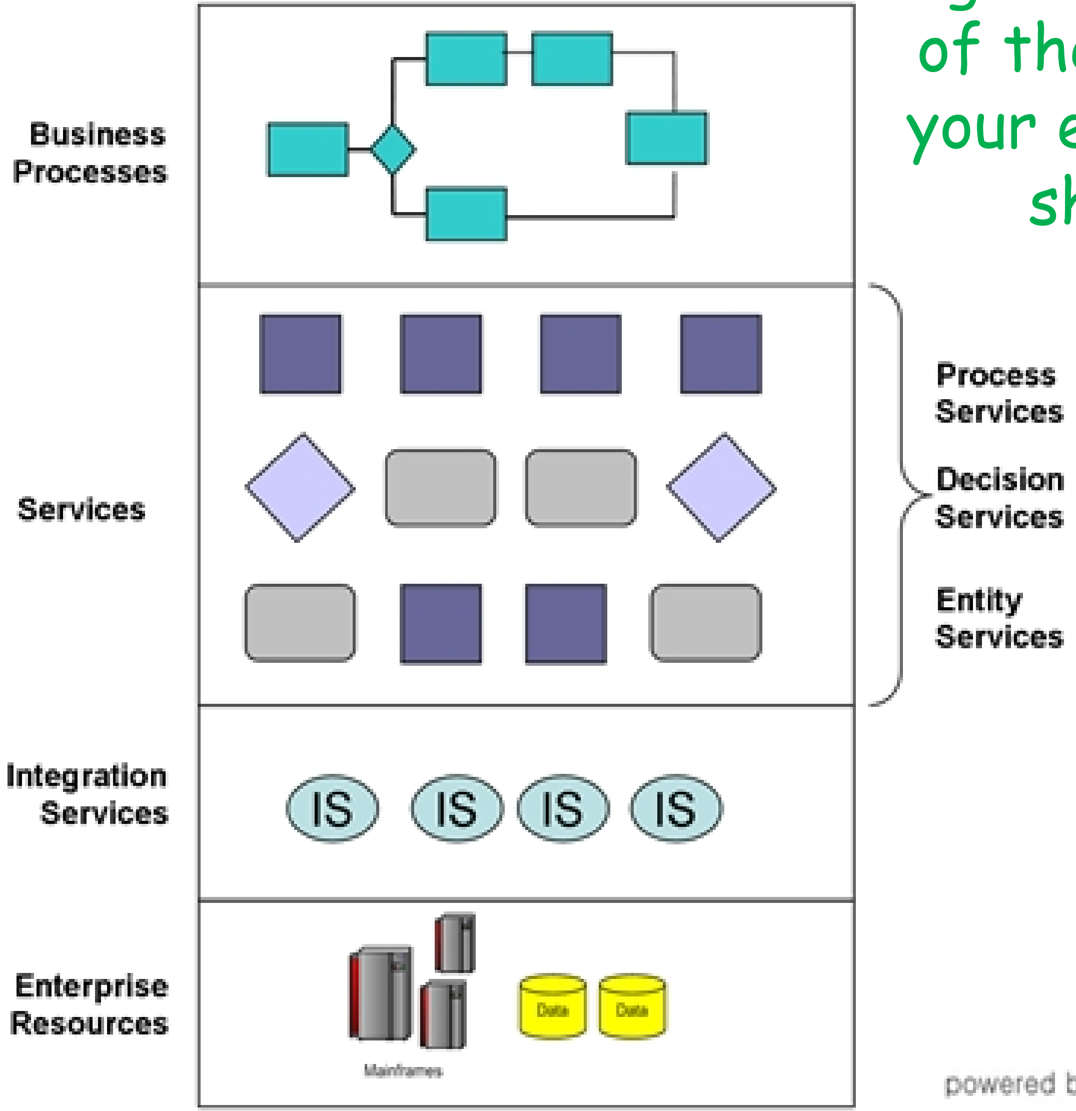
- Test
- Code
- Build
- Document

Subject areas correspond to business process areas. They provide some clues about which data needs to be managed together.

Infrastructure

Layered SOA

The business process model is a good **predictor** of the **services** your enterprise should have.



Infrastructure
Standard Operating Environment

- Hosting (Host hardware / operating system)
- Storage (Database, Back-Up, Retention)
- Messaging

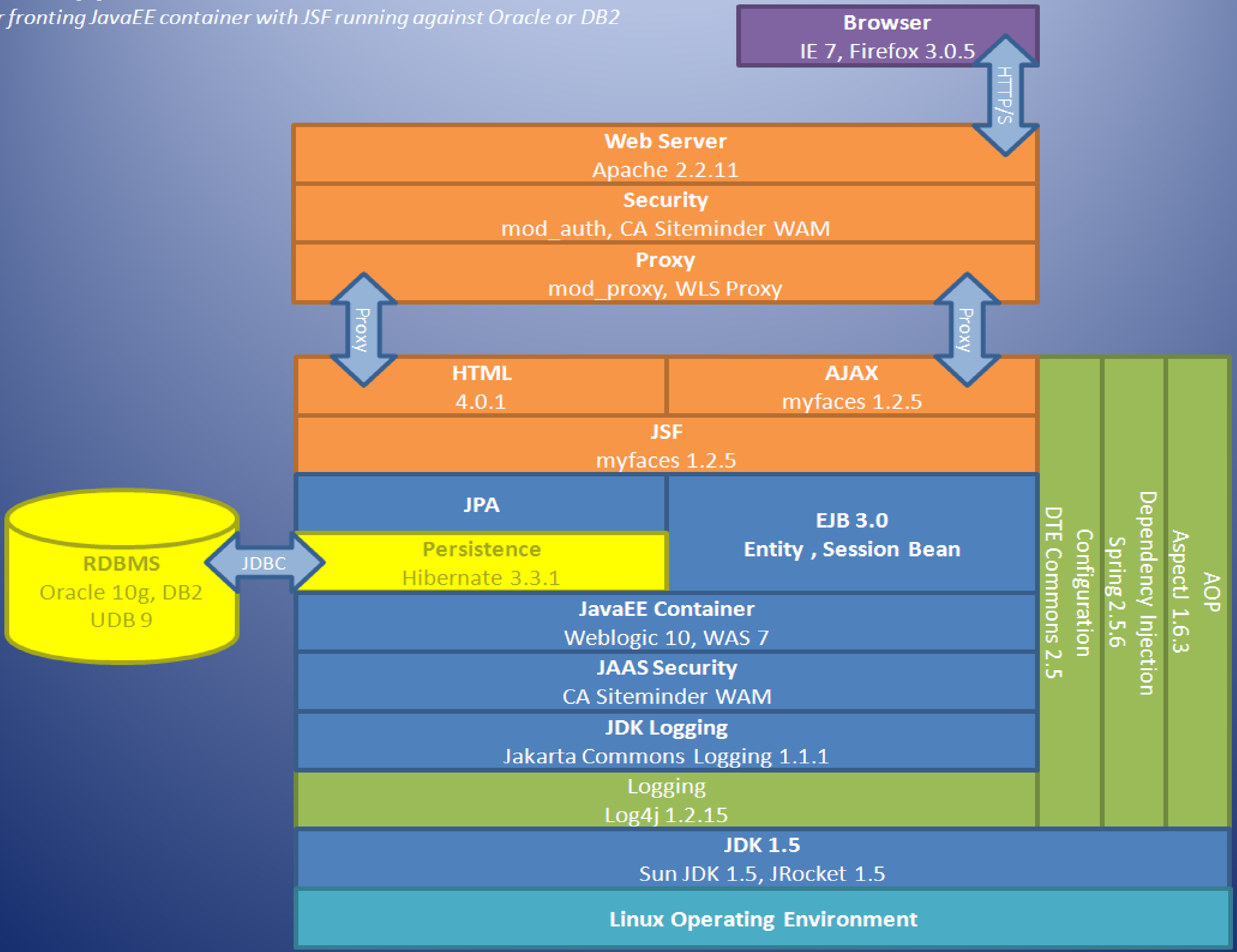
- User Workstations
- Developer Workstations
- Special Workstations

- Web Application Stack
- Management Reporting Stack
- Transaction Processing Stack

Create a **wiki** so that everyone can maintain their own section.

Java Web Application – Non Disaster Tolerant

Web server fronting JavaEE container with JSF running against Oracle or DB2



Standard Architectures for Standard Problems

One project at a time

- Enterprise Architecture
  is like **urban planning**.
- There is a large-scale plan,
  but it is built
  one project at a time.
- How do we do that?

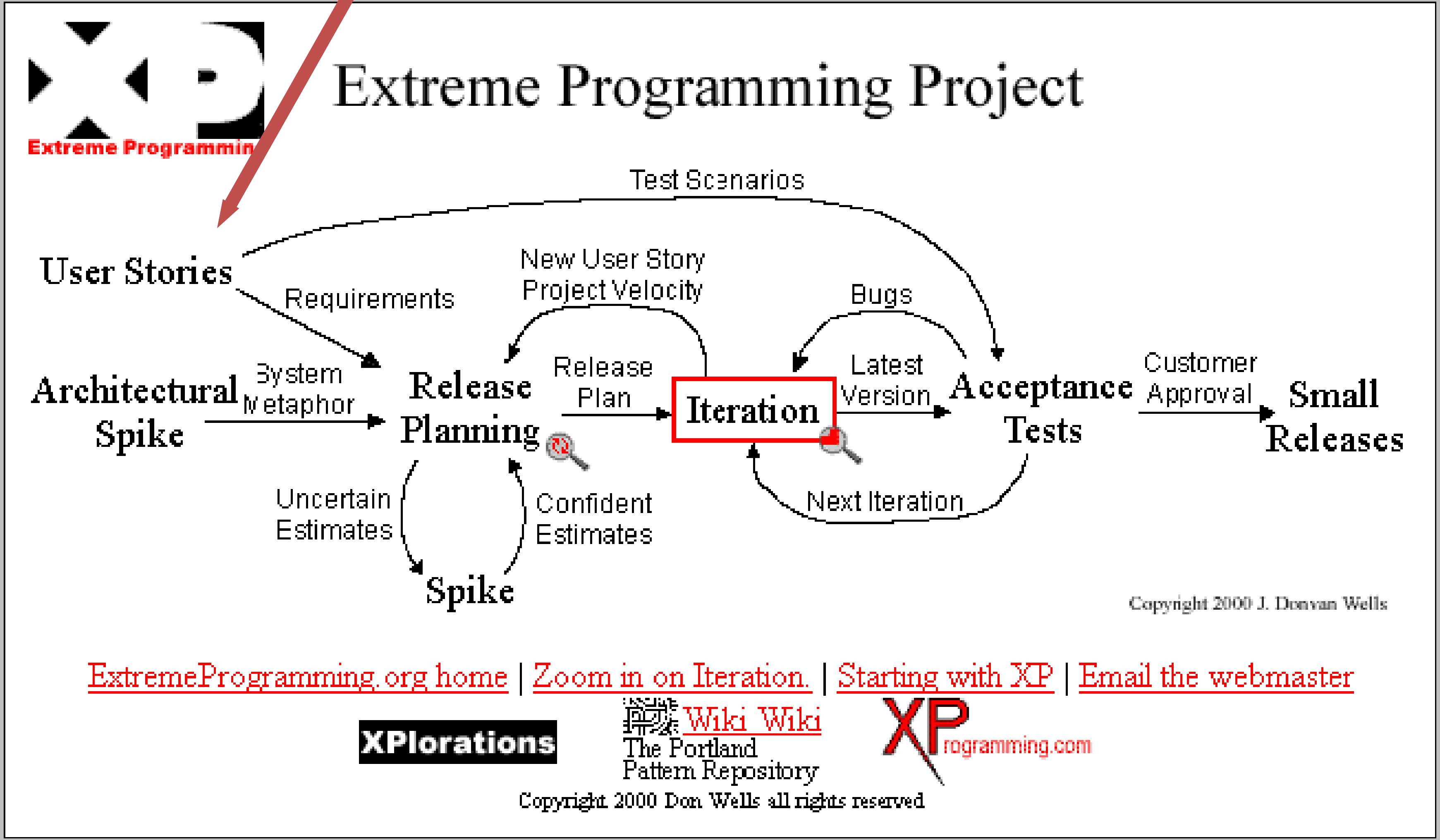
from the **Big Picture** to the **Gig Picture**

- Business processes communicate by sending **messages**.
- The **messages** indicate essential **entities** and implied **capabilities**.
- For project planning, **the message is the feature**.

How are business process interfaces identified?

- **Business level interfaces** are messages sent between business sub-processes.
- A handy technique:
  Express each message in a **complete sentence**.
- Look for four kinds of message:
  - **Request**: Please process work order **W**.
  - **Response**: Work order **W** was completed at time **T**.
  - **Notice**: Person **P** is no longer certified in skill **S**.
  - **Query**: Status **S** = What is the status of work order **W**?

Each message is a user story.



This diagram is cut-and-paste from the www.extremeprogramming.org web site.

Details, Details

Business Process Proxy
(sending application)

Detect business event

Detecting the business event is relatively easy.
The business application will have an API
that publishes the event. Worst case, one needs
to put a trigger on some table to publish the event.

Assemble message object

The message object has a message-sentence
(or some handle) and a few objects representing
the domain objects referenced in the sentence.
These are assembled into the message object.
This follows a simple design pattern.

Deliver Message (asynchronous)

Use a standardized **asynchronous** delivery system.
Simply invoke "Publish message M on topic T"
and let the infrastructure take care of the details.

Receive and dispatch message

Receiving the message
also follows a simple design pattern.

Initiate business process

This is typically where the **hard-work** happens.
The development team needs to translate the
domain objects to what-ever form is required by
the receiving application's API.
Sometimes we have to design and build an API
from scratch (old legacy application).

Business Process Proxy
(receiving application)

Release planning is a multi-person game.

<u>Planning and Budgeting</u>

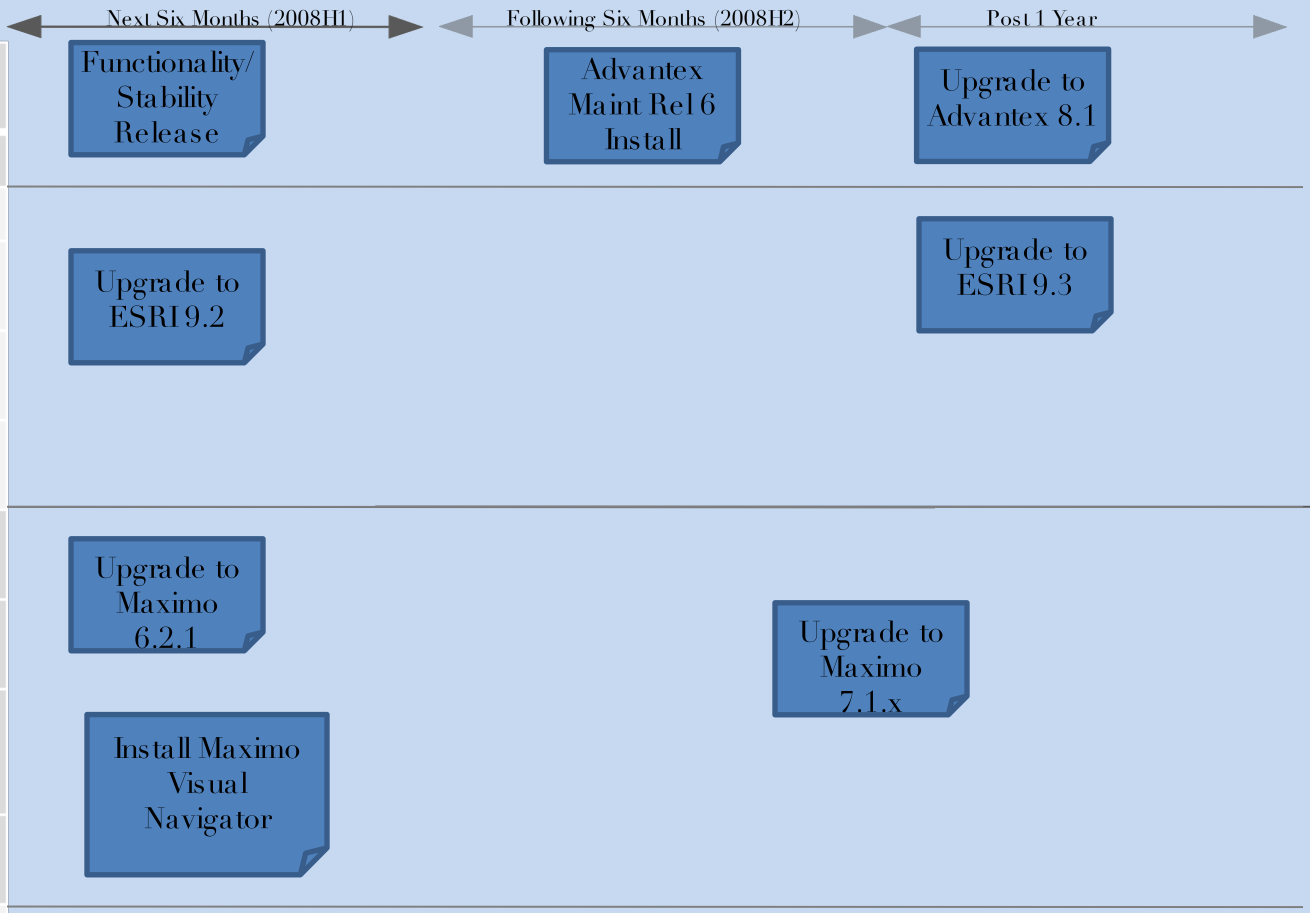
- Business Process - owner / governor
- Application Portfolio - owner /governor
- Component Library - architect / planner

<u>Design and Construction</u>

- Enterprise Architect - architect / planner / coordinator
- Project Manager - planner / coordinator
- Software Developer - build / test / deploy / maintain

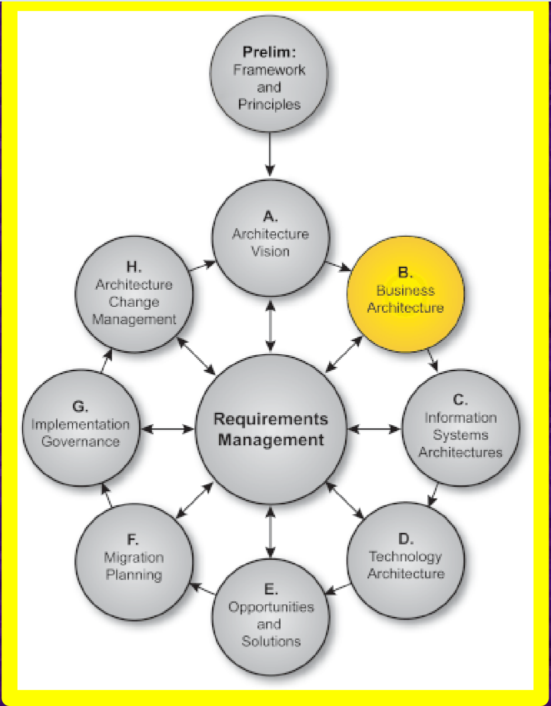
Roadmap

Mobile Field Force Management	Advantex
Crew Management	Advantex
Circuit Design	ESRI
Functionality Used by Gas	ESRI
Functionality Used by Gas	ESRI
Field Services - Design	Miner & Miner Designer
Procurement - Material Requisition	Maximo
Field Services - W/O Tracking	Maximo
Analytical Support - Distribution Management	Maximo
Inv. Mgmt - Installed Equipment	Maximo
…	…



Four Dimensions:

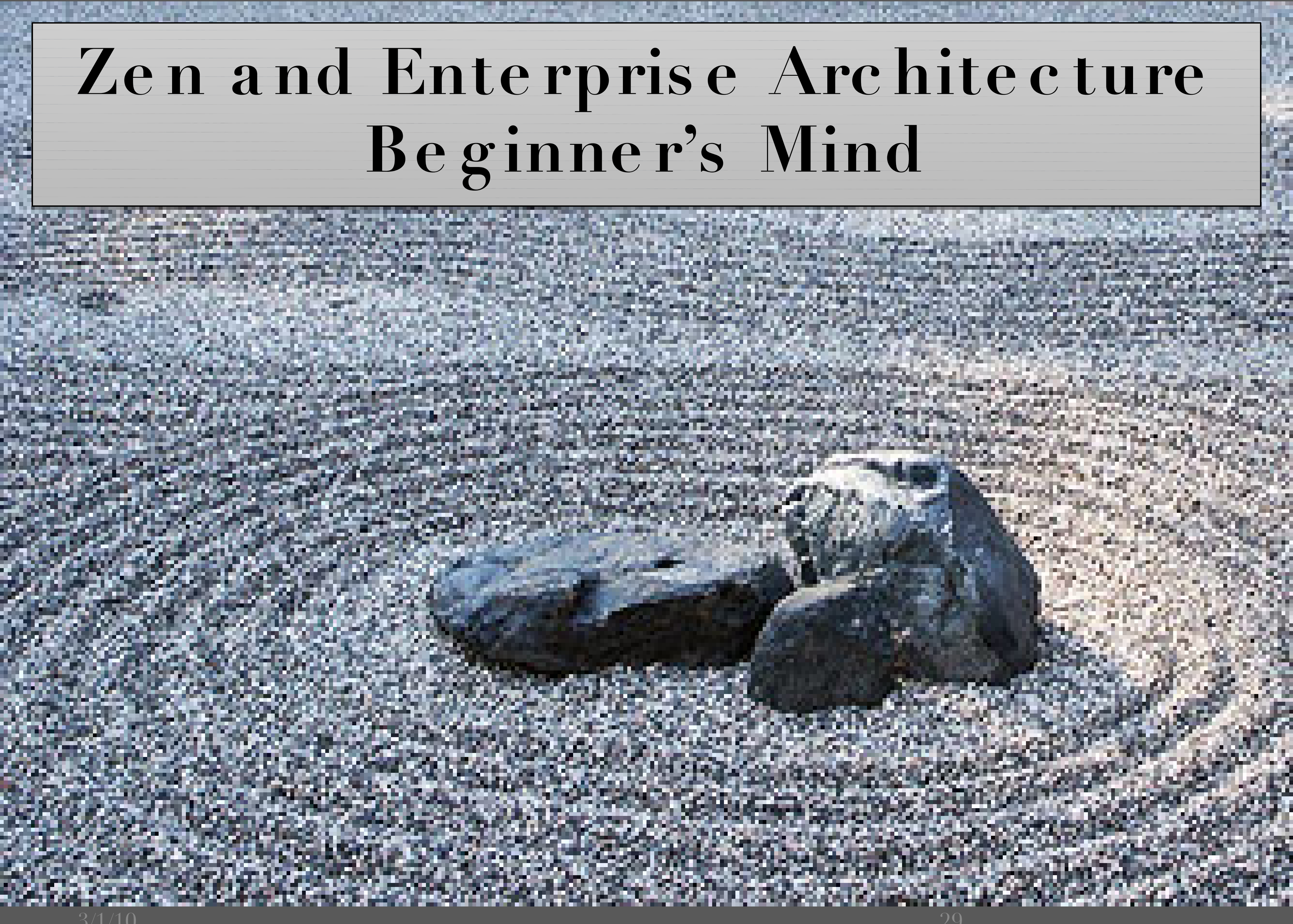
1. Process → Business Architecture
2. Information → Data Architecture
3. Infrastructure → S.O.E.
4. Time → Roadmaps


Prelim: Framework and Principles
A. Architecture Vision
B. Business Architecture
C. Information Systems Architectures
D. Technology Architecture
E. Opportunities and Solutions
F. Migration Planning
G. Implementation Governance
H. Architecture Change Management
Requirements Management

Links

- **TOGAF:**
  - http://www.togaf.info/
  - http://www.opengroup.org/
- **Zackman Framework:**
  - http://www.zachmaninternational.com/
  - http://eacoe.org/
- **Value Chain:**
  - James Martin: ISBN: 9780814403150
    Martin, James (1995). *The Great Transition: Using the Seven Disciplines of Enterprise Engineering*
  - Ken Orr: Business Enterprise Architecture Modeling http://www.cutter.com/workshops/04.html
  - Michael Porter: http://www.isc.hbs.edu/

Zen and Enterprise Architecture Beginner's Mind



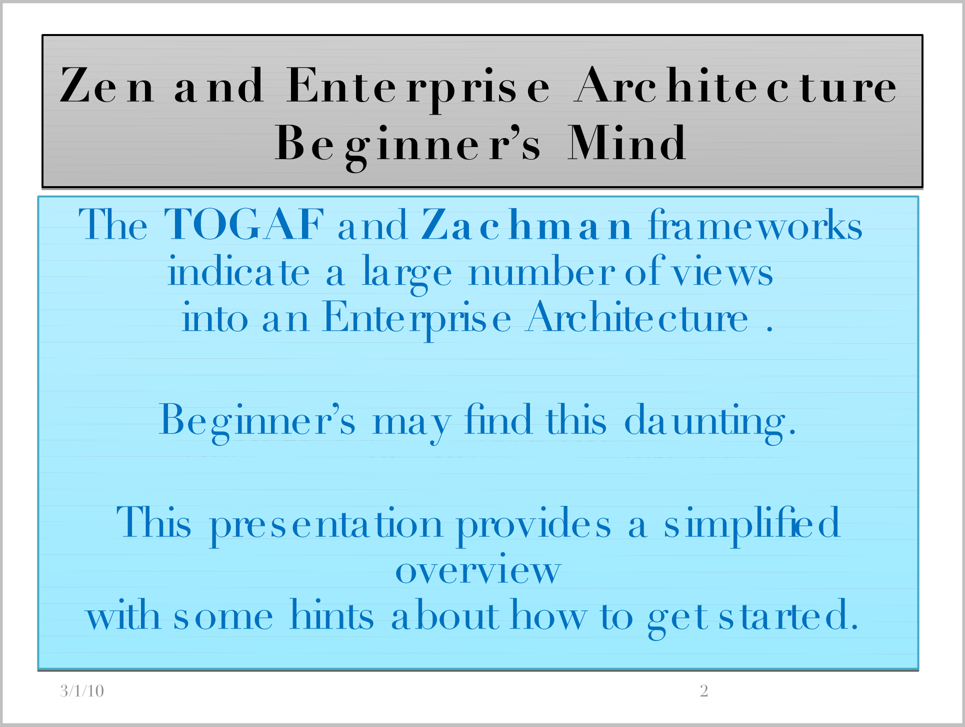
Zen and Enterprise Architecture
Beginner’s Mind

The title of my talk is
“Zen and Enterprise Architecture,
Beginner's Mind”

What does my talk have to do with Zen,
absolutely nothing.

Why is there a rock garden on this slide?
The word Zen is in the title,
so the rock garden is required.

It's in the by-laws someplace.



Zen and Enterprise Architecture Beginner's Mind

The **TOGAF** and **Zachman** frameworks indicate a large number of views into an Enterprise Architecture .

Beginner's may find this daunting.

This presentation provides a simplified overview with some hints about how to get started.

3/1/10 2

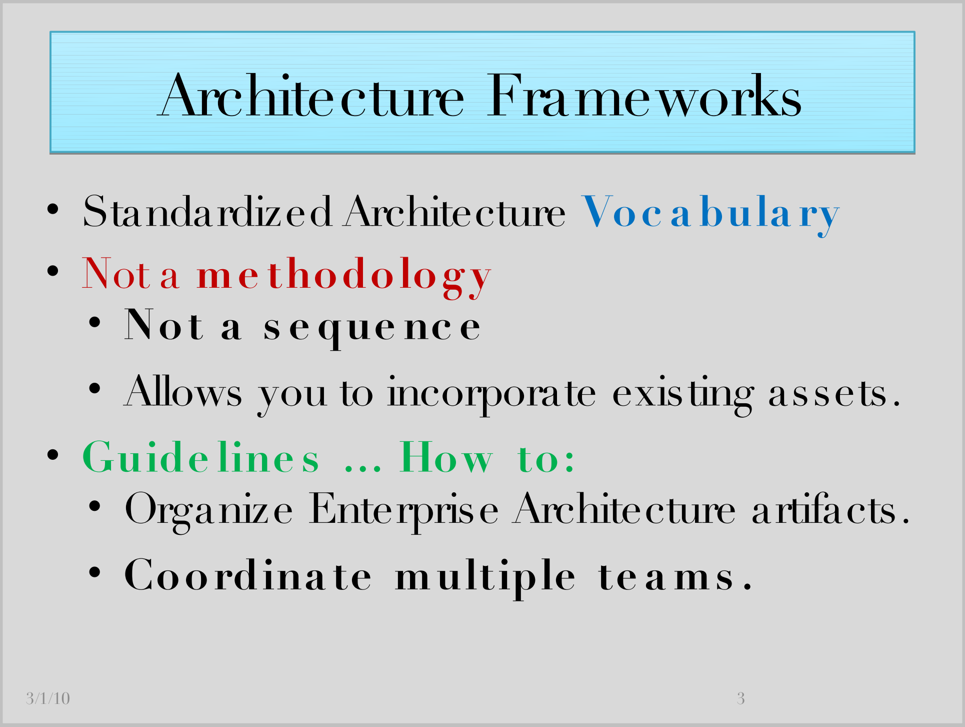
What is my talk really all about?

Architecture Frameworks
Indicate a large number of views and processes.

Beginner's may find this a bit daunting.

In this talk I attempt
to boil enterprise architecture down
to a few fundamental techniques and ideas.

These techniques provide a good place to start.


Architecture Frameworks

- Standardized Architecture Vocabulary
- Not a methodology
  - Not a sequence
  - Allows you to incorporate existing assets.
- Guidelines ... How to:
  - Organize Enterprise Architecture artifacts.
  - Coordinate multiple teams.

3/1/10 3

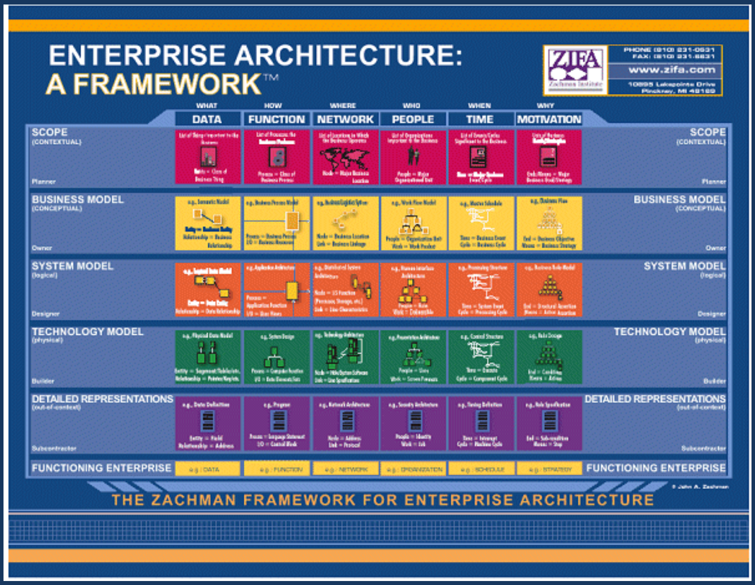
What is an architecture framework?

Basically a framework provides a **vocabulary** for talking about enterprise architecture.

It is not a methodology really.
The procedures that come with a framework are really guidelines, not instructions.

Enterprise Architecture involves coordinating multiple teams.
TOGAF provides some ideas about how to do that.
But the details are left to your group.

Zachman Framework



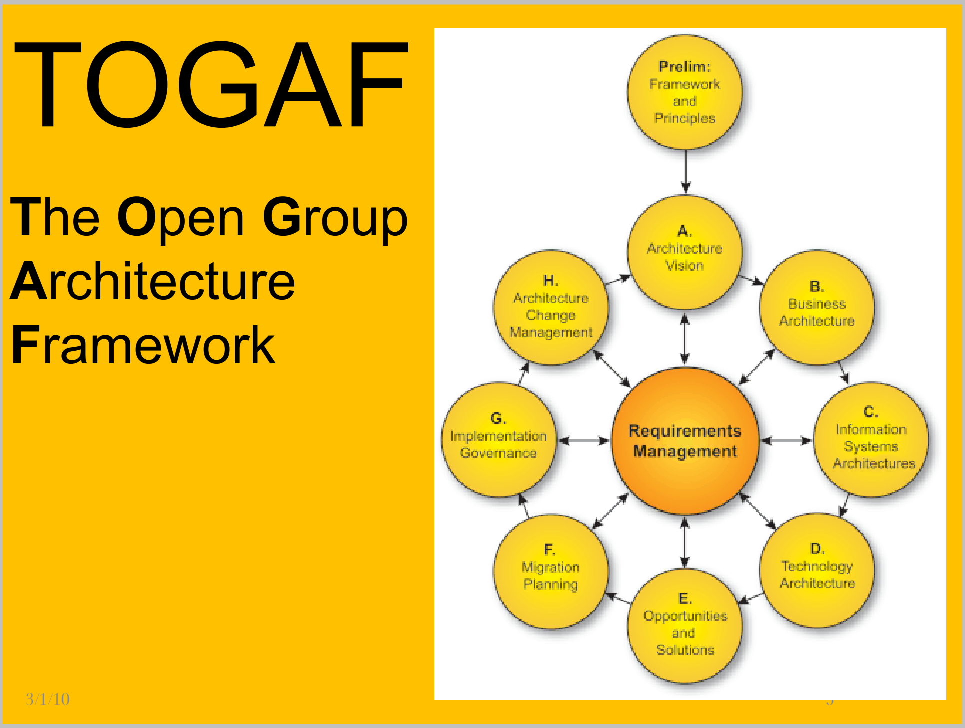
3/1/10 4

This poster for a version the Zachman Framework nicely illustrates the idea that you need many "views" or audience-centric presentations of the Enterprise Architecture.

Enterprise Architects spend most of their time communicating.

Different audiences have different concerns.

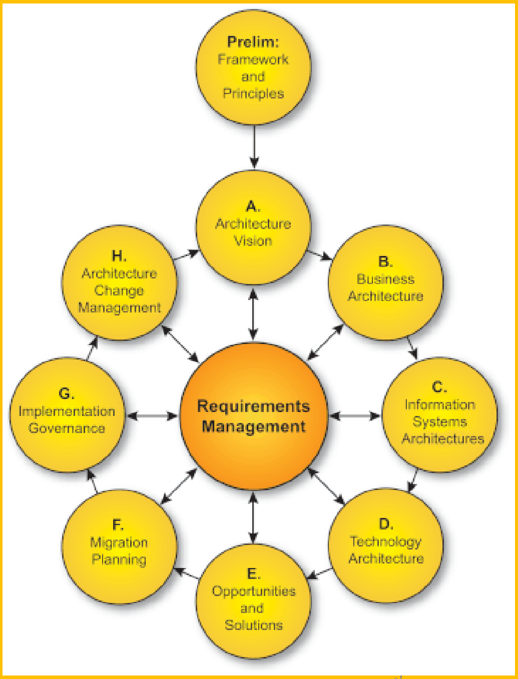
As a result,
there can be a lot of different "**views**" or renderings of the architecture.



TOGAF

The Open Group Architecture Framework

3/1/10

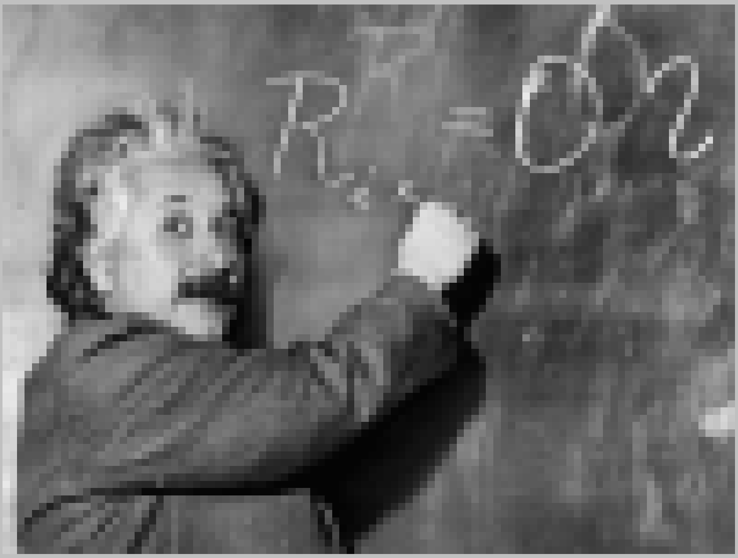

Prelim: Framework and Principles
A. Architecture Vision
B. Business Architecture
C. Information Systems Architectures
D. Technology Architecture
E. Opportunities and Solutions
F. Migration Planning
G. Implementation Governance
H. Architecture Change Management
Requirements Management

This poster for the TOGAF framework indicates that there are a lot of different **processes** involved in developing or working with an Architecture Framework.

The arrows on the diagram might make you think that there is a sequential process involved.
However, the TOGAF framework does not really prescribe the sequence of the processes.
In practice, all of these processes are going on **simultaneously**
And iteratively.

The arrows do represent a good **conceptual sequence**.

A (model) should be as simple as possible, but no simpler.



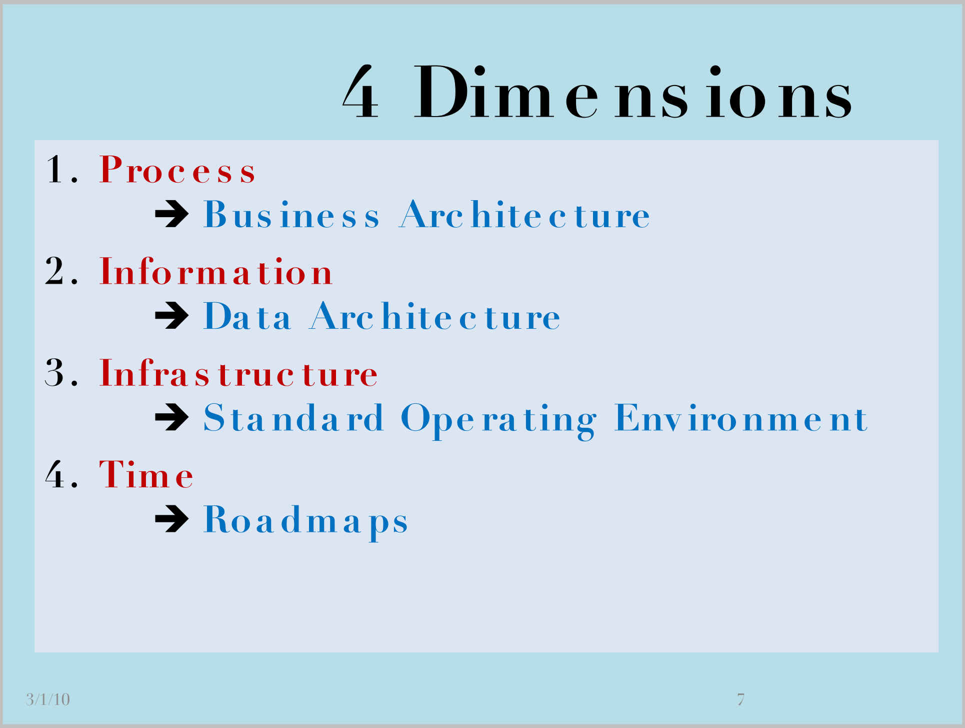
3/1/10

Can these frameworks be simplified?

Not really!
Each component is there for a reason.

But, I can show a minimal subset
that provides a good-enough place
for a beginner to get started.

Additional components can be added
when the situation
and organization
calls for them.



4 Dimensions

1. Process
   - ➔ Business Architecture
2. Information
   - ➔ Data Architecture
3. Infrastructure
   - ➔ Standard Operating Environment
4. Time
   - ➔ Roadmaps

3/1/10 7

Zachman and TOGAF are very good frameworks.

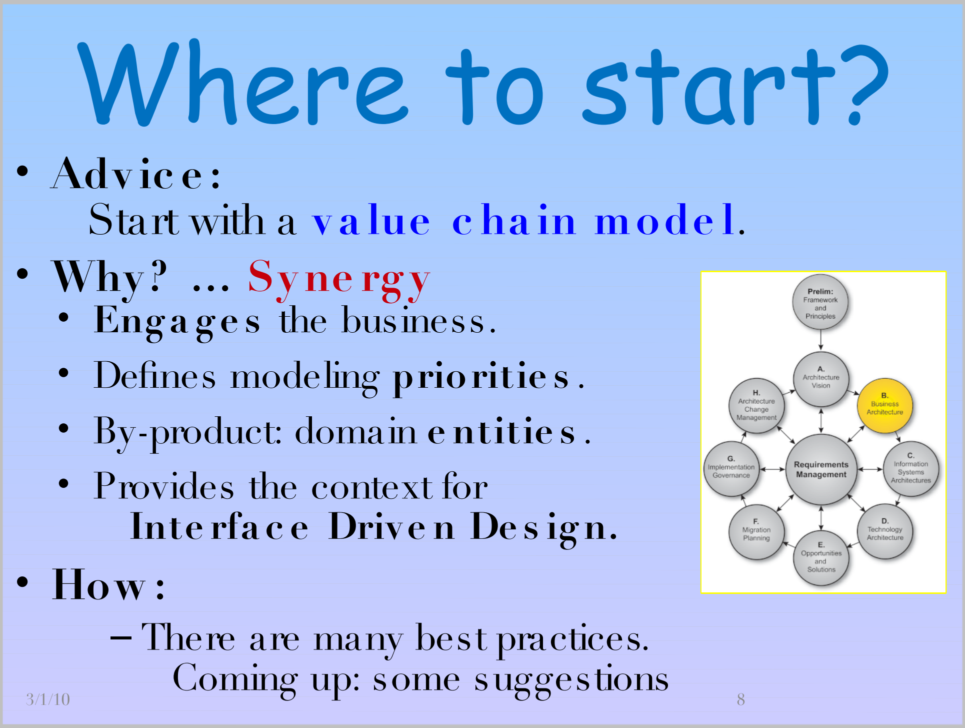
But, when I look at them, I worry that beginners might find them too complicated and give up or get lost.

So I am going to provide a simple framework with just four dimensions.

If you already have some experience with systems analysis, these four dimensions should be very familiar.

I am going to show how to apply analysis techniques at the enterprise level.

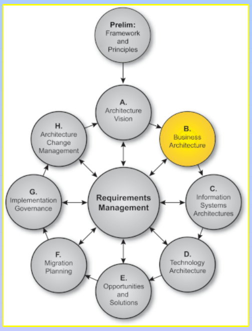
There are some differences from the way things are typically done at the project level.



Where to start?

- **Advice:**
  Start with a **value chain model**.
- **Why? … Synergy**
  - **Engages** the business.
  - Defines modeling **priorities**.
  - By-product: domain **entities**.
  - Provides the context for
    Interface Driven Design.
- **How:**
  – There are many best practices.
  Coming up: some suggestions

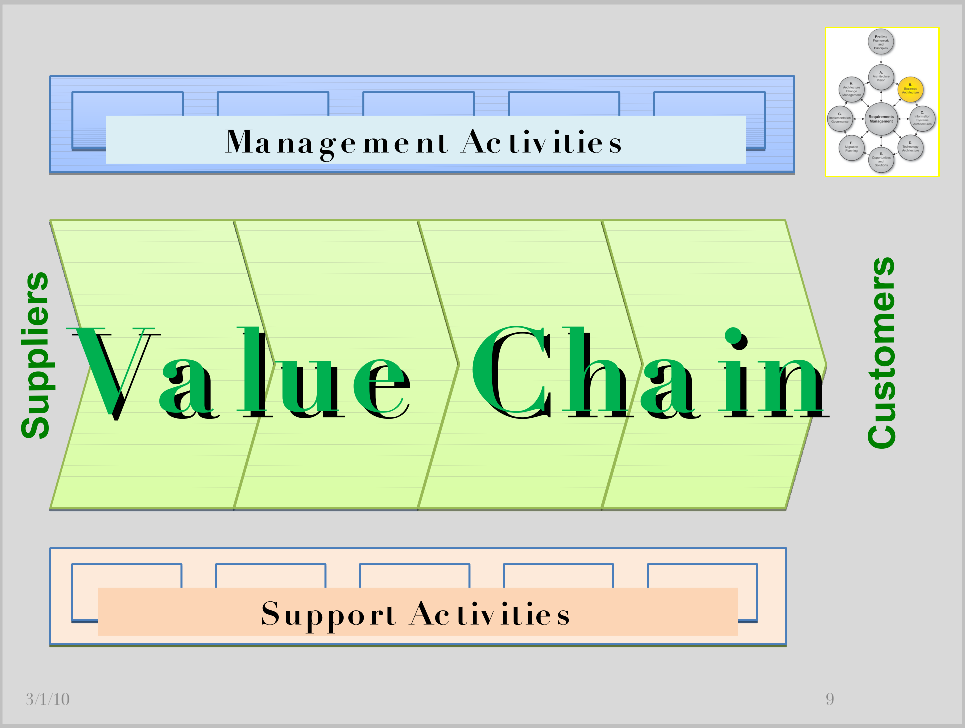
3/1/10 8


Prelim: Framework and Principles
A. Architecture Vision
B. Business Architecture
C. Information Systems Architectures
D. Technology Architecture
E. Opportunities and Solutions
F. Migration Planning
G. Implementation Governance
H. Architecture Change Management
Requirements Management

A enterprise is deep and wide.
Where should we start?

If you can, the best place to start is with a **value chain model**.
A value chain model is kind of high-level business process model.

There are many advantages to starting with this kind of model.
But the main one might be that it helps to define modeling priorities.



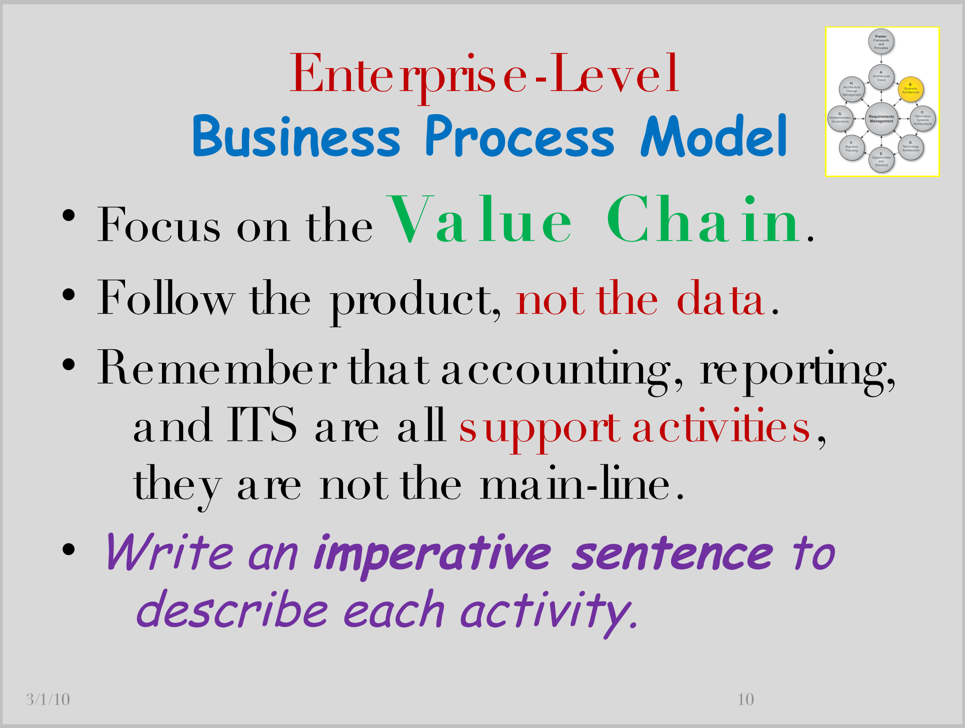
How do you focus your analysis on the things that matter?

Focus on the "value chain."
The value chain includes the activities
that are most directly concerned
with producing the enterprise's product or service.

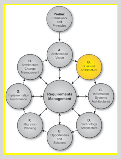
As IT people, we sometimes become side-tracked
by support activities or management activities
because that is where the ITS investment might be
– or the latest question.

But we need to resist the tendency to be side-tracked
and focus on the value-chain.

Use the Management and Support layers
as a **parking place** for those activities.



Enterprise-Level Business Process Model



- Focus on the **Value Chain**.
- Follow the product, not the data.
- Remember that accounting, reporting, and ITS are all support activities, they are not the main-line.
- *Write an **imperative sentence** to describe each activity.*

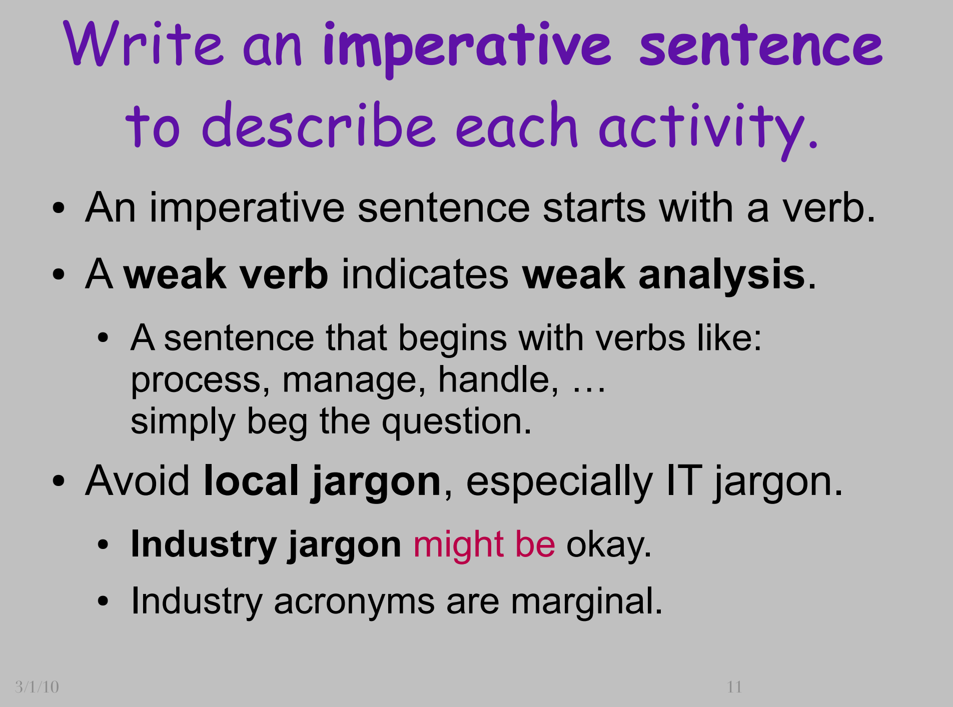
3/1/10 10

Here are some **general guidelines** that always apply when you are doing business process analysis.

• Focus on the business **product flow, not the data flow**.
As IT people we tend to focus on the data but unless the enterprise product is data, that is the often wrong place to focus when doing process analysis and definitely the wrong place to focus when doing enterprise analysis.

• **Frankly, management activities and support activities are a distraction** when you are doing enterprise modeling.
Avoid spending time in those areas.
Use the management and support areas as a **parking place** for those activities.

The last rule here keeps you out of trouble,
and enables synergy:
Describe each business process using an **imperative sentence**.
If you have trouble writing that sentence, there is something wrong.



Write an ***imperative sentence*** to describe each activity.

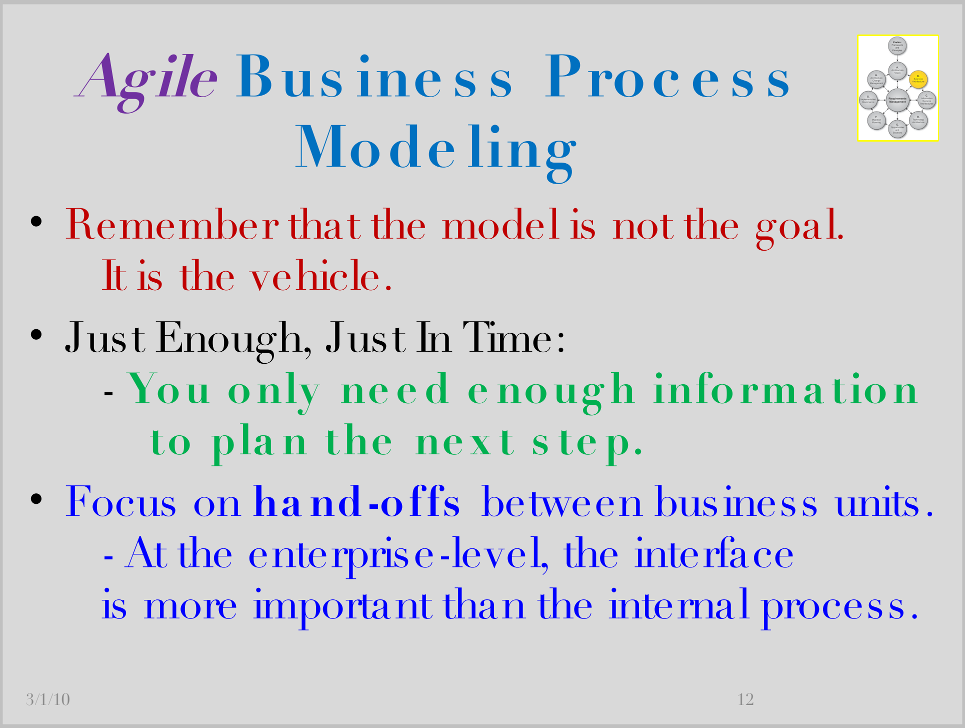
- An imperative sentence starts with a verb.
- A **weak verb** indicates **weak analysis**.
  - A sentence that begins with verbs like:
    process, manage, handle, …
    simply beg the question.
- Avoid **local jargon**, especially IT jargon.
  - **Industry jargon** might be okay.
  - Industry acronyms are marginal.

When we are working on a white board and trying to fit annotation on a diagram, we tend to be **overly terse**.

A diagram without a “bill of materials” is just a “stage prop.”

Good modeling tools provide a place to put text that ties back to the diagram.
You can put the imperative sentence there.

There is a trade-off between innovation and clarity.
Avoid writing sentences that prevent people from thinking outside the box.



Agile Business Process Modeling

- Remember that the model is not the goal. It is the vehicle.
- Just Enough, Just In Time:
  - You only need enough information to plan the next step.
- Focus on **hand-offs** between business units.
  - At the enterprise-level, the interface is more important than the internal process.

3/1/10 12

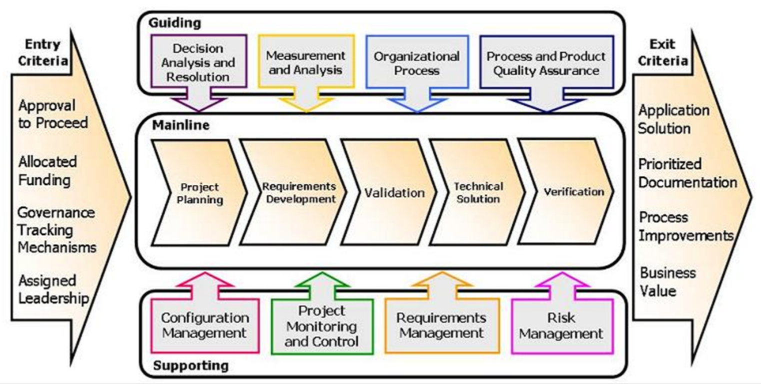
How do you stay agile when you are doing something as big as a model of the enterprise?
How do you avoid **analysis paralysis**?
Actually there is some art involved.
How much detail your team needs is often a judgement call.

Generally, you only need enough information
to plan the next step.

When you are doing Enterprise Architecture,
the details of the various business processes
are **almost irrelevant**.

Focus on the **hand-offs** that occur when a business process hands a partial-product off to another or when a business process delegates some work to another process.

Example: ITS as an enterprise



You cannot model in a vacuum.

Different organizations will have different terms for similar activities.

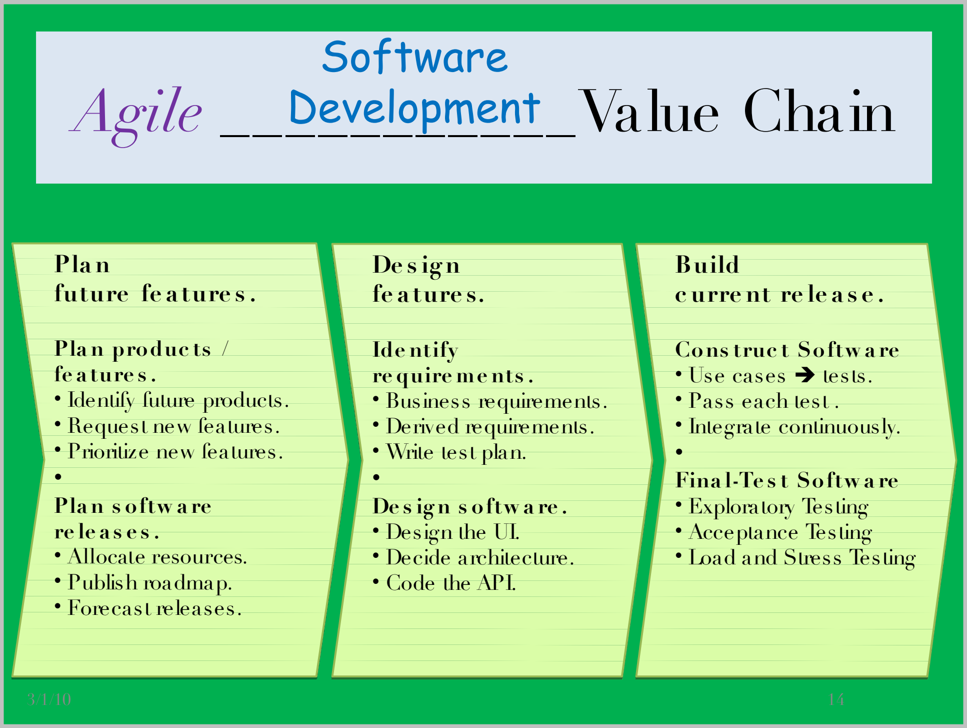
There will be differences in emphasis.

Boxes will be ordered and sized according to the local point of view.

3/1/10 13

Here is a model we use at DTE Energy.
Other enterprise ITS groups are almost certainly performing the same activities. But, those groups probably uses different names for many of these activities and may organize them somewhat differently.

Three points:
[1] There is no right answer.
[2] Names are important.
[3] You cannot construct a model without talking to the process owners because you have to get the names right.
If you don't get the names right, people will simply not recognize the model as pertinent.



Agile Software Development Value Chain

Plan future features.

Plan products / features.
- Identify future products.
- Request new features.
- Prioritize new features.
-

Plan software releases.
- Allocate resources.
- Publish roadmap.
- Forecast releases.

Design features.

Identify requirements.
- Business requirements.
- Derived requirements.
- Write test plan.
-

Design software.
- Design the UI.
- Decide architecture.
- Code the API.

Build current release.

Construct Software
- Use cases ➔ tests.
- Pass each test.
- Integrate continuously.
-

Final-Test Software
- Exploratory Testing
- Acceptance Testing
- Load and Stress Testing

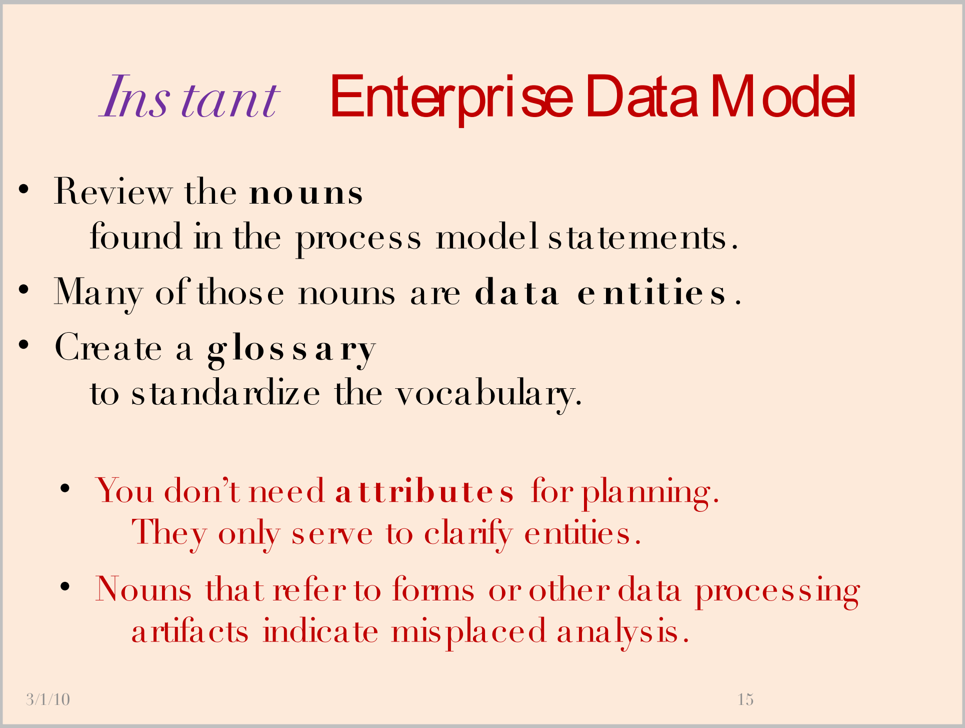
3/1/10 14

This slide is here to demonstrate that there are **value chains within value chains.**

If you have an enterprise level model,
you don't need to create lower level models
until the time comes that you need them.
That is another way of being agile.

However, there is a small risk.
The risk is that you may need to "**re-factor**"
the higher-level model.

But one of the rules of being agile is that you need
to accept, accommodate, and facilitate re-factoring.
It is a simple fact that you cannot know everything.



Instant Enterprise Data Model

- Review the **nouns**
  found in the process model statements.
- Many of those nouns are **data entities**.
- Create a **glossary**
  to standardize the vocabulary.

  - You don't need **attributes** for planning.
    They only serve to clarify entities.
  - Nouns that refer to forms or other data processing
    artifacts indicate misplaced analysis.

3/1/10 15

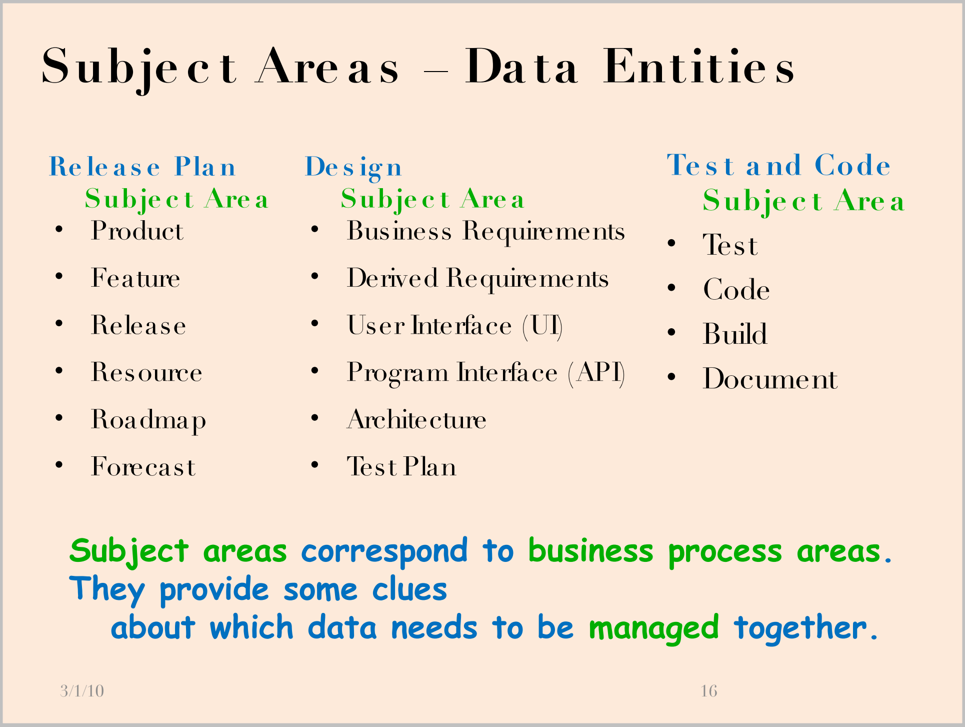
Earlier I said that you should write an **imperative sentence**
to describe each business activity.

If you do that, it will keep you out of trouble,
and it will set you up to create an Information Architecture
as a **quick by-product**.
In fact, many people do process models and data models simultaneously.

Three steps:
[1] **Underline** the nouns in the imperative sentences.
[2] Many of those nouns represent **business objects** (data entities)
that are sufficiently interesting that the business records data about them.
[3] Create a **glossary** that defines how the business uses those terms.
The process of creating and reviewing the glossary will smoke-out problems
with synonyms and homonyms.

Time management:
- Attributes are not important at this level of analysis. They only serve to clarify
the nature of an data entity.

Quality Check:
- Nouns that use the names of forms, acronyms, or data processing objects
may indicate a problem with the process model.


Subject Areas – Data Entities

Release Plan Subject Area
- Product
- Feature
- Release
- Resource
- Roadmap
- Forecast

Design Subject Area
- Business Requirements
- Derived Requirements
- User Interface (UI)
- Program Interface (API)
- Architecture
- Test Plan

Test and Code Subject Area
- Test
- Code
- Build
- Document

Subject areas correspond to business process areas.
They provide some clues
about which data needs to be managed together.

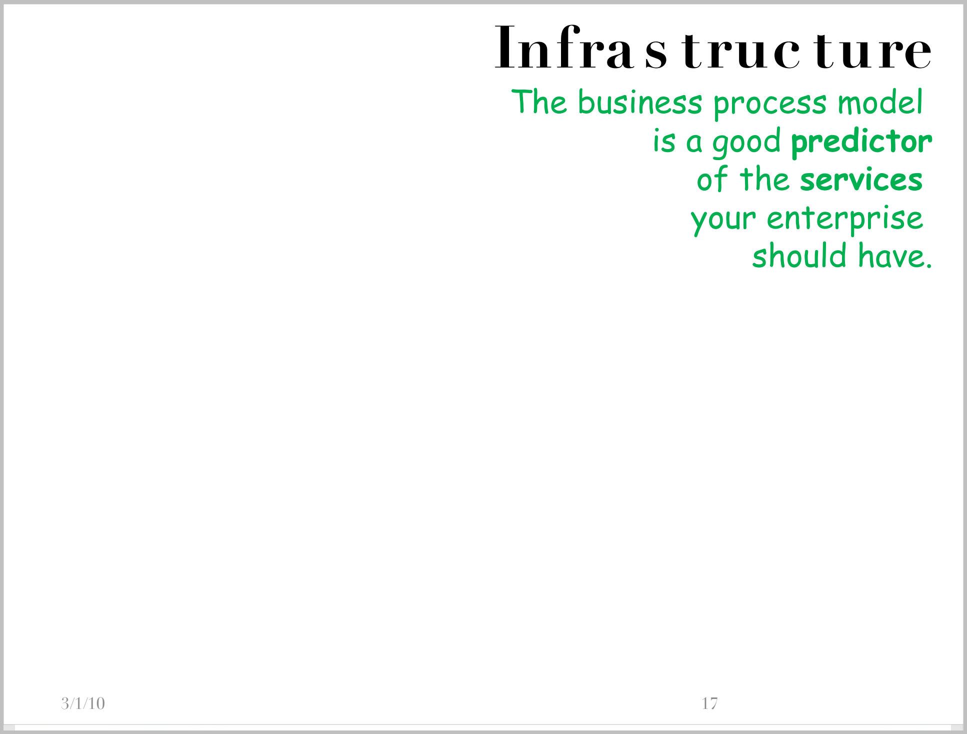
3/1/10 16

This is an **optional step**,
but it typically only takes about an hour
and it is generally worth it.

An enterprise data model typically contains
around **80 data entities**. (kernel entities in ERA terms)
A small one might contain about 20.

You can gain some insight into your Information Architecture
by grouping the data entities into **subject areas**.
Subject areas typically correspond to business process areas.

Subject areas provide some clues about
which data needs to be **stored together**
and which data needs to be **managed together**.



Infrastructure

The business process model is a good **predictor** of the **services** your enterprise should have.

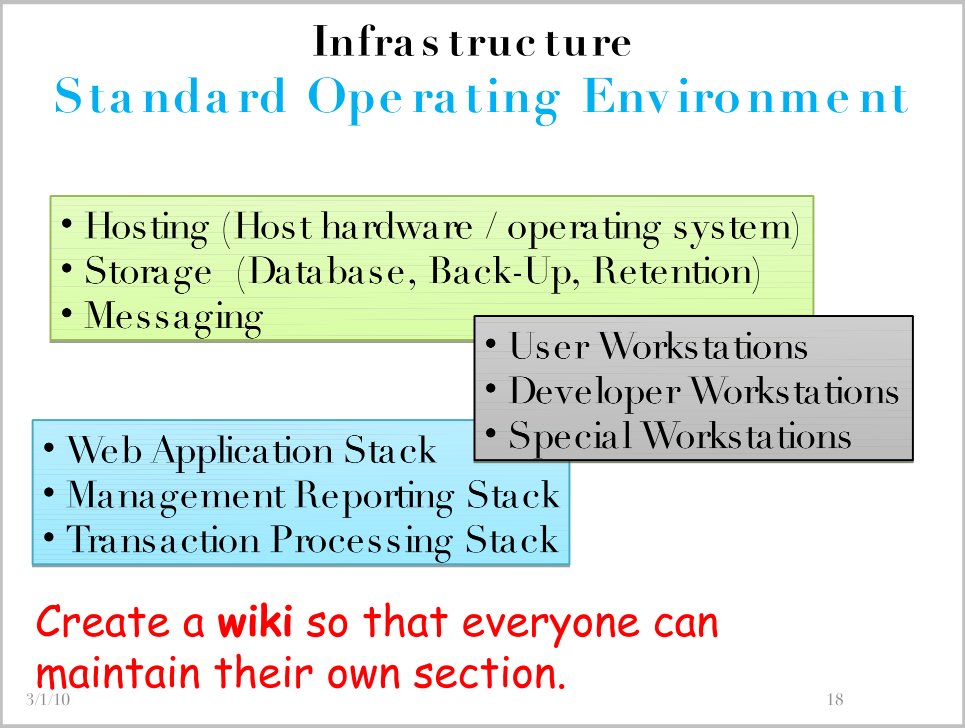
3/1/10 17

I owe Mike Rosen a thank you for this diagram.

The point I want to make here is that …
you can also derive
a plan for the **service architecture**
from the **business architecture**.

• The business processes in your enterprise's business architecture,
may be represented by
process services and **decision services**.

• The **subject areas** in your data architecture
should be packaged as **Entity Services**.


Infrastructure
Standard Operating Environment

- Hosting (Host hardware / operating system)
- Storage (Database, Back-Up, Retention)
- Messaging

- User Workstations
- Developer Workstations
- Special Workstations

- Web Application Stack
- Management Reporting Stack
- Transaction Processing Stack

Create a **wiki** so that everyone can maintain their own section.

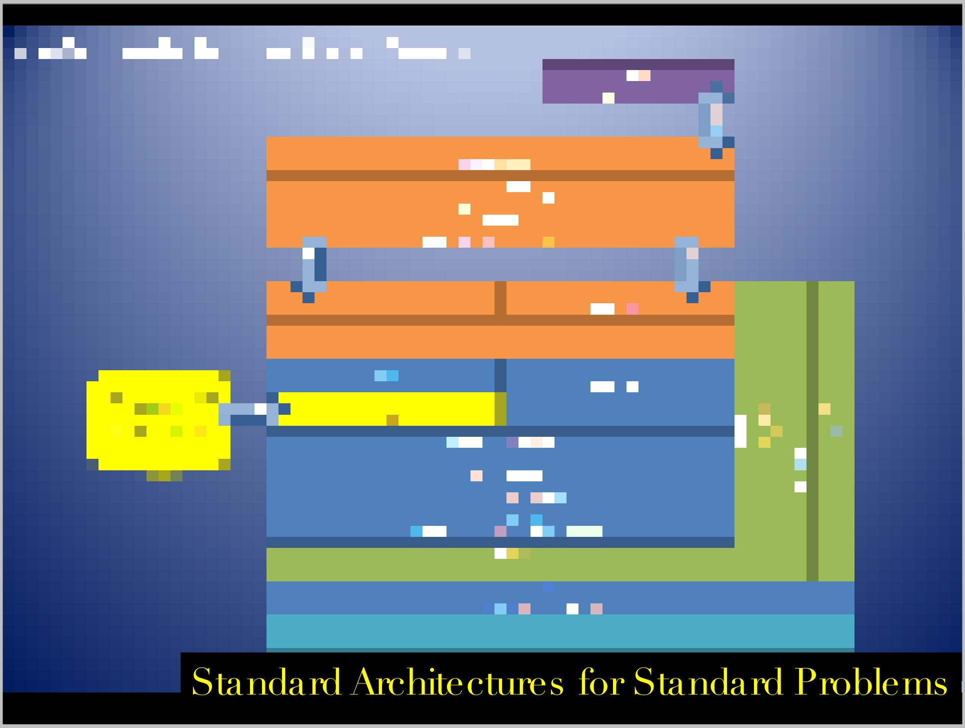
3/1/10 18

Of course, service architecture is **just one view** of the IT infrastructure.

An enterprise of any size will need a **Standard Operating Environment** (SOE).
Having a standard environment reduces problems with **interoperability**.
It also means that projects don't need to design the environment.

You need to get all of the technical practices on board.
I recommend that you create an private enterprise architecture **wiki** inside your firewall.
Each of the technical practices can maintain their own section of the wiki.

You may need to establish some **guidelines** and occasional peer reviews about to shepherd the wiki content.



Here is an example
of a **standard stack**
for a **standard problem**.

This is one of the standard stacks
that our Enterprise Architecture Group specifies at DTE Energy.
You probably can't see the small print,
but the small print specifies which specific implementations
of a component are allowed.

There is another document behind this one
that gives **guidelines** for making choices
when more than one implementation is supported.

For example, two application servers are supported.
Question: Which one should you use?
Answer: It depends.

The guidelines help sort out each case.
In some cases, it is a judgement call
and the Enterprise Architecture Group will make the decision
after consulting all of the interested parties and other experts.

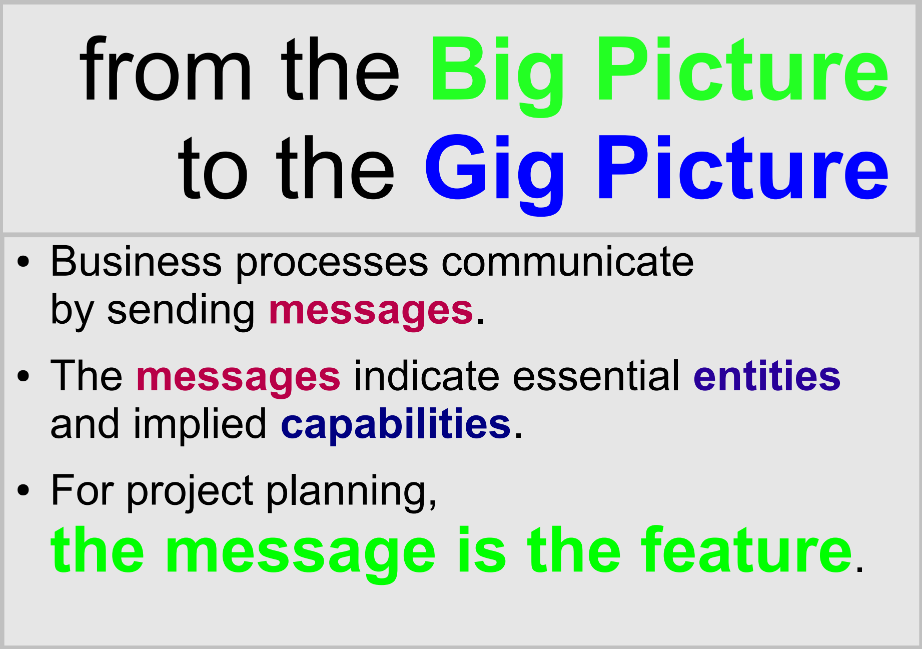
One project at a time

- Enterprise Architecture
  is like **urban planning.**
- There is a large-scale plan,
  but it is built
  one project at a time.
- How do we do that?

3/1/10 20

Now we are moving to the fourth dimension,
the “time” dimension.

- Enterprise Architecture
  is like **urban planning.**
- There is a large-scale plan,
  but it is built
  one project at a time.
- How do we do that?



from the Big Picture to the Gig Picture

- Business processes communicate by sending **messages**.
- The **messages** indicate essential **entities** and implied **capabilities**.
- For project planning, **the message is the feature**.

How do we go from the
Big Picture to the
Gig Picture?

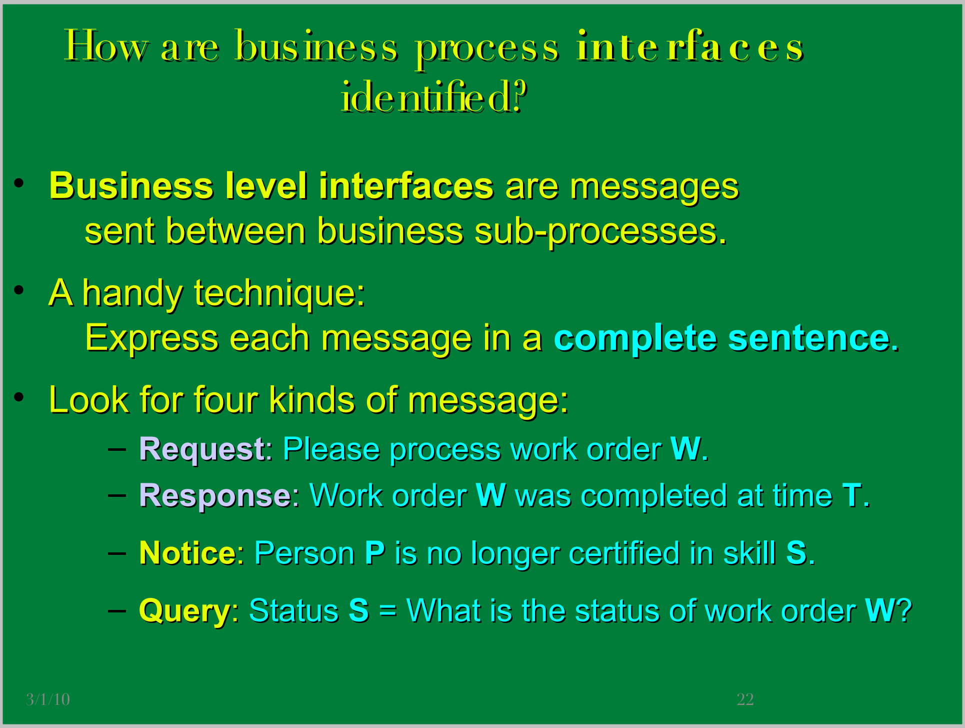
I've discovered a trick.

You can find almost all of the information
that you need for planning purposes
by studying the hand-off's
between business processes.

Business units delegate work to other units
by sending messages.

In the old days, these were seen on sheets of paper.

In a modern system, the messages will be in data. But they are still there.
(read above)



How are business process **interfaces** identified?

- **Business level interfaces** are messages sent between business sub-processes.
- A handy technique:
  Express each message in a **complete sentence.**
- Look for four kinds of message:
  - **Request**: Please process work order **W**.
  - **Response**: Work order **W** was completed at time **T**.
  - **Notice**: Person **P** is no longer certified in skill **S**.
  - **Query**: Status **S** = What is the status of work order **W**?

3/1/10 22

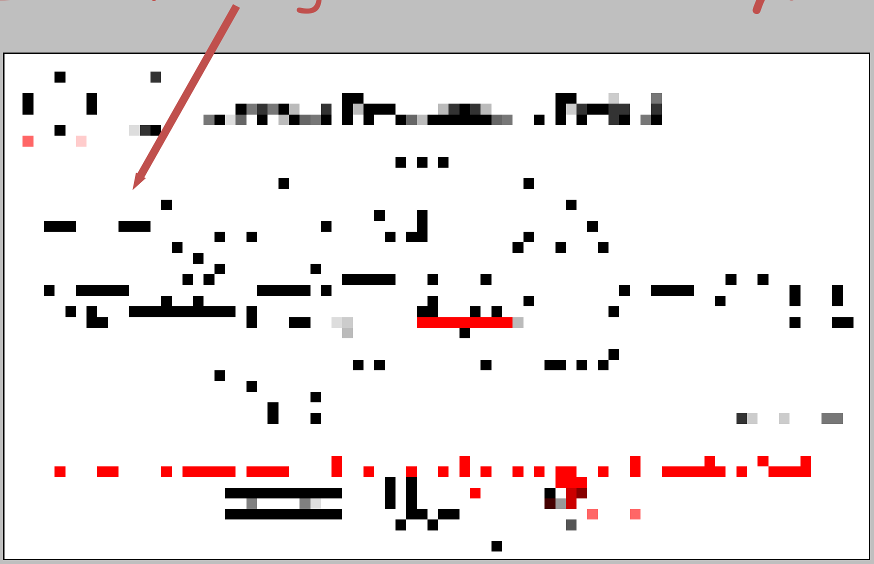
Once again the technique involves writing a complete sentence ... but with some twists.

1. When one business unit is **delegating work** to another, look for a **request** and a **response**
(For planning purposes, just focus on the "happy path" ... you shouldn't need to identify the variants and failure modes.)

2. When a business process is publishing a notice, write the notice in the form of a **present tense assertion**.

3. Write **queries** as an interrogative sentence with the class of the answer made clear.

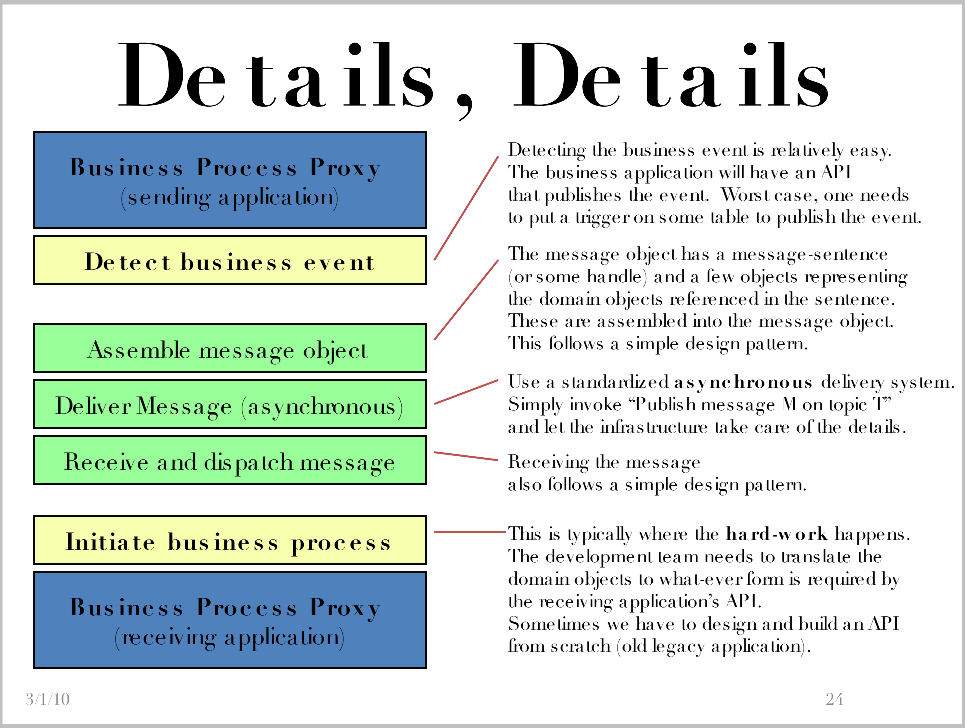
Each message is a **user story.**



This diagram is cut-and-paste from the www.extremeprogramming.org web site.

The diagram in the background comes from: www.extremeprogramming.org.

The point I want to make here is that
Each message can be treated as a “user story”
(use case or feature set)
for release planning.



Details, Details

Business Process Proxy (sending application)
Detect business event
Assemble message object
Deliver Message (asynchronous)
Receive and dispatch message
Initiate business process
Business Process Proxy (receiving application)

Detecting the business event is relatively easy. The business application will have an API that publishes the event. Worst case, one needs to put a trigger on some table to publish the event.

The message object has a message-sentence (or some handle) and a few objects representing the domain objects referenced in the sentence. These are assembled into the message object. This follows a simple design pattern.

Use a standardized **asynchronous** delivery system. Simply invoke "Publish message M on topic T" and let the infrastructure take care of the details.

Receiving the message also follows a simple design pattern.

This is typically where the **hard-work** happens. The development team needs to translate the domain objects to what-ever form is required by the receiving application's API. Sometimes we have to design and build an API from scratch (old legacy application).

3/1/10 24

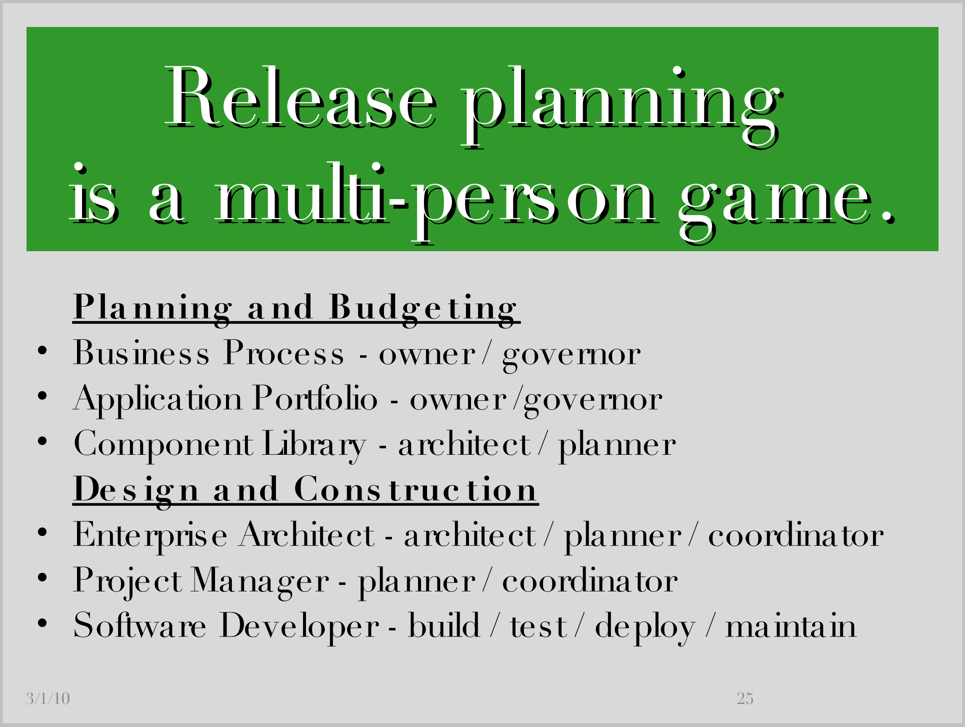
The main point that I want to make here is that ...

You can follow a **standard design pattern**
when you implement **each type of message.**

This will make it easier to **estimate hours**
and **reduce confusion** about
which team should be responsible
for which component.

The design pattern that you select will be partially dependent
on your **standard operating environment.**

One sticky question concerns:
How will you represent business objects in message payloads?



Release planning is a multi-person game.

Planning and Budgeting

- Business Process - owner / governor
- Application Portfolio - owner /governor
- Component Library - architect / planner

Design and Construction

- Enterprise Architect - architect / planner / coordinator
- Project Manager - planner / coordinator
- Software Developer - build / test / deploy / maintain

3/1/10 25

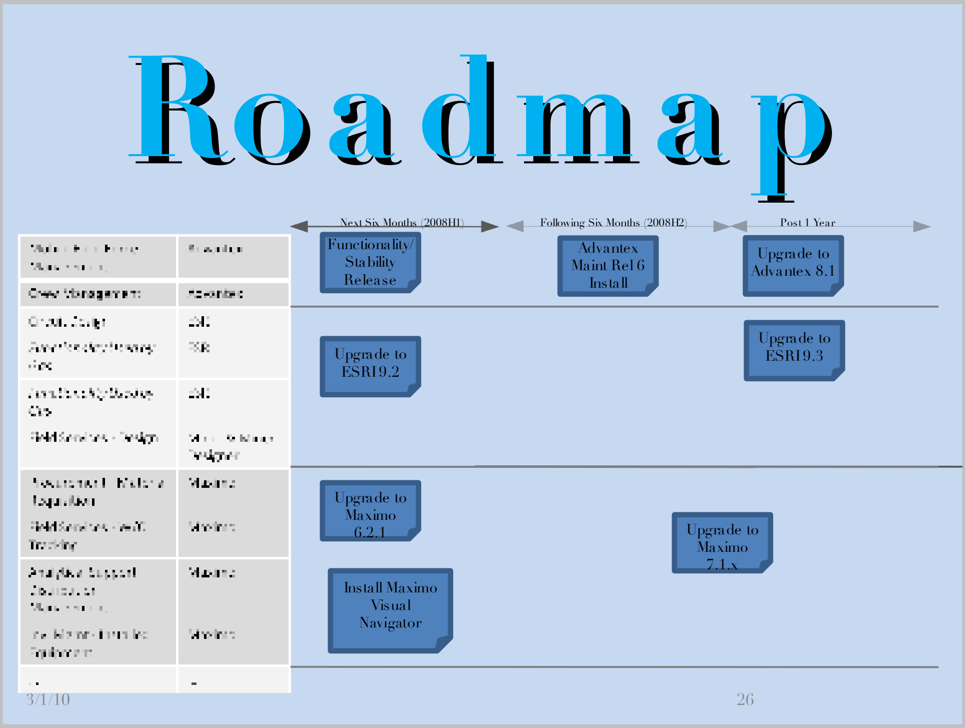
The "Planning Game",
as it is called in Extreme Programming practice,
can be two or three times as complicated
when you are planning enterprise integration.

The reason is that there is a sender and a receiver for every message.
The sender process and the receiver process
will each have a full complement of project roles.
In addition, you may need to involve data service providers.

Thus, you will have at least twice the number of players at the planning table.

I strongly recommend that start thinking
about how you will do this if you have not already done so.
The biggest hazard to becoming agile in an enterprise integration context
revolves around how the planning process works.

*I have seen cases where the effort required for planning
greatly exceeded the effort required for implementation.*



Roadmap

[illegible]	[illegible]
Crew Management	Advantex
Circuit Design [illegible]	ESRI ESRI
[illegible] Field Services - Design	ESRI [illegible] Designer
Procurement / Materials Acquisition Field Services - Asset Tracking	Maximo Maximo
Analytics Support [illegible] [illegible] Equipment	Maximo Maximo
[illegible]	[illegible]

3/1/10 26

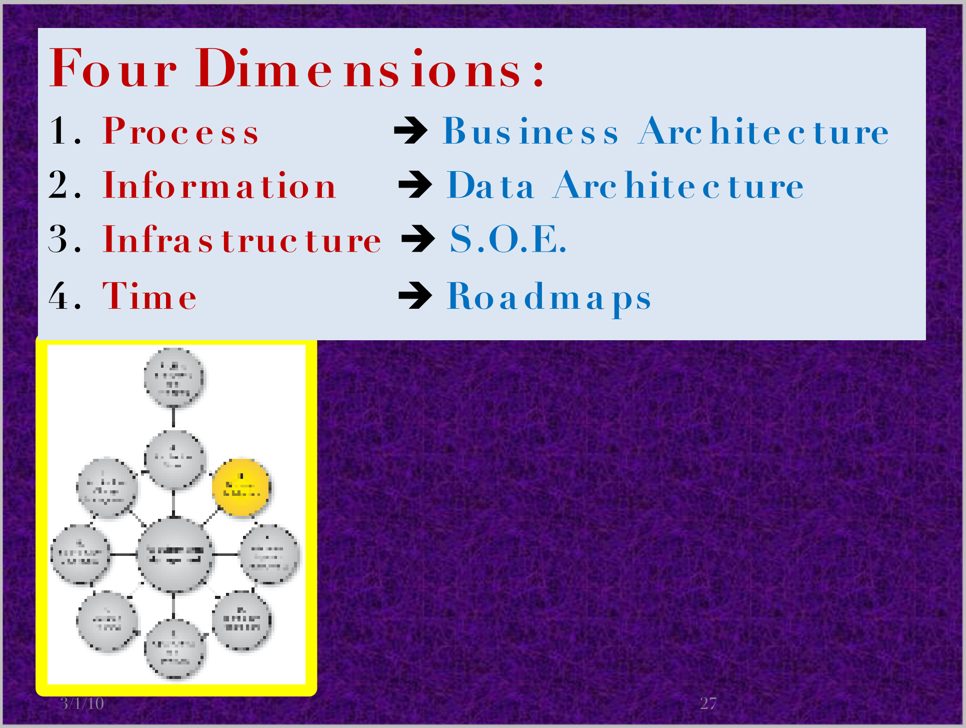
For longer-term planning,
we use a document we call a **roadmap**.

We are primarily a regulated energy utility.
Software is just something our ITS group does to support the main business.
Nevertheless,
we encourage our Information Officers and Portfolio Leads
to think like **product managers**.

Of course, we own a lot of **purchased software**
in addition to our home-built software.
In the case of the purchased software,
the roadmap tends to be driven by the **vendor's updates**.

The road map has time on the horizontal axis.
The vertical axis has a breakdown of the portfolio.

Again, the **business architecture**
helps to **structure** the **portfolio**.



Four Dimensions:

1. Process → Business Architecture
2. Information → Data Architecture
3. Infrastructure → S.O.E.
4. Time → Roadmaps

3/1/10 27

Here is where we've been:
We've seen four fundamental dimensions of Enterprise Architecture.
I've provided a brief overview of some key concepts and techniques in each of those four dimensions.

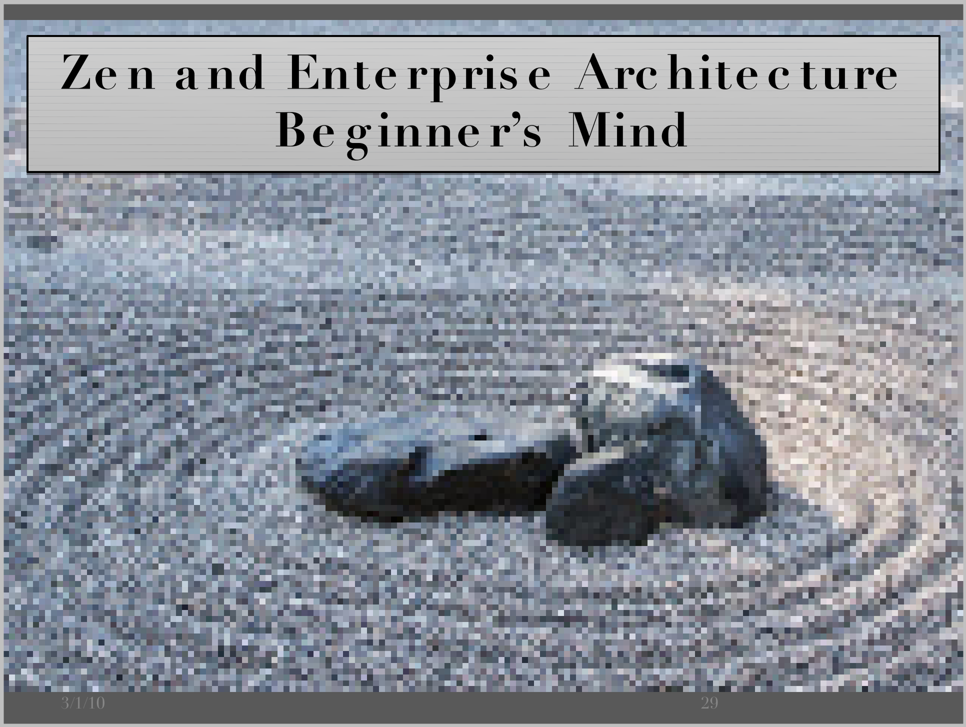
As you can see, I have only touched on **four** of the **ten** activities in the TOGAF.
The other six activities are **organizational**, and not to be under-estimated.
However, if you understand those four, you can get started and the rest will come.

Links

- **TOGAF:**
  - http://www.togaf.info/
  - http://www.opengroup.org/
- **Zackman Framework:**
  - http://www.zachmaninternational.com/
  - http://eacoe.org/
- **Value Chain:**
  - James Martin: ISBN: 9780814403150
    Martin, James (1995). *The Great Transition: Using the Seven Disciplines of Enterprise Engineering*
  - Ken Orr: Business Enterprise Architecture Modeling http://www.cutter.com/workshops/04.html
  - Michael Porter: http://www.isc.hbs.edu/

If you want to learn more about this subject, these **links** are some good places to start.

If you give me your business card, or anything with your e-mail address, I will send you a link to my presentation notes.



Zen and Enterprise Architecture Beginner's Mind

I hope that this talk has made
Enterprise Architecture look easy.
The concepts **are** relatively easy.

But it is kind of like learning object-oriented programming or test-driven development.
The **concepts** are relatively easy to acquire, but acquiring the **skill** requires
weeks of **individual practice**, and
months of **team practice**.

The really really really hard part involves getting all of your teams trained and practicing.