Enterprise Architecture Frameworks

Richard Martin
Tinwisle Corporation
Bloomington IN

Edward Robertson
Computer Science Department
Indiana University

I keep six honest serving men
    (They taught me all I knew);
Their names are What and Why and When
    And How and Where and Who.
I send them over land and sea.
    I send them east and west;
But after they have worked for me,
    I give them all a rest.
I let them rest from nine till five,
    For I am busy then,
As well as breakfast, lunch, and tea,
    For they are hungry men.
But different folk have different views,
    I know a person small–
She keeps ten million serving-men,
    Who get no rest at all!
She sends 'em abroad on her own affairs,
    From the second she opens her eyes–
One million Hows, two million Wheres,
    And seven million Whys!

Rudyard Kipling
"The Elephant's Child"
Just So Verses, 1902

Outline – Enterprise Architecture Frameworks

- Background
- Concepts
- Formalism
- Support for Methodologies
- Example: IBM's ESS
- Conclusions

Background

Zachman's Generalizations

In the 1980's, John Zachman observed that

1. construction, aircraft manufacture, information systems, $\cdots$ asked the same questions
2. construction, aircraft manufacture, $\cdots$ clearly classified answers by ordered roles, while information systems did not

He postulated a framework grid organized by

- questions
- roles

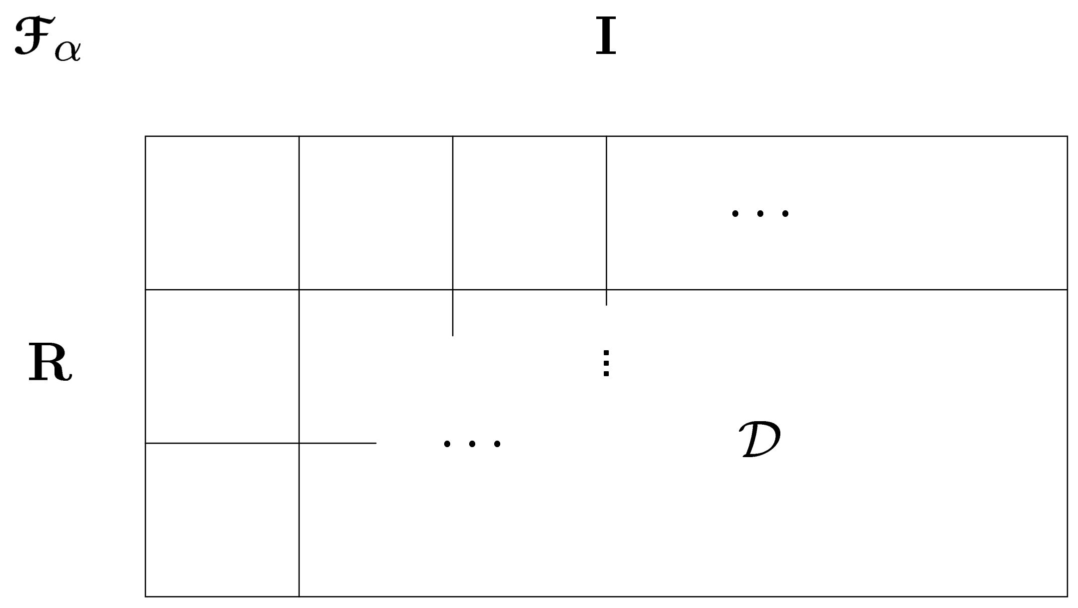
see `www.zifa.com`

ENTERPRISE ARCHITECTURE - A FRAMEWORK

— from Zachman International, `www.zifa.org`

Framework Structure

$\mathbf{R}$	$\{\mathsf{r}_1, \cdots, \mathsf{r}_n\}$, a set of *roles*
$\mathbf{I}$	a set of *interrogatives*
$\mathcal{D}$	a finite set, the *descriptors*
$\mathfrak{F}_\alpha$	a *frame*



Roles

$\mathbf{R}$ is ordered $\{\mathsf{r}_1, \cdots, \mathsf{r}_n\}$

$\mathbf{R}$ typically fixed by methodology

$\mathbf{R}$ specifies rows within a frame

examples of $\mathbf{R}$:

{ "specification", "design", "implementation" }

{ "owner", "designer", "builder" }

{ "enterprise", "system", "technology" }

$\mathbf{R}$ is extended with two "contexts":

r_0 before (typically written "context")

r_{n+1} after (typically "out of context")

extended $\mathbf{R}$ is therefore:

$\{\mathsf{r}_0, \mathsf{r}_1, \cdots, \mathsf{r}_n, \mathsf{r}_{n+1}\}$

Interrogatives

I is not ordered

I typically fixed by methodology

I specifies columns within a frame

examples of **I**:

{ "what", "how", "where", "who", "when", "why" }

{ "data", "function", "network", "people", "time", "motivation" }

{ "things", "activities", "places", "roles", "events", "rules" }

Descriptors

$\mathcal{D}$ is unordered

$\mathcal{D}$ localized to application

$\mathcal{D}$ captures "domain knowledge"

examples of $\mathcal{D}$:

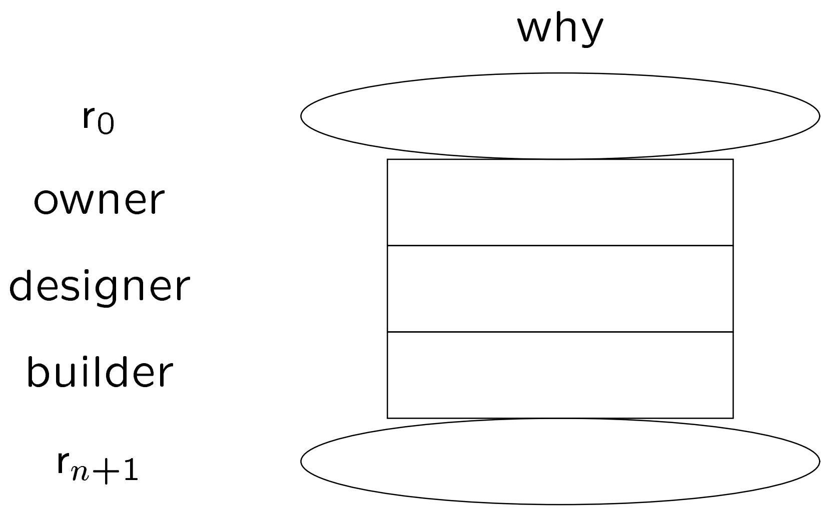
{ "Department", "Employee", "Hire", ⋯}

{ "Accommodate Legacy", "Account", "Actor", "Actual", "Address", "Analyze", "Arbiter", "Architecture", "Architecture Aspect", "Aspect Dependent", "Aspect Difference", "Aspect Integration", "Asset", "Association Notation", "Attitude", "Availability", "Behavior", ⋯}

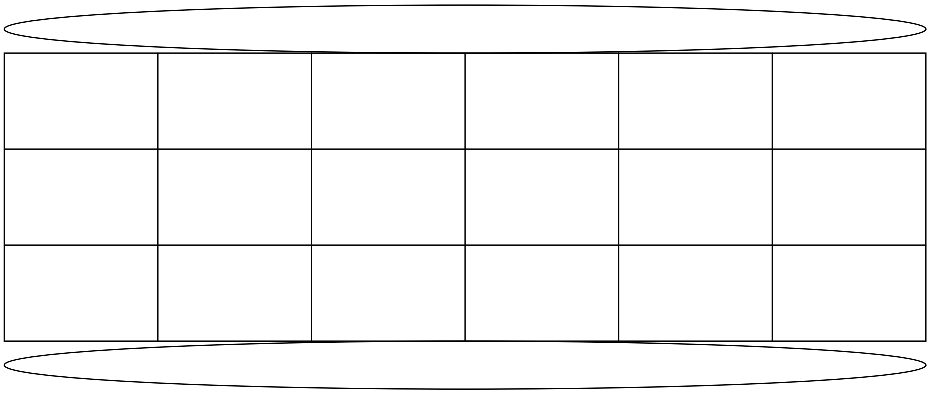
– IBM ESS terminology, 200+ terms

subsets of $\mathcal{D}$ specify the "visible interface" of a frame $\mathfrak{F}_\alpha$

A Single Column



A Single Frame



Concepts

Principles

The significant principles that govern our formalization are:

1. formalize the framework approach, not one particular framework
2. naming as important as structure
3. abstraction↔concreteness and generality↔detail are distinct dimensions
4. the framework should provide mechanism, not policy
5. recursion is decomposition and successive refinement, not reoccurrence
6. multiple views are inherent and essential in system analysis and design
7. roles are ordered, interrogatives are not
8. entire preceeding role is relevant for each cell of a frame

Why Formalize:

Compelling advantages of formalization:

1. it exposes commonalities in the variety of modeling methodologies
2. it challenges us to be precise in our concepts and the use of these concepts
3. it enhances our understanding of the issues
4. it promotes coherence among models along the spectrum from enterprise-wide to full detail
5. it allows communication across various methodologies and approaches
6. it provides conceptual metrics for evaluating particular methodologies
7. it enables precise specification of constraints and views
8. it scales
9. it facilitates toolkit development

The Heart of the Meta

- real data is from the real world
- the structure we impose on the real world is the model
- the methodology reflects the structure we impose on models, hence it is a meta-model
- the framework reflects the structure we impose on methodologies, hence it is a meta-meta-model

- meta-data represents the model, hence meta-meta-meta-data represents a framework

Enterprise/Information Systems Perspective

Lifecycle

- more iterative
- less predictable

Maintenance

- different modes
  - operational (ϵ)
  - corrective ($< 25\%$)
  - adaptive ($< 25\%$)
  - supplementative ($> 50\%$)
- dominates costs

lifecycle

linear time

atemporal

Architecture in the New e-Conomy

- technology is the strategy *driver*
- risk is a requirement

  ⇒ design to manage risk
- old optimizations are insufficient
  - cost, performance, reliability

  ! must itemize for adaptability
- change was an event, now it's a process
- adaptable architecture determines business success

- after Vinod Khosla, 2000

Formalism

Overview

Base:

- frames

Structure:

- recursively constructed from frames
- accessed by $\mathcal{PATHS}$

Semantics:

- row to row linking *via* Φ relationship

Support for methodologies:

- special naming and structuring considerations
- structural and semantic constraints
- views
- implementation depends upon path abbreviations

Frames and Frameworks

Frames:

- indexed by $\mathbf{I} \times \mathbf{R} \times \mathcal{D}$
- two internal structures, either:
  - *branch* frame
  - *leaf* frame

Frameworks:

- constructed recursively from frames

$\mathfrak{F}$ — a *framework* is a finite set of *frames* $\{\mathfrak{F}_\alpha : \alpha \in \mathcal{L}\}$

$\mathfrak{F}_\epsilon$ — the "root" frame

Do not require $\alpha \neq \beta$ implies $\mathfrak{F}_\alpha \neq \mathfrak{F}_\beta$

Hence, $\mathfrak{F}$ comprises a rooted directed acyclic graph

Labels and Paths

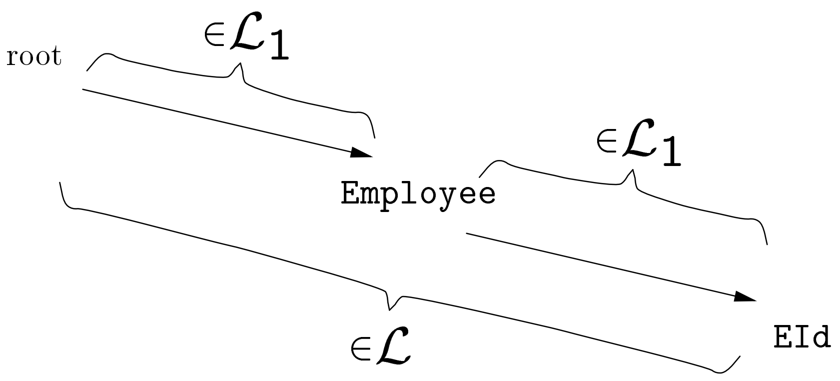
edge labels:

$\mathcal{L}_1 \quad = \{\langle r, i, d\rangle : r \in \mathbf{R},\ i \in \mathbf{I},\ \&\ d \in \mathcal{D}\}$

path labels:

$\mathcal{L}$ defined by BNF $\mathcal{L} := \epsilon | \mathcal{L}\,\mathcal{L}_1$

(ϵ denotes the empty path)



$\mathcal{PATHS}_{\mathfrak{F}} = \{\alpha : \mathfrak{F}_\alpha$ is defined$\}$

- no gaps
- acyclic

Tree *versus* DAG

$$\alpha\beta\delta \neq \alpha\gamma\zeta$$

but

$$\mathfrak{F}_{\alpha\beta\delta} = \mathfrak{F}_{\alpha\gamma\zeta}$$

$$\begin{tikzcd}
\mathfrak{F}_{\epsilon} \arrow[dr, dash, "\alpha"] & & \\
& \mathfrak{F}_{\alpha} \arrow[dl, dash, "\beta"'] \arrow[dr, dash, "\gamma"] & \\
\mathfrak{F}_{\alpha\beta} \arrow[dr, dash, "\delta"'] & & \mathfrak{F}_{\alpha\gamma} \arrow[dl, dash, "\zeta"] \\
& \mathfrak{F}_{\alpha\beta\delta} = \mathfrak{F}_{\alpha\gamma\zeta} &
\end{tikzcd}$$

Branch Frame Structure

A *branch frame* $\mathfrak{F}_\alpha$ is characterized by the 4-tuple:

$$\mathcal{IC}_\alpha \subseteq \mathcal{D}$$
$$\mathcal{OC}_\alpha \subseteq \mathcal{D}$$
$$\mathcal{SF}_\alpha : \mathbf{R} \times \mathbf{I} \times \mathcal{D} \rightarrow (\mathfrak{F} \cup \mathcal{L})$$
$$\Phi_\alpha \subseteq outputs \times inputs \text{ (details below)}$$

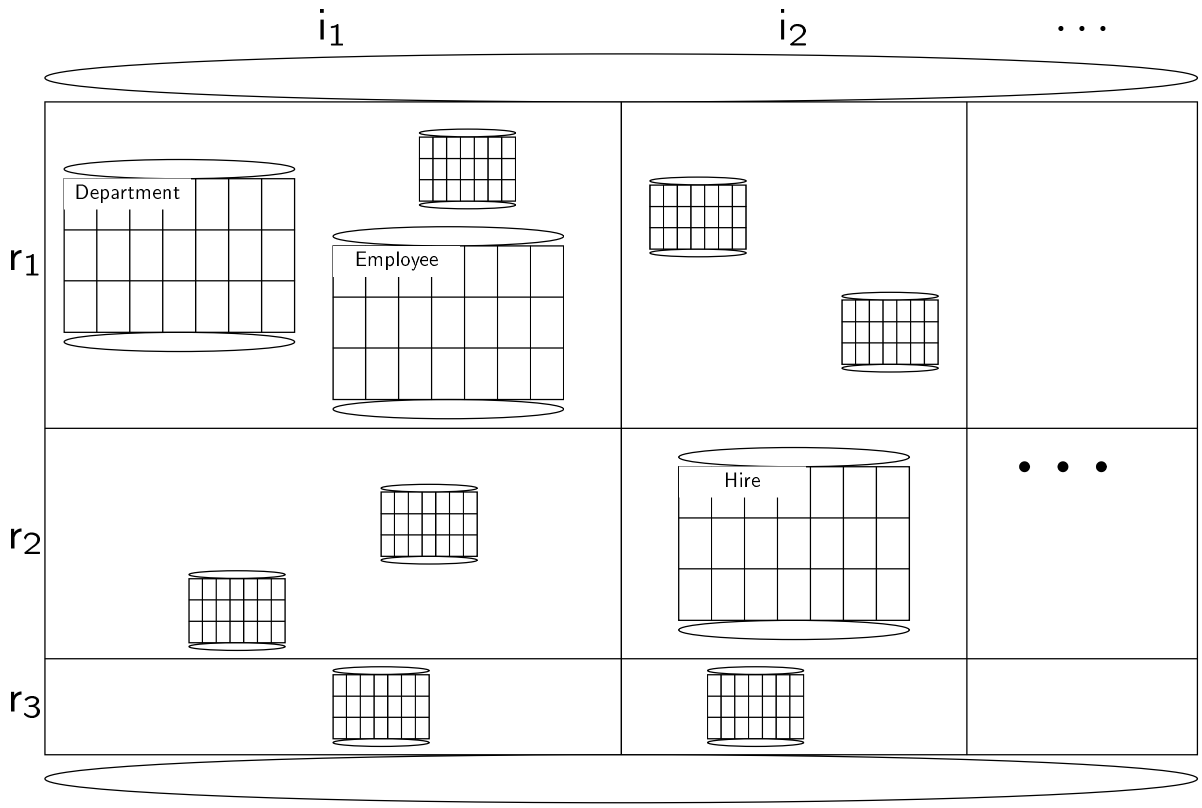
$\mathcal{IC}$, $\mathcal{OC}$ provide scoping

$\mathcal{SF}$ provides framework structure

Φ provides semantic connections

Branch Frame Structure: Example

$\mathbf{R} \times \mathbf{I} \times \mathcal{D}$ indexes subframes



- typically $\mathbf{R}$ and $\mathbf{I}$ are implicit

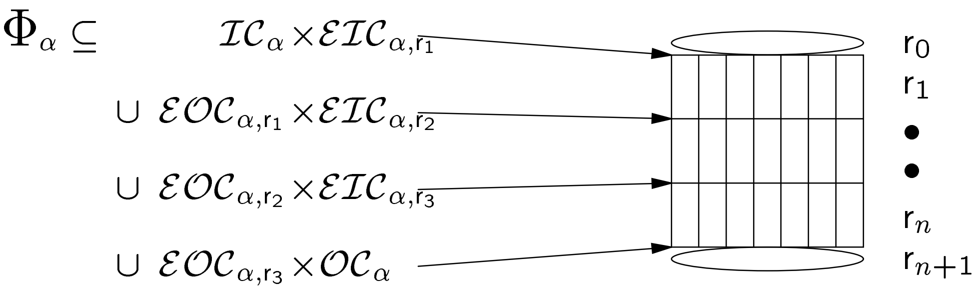
Φ Details

for $r \in \mathbf{R}$, $\alpha \in \mathcal{PATHS}$:

$$\begin{aligned}
\mathcal{EIC}_{\alpha,r} &= \{\alpha\langle r,i,d\rangle\, d' : i \in \mathbf{I} \ \&\ d \in \mathcal{D} \\
&\qquad \&\ d' \in \mathcal{IC}_{\alpha\,\langle r,i,d\rangle}\} \\
\mathcal{EOC}_{\alpha,r} &= \{\alpha\langle r,i,d\rangle\, d' : i \in \mathbf{I} \ \&\ d \in \mathcal{D} \\
&\qquad \&\ d' \in \mathcal{OC}_{\alpha\,\langle r,i,d\rangle}\}
\end{aligned}$$

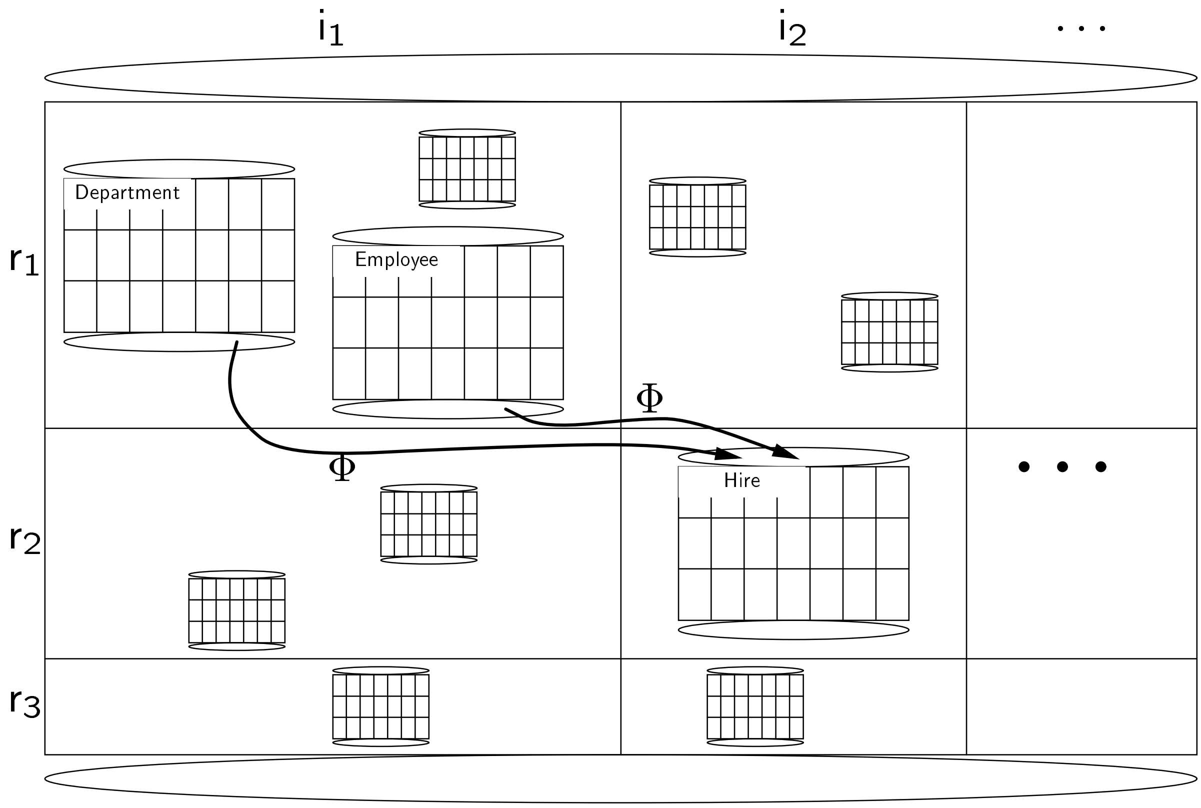
$$\begin{aligned}
\mathcal{EOC}_{\alpha,\mathrm{r}_0} &= \{\alpha\, d : d \in \mathcal{IC}_\alpha\} \\
\mathcal{EIC}_{\alpha,\mathrm{r}_{n+1}} &= \{\alpha\, d : d \in \mathcal{OC}_\alpha\}
\end{aligned}$$

$$\Phi_\alpha \subseteq \bigcup_{0\le i\le n}(\mathcal{EOC}_{\alpha,r_i} \times \mathcal{EIC}_{\alpha,r_{i+1}})$$



Φ Details: Example

Φ_α connects subframes



- Φ connects
  - to next role
  - from output to input

Leaf Frame Structure

A *leaf frame* $\mathfrak{F}_\alpha$ is characterized by the 3-tuple:

$\mathcal{IC}_\alpha \subseteq \mathcal{D}$

$\mathcal{OC}_\alpha \subseteq \mathcal{D}$

$\mathcal{S}_\alpha$ (details below)

$\mathcal{S}_\alpha$ provides the real world connections which express the meaning of the instantiated model

$\mathcal{E}$ the completely empty leaf $\langle \emptyset, \emptyset, \emptyset \rangle$

Leaf Frame Details

$$Types_\alpha = \begin{array}{l} \mathcal{IC}_\alpha \cup \{\text{“SET OF } d\text{”} : d \in \mathcal{IC}_\alpha\} \cup \\ \mathcal{OC}_\alpha \cup \{\text{“SET OF } d\text{”} : d \in \mathcal{OC}_\alpha\} \end{array}$$

$Types$ allows actual framework instance to deal with individuals or sets

$$\mathcal{S}_\alpha \quad : \mathcal{D} \rightarrow \bigcup_{n \in \mathbb{N}} {Types_\alpha}^n$$

$\mathcal{S}_\alpha$ gives only syntax!

cannot prescribe or proscribe connections to real world

- “interflow” of ENV 12204

Abbreviations

Goal:

use abbreviations to facilitate formalism, providing

- familiarity
- uniformity
- structure

- a single name $\mathcal{N}$ can abbreviate
  - a path: $\mathcal{N} \rightarrow \langle r, i, d \rangle \langle r', i', d' \rangle$
  - $\mathbf{R} \times \mathbf{I}$ cell coordinates: $\mathcal{N} \rightarrow \langle r, i, \ \cdot \ \rangle$
  - a path template with multiple substitutions: $\mathcal{N} \rightarrow \langle r, i, \ \cdot \ \rangle \langle r', i', \ \cdot \ \rangle$
- use to shorten complicated paths

Abbreviations and Methodologies

Abbreviations and related constraints specify

- methodologies
- generic ↔ partial ↔ particular

$$\begin{array}{c} \text{set of} \\ \text{abbreviations} \end{array} \simeq \begin{array}{c} \text{meta-model of} \\ \text{framework} \end{array}$$

Abbreviation Examples

1. unique and meaningful names given to each cell of the $\mathbf{R} \times \mathbf{I}$ grid

conceptual,data cell called the "Semantic Model"

2. names associated with the components of a cell labeled by an $\mathbf{R}$, $\mathbf{I}$ pair.

object within conceptual,data cell called "Entity" or "Relationship"

3. $\mathbf{R}$ and $\mathbf{I}$ components of a collection of paths separated from the $\mathcal{D}$ components.

conceptual,data cell called "Entity" at the first level and "attribute" at the second.

"Entity Employee has attributes EId and name" means that there are frames with labels
"⟨conceptual,data,Employee⟩⟨conceptual,data,EId⟩"
"⟨conceptual,data,Employee⟩⟨conceptual,data,name⟩"

Abbreviation Examples (cont)

4. if a name in $\mathcal{D}$ appears in only one frame, then entire paths are abbreviated.

if Employee appears only at the top level, then Entity Employee indicates a unique subframe

5. a succession of Φ connections is often abbreviated by a single connection

Constraints

Goal:

use restrictions on formal relations ($\mathfrak{F}$, $\mathcal{SF}$, Φ, $\cdots$) to:

- insure consistency (interior)
- enforce business rules (leaves)

- analogous to foreign key constraints in RDB implementation
- abbreviations constrain legal path labels to provide a methodology's structure

Constraints Defining a Model

- abbreviations define the meta-model:

  "Entity" and "Relationship" designate subframes in conceptual$\times$data cell

- vocabulary in $\mathcal{D}$ corresponds to the ER model itself

  $\mathcal{D}$ for HR ER includes `Employee`, `EId`, $\cdots$

- Thus the constraint

  $$\mathfrak{F}_{\langle \text{conceptual}, \text{data}, \texttt{Employee} \rangle}.\mathcal{IC} \supseteq \{\texttt{EId}, \texttt{name}, \texttt{hiredate}, \texttt{skill}, \cdots\}$$

  corresponds to

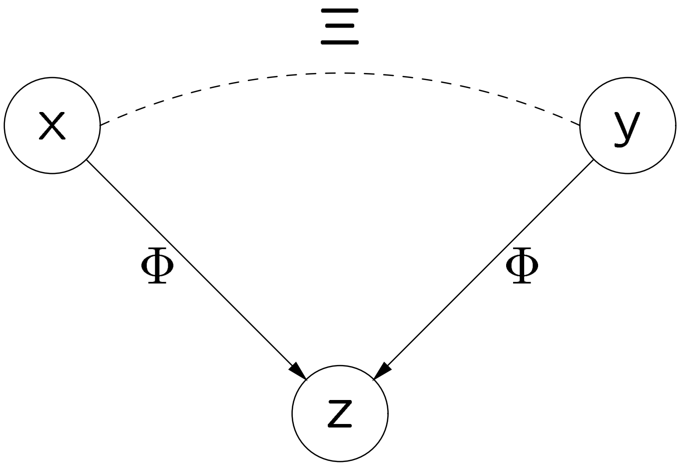
  Entity `Employee` includes `EId`, `name`, `hiredate`, `skill`, $\cdots$

Constraint Example

Carry-forward consistency:

$$\Xi_\alpha \qquad \subseteq \bigcup_{r \in \mathbf{R}} \mathcal{OC}_{\alpha,r} \times \mathcal{OC}_{\alpha,r}$$

$x \, \Phi_\alpha \, z$ and $y \, \Phi_\alpha \, z$ imply $x \, \Xi_\alpha \, y$.



In order to fully capture the world being modeled, it is expected that there will be additional *ad hoc* constraints.

Views

Goal:

rearrange framework components in order to

- focus on certain elements
- match different perspectives
- validate structure

- achieve this by rewriting path labels
  - omit segments
  - rearrange segments

- full tree rewriting too powerful

- more structural than semantic
  - difficult to carry along Φ

Views

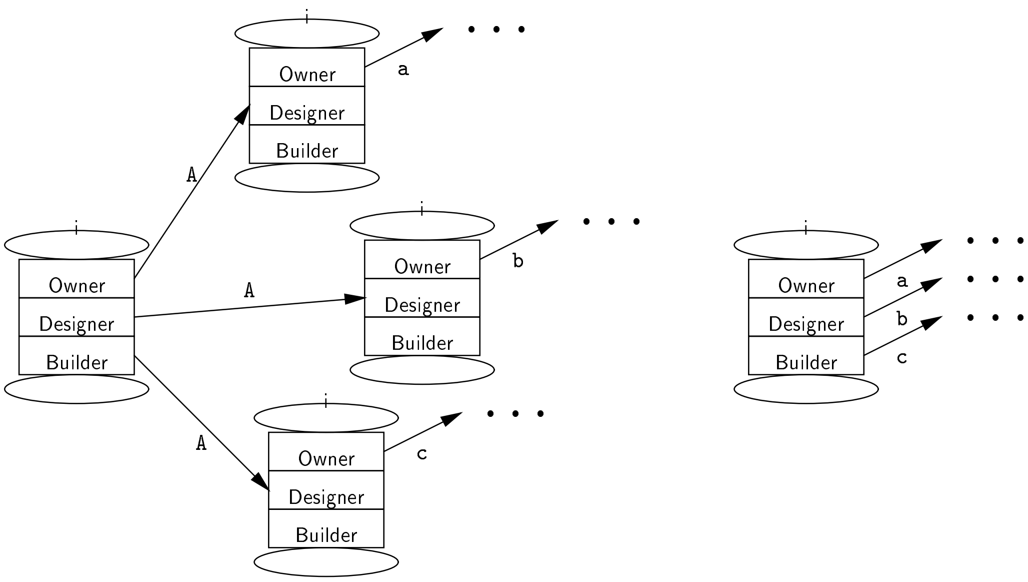
A view of $\mathfrak{F}_\alpha$ is merely a subgraph of $\mathfrak{F}_\alpha$

As with relational views, complexly defined views may have update anomalies.

collapse Φ along paths ... problems with indirections

A view is *coherent* if any collapsing happens consistently across all paths.

A view may be used to conceptually rearrange a framework:

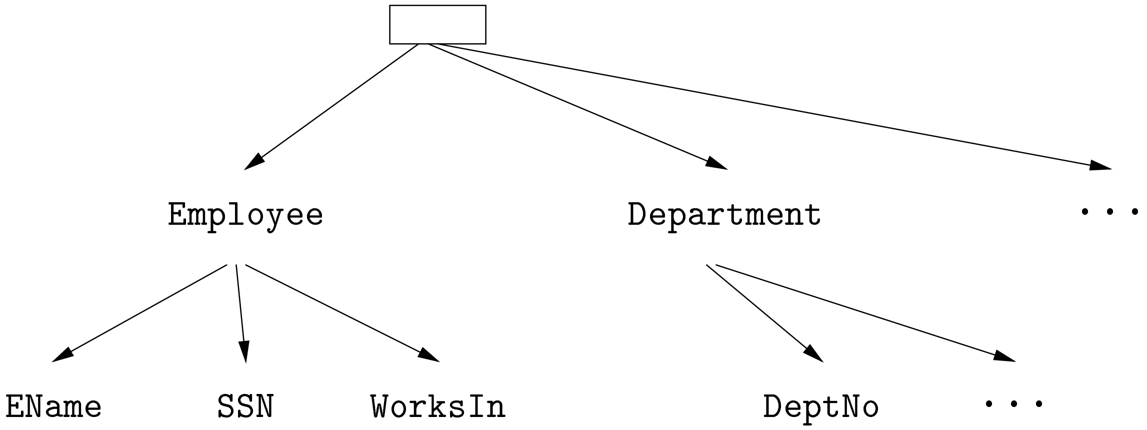


Views in ER Example

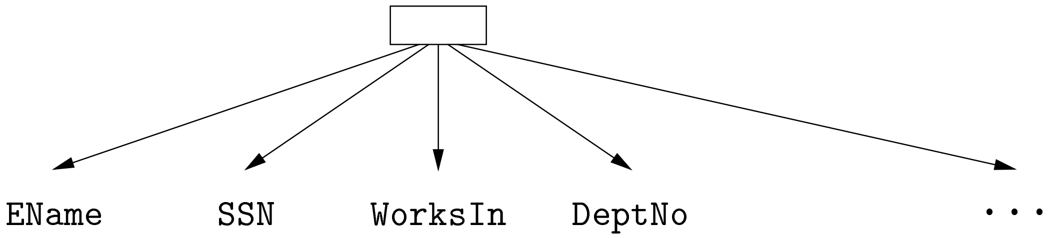
Form data dictionary using rule

$$\langle r_1, i_1, X\rangle.\langle r_1, i_1, Y\rangle \Rightarrow \langle r_1, i_1, Y\rangle$$

so



becomes



VIEW EXAMPLE: BY LIFECYCLE

Define $Stages$ =
{"Identification", "Concept", "Requirement", "Preliminary Design", "Detailed Design", "Implementation", "Operation", "Decommission"}

Rewrite using

$$\alpha\langle r, i, \ell\rangle \Rightarrow \langle \ell, i, r\rangle, \text{where } \ell \;\in Stages$$

This collects together components from respective lifecycle stages:

Identification			
Concept		$\cdots$	
$\vdots$			
Decommission			

SUPPORT FOR METHODOLOGIES

"IDENTITY" LEAF

$\mathfrak{I}$ any leaf frame which merely connects its input to its output

conditions on $\mathfrak{I}$:

- $\mathcal{IC} = \mathcal{OC}$

$\mathfrak{S}$

- is $\mathcal{IC} \times \mathcal{OC}$ with a fixed interpretation as the identity relation

Formally, such usage is ambiguous because there is an identity frame for each distinct $\mathcal{IC}$, $\mathcal{OC}$ pair but practically it is easy to handle.

Operations of Frameworks

Example: Merging

$\mathfrak{H}$ is the merger of frameworks $\mathfrak{F}$ and $\mathfrak{G}$, written $\mathfrak{F} \cup \mathfrak{G}$, if,

for all $\alpha \in \mathcal{PATHS}_{\mathfrak{F}} \cup \mathcal{PATHS}_{\mathfrak{G}}$,

$$\mathfrak{H}_{\alpha}.\Diamond \;=\; \mathfrak{F}_{\alpha}.\Diamond \cup \mathfrak{G}_{\alpha}.\Diamond$$

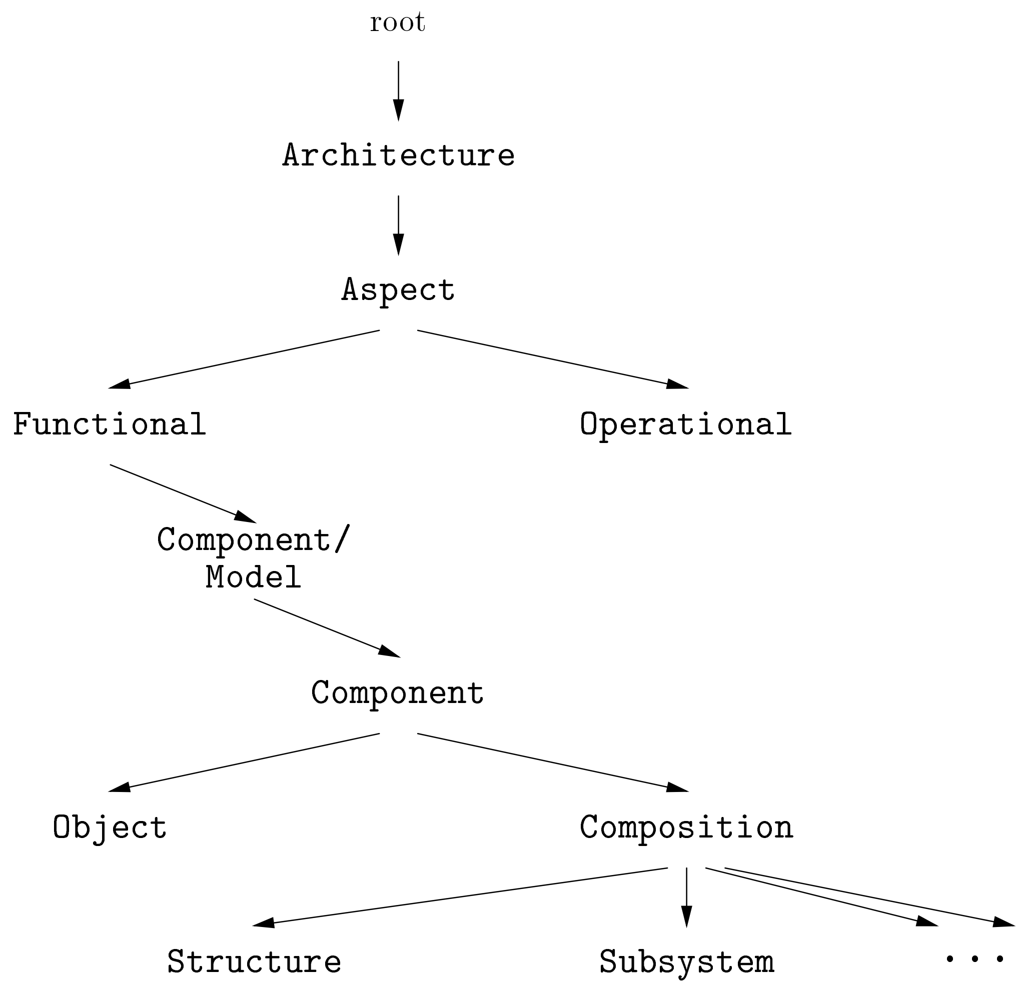
Example: IBM's ESS

ESS Background

Goal: validate formalism with application to a real model

- begin with IBM's Enterprise Solutions Structure (*IBM Systems Journal*, 1999)
- applied Doug McDavid's semantic analysis technique to obtain vocabulary
- mapped into our formalism

Fragment of ESS Meta-hierarchy



Note: **R** and **I** indices omitted

ESS Equivalences for I

what	≈	resource
how	≈	behavior
where	≈	location
who	≈	delivery
when	≈	______
why	≈	drivers

Vocabulary for One Frame

Composition – $\mathfrak{F}_\alpha$

what	how	where
Structure	Dynamic_Behavior Enablement	Subsystem
Object_State Partitions Allocations	Object_Interface Optimizing Templates	Grouping Overlap Platform_ Span
UML_Class_ Diagram	Component_ Interaction_ Diagram	UML_Deployment_ Diagram

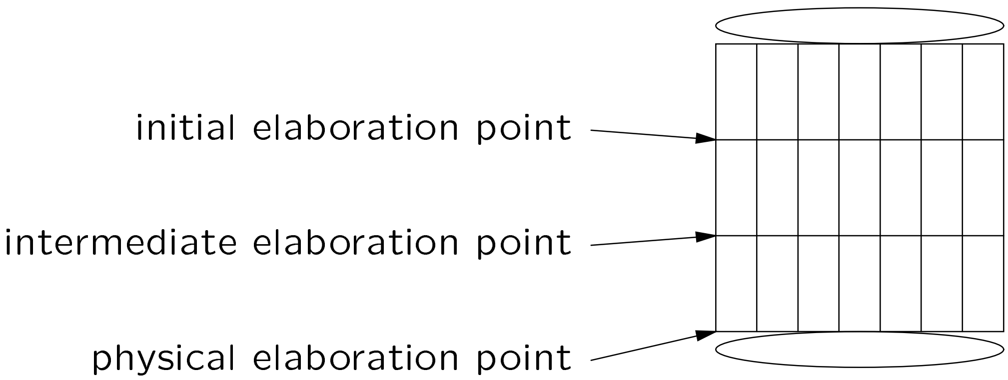
path α = ⟨conceptual, what, **Architecture**⟩.
⟨conceptual, what, **Aspect**⟩.
⟨logical, how, **Functional**⟩.
⟨logical, what, **Component_Model**⟩.
⟨physical, what, **Component**⟩.
⟨logical, how, **Composition**⟩

Vocabulary for One Frame – continued

who	when	why
Collaborations Role_Player	Sequence	Use_Case Situation Purpose
Performing_ Component Requesting_ Component Workflow	Interactions Operation_ Sequence Task_Sequence	Interface_Use Service_Level Commitment
UML_Collaboration_ Diagram	UML_ Sequence_ Diagram	UML_ Scenario

Elaboration Points

ESS makes heavy use of $\mathcal{EOC}$ during modeling process



Future Work

- gain experience
- semantics
  - make use of Φ less awkward
- constraints
  - extend concepts to capture different flavors of frameworks
  - more rigorous formal characterization
- views
  - formalize and investigate formal properties
  - better ways for handling ambiguities
- methodologies & toolkits

Technical report available at

`http://www.cs.indiana.edu/database/Publications/TR522.html`

Cross Fertilization

- information systems architectures
  - chaotic growth
  - desperate needs
  - driven by toolkits
- IFIP-IFAC
  - has gone further in promoting standardization
  - we need your experience in applying enterprise architecture standardization efforts
- enhance interaction between these two tracks
  - formalisms can facilitate