Science of Enterprise Modeling: An Informatic Perspective

Richard Martin
Tinwisle Corporation
Bloomington, Indiana

Edward Robertson
Computer Science Program
Indiana University and
Persistent Systems

General Principles

1. Models are formal artifacts developed and used by people.
2. A complexity tradeoff exists between modeling medium and model instances in that medium.
3. Naming serves as the bridge between the formal and the human.
4. Do not confuse meta-levels - separate model and instance decompositions
5. Dependency is not chronology
6. Don't hide architecture in methodology.

Framework Principles

7. Frameworks organize artifacts to facilitate understanding.
8. To improve quality, distinguish structure from connectivity.
9. Separate policy from mechanism.
10. Both grid (ordinant) and tree (decomposition) structures appear in models.
11. Scale dimensions include:
    abstractness (abstract to concrete),
    scope (general to special) and
    refinement (coarse to fine).

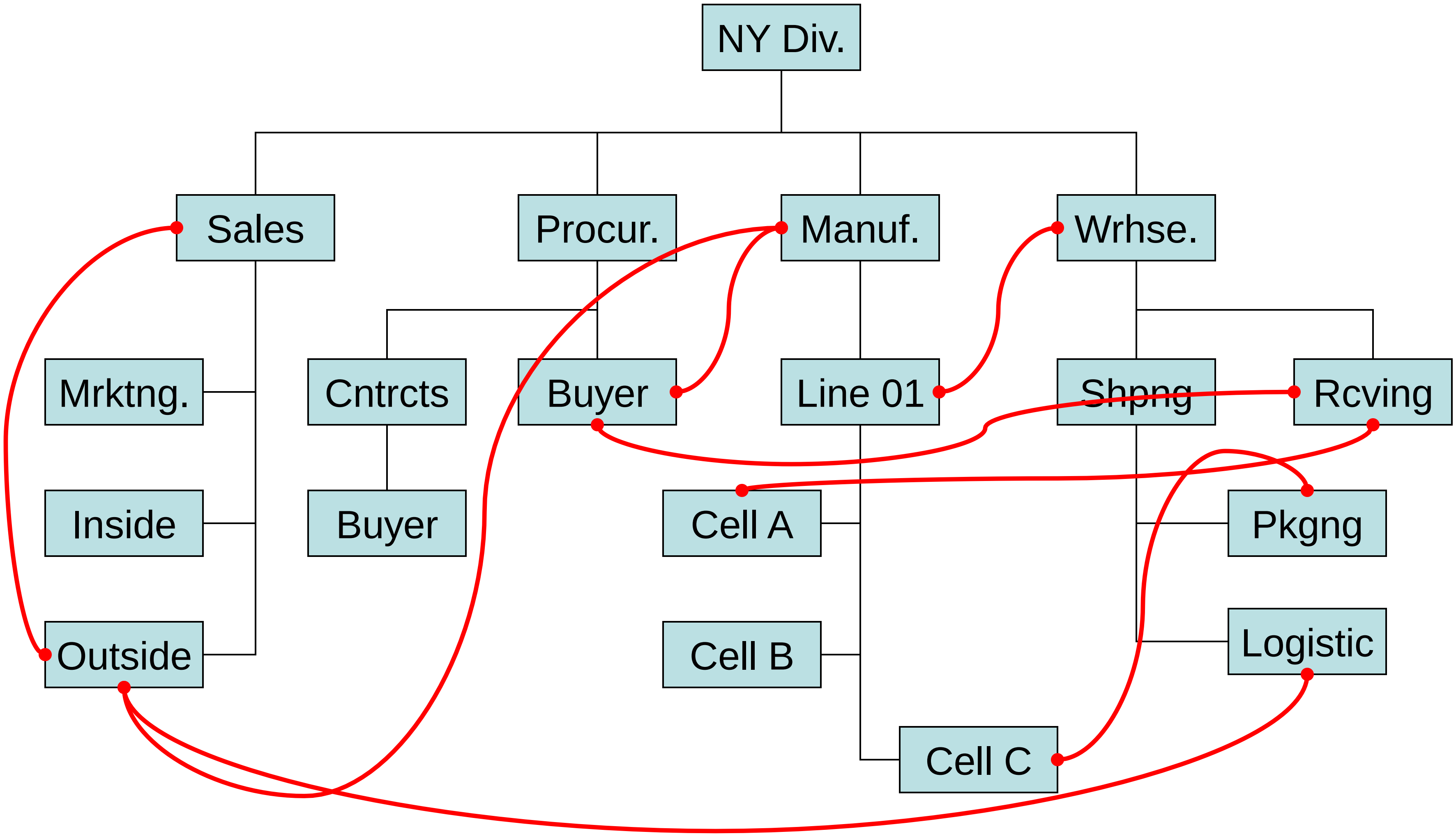
Framework Principles

12. Within a framework, use of components are driven along one ordered dimension.
13. Along this ordered dimension, all prior context is relevant.
14. Refinement is recursive.
15. Connections can be of arbitrary arity.
16. Views are important in standards and methodologies.
17. Views are used both to "see" contents and to "create" contents.
18. Separate model and instance constraints.

Meta Process Principles

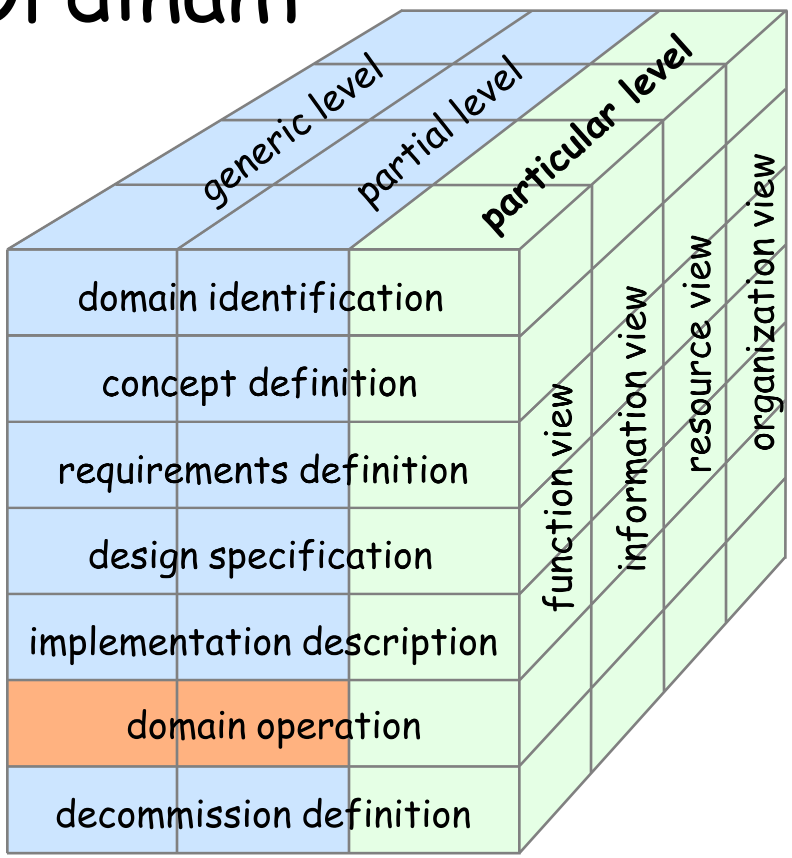
19. Framework are developed by transformations.
20. Transformations include:
    - Projection
    - Instantiation
    - Refinement
    - Specialization
    - Derivation
    - Linking
21. Transformation use depends upon stakeholders, meta-pahse, context,…

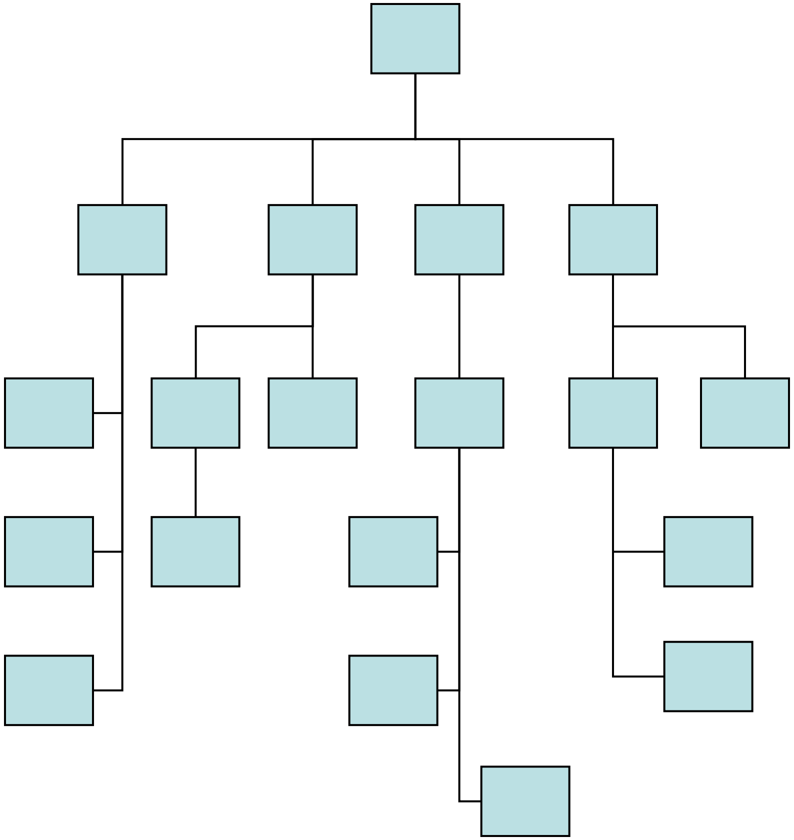
Distinguish structure from connectivity



Two structural aspects

Ordinant



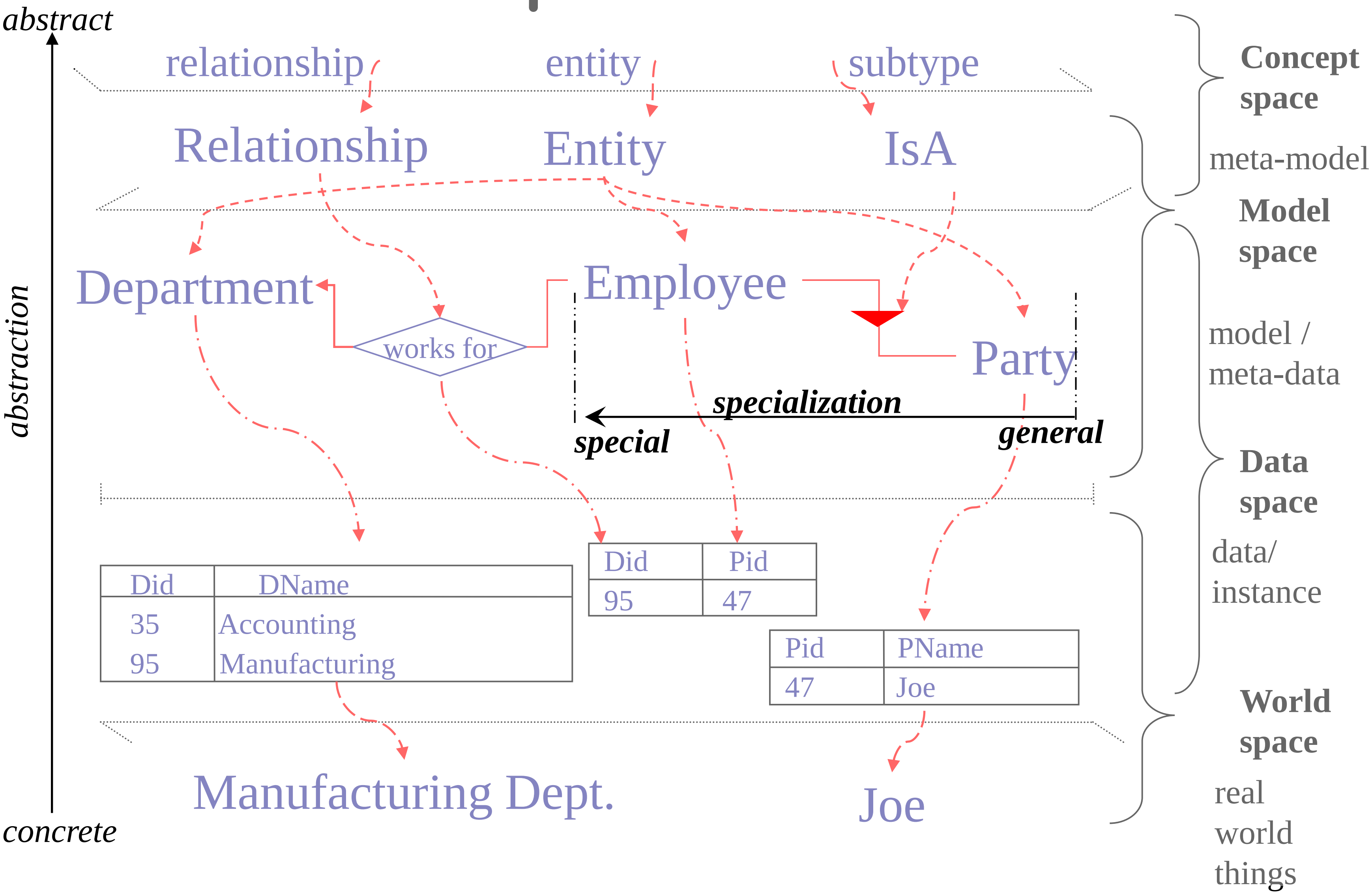


Decomposition

Three aspects of scale

- Abstractness, scope, and refinement
- Examples of dimensional independence:
  - E-R diagrams are abstract but have rich refinement when fully populated.
  - 19439 Genericity contains constructs for use along a generalization gradient with a range of phase abstractions.
  - Zachman interrogative proto-types are abstract with concrete model contents.
  - DoDAF views span operational abstractions with technical refinement.

Scope Dimensions



Zachman Framework for Enterprise Architecture

ENTERPRISE ARCHITECTURE - A FRAMEWORK ™

	DATA *What*	FUNCTION *How*	NETWORK *Where*	PEOPLE *Who*	TIME *When*	MOTIVATION *Why*	
SCOPE (CONTEXTUAL) *Planner*	List of Things Important to the Business; ENTITY = Class of Business Thing	List of Processes the Business Performs; Function = 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 Organizations	List of Events Significant to the Business; Time = Major Business Event	List of Business Goals/Strat; Ends/Means=Major Bus. Goal/ Critical Success Factor	SCOPE (CONTEXTUAL) *Planner*
ENTERPRISE MODEL (CONCEPTUAL) *Owner*	e.g. Semantic Model; Ent = Business Entity, Reln = Business Relationship	e.g. Business Process Model; Proc. = Business Process, I/O = Business Resources	e.g. Logistics Network; 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	ENTERPRISE MODEL (CONCEPTUAL) *Owner*
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	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; Ent = Segment/Table/etc., Reln = Pointer/Key/etc.	e.g. "System Design"; Proc.= Computer Function, I/O = Screen/Device Formats	e.g. "System 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 CONSTRAINED MODEL (PHYSICAL) *Builder*
DETAILED REPRESEN- TATIONS (OUT-OF- CONTEXT) *Sub- Contractor*	e.g. Data Definition; Ent = Field, Reln = Address	e.g. "Program"; Proc.= Language Stmt, I/O = Control Block	e.g. "Network Architecture"; Node = Addresses, Link = Protocols	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*
FUNCTIONING ENTERPRISE	e.g. DATA	e.g. FUNCTION	e.g. NETWORK	e.g. ORGANIZATION	e.g. SCHEDULE	e.g. STRATEGY	FUNCTIONING ENTERPRISE

Zachman Institute for Framework Advancement - (810) 231-0531

Roles ↓

Interrogatives →

(used with permission)

Aspects of Formalization

- **Structure**:
  - both tree (decomposition) and grid (ordinant)
  - frames and sub-frames
- **Connections**:
  - between frame components
  - respects purposive order
- **Constraints**:
  - model and instance
  - beyond structure and connection
- **Views**:
  - generalizes "view" in existing frameworks
  - defined on structure
  - attempts to carry forward connections and constraints

Zachman meta-meta model

branch frames:

$\mathcal{F}_\alpha \quad \langle IC_\alpha, OC_\alpha, \mathcal{SF}_\alpha, \Phi_\alpha \rangle$

leaf frames:

$\mathcal{F}_\alpha \quad \langle IC_\alpha, OC_\alpha, \quad S_\alpha \rangle$

where

$IC_\alpha \quad \subseteq \mathcal{D}$

$OC_\alpha \quad \subseteq \mathcal{D}$

$\left.\begin{array}{l} \mathcal{E}OC_{\alpha,r} \\ \mathcal{E}IC_{\alpha,r} \end{array}\right\} \subset \mathcal{D}$ restricted to row r

$\mathcal{SF}_\alpha \quad : \mathcal{R} \times I \times \mathcal{D} \rightarrow \mathcal{F} \cup \mathcal{VF}$

$\Phi_\alpha \quad \subseteq \cup_{r \in \{\theta\} \cup \mathcal{R}} (\mathcal{E}OC_{\alpha,r} \times \mathcal{E}IC_{\alpha,r'})$

$\mathcal{T}ypes \quad \mathcal{D} \cup \{$SET OF $d : d \in \mathcal{D}\}$

$S_\alpha \quad : \mathcal{D} \rightarrow \cup_{n \in \aleph} \mathcal{T}ypes_\alpha{}^n$

Aspects in Zachman model

- **Structure**:
  - ordinant: $\varepsilon_{OC}, \varepsilon_{IC}$
  - tree: $\mathcal{SF}$
- **Connections**:
  - Φ
- **Constraints**:
  - predicates using defined constructs
- **Views**:
  - sets satisfying predicates

Abbreviations

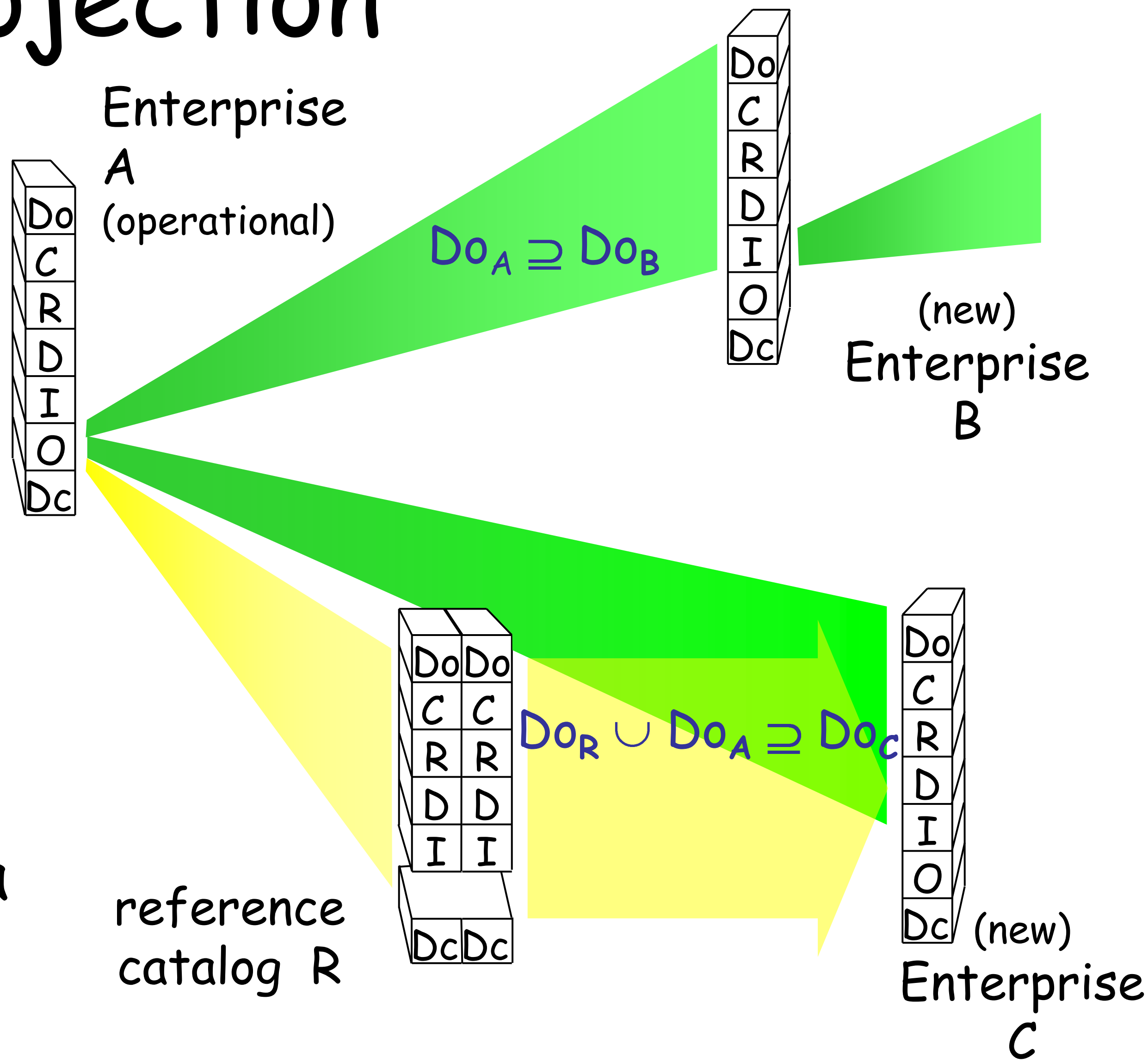
- Bridge from formalism to human use, providing
  - familiarity
  - uniformity
  - structure
- A single name $\mathcal{N}$ can abbreviate
  - a path: $\mathcal{N} \rightarrow \langle r,I,d\rangle \langle r, I, d\rangle$
  - R x I cell coordinates: $\mathcal{N} \rightarrow \langle r, I, .\ \rangle$
  - a path template with multiple substitutions: $\mathcal{N} \rightarrow \langle r, I, .\ \rangle \langle r, I, .\ \rangle$
- Use to shorten complicated paths

Transformations

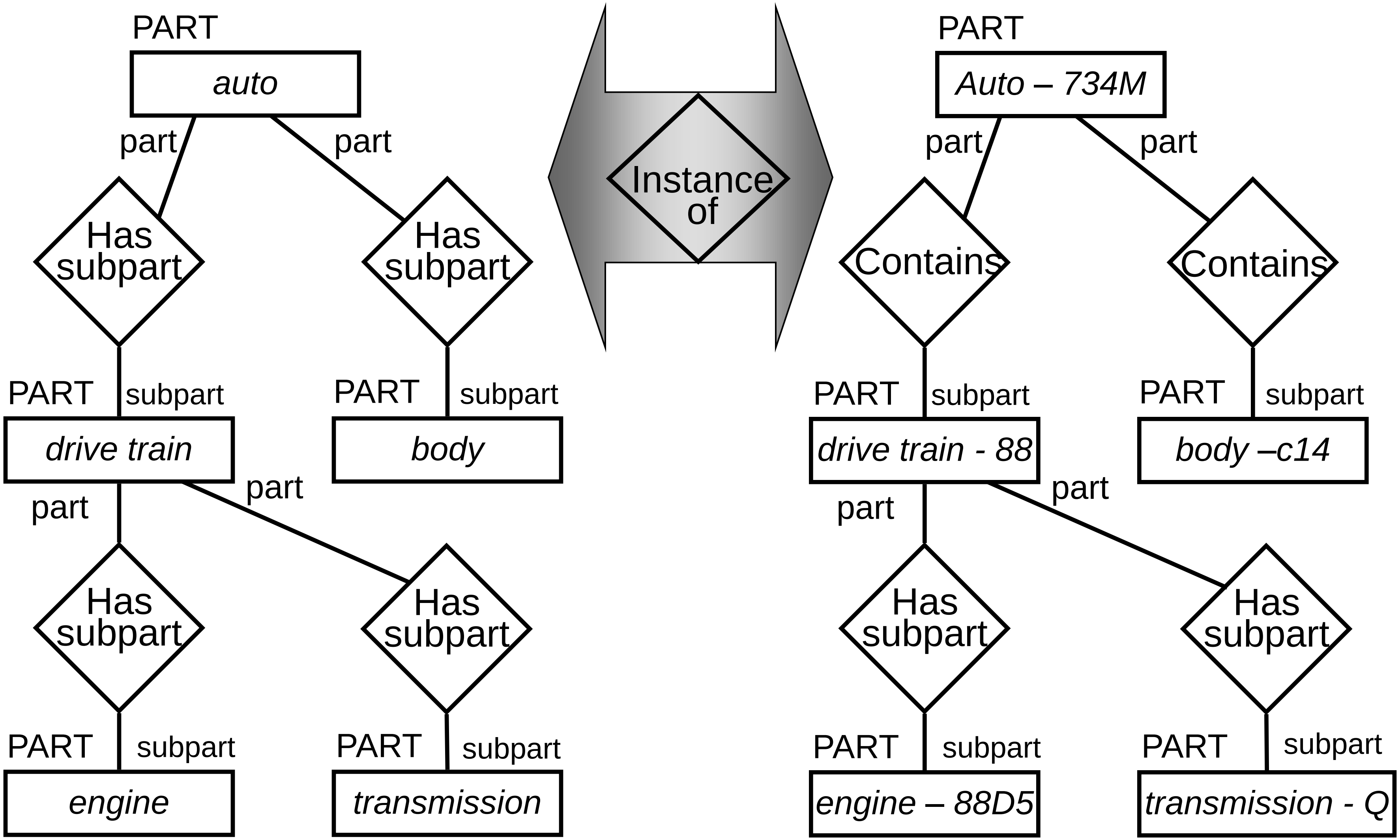
- Projection
  - broadly applied math notion
- Instantiation
  - only applies to abstraction
- Refinement
  - decomposition may be recursive for complicated objects
- Specialization
  - adds attributes
- Derivation
  - domain-specific transforming concept
- Linking
  - graph-like edges

Projection

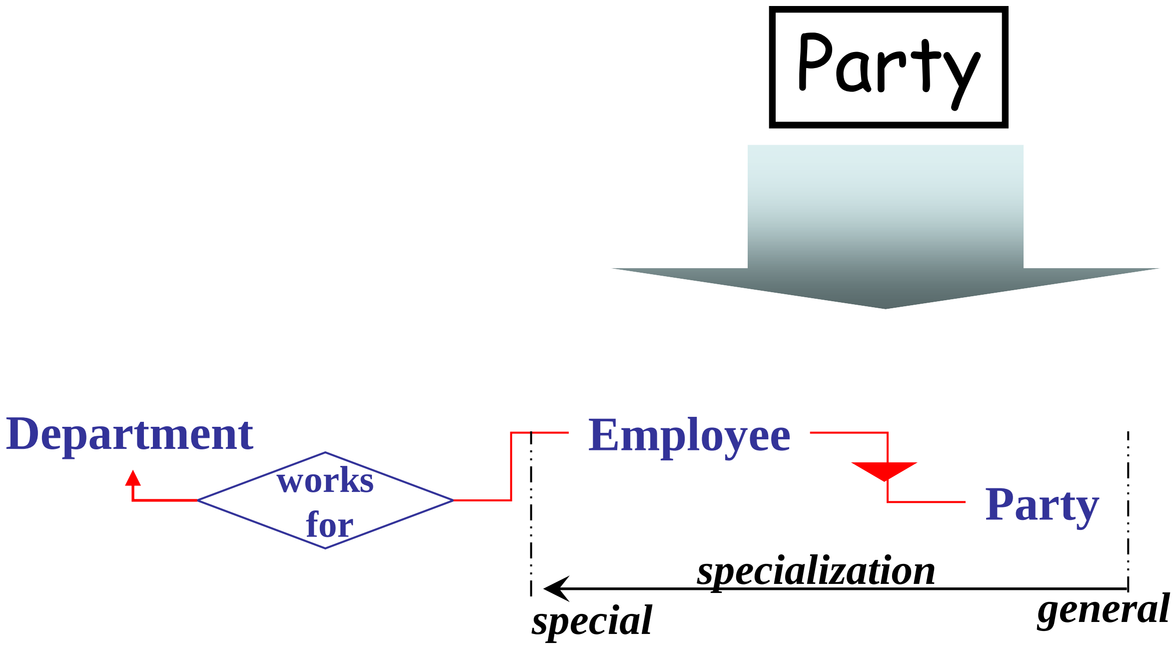
From the set of architectural models, select a sub-set that is useful to a set of tasks during a life cycle phase.



Instantiation

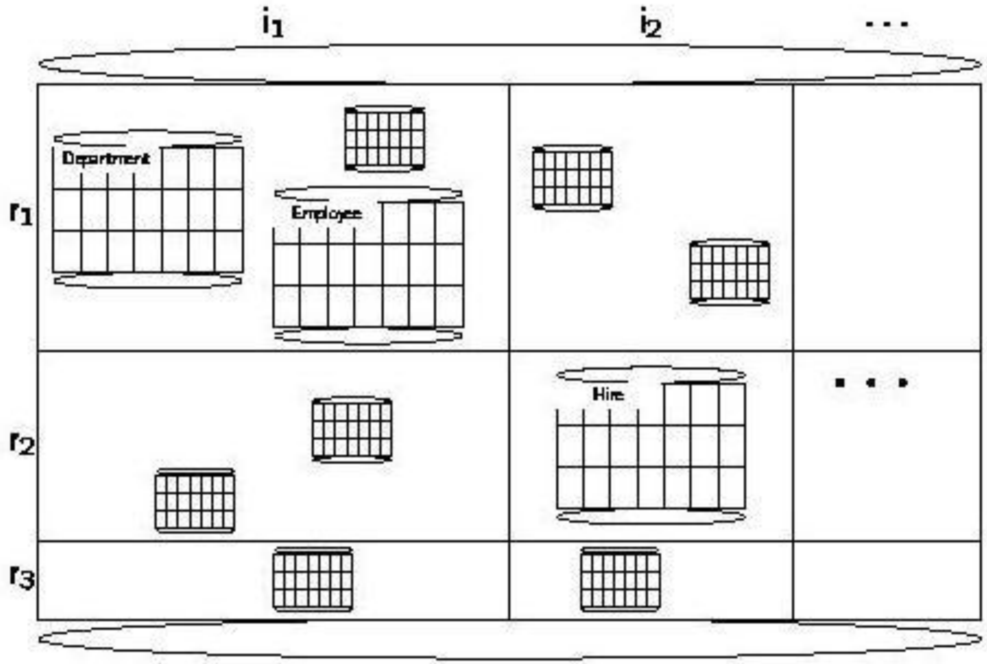


Specialization

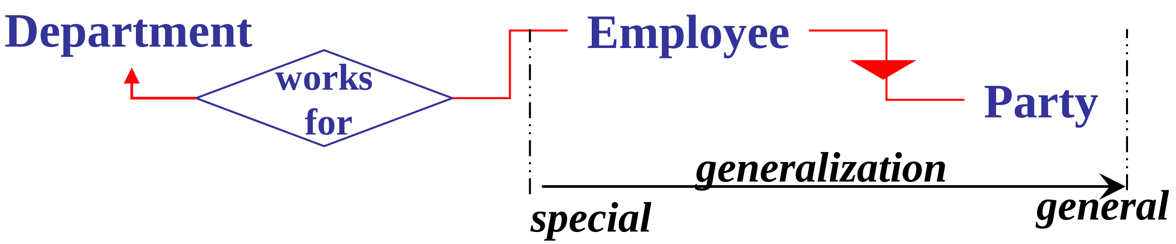


Refinement

- Refine an architectural model by addition of significantly more detail to ensure its use for a task during the life cycle



Derivation



Did	DName
35	Accounting
95	Manufacturing

Did	Pid
95	47

Pid	PName
47	Joe

Linking

- Take elements from different architectural models to satisfy data or decision needs for a task during the life cycle phase.