From Small-Grained to Large-Grained Semantic Interoperability

Université A. Mira
Béjaïa, Algeria,
April 2015

Nacer.Boudjlida@loria.fr
U. de Lorraine-LORIA, http://www.loria.fr/~nacer

Presentation Outline

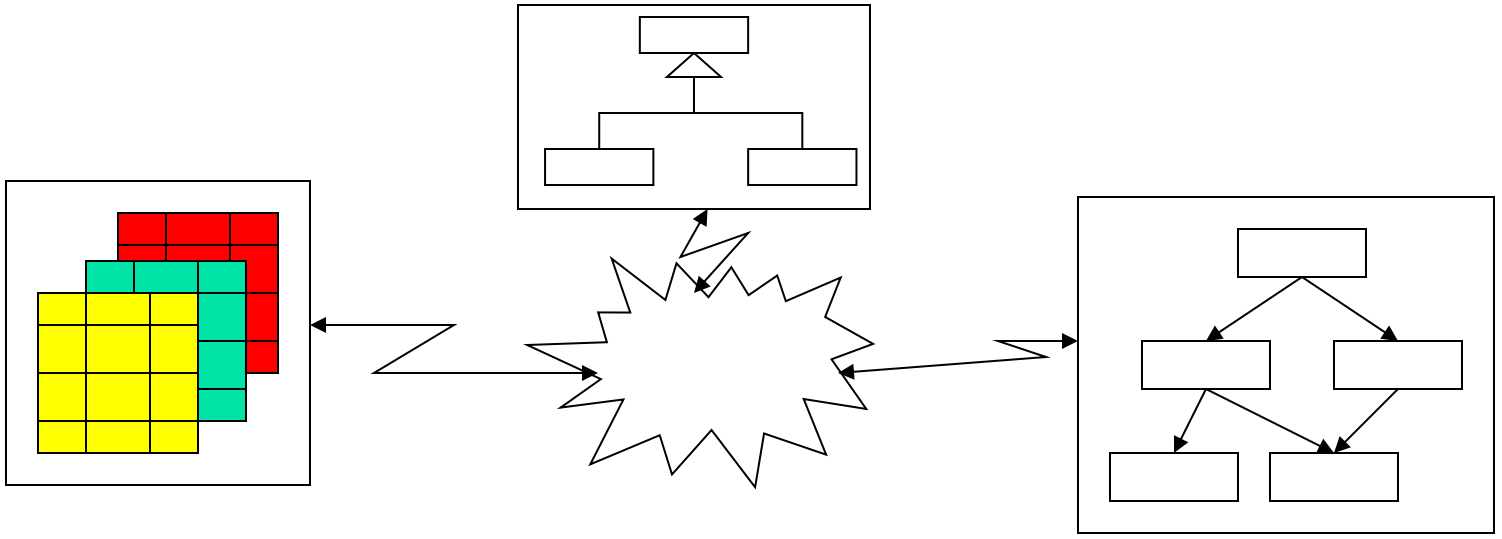
I. Integration and Interoperability
   1. Problem Statement
   2. Approaches (“process” vs “objets”)
   3. Architectures

II. Ontology and Semantic Annotations

III. Enterprise Modeling and Large scale Semantic Interoperability

Integration and Interoperability

- *Large scale Integration*
  - Locally/widely distributed enterprises
  - Federated enterprises, Virtual Enterprises, etc.
  - Enterprise Alliances, Merges, and so on.



Integration and Interoperability

- *Small Scale* : Illustrative *Example*
  - concat : string x string → string
  - string1: < size, value > (<4, ABCD>)
  - string2: < value, ending mark > (<EF, #>)
  - Result:
    - <5, ABCDEF>
    - OR <ABCDEF, #>

- *Solutions:*
  1. Process-Oriented Interoperability
  2. "Object" Oriented Interoperability

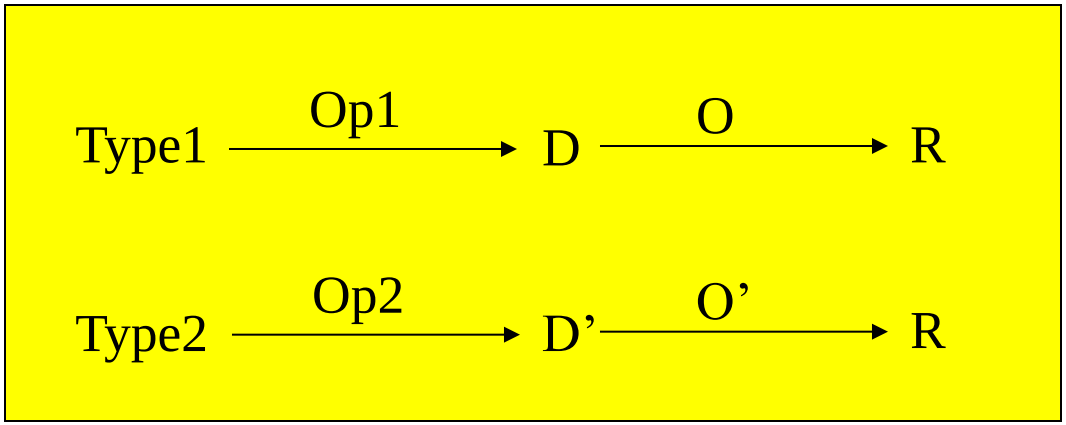
Process vs Object-Oriented Interoperability

- *Syntax:*
  - Concat : string x string $\rightarrow$ string
- *Representation*:
  - string1: < size, value >
  - string2: < value, ending mark >
- *Solution 1* : Process-Oriented Interoperability
  - Concat1 : string1 x string1 $\rightarrow$ string1
  - Concat2 : string2 x string2 $\rightarrow$ string2
  - One or Two implementations of concat

Process-Oriented Interoperability

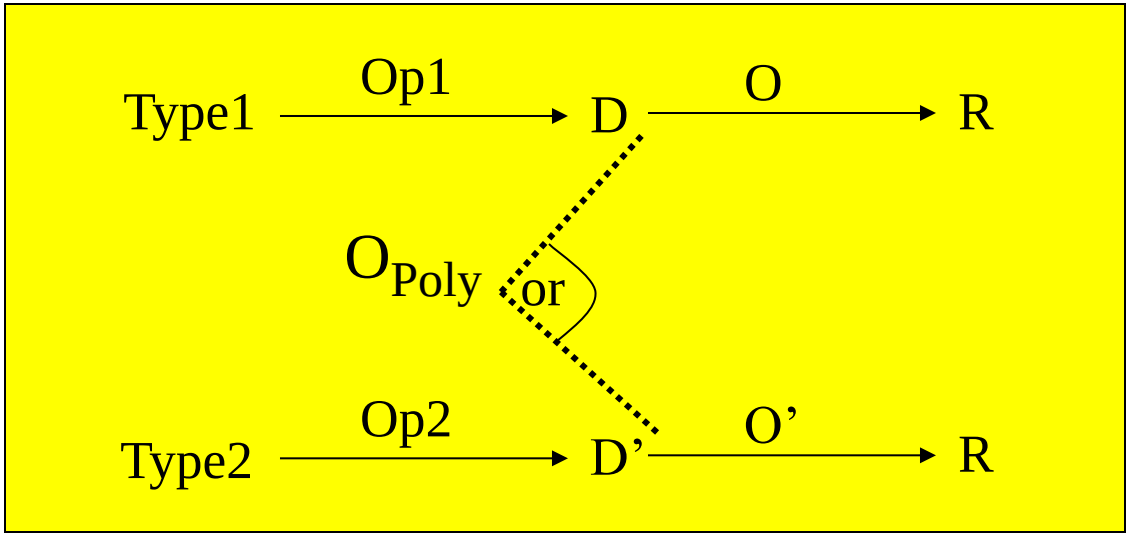
1/3. *Polymorphism with no overloading nor coercion*

- Op1: Type1 $\rightarrow$ D
- Op2: Type2 $\rightarrow$ D' and D $\cong$ D'
- O: D $\rightarrow$ R
- How to apply O on Op2(x : Type2)?
- Two implementations of O (O and O')



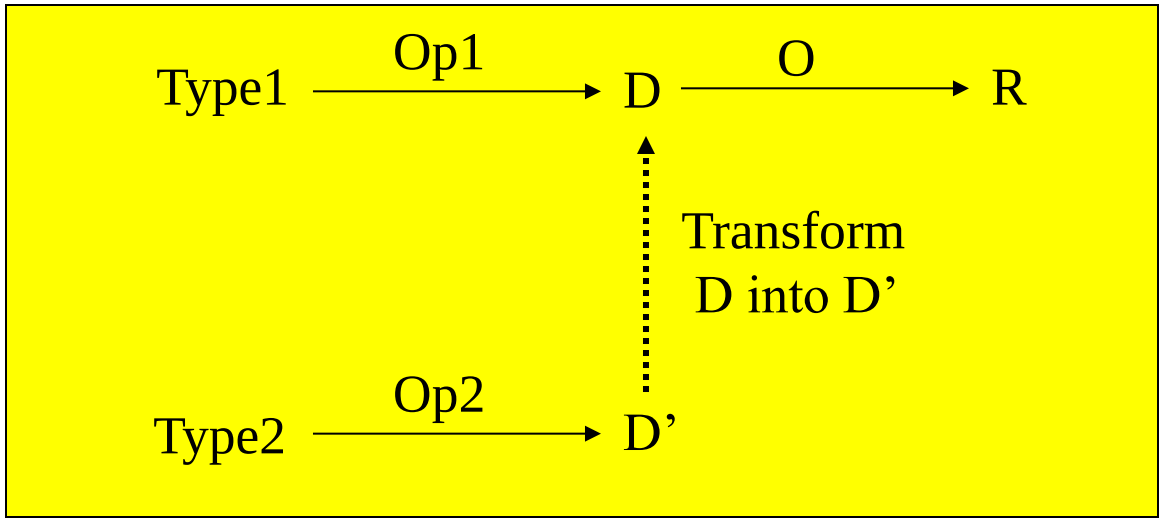
2/3. *Polymorphism without coercion, with overloading*

- Apply O on Op2(x : Type2): A unique name (O_{poly})
- Invoke O or O' depending on the type of the parameter



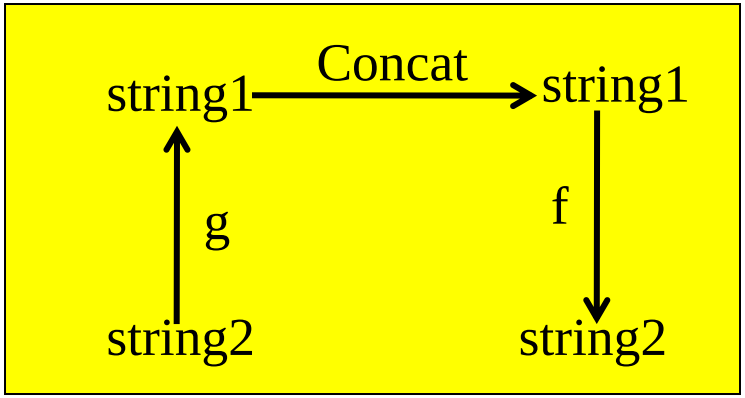
3/3. *Polymorphism with type coercion*

- Apply O to Op2(x : Type2)



Object-Oriented Interoperability

- *Solution 1* : Process-Oriented interoperability (done)
- *Solution 2* : Object-Oriented interoperability
  - Concat : string1 x string1 → string1
  - Concat : **g**(string2) x **g**(string2) → **f**(string1)
  - **g** : string2 → string1
  - **f** : string1 → string2



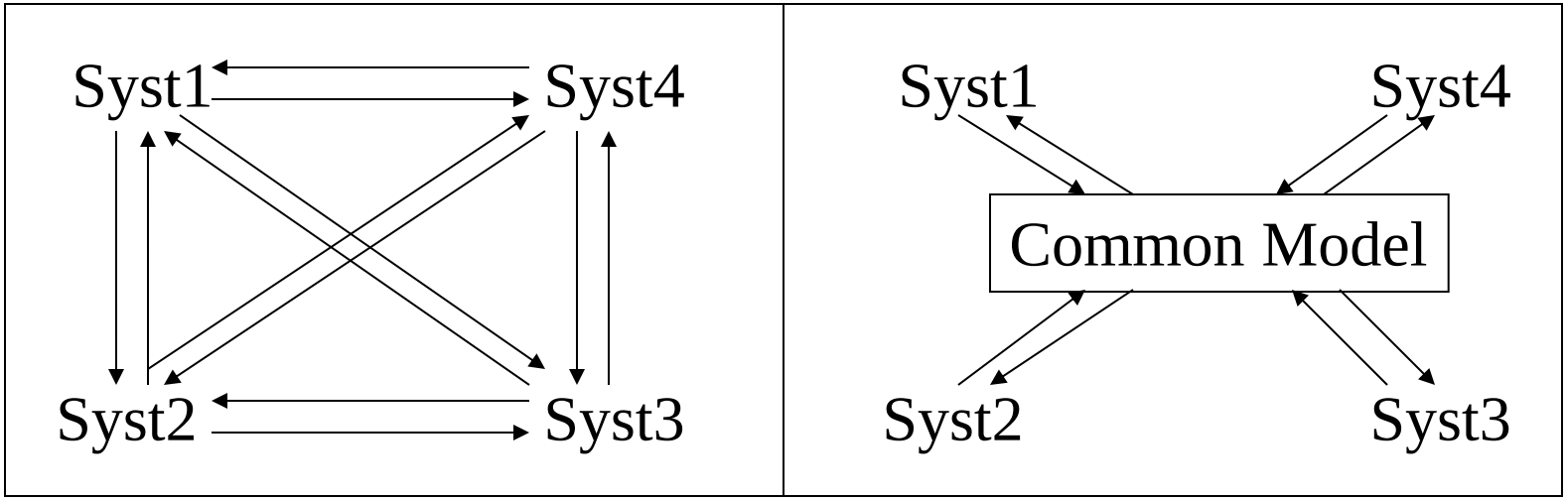
- A single implementation but Structure Transformation/Mediation

Object-Oriented Interoperability

- How to determine g and f?
- Data structure complexity?
- Structured vs (semi-)/unstructured?
- Data Semantics?
- Meta-data availability?

Object-Oriented Interoperability

1. Naïve Approach : n*(n-1) converters
2. Common Model: 2*n converters



Object-Oriented Interoperability

1. What a common Model?
2. What an architecture?
3. What a process?

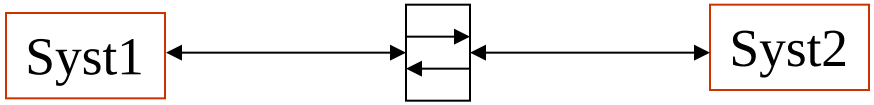
1/3: What a "common model"?

- Ad hoc
  - *Example* : Data Model of a common repository
- Specific
  - *Example* : Relational, JDBC, ODBC, UEML
- Standard
  - OMG/ODMG Object Model
  - XML
  - WfMC Interoperability Interface
  - OMG/CORBA IDL

2/3: What an "architecture"?

- Encapsulation/Wrapping

- Mediation ("Format negociators")



3/3: What a process?

1. Structure Traversal/Comparison
2. Conflict Solving
3. Integrate-Restructure
4. Convert/transform the values

- Case of :
  1. Strongly typed data
  2. Un-structured or semi-structured data

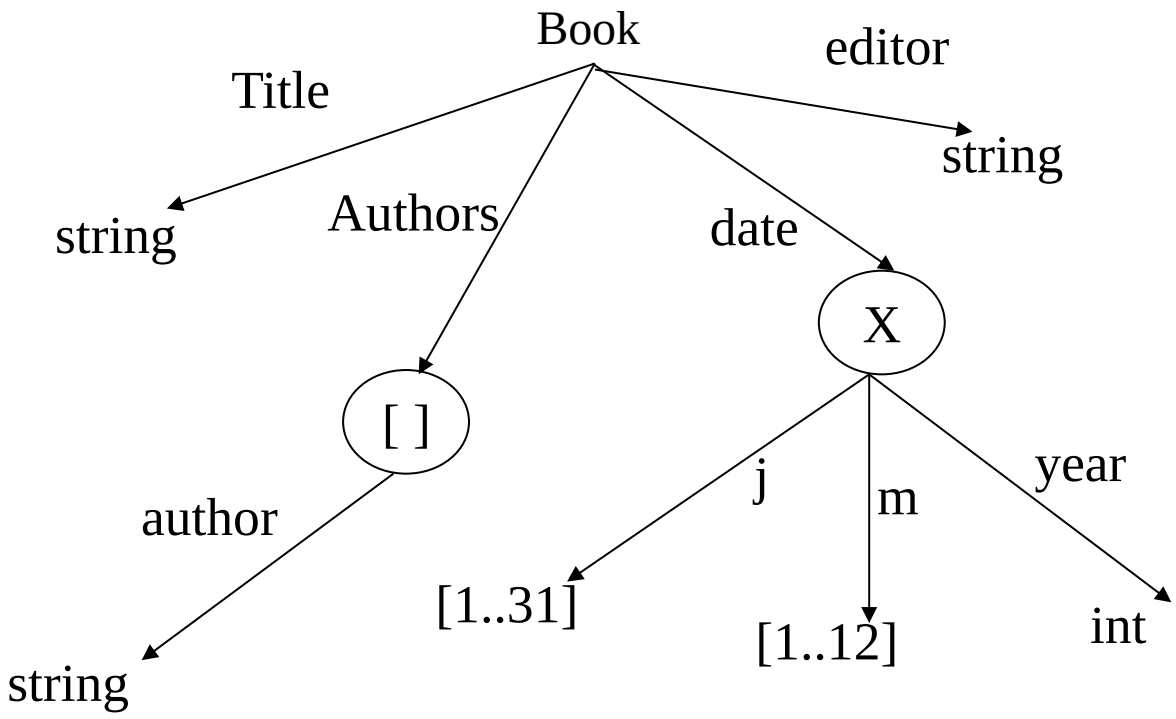
1/2 Strongly typed data

Note : Data tree/graph (Type & instance)

- *Leaves*: data types
- *Internal nodes*: Datatype constructors
- *Edges*: labelled with attribute names
- Notations:
  - <X1 : T1, …, Xn : Tn> : cartesian product (record)
  - [E : T] : list of E elements of type T
  - * (E : T): set of E elements of type T
  - [i..j] : Values range

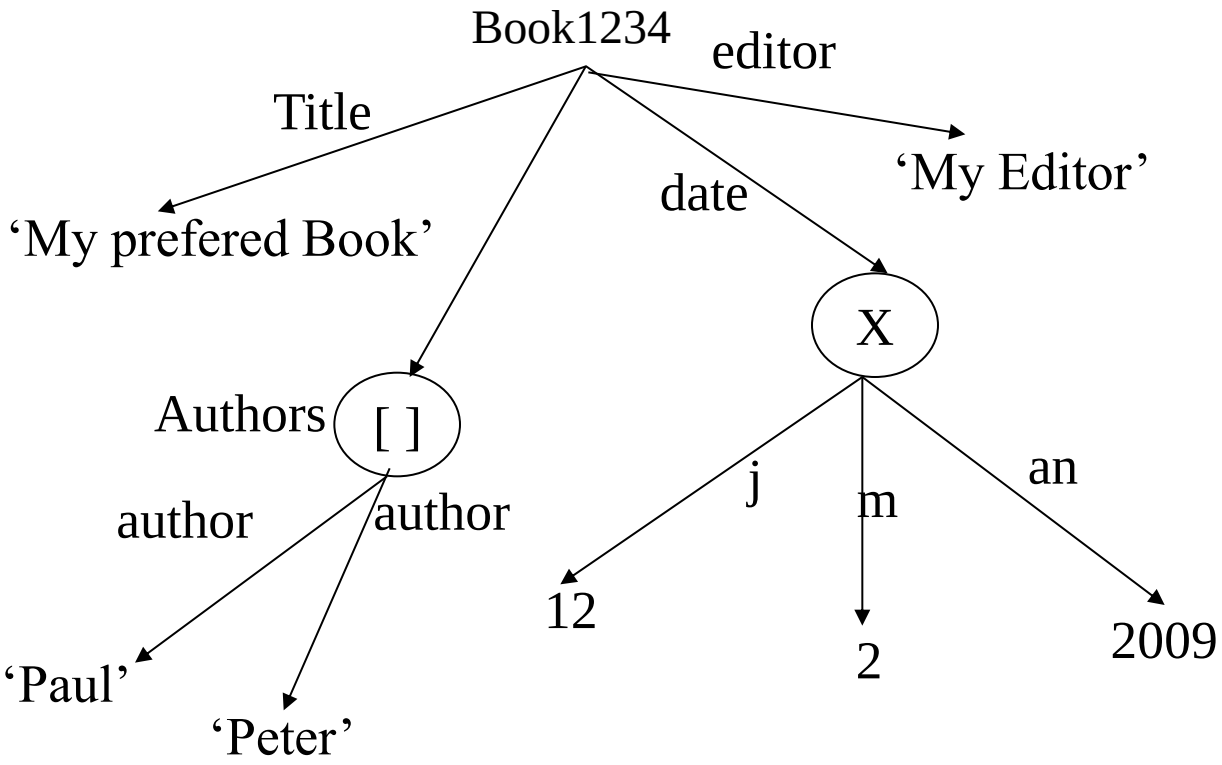
1/2 Strongly typed data (cont'd)

- Book:< Title: string, Authors: [author: string],
  date : < j: [1..31], m: [1..12], year: int>, editor : string>
- Data Type Tree:



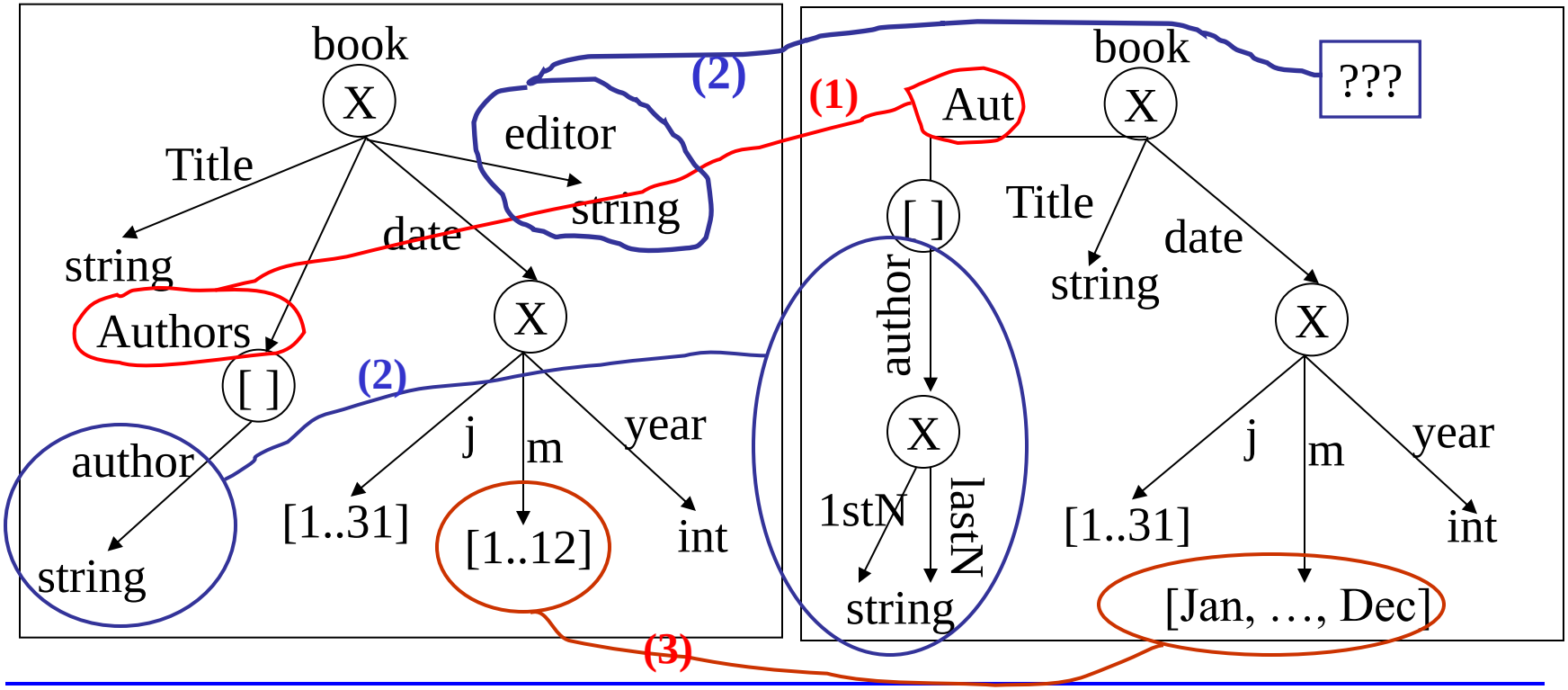
1/2 Strongly typed data (cont'd)

- Instance (data) tree:



1/2 Strongly typed data (cont'd)

- Integration & 'Conflicts' [cf. Sheth92]
  - Naming (1) , Structures (2)
  - Domains (3) , etc.



1/2 Strongly typed data (cont'd)

1. Comparison

- Naming: 'Aut' → 'Authors'
- Structure : <author: string> →
  <author : <LastN: string, 1stN: string>>
- Domain : <m : [1..12]> → <m : [Jan..Dec]>
- Missing element : <editor : string> → ∅

1/2 Strongly typed data (cont'd)

2. Conflict Solving

- Aut $\rightarrow$ Authors
- <author: string> $\rightarrow$
  <author : <LastN: string, 1stN: string>>
- <m : [1..12]> $\rightarrow$ <m : [Jan..Dec]>
- <editor : string> $\rightarrow$ $\varnothing$

1/2 Strongly typed data (cont'd)

3. Integration

3.a. 'Rewrite' the data structures

- <Authors: [author: string], Titre: string, … >

→ <Titre: string, Aut: [author: <LastN, 1stN: string >]..>

- Rewriting Rule: X(A1: [A2: T2], A3: T3, …)

→ X(A3: T3, B1: [A2: X(B2: T4, B3: T5)], …)

- Thanks to a kind of Algebra on Datatype Constructors

1/2 Strongly typed data (cont'd)

3.b. 'Rewrite' the data values

- <12, 2, 2009 > → <12, Feb, 2009 >
- author : string → author: <LastN, 1stN: string> ?

- [A, B, C, D] → [0..20] ?

- Finite set of datatype constructors
- Finite set of rewriting rules
- Many often, *ad hoc* solutions

1/2 Strongly typed data (end)

- Meta-data presumed available
- Common Representation Model: Datatype constructors and datagraphs
- Domains of the data values (usually) simple
- *Case of* :
  - No existing meta-data
  - Complex Values (text, sounds, images, etc.)
  - Fragmented and distributed values (ex: Web)
- Same kinds of problems and mechanisms?

2/2 Unstructured Data

- Self described objects
- No strong typing (structure)
- Data trees or graphs
  - Non cyclic data
  - Cyclic data (Recursive Data Structures)

2/2 Unstructured Data (end)

- Datagraph Equality/Comparison/Merging
- Complex semi-structured data transformation thanks to:
  - Data graph traversal
  - Resulting in a new graph, when needed

1. Case of data trees: Structural Recursion
2. Case of data graphs (cyclic data): bi-simulation

cf. S. Abiteboul, P. Buneman, and D. Suciu. Data on the Web; From Relations to Semi-Structured Data and XML. *Morgan Kaufmann Publishers*, San Francisco, CA, 2000.

Let's go further: Interoperability Dimensions

- A software engineering perspective (to enlarge the view)
- (as an example) Software System : {Components}
  - Possibly heterogeneous and distributed components
  - That communicate
    - Data, Results, Events, Requests for services, etc.
  - AND Cooperate in task achievement
    - Synchronization
    - Sharing
    - etc.

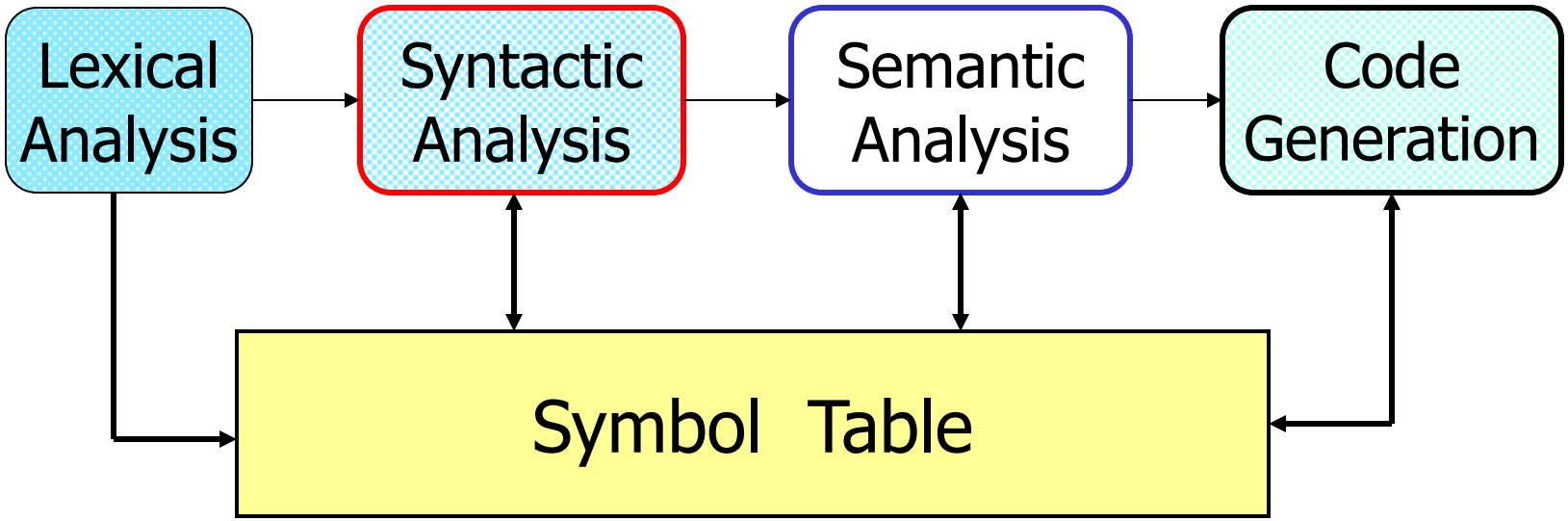
Interoperability Dimensions

- Dimensions (Soft. Eng. Environments):
  1. Interfaces : *look and feel*
  2. Platforms : Transparency
     - Technologies, Operating Systems
     - Distribution, Protocols, etc.
  3. Data :
     - Communication and sharing (despite the possible heterogeneity)
  4. Control : Notification/Reaction to events
  5. Process Models : Rules that govern the behaviour of the software systems

Integration and Interoperability Models

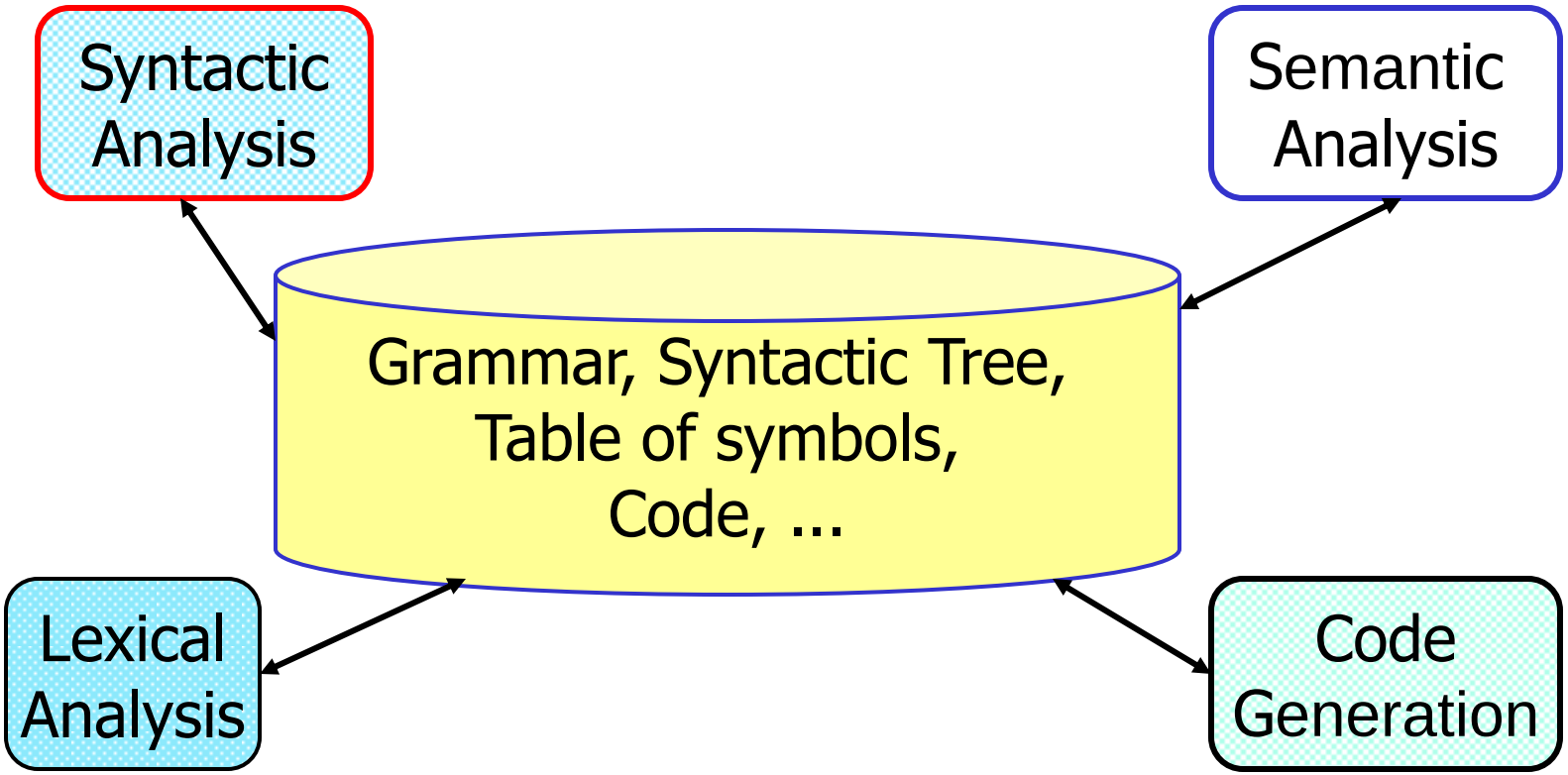
A. *(Some) Models for the* communication and the sharing of data/information
   1. Data flow *(pipe and filter)*
   2. Repository (private, shared)
   3. Client/Server
B. Control Model
C. Architectures
   1. Generic
   2. Reference and Standard

1. Data flow *(pipe and filter): Compiler*



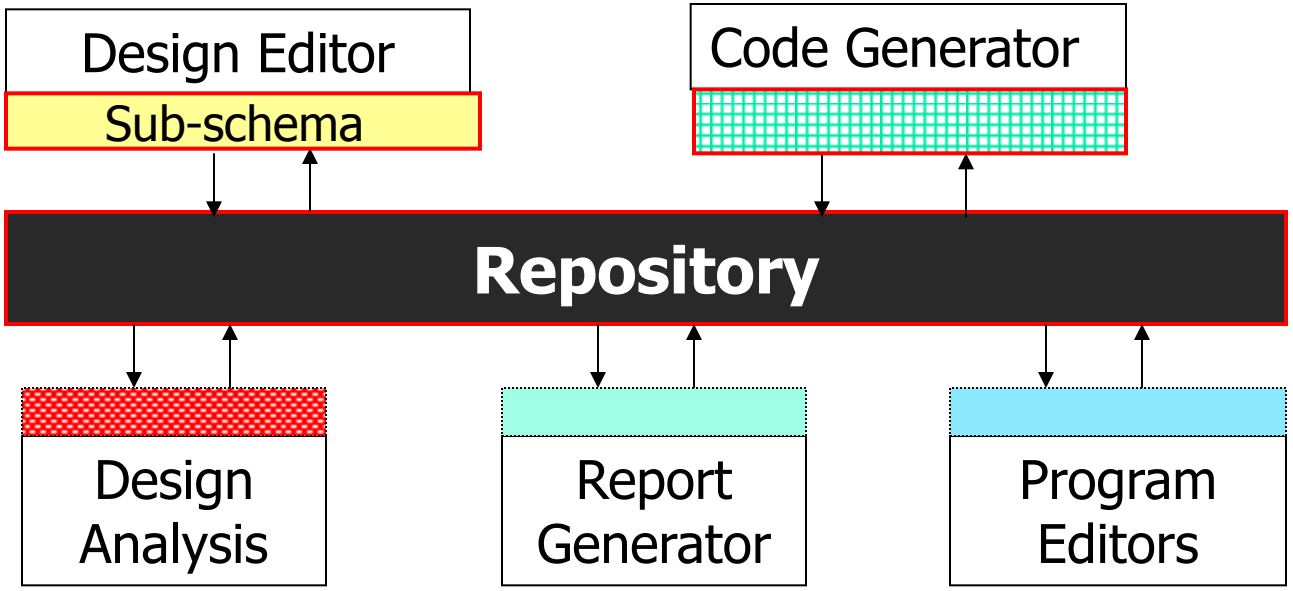
Models for sharing and exchange

2. *Repository* : Common « Database »
   - Example: Compiler (tight integration)



2.a) Shared Repository Model

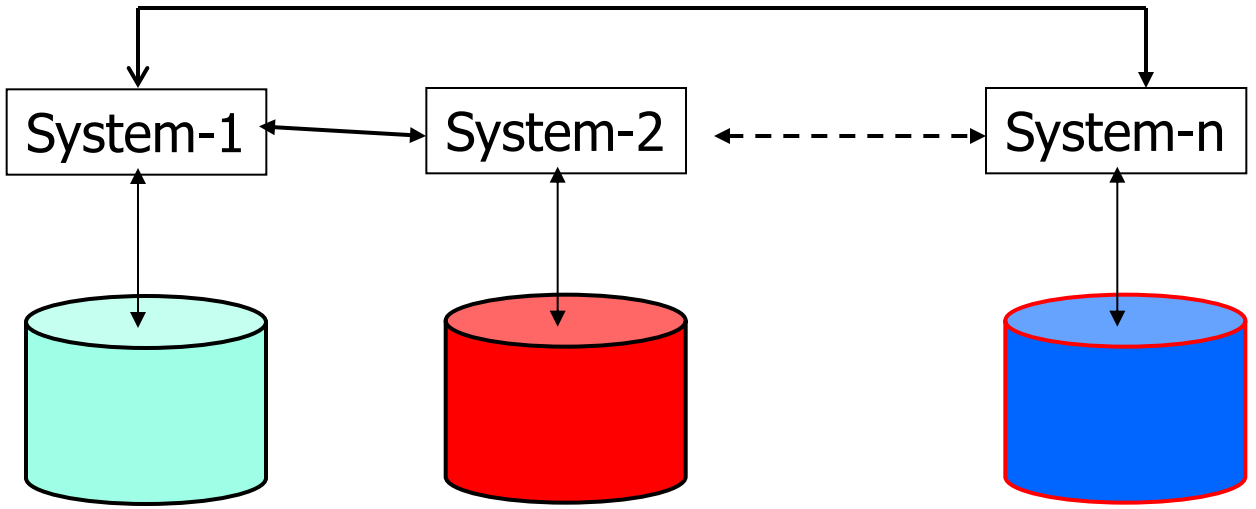
- Example: CASE Environment
- 'Enveloppes' (Wrappers)



2.b) Private Repository Model

b. Private Repository

- Communication by message passing



Private Repository Model (cont'd)

- Private Repository

(+) Independent sub-systems: production of data without carying about their usage by others

(-) Negociation/Mediation of formats (often) required

Repository-based Model (cont'd)

- *Common repository*

(+) Efficient mean for exchange (no transfer)

(+) Protection/Sharing thanks to sub-schemas

(+) Repository Manager services

- Back-ups
- Recovery in case of failure
- Security
- Access Control

Repository-based Model (end)

- *Common repository*

(-) Common Data Format/Model or mediators

(-) Specific security, back-up, etc. policies

(-) Evolutivity : Integration of new systems

(-) Distribution of the Repository

Integration and Interoperability Models

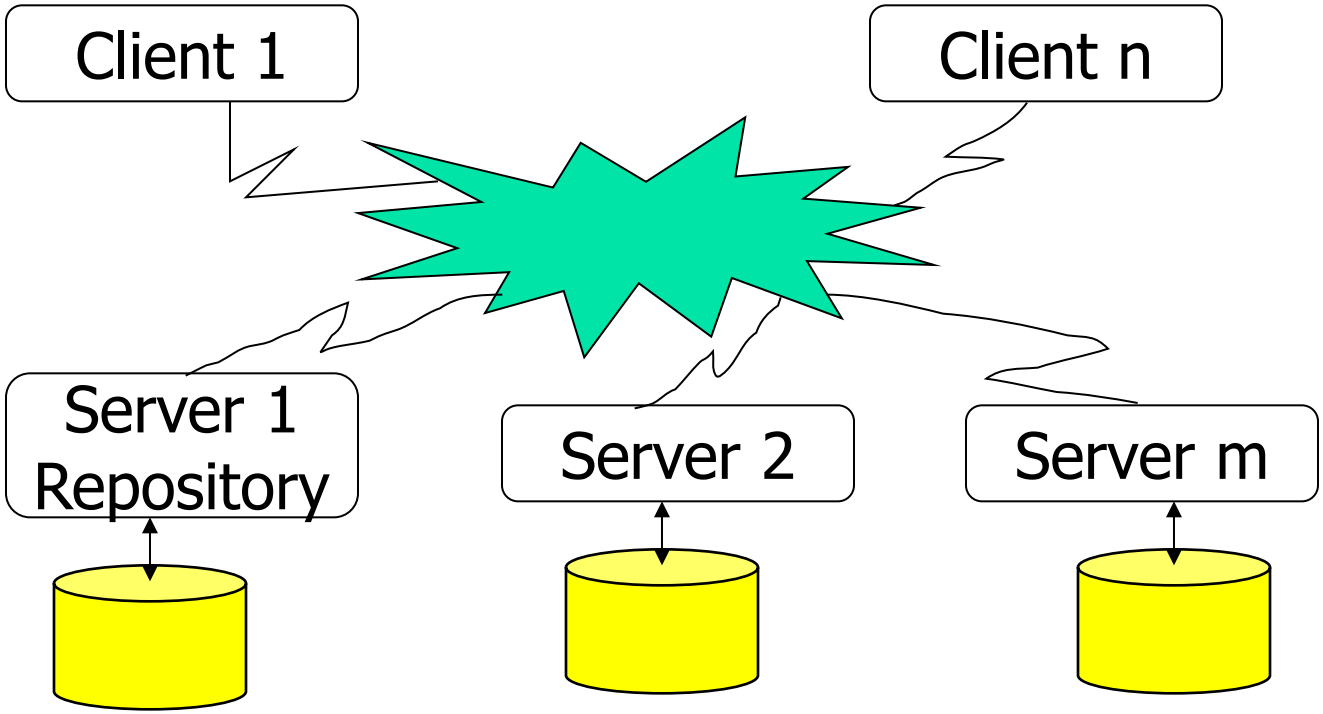
A. *(Some) Models for the* communication and the sharing of data/information
   1. Data flow *(pipe and filter)*
   2. Repository (private, shared)
   3. Client/Server
B. Control Model
C. Architectures
   1. Generic
   2. Reference and Standard

A.3. Client/Server Model

- Client: Service requester
- Server: Service offerer
- Middleware (/): the “Glue”
- Client (server resp.) may be a server (client resp.)
- Client & Server on same or distinct machines

A.3. Client/Server Model

- Example: Client/Server and repositories



Integration and Interoperability Models

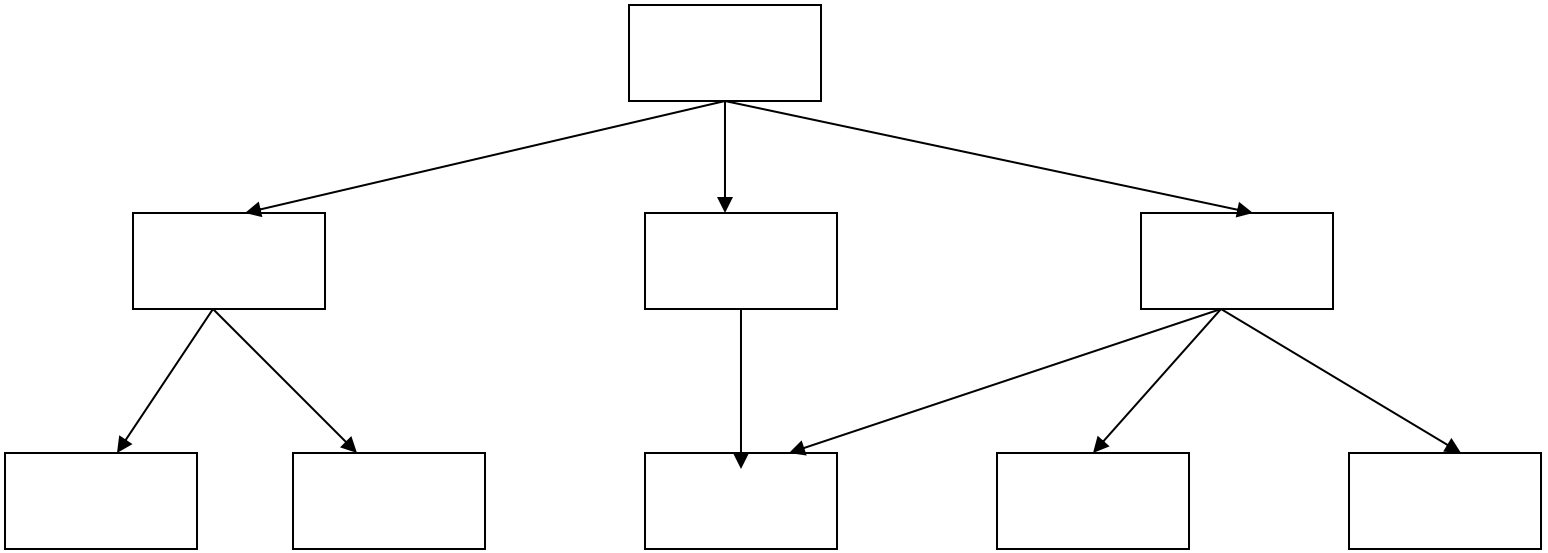
A. *(Some) Models for the* communication and the sharing of data/information
   1. Data flow *(pipe and filter)*
   2. Repository (private, shared)
   3. Client/Server
B. Control Model
C. Architectures
   1. Generic
   2. Reference and Standard

Control Models

- Control the inter-actions between the sub-systems
  - Centralized
  - Event Based

Centralized Control Models

- *call-return model* : cf.
  - WS orchestration
  - Functional design methods



Event-Based Control Models

- *Broadcast*
- *Push* :
  - Event producer is the initiator
  - Event transferred to the consumer
- *Pull* :
  - Consumer “worries” about event arrivals

Integration and Interoperability Models

A. *(Some) Models for the* communication and the sharing of data/information
   1. Data flow *(pipe and filter)*
   2. Repository (private, shared)
   3. Client/Server
B. Control Model
C. Architectures
   1. Generic
   2. Reference and Standard

Architectures for interoperability

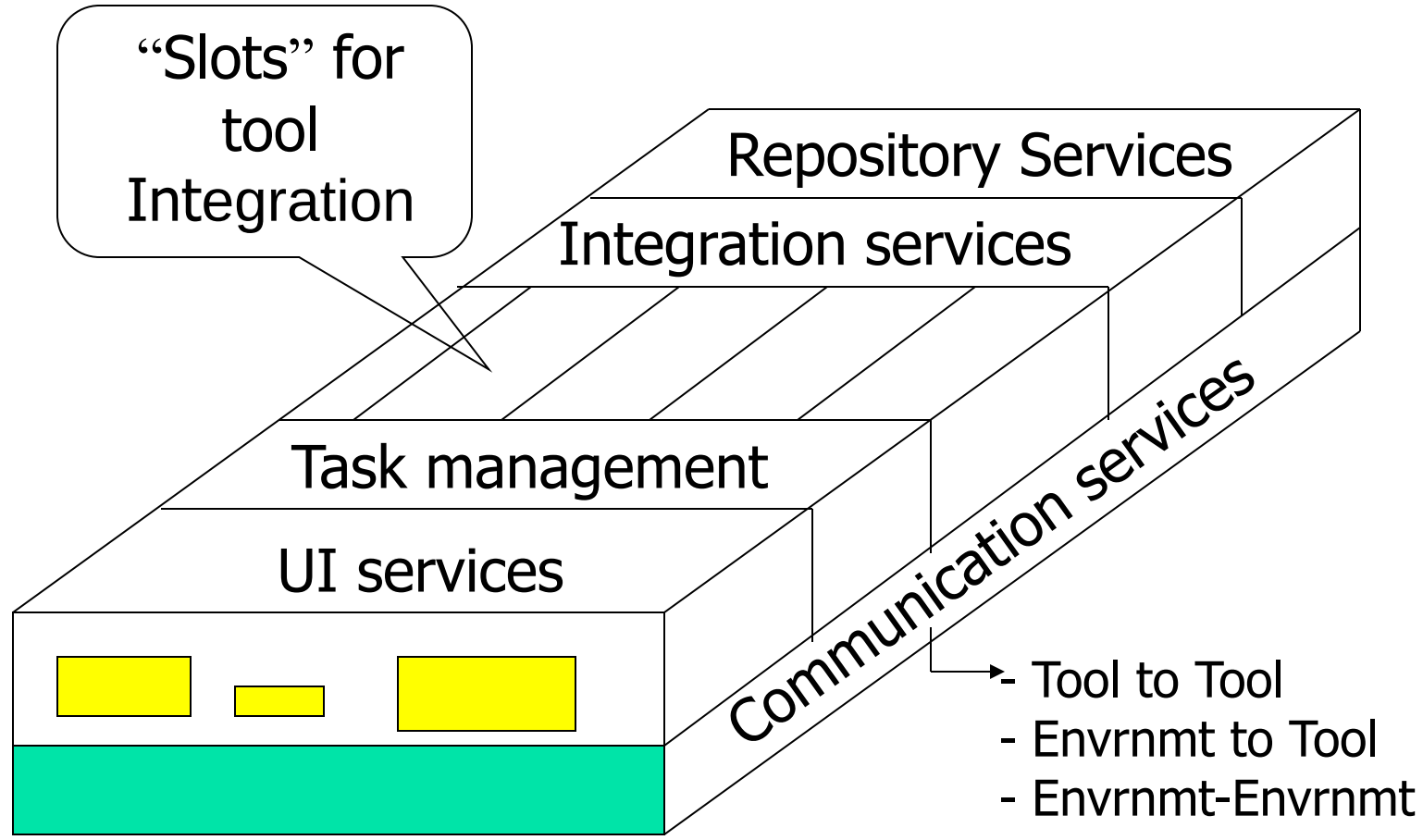
- Generic Architectures

- Reference/Standard Architectures:
  - To describe a class of systems
  - To be able to compare the systems of a domain

2/2. Reference Architectures

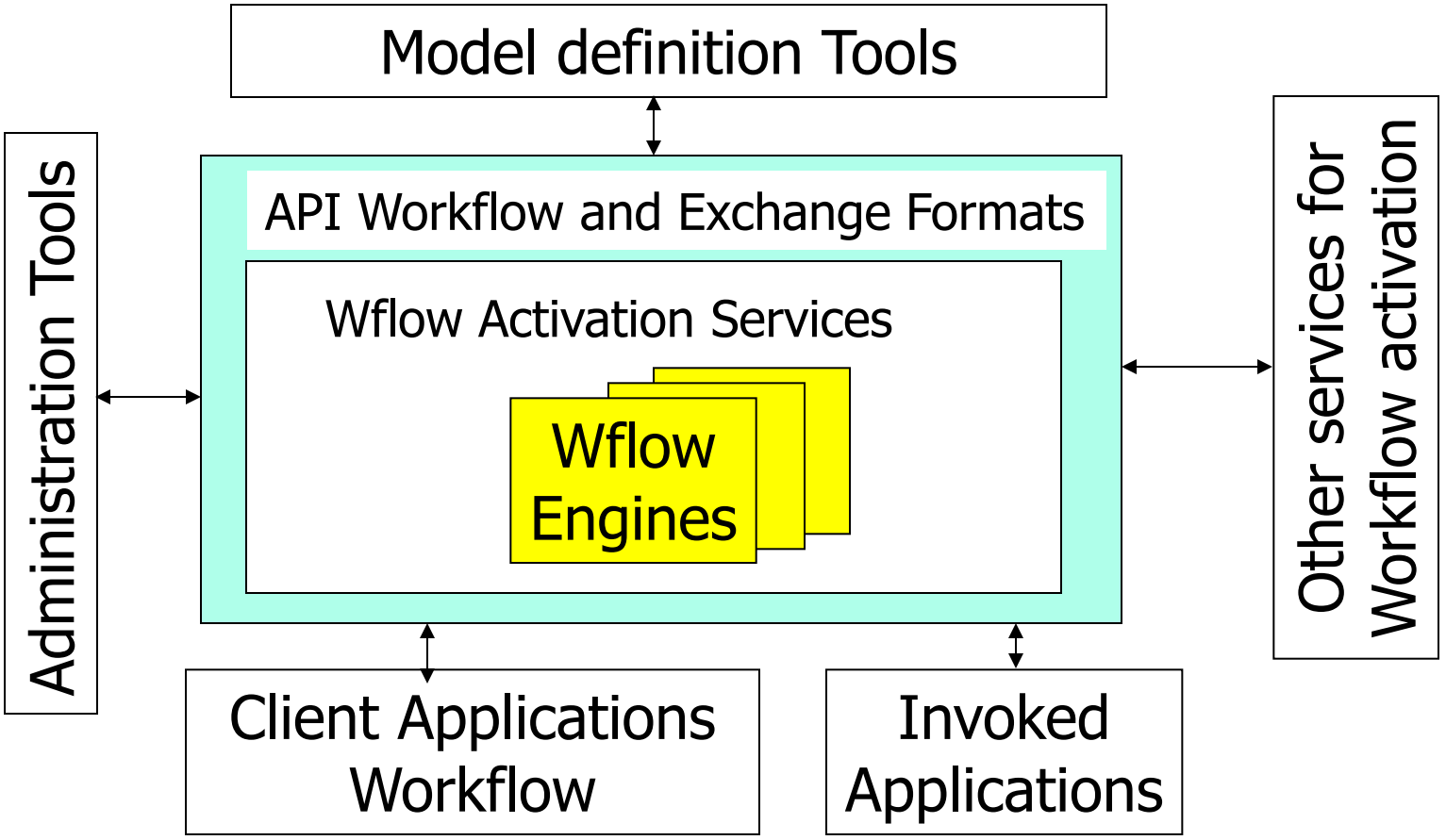
- Integrated Development Environments
- NIST/ECMA
  - National Institute of Standard and Technology
  - European Computer Manufacturers Association

ECMA Reference Architecture



2/2. Reference Architectures: Example

- WfMC : Workflow Management Coalition

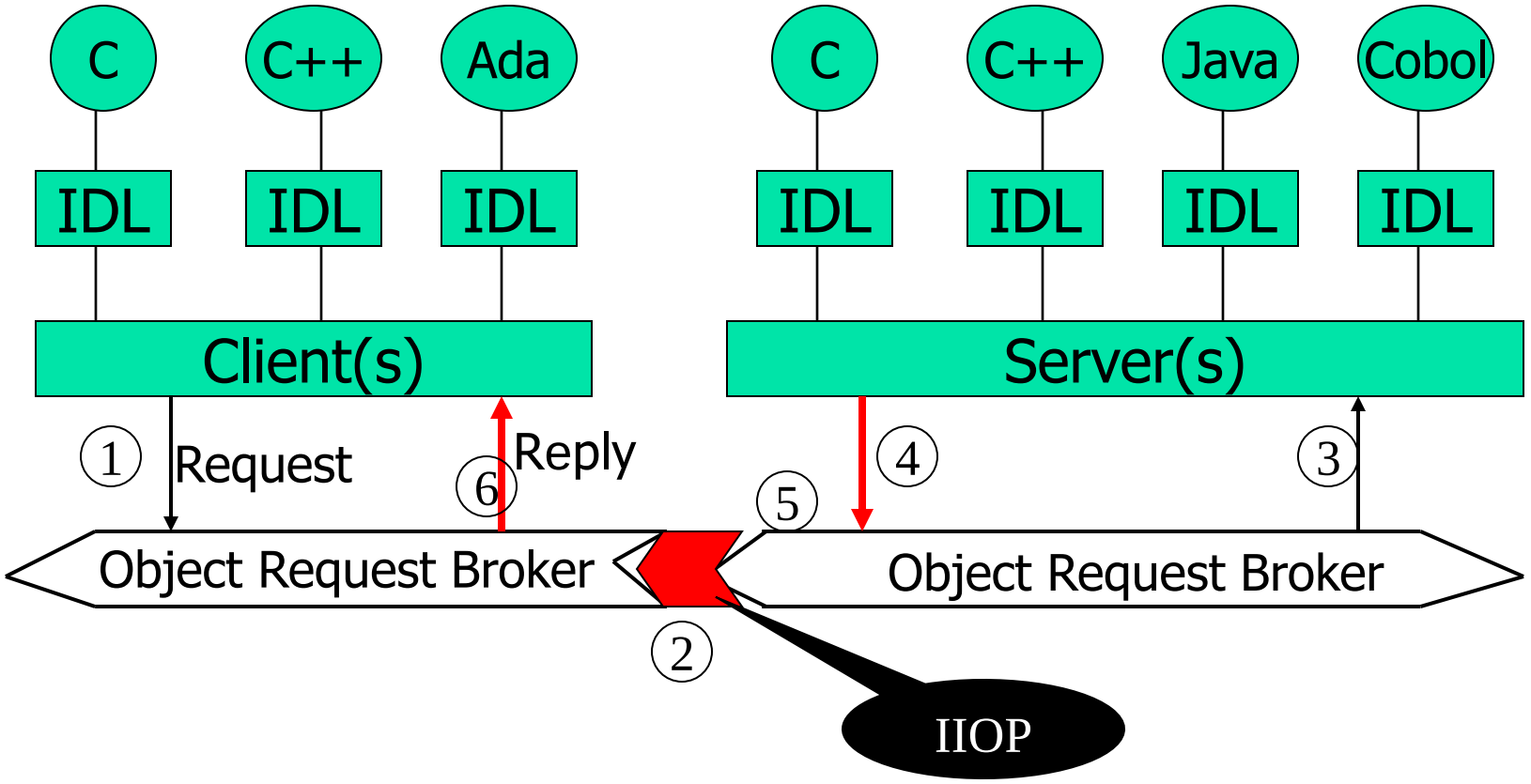


Standard/Reference Architectures

- CORBA : Common Object Broker Architecture
- Object Management Group (http://www.omg.org)
  - Standards for the integration of distributed applications (Specifications: NO code)
  - CORBA Distributed Object: {Binary Components remotely accessibles by the clients}
  - Server Objects "known" by their interfaces
  - Interface Definition Language
    - "Neutral" Language
    - Bindings: C, C++, Java, Smalltalk, Cobol, …

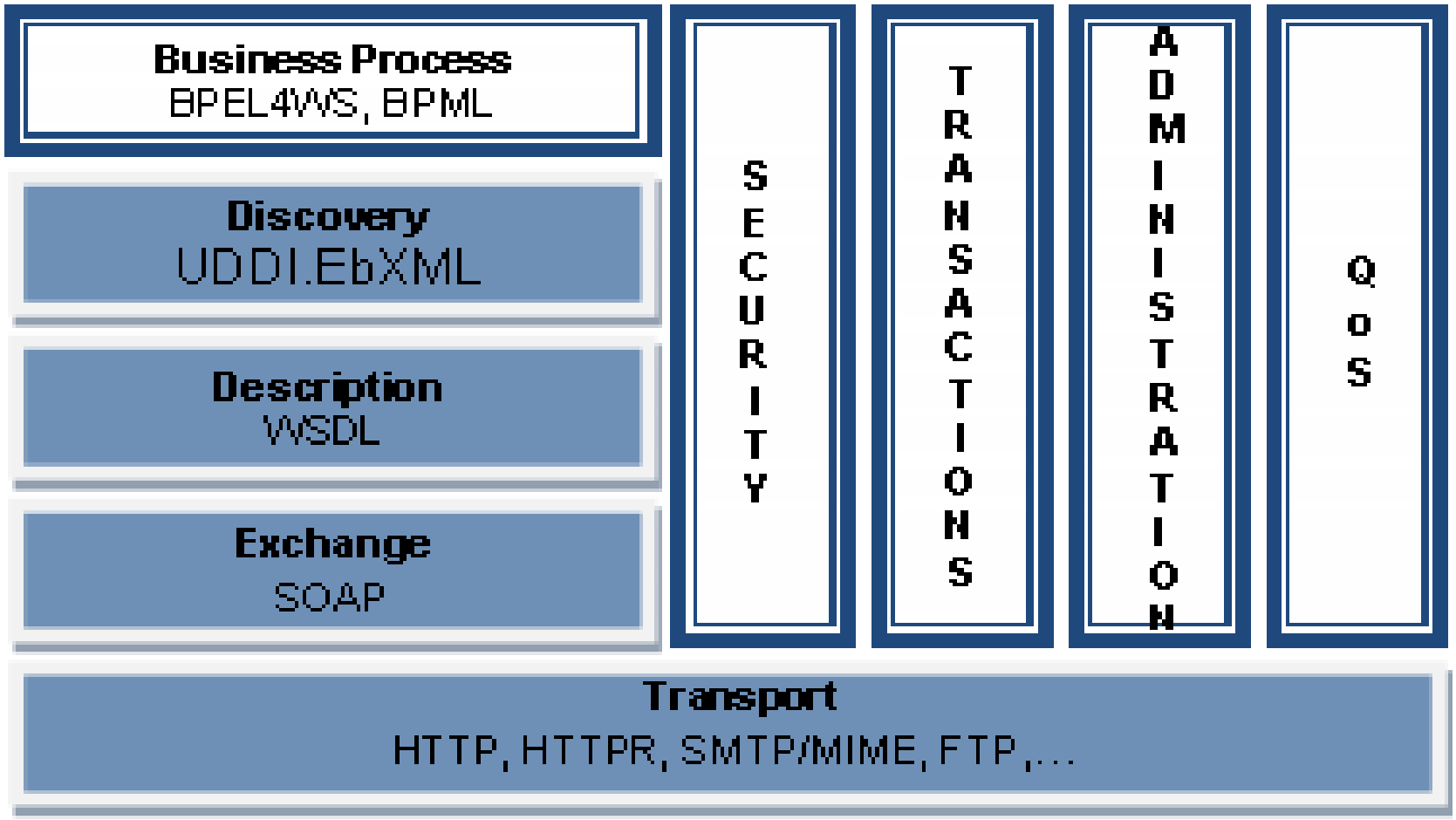
Standard/Reference Architectures

- *CORBA*



Standard/Reference Architectures

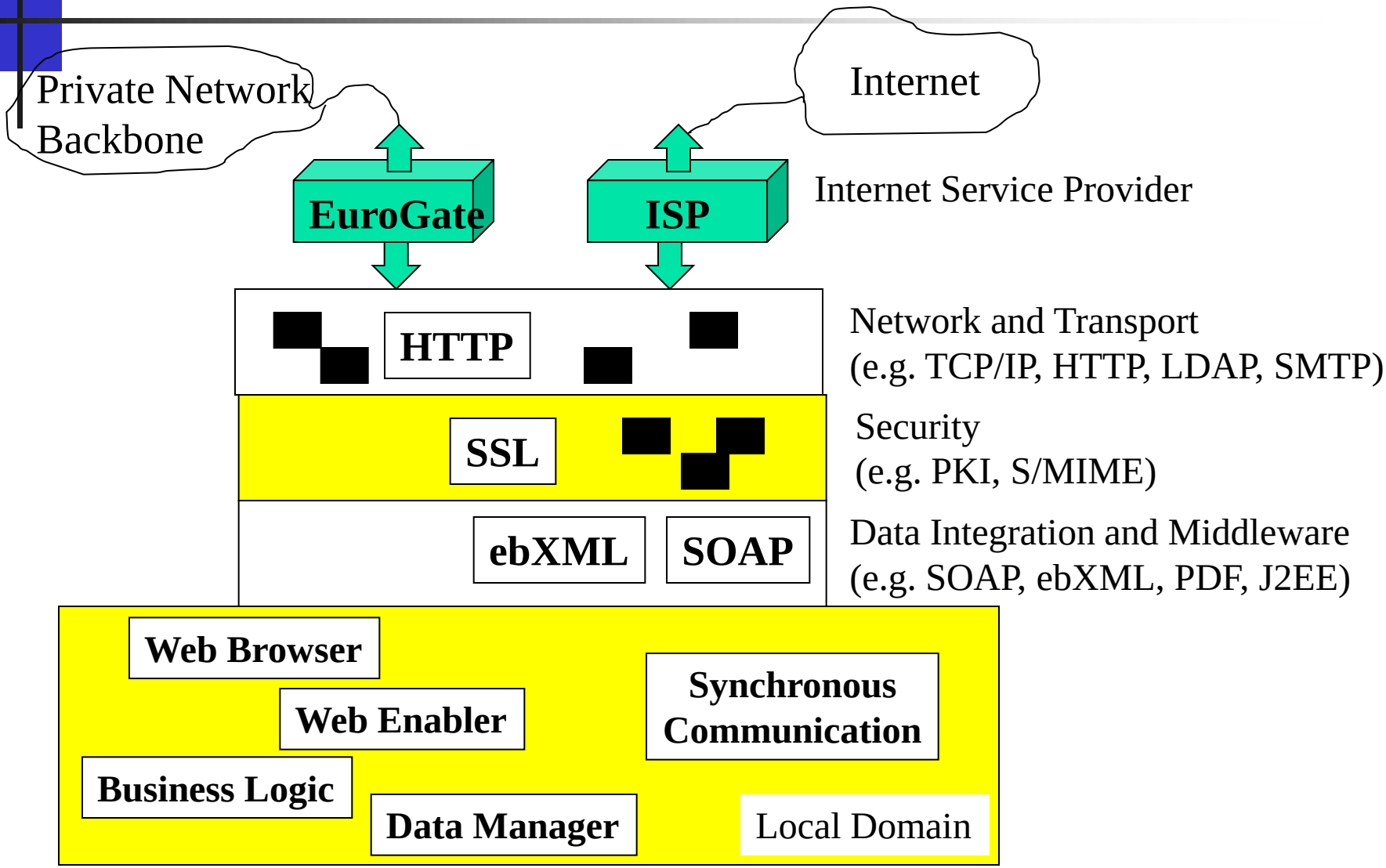
- Service Oriented Architecture



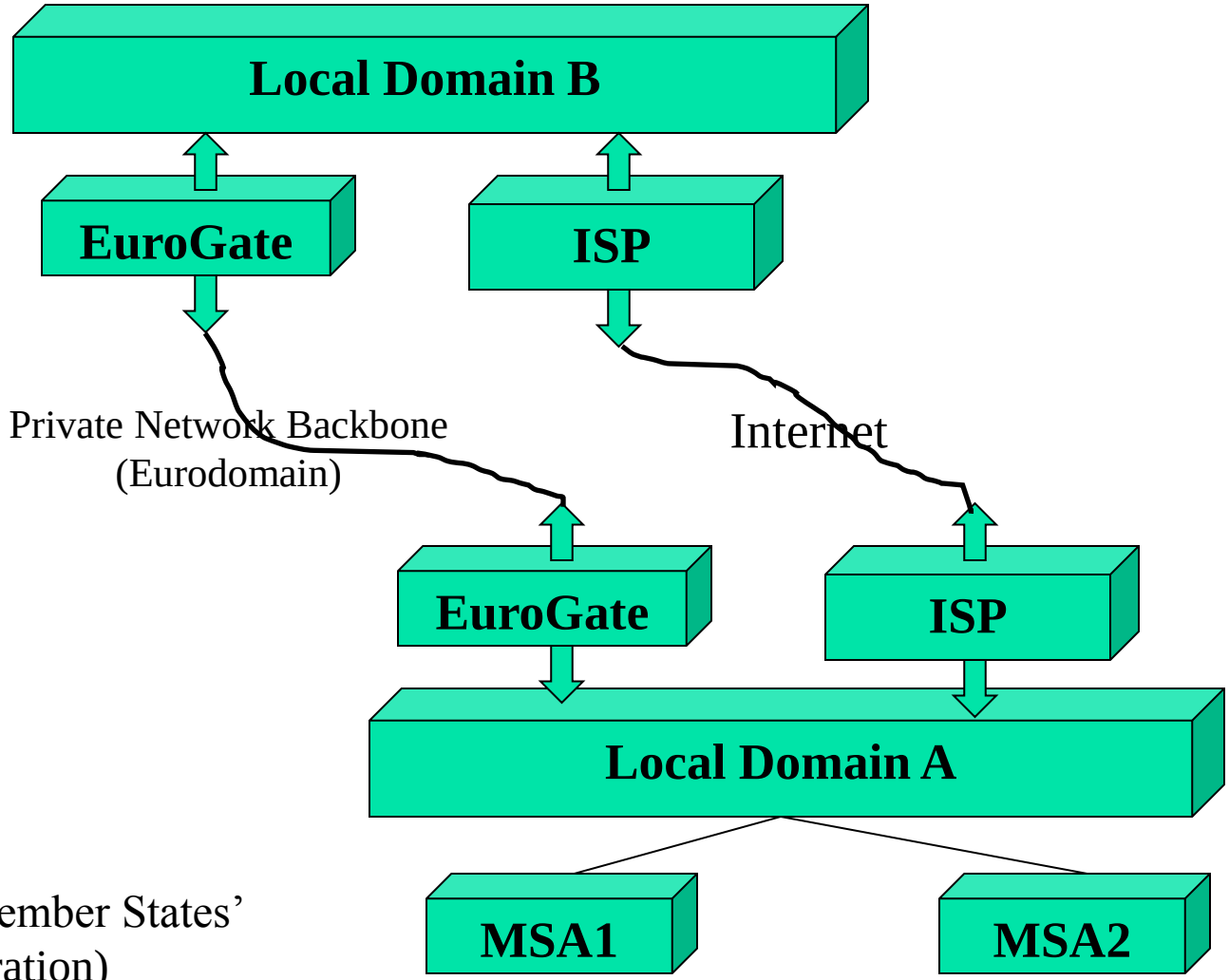
Standard/Reference Architectures

- LARGE SCALE INITIATIVES: European Interoperability Framework for European Public Services (EIF) and Interchange of Data between Administrations (IDA) guidelines:
  1. To promote and support the delivery of European Public Services by fostering cros-border and cross-sectoral interoperability
  2. To guide public adminsitrations' efforts in providing European Public Services to businesses and citizens
  3. To complement and tie together the various National Interoperability Framework in a European dimension

IDA Architecture Basic Components



IDA Reference Networking Model



(MSA: Member States' Administration)

IDA Reference Networking Model

- EuroDomain:
  1. Common internetworking platform in terms of requirements, specifications and functionality for trans-European information services
  2. Set of interface definitions (protocols, formats, APIs) for the backbone infrastructure of trans-European telematics applications
  3. Services that implement these features
- Note:
  - EIF: Open Source recommandations and software vendors lobbying

Integration and Interoperability : Concluding Remarks

- Exchange, Communication between heterogeneous systems
- Middlewares, Formats and protocols
  - common, ad hoc, standards
- Do not prevent from mediation
- Drawback: Lack of semantics
  - For operations/services
  - For objects (content)

Integration and Interoperability : Concluding Remarks

- Benefits from adding semantics:

  'understand' (and consistently interpret) not only the structure but also the content

- Currently (mainly over the Web):
  - Semantic interoperability
  - Thanks to Ontologies

Presentation Outline

I. Integration and Interoperability
   1. Problem
   2. Approaches (« process » vs « objets »)
   3. Architectures

Questions?

Presentation Outline

I. Integration and Interoperability
   1. Problem
   2. Approaches (« process » vs « objets »)

II. Ontology and Semantic Annotations

III. Enterprise Modeling and Semantic Interoperability

Semantic Interoperability

- An approach: Use of ontologies
- Ontology = branch of philosophy about «essence of things» that exist in world, in a domain (17th century!)

- Definition :
  1. [Gruber] : Formal and explicit Specification *of a shared conceptualization*
  2. [W3C] : defines the terms used to describe and represent a domain of knowledge

Semantic Interoperability

- *Purpose of an ontology*: to be used by persons, applications, information bases, … that need to share information about a domain

- Examples:
  - Biology, Genomics
  - Automotive industry
  - Finances
  - …

Semantic Interoperability

- Ontology Anatomy : Types of components

1. Concepts or classes:
   - Represent a set of entities within a domain
   - Organized as taxonomy(ies)
2. Relations:
   - Represent inter-actions among concepts
   - May be organized as a taxonomy

Ontology Anatomy : Types of components

3. Axioms:
   - 'Sentences/Assertions' always true
   - Constrain class values
   - Define arguments of the relations

4. Instances:
   - Representation of individuals of the classes

Semantic Interoperability

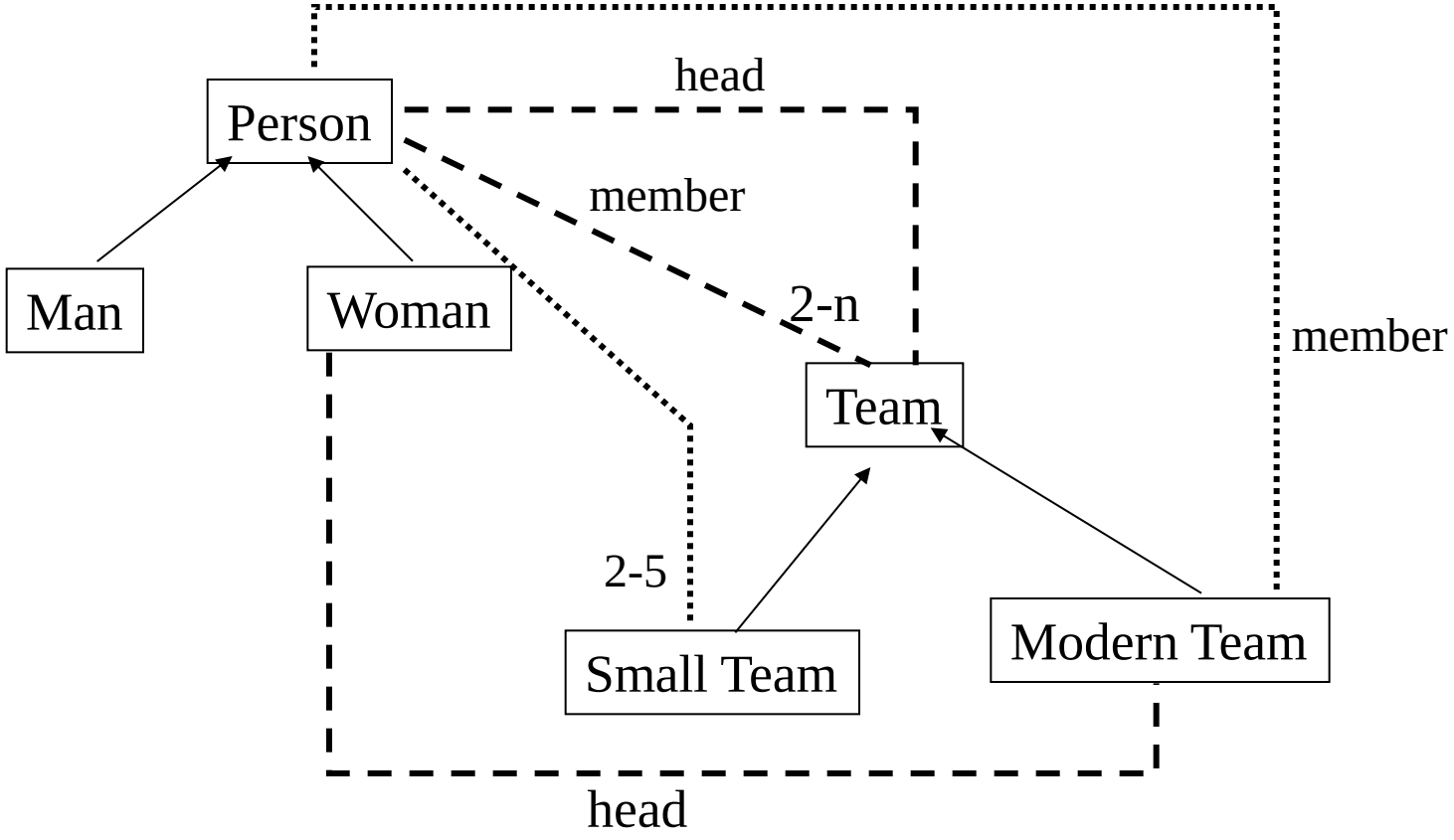
Levels	Concepts	Examples
Higher	The most general	▪ Actor, process ▪ Event, Goal ▪ Business Rule
Domain/ Application	Specific to an application domain	▪ Employee ▪ Bill ▪ Order
Lower	Specific attributes and concepts used for composing the concepts in the domain level	▪ Date, duration ▪ Name, Address ▪ Amount, Cost ▪ …

Ontologies : Languages

- Typology of Ontology description languages
  - *Unformal* : natural language (cf. glossaries)
  - *Unformal structured* : « restricted » natural language
  - *Semi-formal* : Artificial language usually without theoretic foundations
  - *Formal* : terms defined with
    - Formal Semantic
    - Theorems, Proofs
    - Properties (completeness, consistency, …)

Ontology: Example

- Structuration into Research Teams (A. Napoli)



Ontology: Example

- Person $\subseteq$ TOP
- Set $\subseteq$ TOP
- Woman $\subseteq$ *and* Person (*not* Man)
- member $\subseteq$ toprole
- head $\subseteq$ member
- Team $=_{def}$ (*and* set (*all* member Person
  atleast 2 member))
- Small team$=_{def}$ (*and* Equipe (*atmost* 5 member))
- Modern team$=_{def}$ (*and* Team (*all* head Woman))

Ontologies : Expected Benefits

- *Idea* : Improve the communication between humans and machines for:

1) Assisting the communication between persons
2) Interoperability between systems
3) Improving the process and/or the system engineering quality

Ontology and Interoperability

1) Systems Interoperability: thanks to 'translators' between
   - Different Modeling Methods
   - Different paradigms
   - Different languages and Sofwtare Tools

- In this situation: Ontology ≈ Common Exchange Format

2) Improve the process and/or the system engineering quality

1) Re-use:

- Ontology = basis for a formal encoding of important « things » (entities, attributes, processes, relations) within a domain
- Ontology = Formal Representation that might
  - Be re-used
  - Become a shared component

in Software systems

Ontology and System Engineering

2. Information Retrieval:
   - Ontology ≈ meta-data for
     - Indexing
     - Accessing

   an information repository
3. Safety/Reliability:
   - Base the verification of constraints on the formal specification of the ontology
4. etc.

Ontology: Conclusion

- Approach for Semantic Interoperability
- Mainly 'around the Web'
- Similarities to the structural integration and interoperability
  - Integrating, Merging, mapping ontologies
  - Heterogeneous Ontologies, …
- Additional feature: Semantic Annotations

Semantic Annotations

- ***Annotate***: to add a brief explanation or opinion to a text or drawing (*Cambridge advanced learner's dictionary, http://dictionary.cambridge.org/)*

- ***Annotation***: A comment attached to a particular section of a document. Many computer applications enable you to enter annotations on text documents, spreadsheets, presentations, and other objects. This is a particularly effective way to use computers in a workgroup environment to edit and review work… (*Webopedia, http://www.webopedia.com*).

Annotations for Semantic Enrichment

- Purpose of annotations:
  - Describe the content of "something" (annotated object)
- Annotations *ako* meta-data
- Used for a while for
  - Text books,
  - Articles,
  - Hypertext documents
  - Communities (e.g. biologists)

Forms of Annotations?

- Links
- Paths
- Notes in and around a text
- Comments
- Highlights of important terms and definitions
- Numbered steps in a process
- etc.

Annotation Usage

- *Example*: Annotation of Web resources and services for
  - Retrieval,
  - Discovery,
  - Composition,
  - Reasoning,
  - etc.

Annotation Typology

1. *Textual annotations*: added notes and comments to the annotated "object"
2. *Link annotations:*
   - extend the textual annotation notion
   - annotation content is reachable through a provided link
3. *Semantic annotations*:
   - 1 & 2 primarily intended toward humans
   - 3: intended to be human readable and machine readable and processable.

Annotation Language(s)

- *Informal vs formal*

- Annotation expressions may range
  - From a given structural standard (like RDF and RDF Schema)
  - To sound and well-founded language(s) (like First Order Logic, Description Logic, etc.).

Annotation Language(s) (cont'd)

- Formality assumes
  - *no implicit assumptions nor ambiguity*
  - *common understanding* of the provided annotations

- Common understanding thanks to one or several ontologies agreed by
  1. The authors of the annotation (*annotation providers*)
  2. The ones who exploit the annotations (*annotation consumers*)

  $\Rightarrow$ Annotation content refers to concepts in the ontology(ies).

Semantics of Annotations

- A*nnotation consumers* need to understand the meaning of *the variety of types of annotations* (semantics of the annotations)

"… meaningless data cannot acquire meaning by being tagged with meaningless metadata…" [Sowa2000]

Semantics (Types) of Annotations (examples)

- Example [Bechhofer2002]

1. *Decoration*: annotations = comments associated with the resource

2. *Linking*: annotations = links

3. *Instance Identification*: the annotated object is an instance of a given class and the annotation content may be a link to that class;

Semantics (Types) of Annotations (examples)

4. *Aboutness*: no assertion is made about the existence of an instance of the concept C, but there is a loose association with the concept;

5. *Pertinence*: the target of the annotation may be of interest to the annotated object.

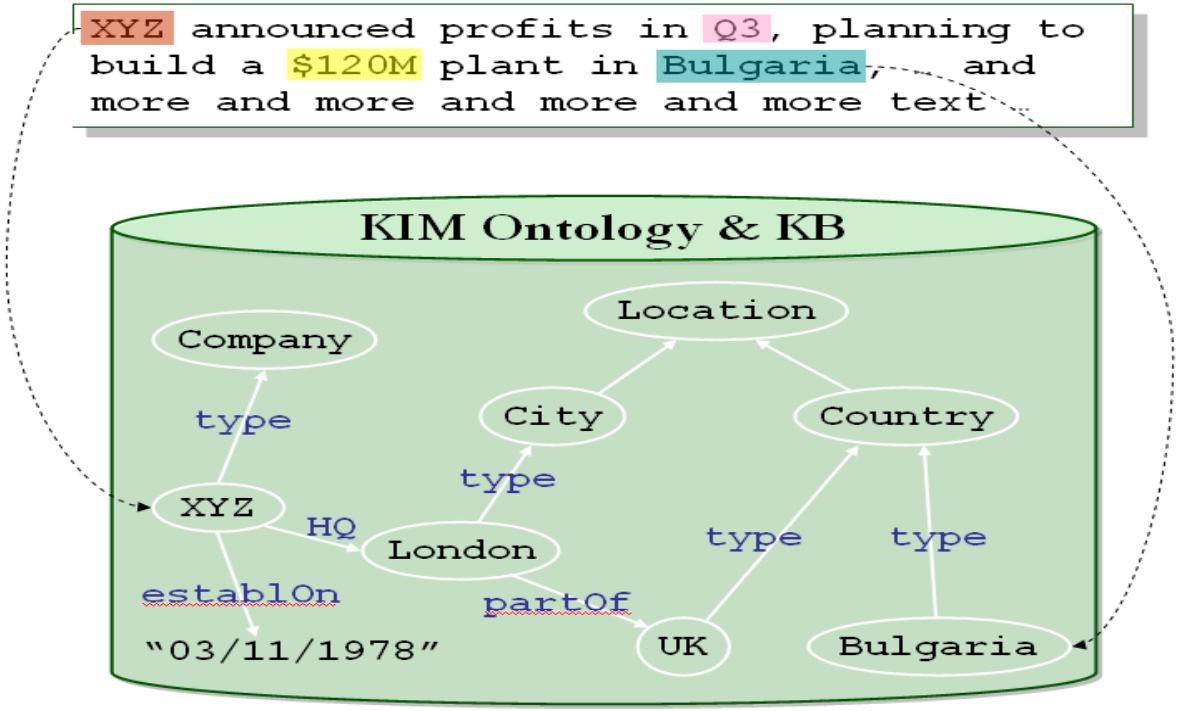
- Types of annotations to classify query results

Annotation Services

- Progressive move *from manual to automatic or semi-automatic* annotation provision
- Services to provide ("write") annotations?
- Content of some types of annotations relies on given ontologies $\Rightarrow$ ontology services (like querying or browsing an ontology) have to be coupled with annotation services

Knowledge and Information Management (KIM)

- Infrastructure and services for automatic semantic annotation, indexing, and retrieval of unstructured and semi-structured content



Semantic Annotations (end)

- Semantic annotation
  - Mainly for Web Documents and Web Services
  - Concepts and Tools available

- To what extent the current advances/approaches might be re-used/adapted for annotating enterprise models (or Entreprise Architectures) for interoperability purposes?

Presentation Outline

I. Integration and Interoperability
   1. Problem
   2. Approaches (« process » vs « objets »)

II. Ontology and Semantic Annotations

III. Enterprise Modeling and Large Scale Semantic Interoperability

What next?

- Enterprise Modelling
- Enterprise Ontologies
- Enterprise Models Annotations and their possible role in Interoperability Solutions

Enterprise Models: Why

- Understand, Analyze, Simulate, …
- Diagnosis (disfunctioning: material, flows, organization, etc.)
- Re-structuring
- Integration/Interoperation
- Evolution
- Conformity to standard (example: Quality)
- …
- Define a common understanding

Enterprise Models: What is it?

- Describe the "things" of an enterprise
  - Functions
  - Behaviour
  - Information
  - Resources
  - Organization
  - …

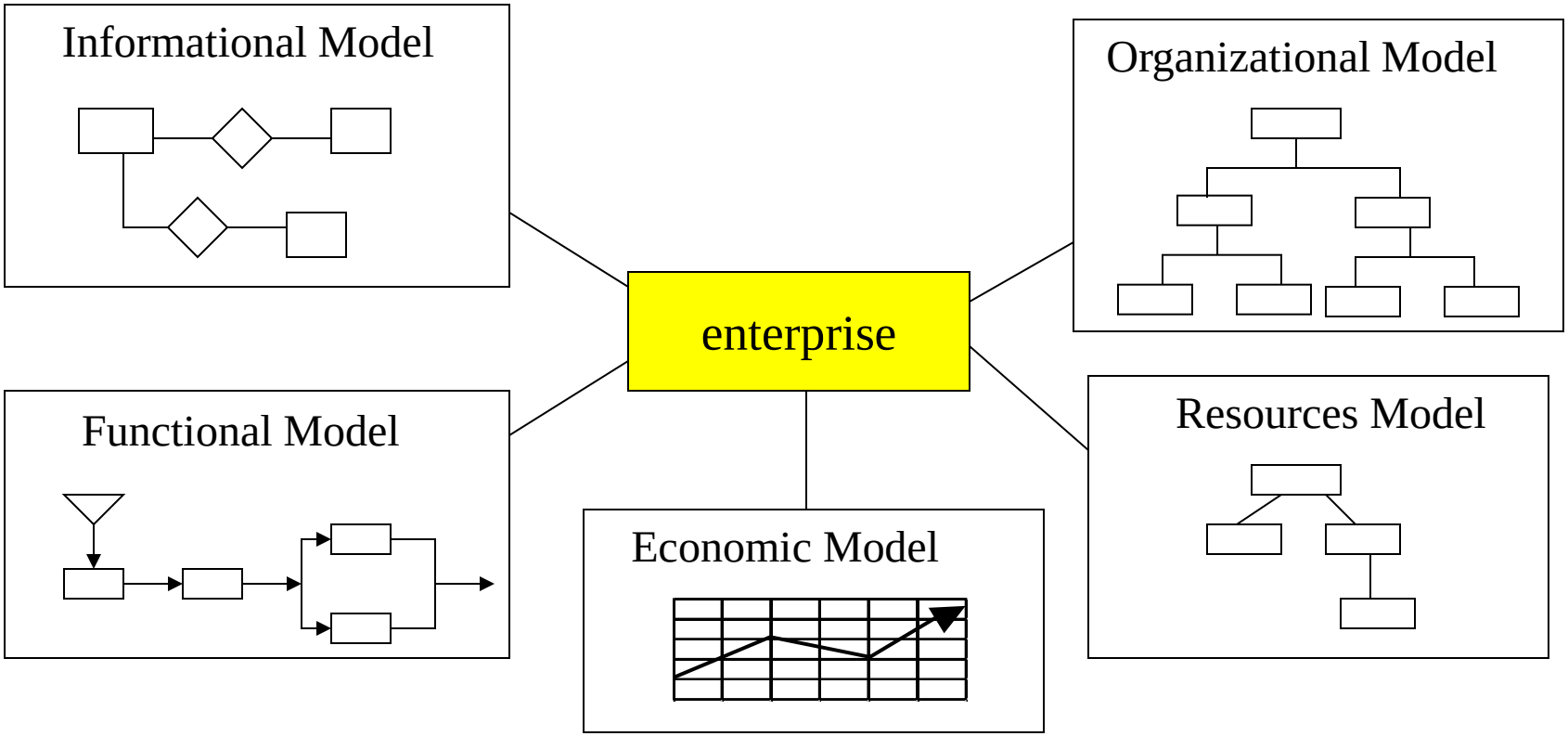
Enterprise Models: What is it?

- Representation (*artefact*)

- Multiple points of views
  - Operational,
  - Decisional,
  - Informational,
  - Strategic,
  - Economic
  - etc.

Enterprise Models

- Model or Models

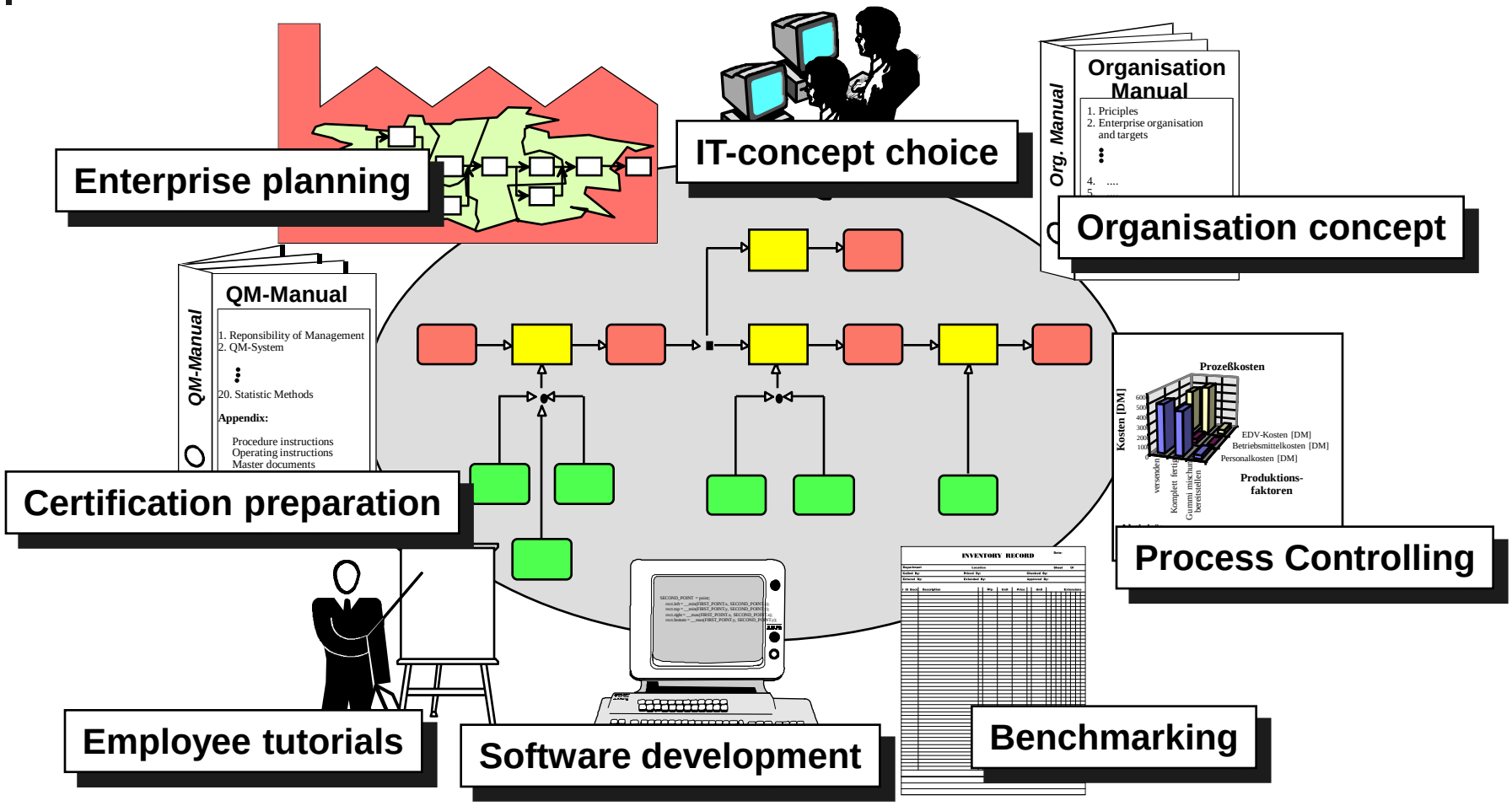


Enterprise Models

- Diversity of vocabularies
  - Person, employee, agent
- Diversity of notations
  - Entity-Relationship
  - Objects (example: UML)
  - Petri net
  - Transition Diagrams
  - Logic(s)
  - etc.
- Diversity of modelling tools and software

The many facets

© IPK, Berlin, 1999

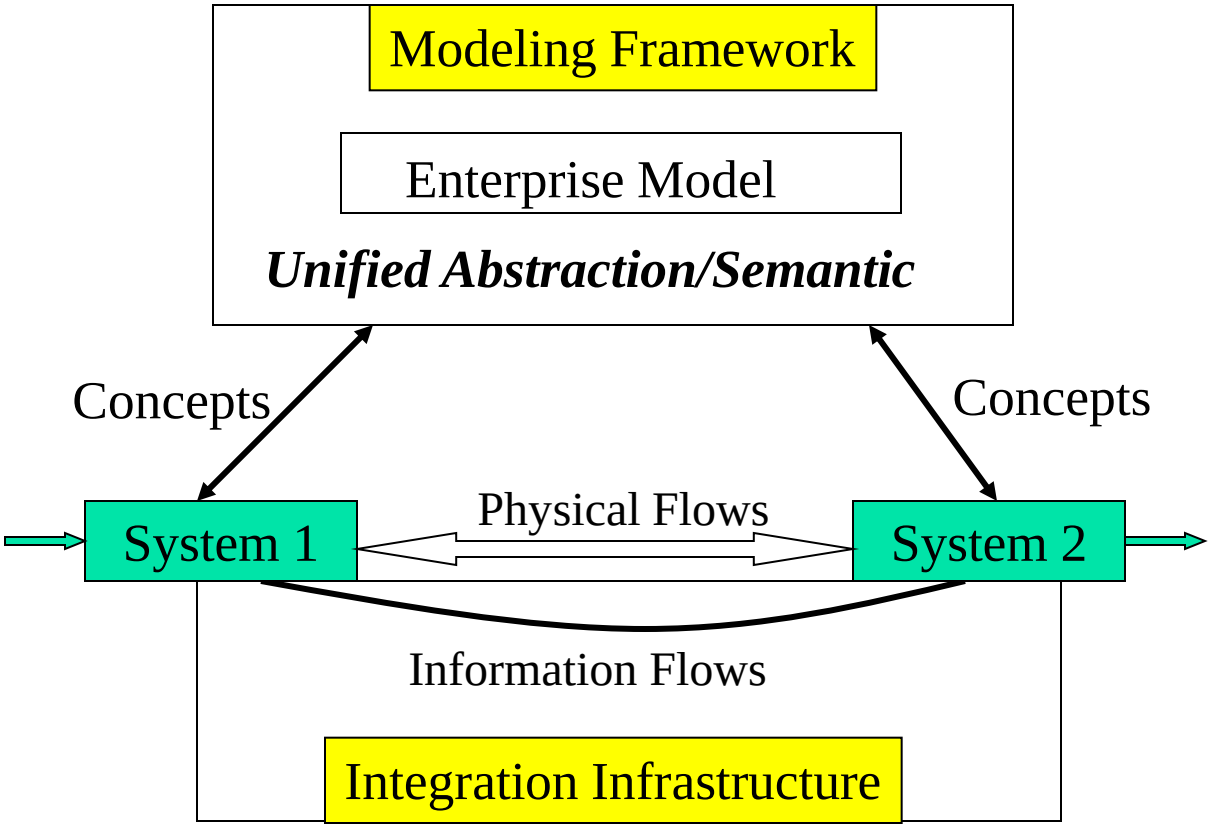


EO: Edinburgh Enterprise ontology

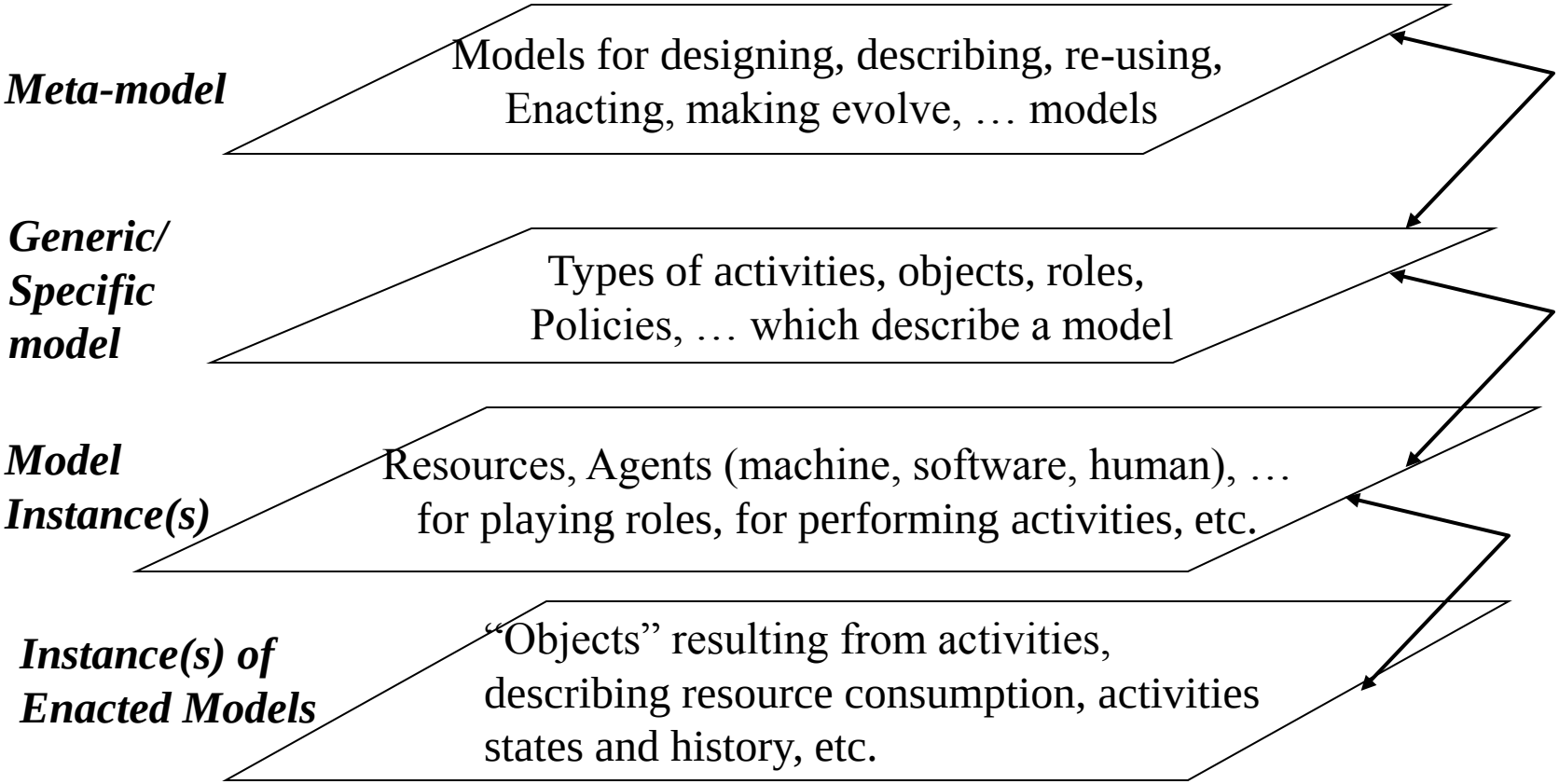
- What an Entreprise Ontology?
- Example: Edinburgh Enterprise ontology
  - 92 Classes
  - 68 Relations
  - 483 axioms
  - 10 individuals
- Fewly or not in use at all!

Enterprise Models

- How?



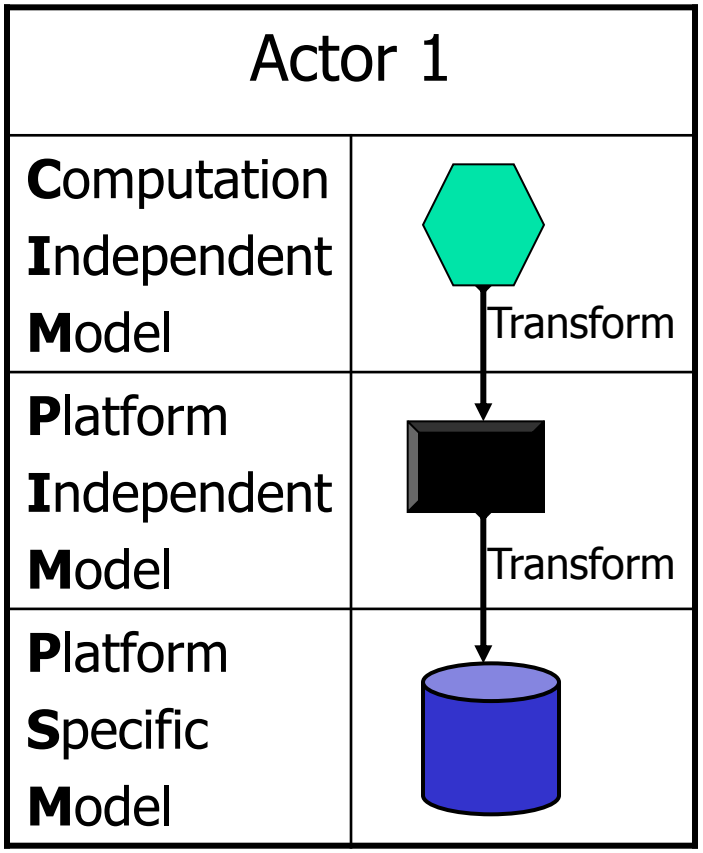
Models, Instances and Annotations

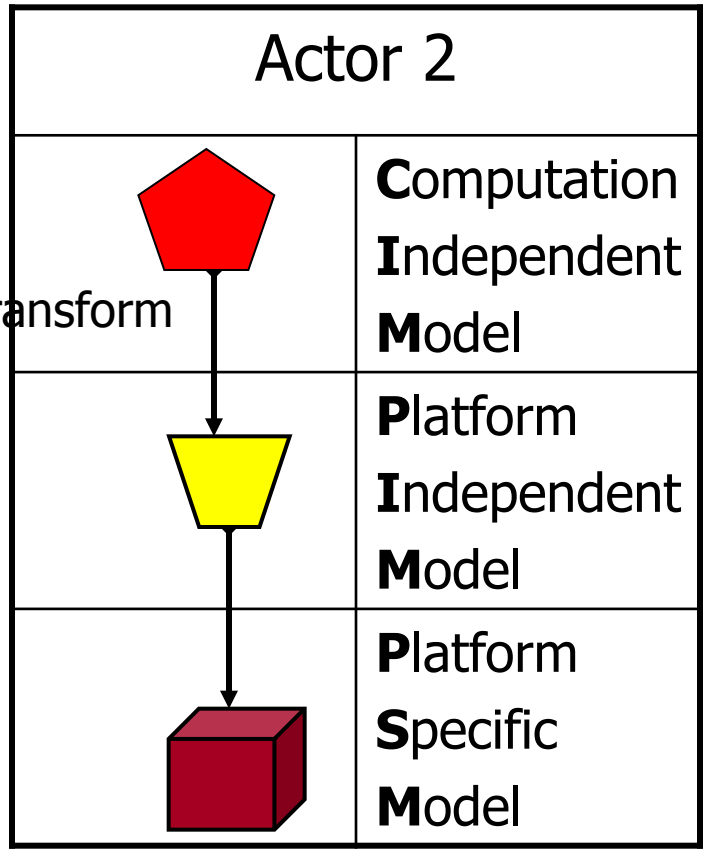


⇒ Intra-levels and inter-levels semantic enrichment

Models, Instances and Annotations

- The Model Driven Architecture (MDA) View





Enterprise Systems and Applications

- "Co-existence" of concepts and "supports":
  - For (meta-)modeling
  - For instanciating
  - For functioning (activation/enactment)
    - Integrated software systems (example: ERP)
    - Proprietary software
    - "House made" Software
    - etc.
- Cooperation and communication Purposes $\Rightarrow$ interoperability
  - Intra-level
  - Inter-levels

Semantic Annotations of Enterprise Models

- Types, forms, … of Semantic Annotations of Enterprise Models?

- Tentative Pragmatic Identification thanks to
  1. Literature (poor concerning enrichment of models)
  2. Case studies:
     - Various Perspectives of Models
  3. Common framework (Enterprise ontology, Enterprise Modeling and Ontology Management tools)

Annotation Scheme

- *How to actually provide annotations?*

- *Annotation Scheme (actual content):*
  - To be agreed on by the producer and the consumer
  - Without concrete syntax (cf. W3C rdf scheme and SAWSDL)
  - About the '*constraints*' attribute in the scheme: indicates matching and mismatching between the annotated 'object' and the corresponding concepts

An Annotation Scheme

<Annotation

Annotation-Id: Annotation identifier

Unformal Content = Intent of the annotation;

Annotation Type = (not limited to): Decoration, Linking, etc.

Ref2Ontology = reference (assumed to be a uri) to the ontology concept(s) that match(es) the concept in the model

Constraints = (might be written in OCL, with references to the ontology or to the meta-model, when these ones are expressed using UML).

/>

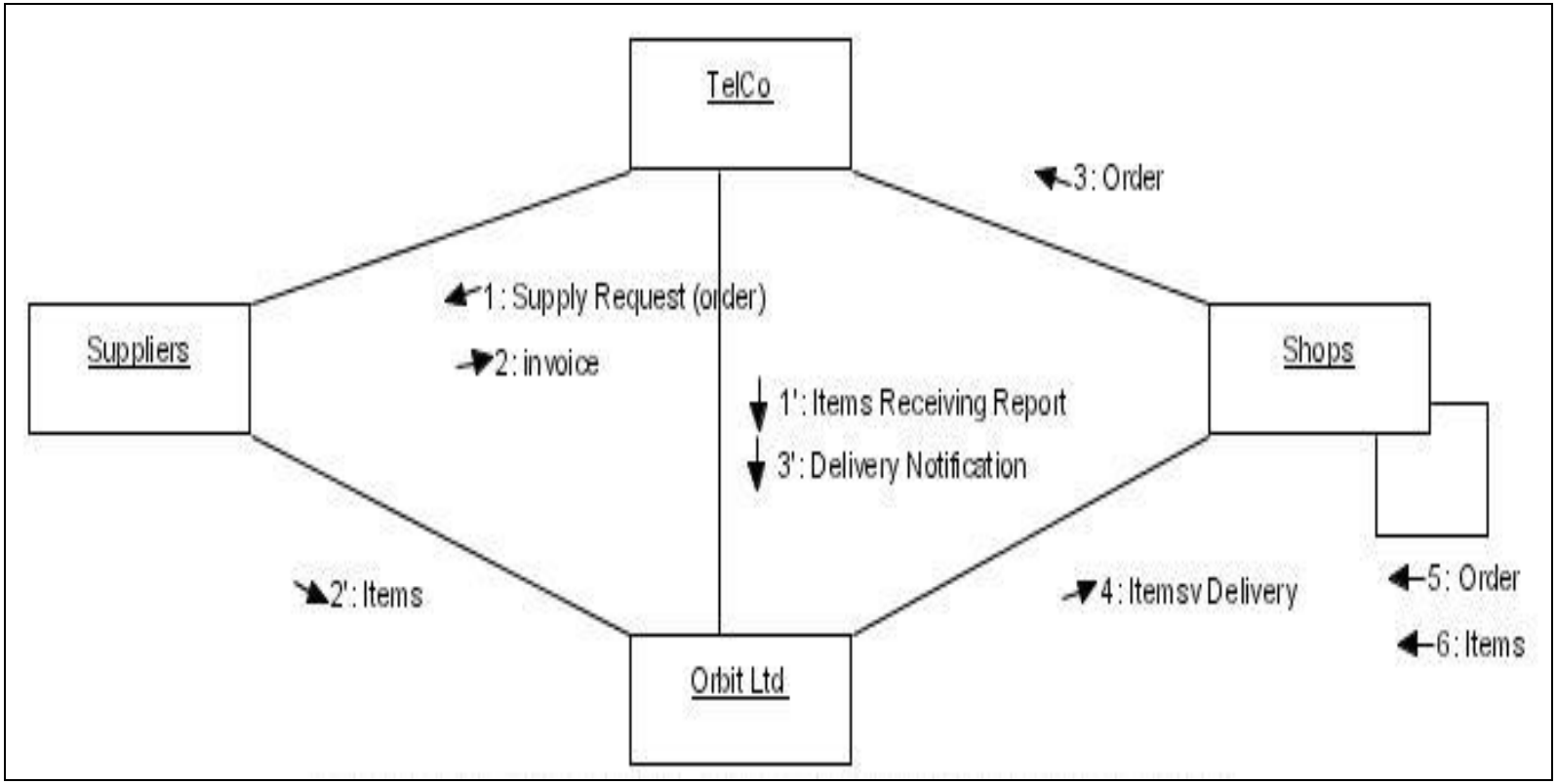
Annotation Scheme: About the Constraints Attribute

1. A concept in the model *exactly matches/is more specific/more general/part of/ instance of /…*a concept in the ontology (*and vice versa*)

2. A concept in a model matches (*exactly, partly*) *not a single concept, but possibly more than one concept and relationships* (and conversely)

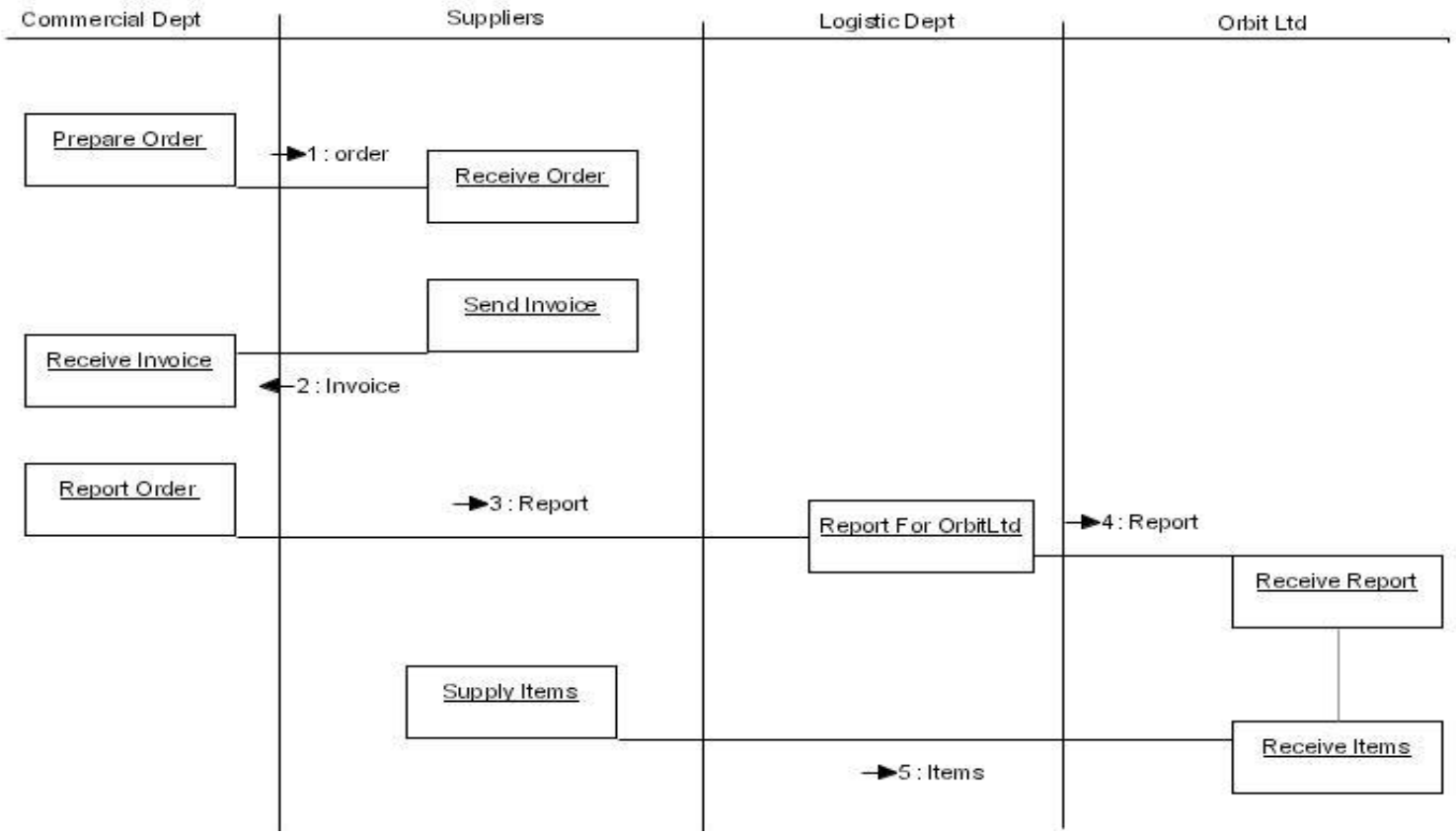
3. *No concept* in the ontology matches a concept in the model

Annotation Content: Example



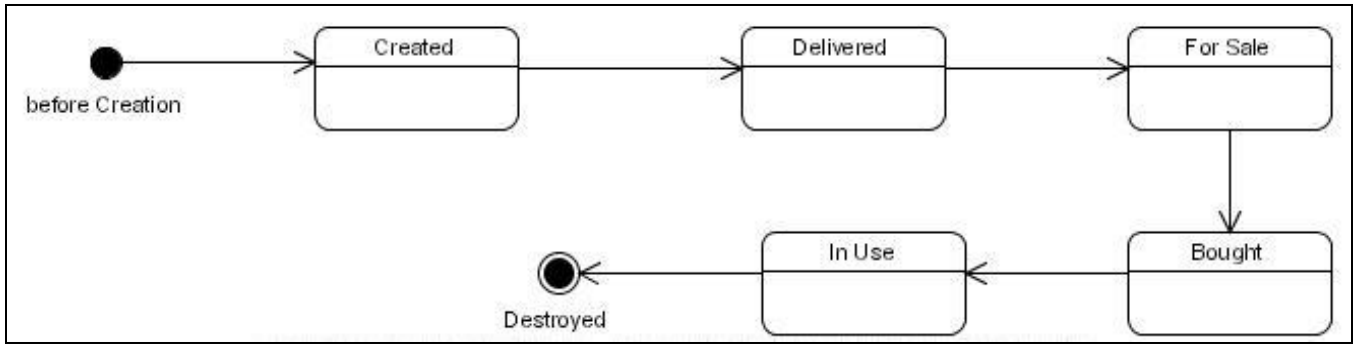
Items Receiving and Delivery Activity Diagram (in the studied enterprise)

Annotation Content: Example



Activity diagram of the process "Ordering & Receiving Items"

Annotation Content: Example



Items Receiving and Delivery Activity Diagram (in an Ontology)

<Annotation

Type = 'Instance identification'

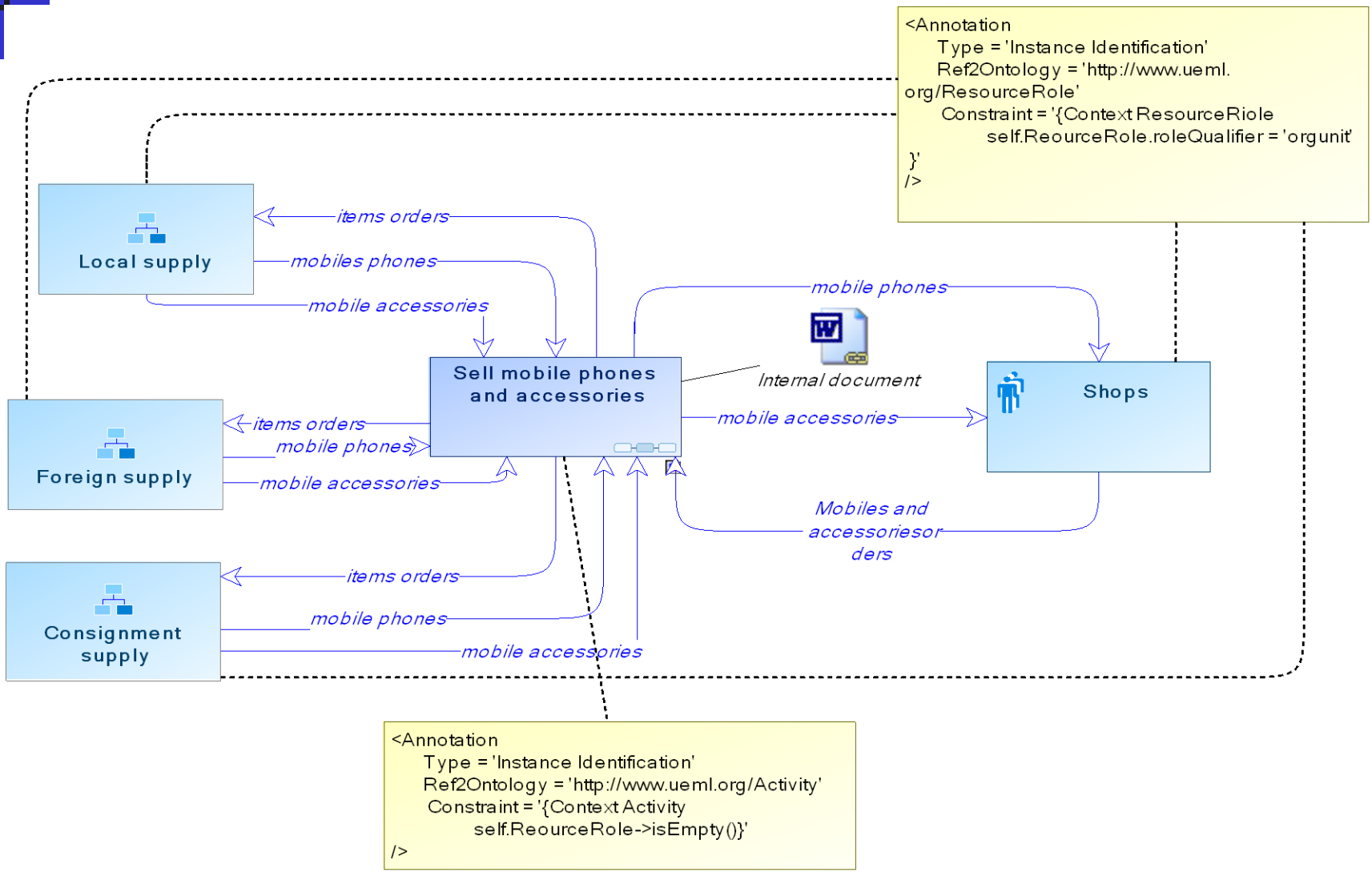
Ref2Ontology = 'http://www.ueml.org/Activity'

Constraint = '{Context Activity

Self. ResourceRole->isEmpty()}'

/>

Annotation Content: Example

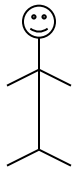
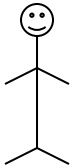


Model Annotation Purposes

- Main Purpose: Model-based interoperability

1. Exchange Models
2. Transform Models
3. Trace models

Annotations for Model Exchange

Meta-Meta	Unifying meta-model (?)	
Meta	Meta-Model MM1 Onto. O1	Meta-Model MM2 Onto. O2
Model	Model M1 • Annotation scheme A1 • Ontology O1 for A1 • MM1 for M1 and A1	Model M2 • Annotation scheme A2 • Ontology O2 for A2 • MM2 for M2 and A2
Instance	 Application 1 (eg. ERP)  Cooperating	 actors Application 2 (eg. CRM)

Annotation Categories

- For a *deep semantics*

Category	Annot° Target	Example
Structural	Meta-Model (specific/generic)	P_01: Mega Process
Lexical/ Terminological	Thesaurus, Taxonomy, Ontology	P_01: SCOR.Plan Supply Activity
Behaviour (Optional)	Business Logic, Rules, etc.	UML use cases, interaction diagrams, Petri nets, …

Example: Annotation for Model Exchange

- From GRAI to MEGA (UEML as a meta-model)

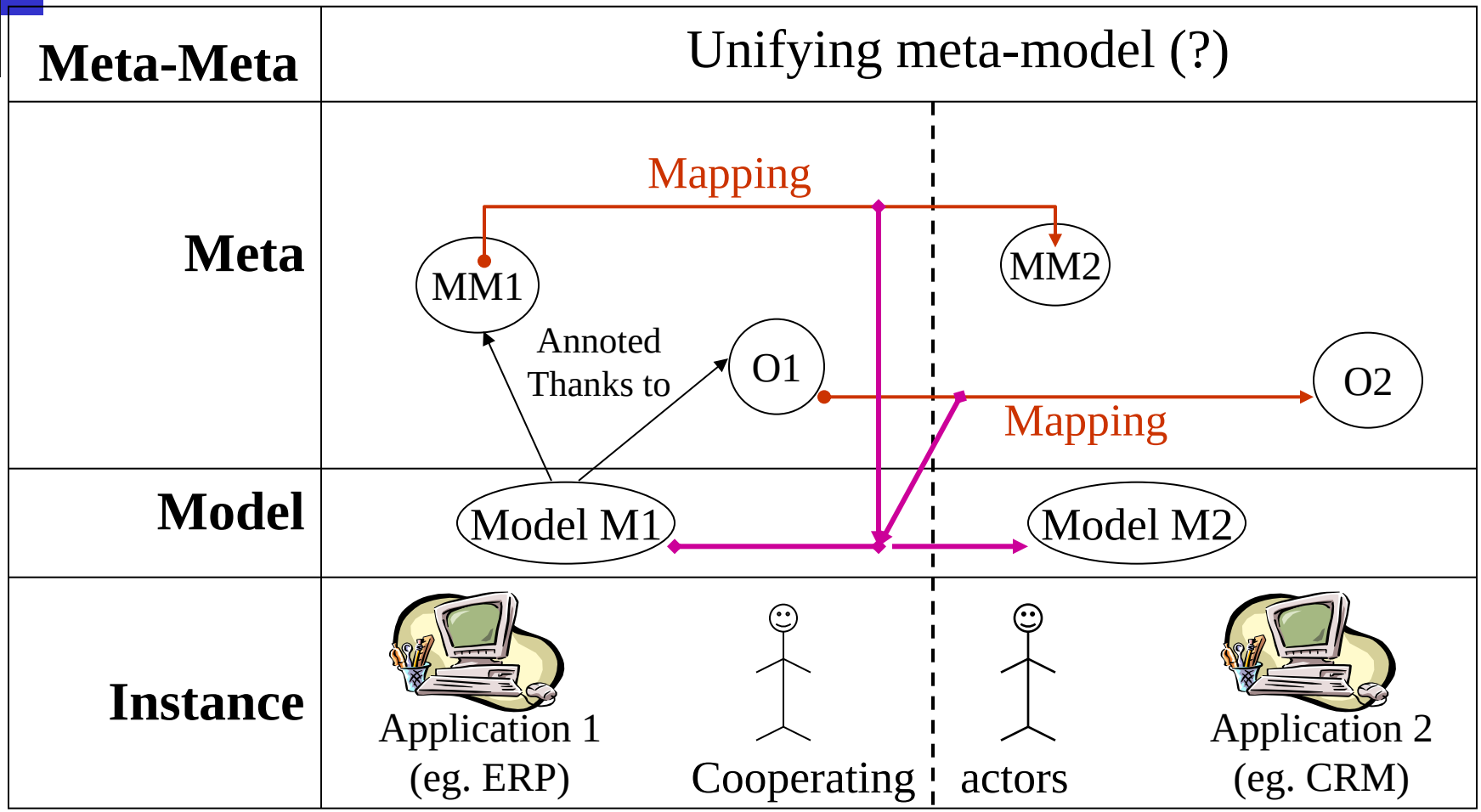
1. Structural Annotation
   - X: GRAI Decision Centre
   - Mappings wrt Meta-Model:
     - Decision Centre $\subset$ UEML.Activity $\subseteq$ Mega Activity
2. Terminological Annotation
   - X: SCOR.Plan Supply Chain (Reference Model)
   - http://www.scor.org/Plan
3. Behavioural Annotation
   - Planing business rules

Example: Annotation for Model Exchange

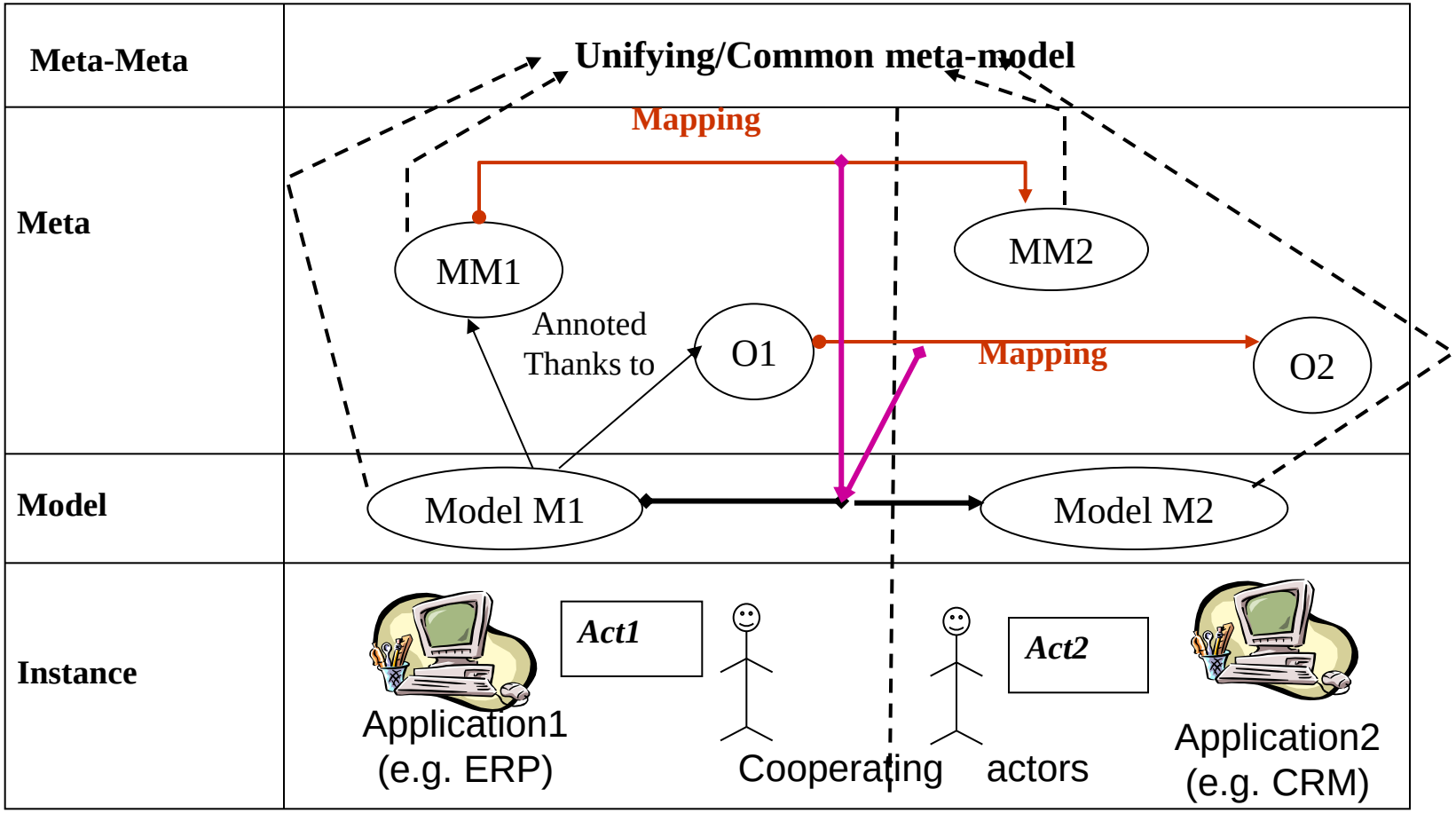
- From GRAI to MEGA (without a meta-model)
- Structural annotation: "Direct Mappings"

GRAI Concepts		**Mega Concepts**
Decision centre	$\subseteq$	Activity
Decision frame	$\subseteq$	Message
Information link	$\subseteq$	Message
Decision level		N/A
….		

Annotations for Model Exchange

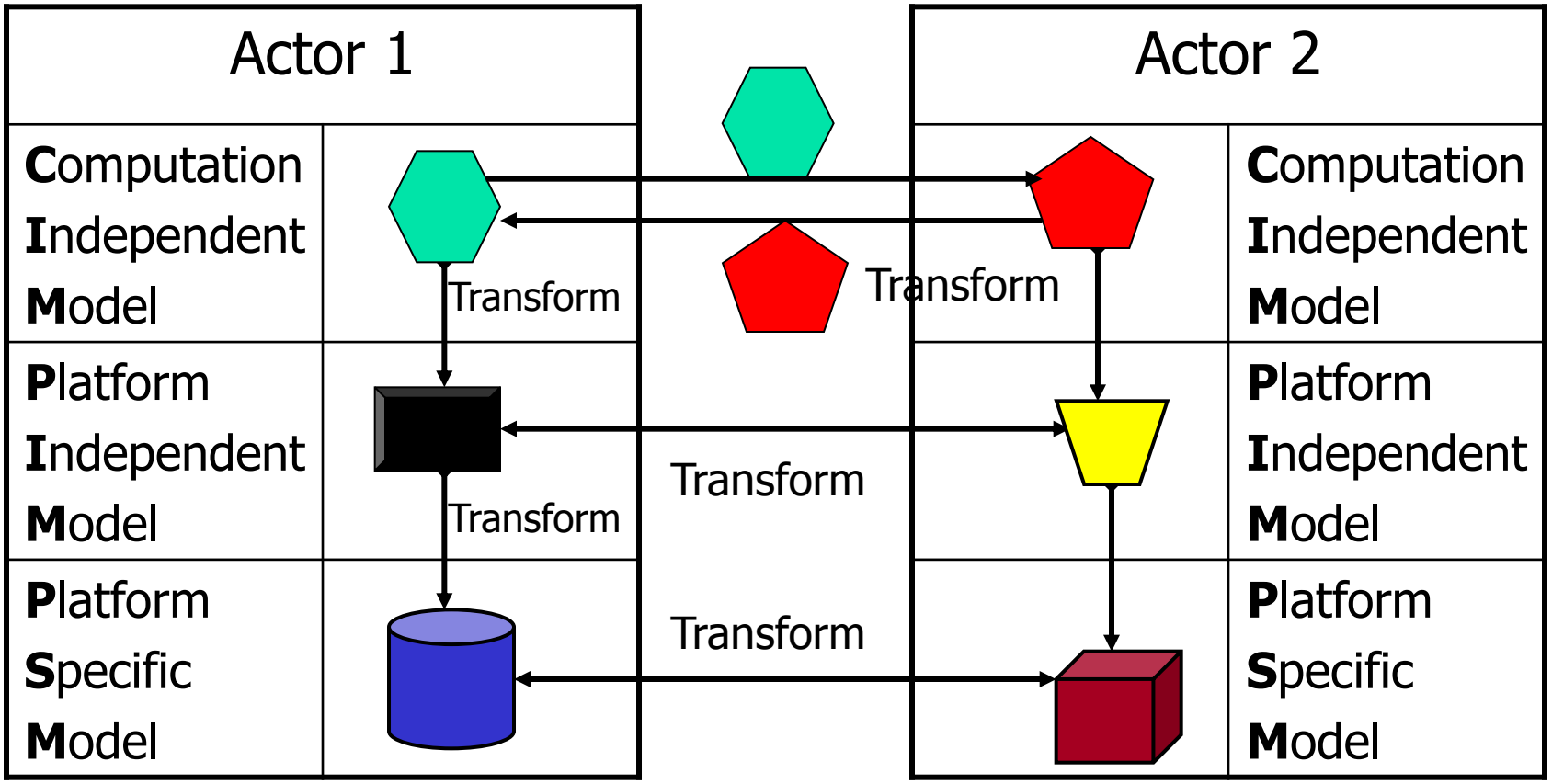


Annotation for Model Exchange



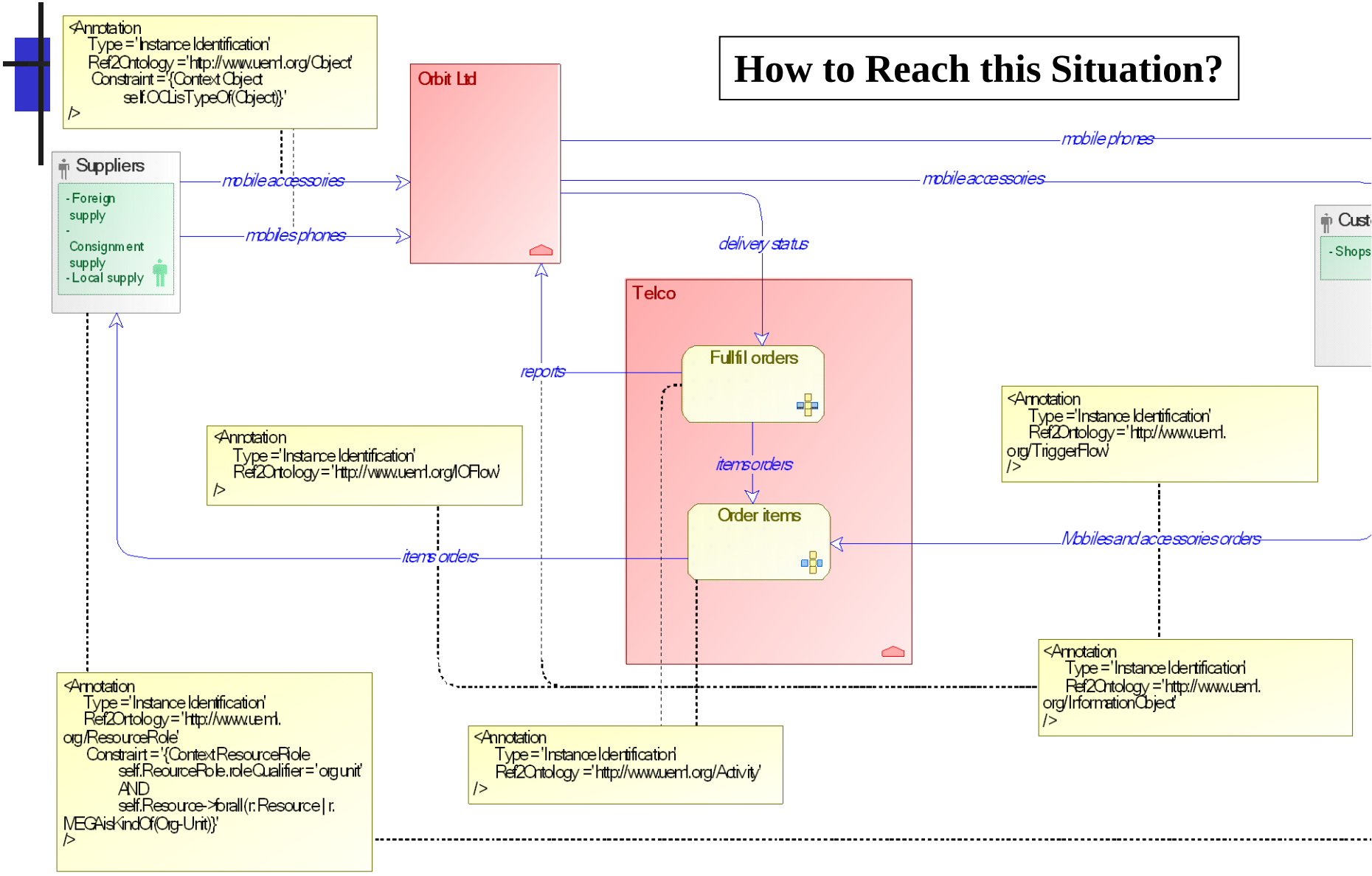
For Model Transformation and Traceability

- The Model Driven Architecture (MDA) View

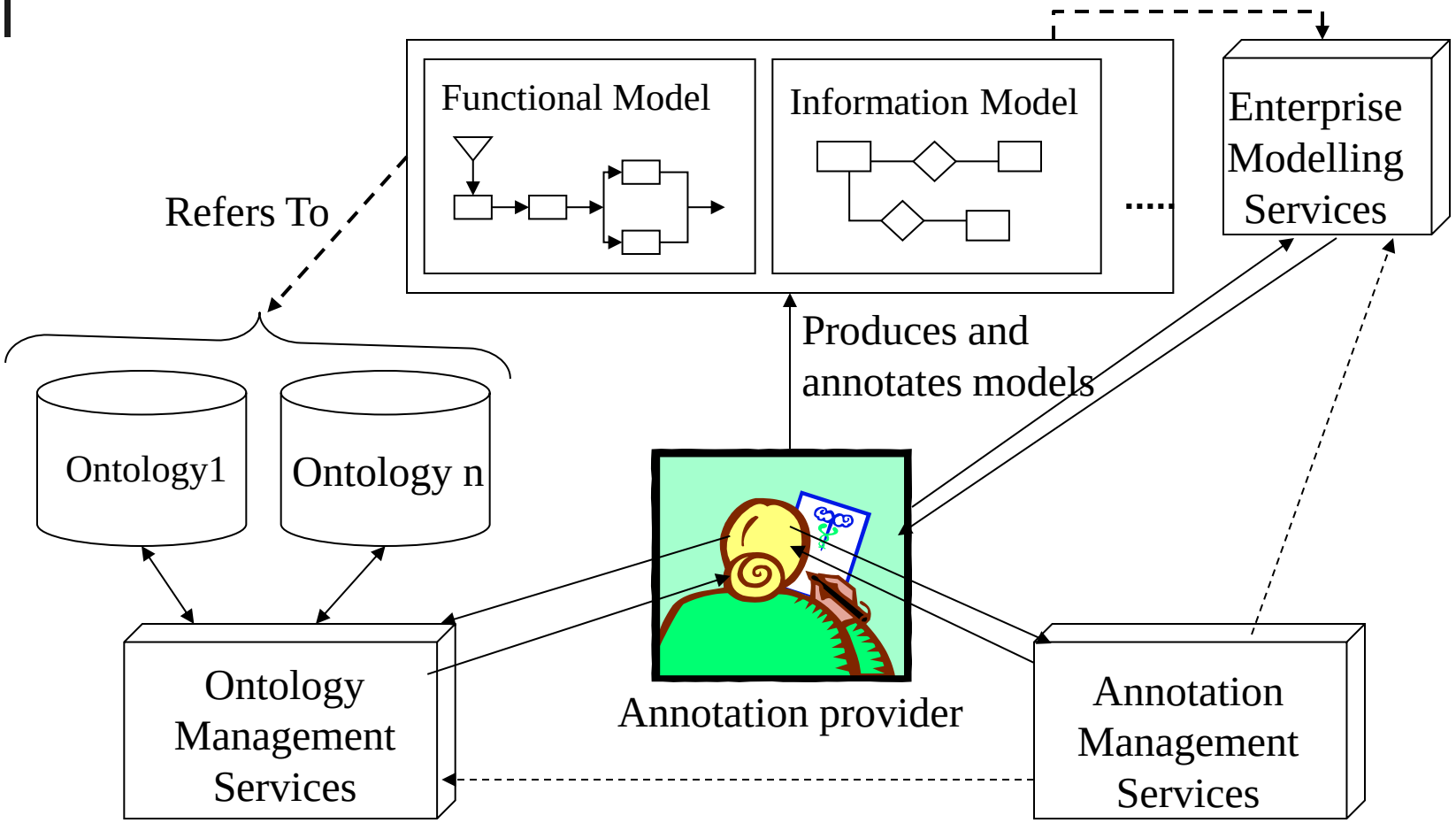


Model Annotation Support

How to Reach this Situation?



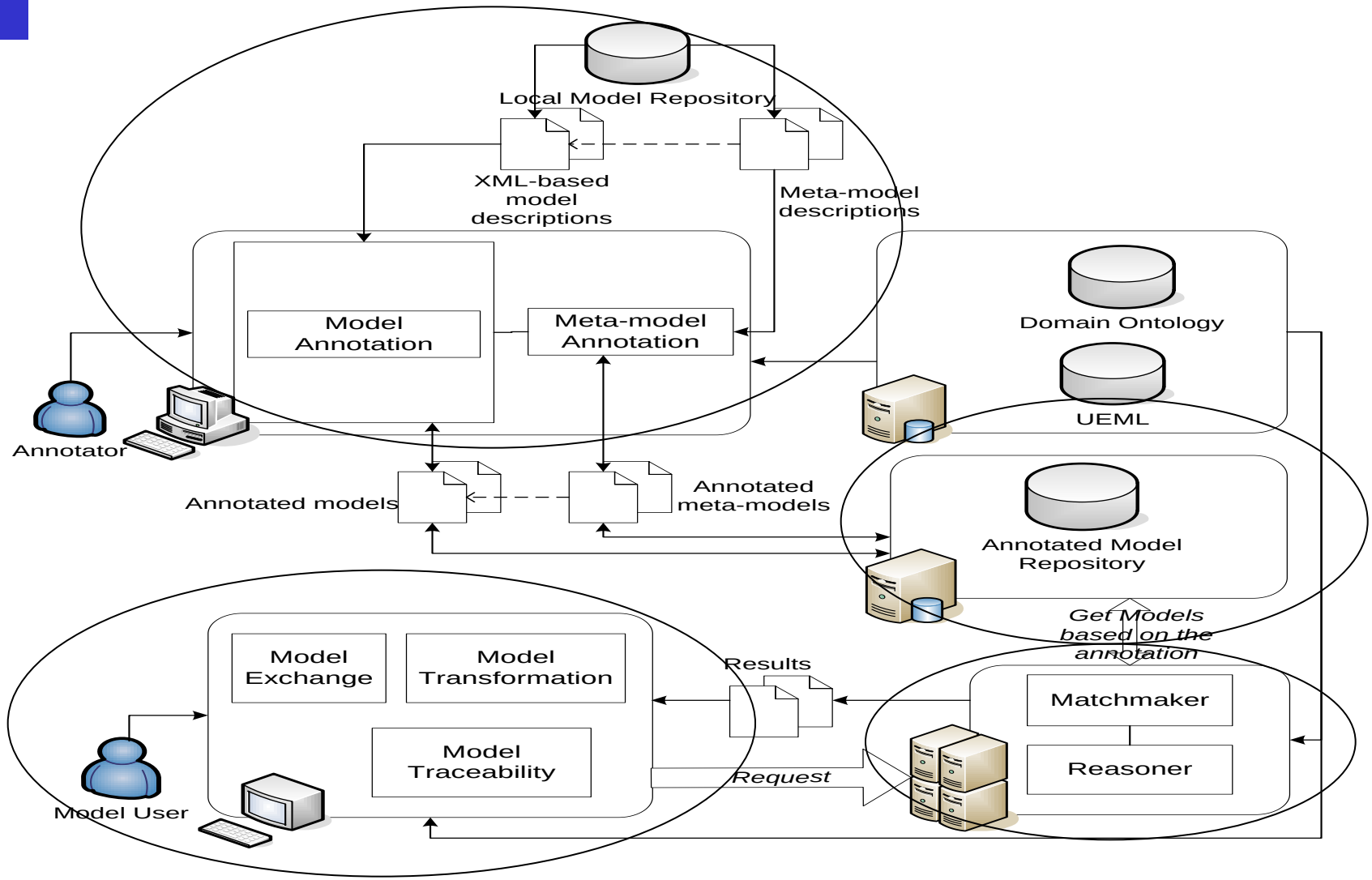
What Types of Annotations Services for EM?



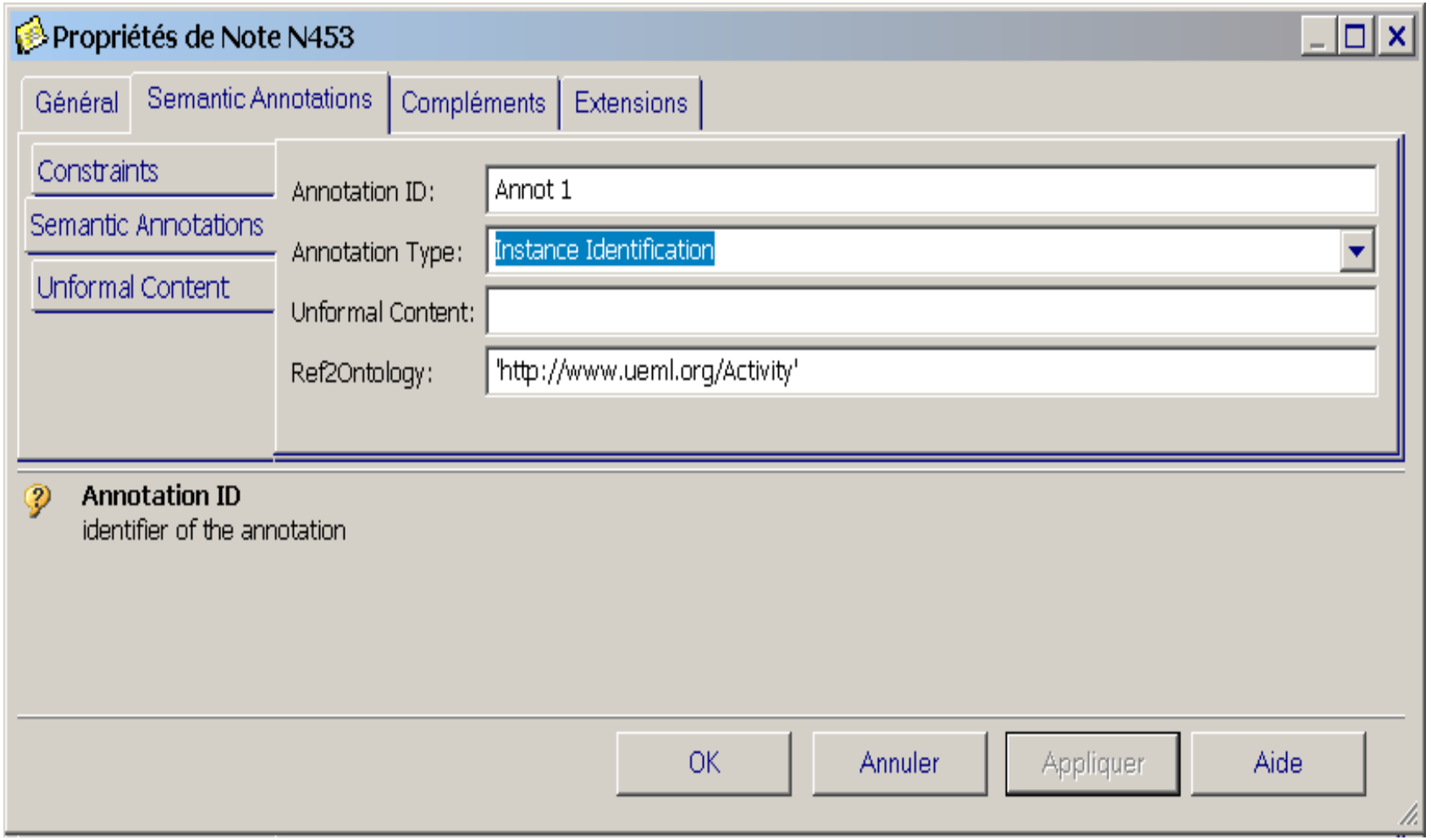
Model Annotation Support

1. Ontology management: define, store, browse, query, ...
2. Model management: model, store, retrieve, browse and "export" models under various notations, etc.
3. Definition of a variety of annotation schemes and content types with regard to a variety of annotation types
4. Actual provision and management of the annotations:
   - *Matching facilities:* relate concepts in a model to concepts in an ontology (*lexical annotations*) or in a meta-model (*structural annotations*).
   - *Annotation and annotated grain identification*
   - *Annotation re-use:* attach a given annotation to multiple grains

Model Annotation Support



Support for EM annotation



Annotations of EM: Problems

- Annotation grain in a model(?):
  - Size? Identification (uri, text anchor, …?)
  - Multiple (types) annotations
- About the target of the annotation content
  - Reference to (multiple) ontology
  - Reference to the Meta-model of the modeling language: (≈ lexical/structural annotation)

Open problems

- Formal and/or technical points of view:
  - Formal properties of a set of annotations?
  - Knowledge inference from annotated models?
  - (Semi-)automatic support for model annotation
    - Model exploration (cf. Information Extractors)
    - Matching a grain in a model onto an ontology
    - No such a support is available today

Who is working on (?)

- EIF/IDA: EU Semantic Interoperability Initiative (SEMIC.EU: Semantic Interoperability Centre Europe)
- INTEROP Virtual Laboratory (interop-vlab.eu)
  - Follow-up of the INTEROP Network of Excellence
  - Organized into Scientific Poles
  - European Poles + Chinese Pole
- INTEROP-GrandeRegion.eu Pole
  - Organized as a Scientific Interest Group (GIS)
  - U. Lorraine (LORIA, CRAN), U. Paris 1, La Sorbonne, U. Namur (Belgium), U. Lyon 1, INSA Lyon.

Concluding Remarks

Thank you for paying attention

Questions?

Elements of bibliography

- Yongxin Liao, Mario Lezoche, Eduardo Rocha Loures, Hervé Panetto, Nacer Boudjlida. A semantic annotation framework to assist the knowledge interoperability along a product life cycle. In Advanced Materials Research, Trans Tech Publications, 2014, pp.424-429.
- Hamri Salah, Boudjlida Nacer, Boufaida Mahmoud. Semantic Annotations for Workflow Interoperability. In Informatica : An International Journal of Computing and Informatics, 2014, 38 (4), pp.347-366.
- H. Talantikit Nacer, D. Aissani and N. Boudjlida. Semantic annotations for web services discovery and composition. in Computer Standards and Interfaces 31 (1108-1117), Elsevier, (doi: 10.1016/j.csi.2008.09.041), 2009.
- M. Uschold, R. Jasper. A Framework for Understanding and Classifying Ontoloy Applications. *Procedings of the IJCAI'99 Workshop on Ontologies and Problem Solving Methods,* Stockholm, Sweden, August 2, 1999.
- K. Sivashanmugam, K. Verma A. Sheth and J. Miller, Adding Semantic to Web Services Standards, *1rst Intern'l Conference on Web Services (ICWS'03),* Las Vegas, Nevada, June 2003, pp. 395-401

Elements of bibliography

- S. Abiteboul, P. Buneman, and D. Suciu. Data on the Web; From Relations to Semi-Structured Data and XML. *Morgan Kaufmann Publishers*, San Francisco, CA, 2000.
- M.S. Fox, M. Barbuceanu, and M. Gruninger. An organization ontology for enterprise modeling: Preliminary concepts for linking structure and behaviour. *Computers in Industry*, 29(1-2):123–134, July 1996
- D. Konstantas. Object Oriented Interoperability. *In ECOOP'93, 7th European Conference on Object Oriented Programming,* Lecture Notes in Computer Science 707, pages 80–102, Kaiserslautern, Germany, July 1993.
- W. Inmon, J.A. Zachman and J.G. Geiger. *Data stores data warehousing and the Zachman framework, Managing enterprise knowledge*. McGraw-Hill, 1997.
- S. Bechhofer, L. Carr, C.A. Goble, S. Kampa and T. Miles-Board, *The semantics of Semantic Annotation.* In CoopIS/DOA/ODBASE'2002, First International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems, Irvine, California. Springer-Verlag LNCS Vol. 2519, pp. 1151--1167. 2002. METEOR-S:
- Semantic Web Services and Processes, http://swp.semanticweb.org