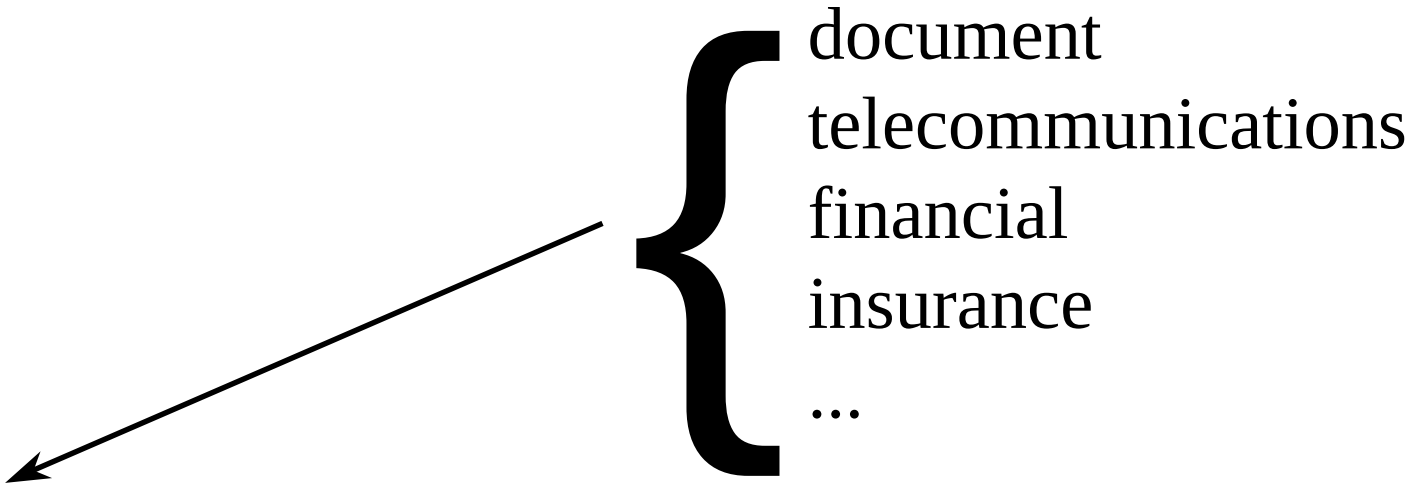
International standards define semantics:
RM-ODP and GRM

Haim Kilov
Merrill Lynch
haim_kilov@ml.com
haim_kilov@omg.org

Good patterns of thought

- ***separate*** business and system ***concerns***
- specify declaratively (use ***invariants*** and operation contracts)
- look for reusable constructs (large and small)
- be explicit and precise (so your statements can be challenged *and* defended)
- be simple and elegant (16 fragmentary views is too many)
- … and use the same concepts throughout

… And use the same concepts throughout



Business analysis — producing and managing business specifications — can become as disciplined as programming, by using specification concepts that are *understood* and *explicitly formulated*

Let's reuse what good programmers do...
Let's keep what good modelers have done...

Some international standards do just that

- ISO Reference Model for Open Distributed Processing (RM-ODP)
- ISO General Relationship Model (GRM)

Why ODP?

- Promotes understandability
- ***Common generic concepts*** throughout lifecycle
  - from specification to implementation
- **Emphasis on semantics, not syntax**
- Well-written and **short**!

Object-orientation = **abstraction** + precision

Abstraction — suppression of irrelevant detail

- **to enhance understanding**
- **by establishing a simplified model**

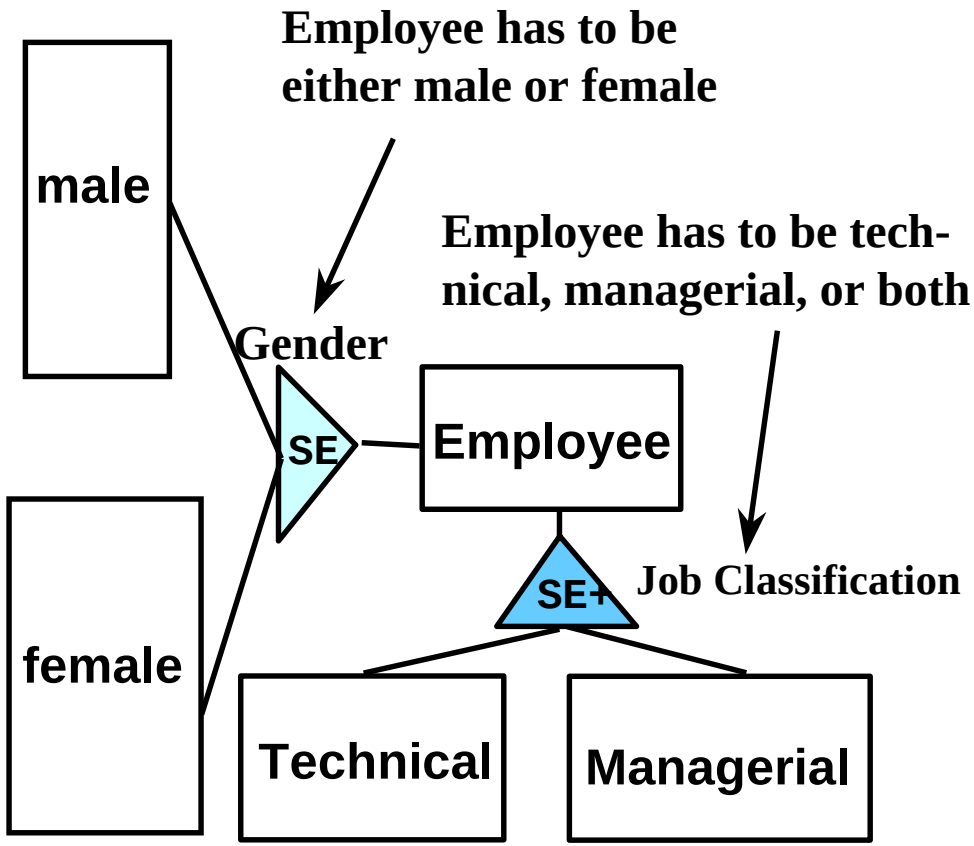
(ISO RM-ODP, Part 2)

(To get a precise (over)specification — of the system rather than of the business — just read the code)

Business rules

A business rule is a proposition about business *things*, *relationships* between them and *operations* applied to them, from the business enterprise viewpoint.

A proposition is an observable fact or state of affairs involving one or more entities, of which it is possible *to assert or deny* that it holds for those entities. [RM-ODP]



Common generic concepts (in RM-ODP)

abstraction (see above!)
proposition (see above!)
system (and subsystem)
Viewpoint (on a system): "a form of abstraction achieved using a selected set of architectural concepts and structuring rules, in order to focus on particular concerns within a system"
architecture: "a set of rules to define the structure of a system and the interrelationships between its parts"

Common generic concepts (in RM-ODP) (cont.)

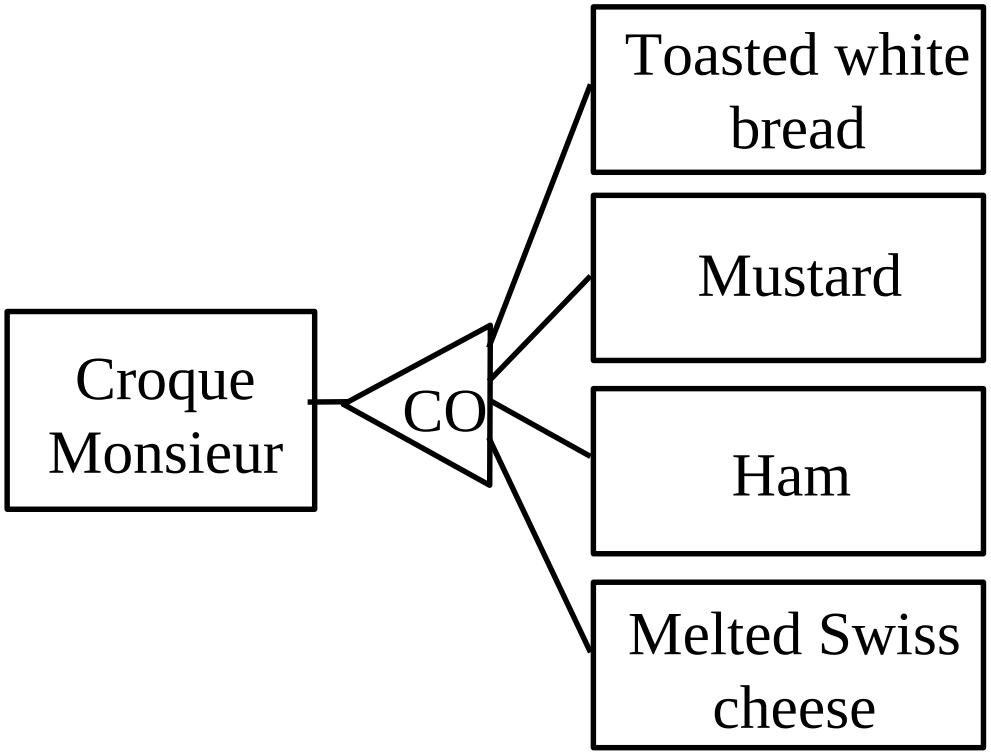
object ("...characterized by its behaviour and, dually, by its state…")
environment: "the part of the model which is not part of that object"
action
behaviour: "a collection of actions with a set of constraints on when they may occur"
state: "the condition of an object that determines the set of all sequences of actions in which the object may take part"

Common generic concepts (in RM-ODP) (cont.)

composition: of objects: “a combination of … objects yielding a new object, at a different level of abstraction… The characteristics of the new object are determined by the objects being combined and by the way they are combined.”

… “The composition of a collection of objects implicitly yields an equivalent object representing the composition. The behaviour of this object is often referred to simply as the behaviour of the collection of objects.”

Composition: different abstraction levels



… **`Ham`** sends the message

`Melted_Swiss_Cheese->lay_down_on_me()`

Representation...

Generating a good icon, file name, explanation or metaphor, and arranging text and graphics, all involve moving signs from one system to another. … In determining what makes one representation better than another, the entire structure [including complexity] of a sign system should be considered.

Joseph Goguen, Semiotic Morphisms, May 31st, 1997.

Common generic concepts (in RM-ODP) (cont.)

refinement: “the process of transforming a specification into a more detailed specification… specifications and their refinements typically do not coexist in the same system description”

type (of an <X>): “a predicate characterizing a collection of <X>s. An X is of the type, or satisfies the type, if the predicate holds for that <X>. … Types are needed for, at least, objects, interfaces, and actions.”

Common generic concepts (in RM-ODP) (cont.)

class (of <X>s): “the set of all <X>s satisfying a type” <<not an OOPL class!!!>>

<X> **template**: “the specification of the common features of a collection of <X>s in sufficient detail that an <X> can be instantiated using it.”

Invariant; precondition; postcondition

epoch: “a period of time for which an object displays a particular behaviour”

contract: “an agreement governing part of the collective behaviour of a set of objects”

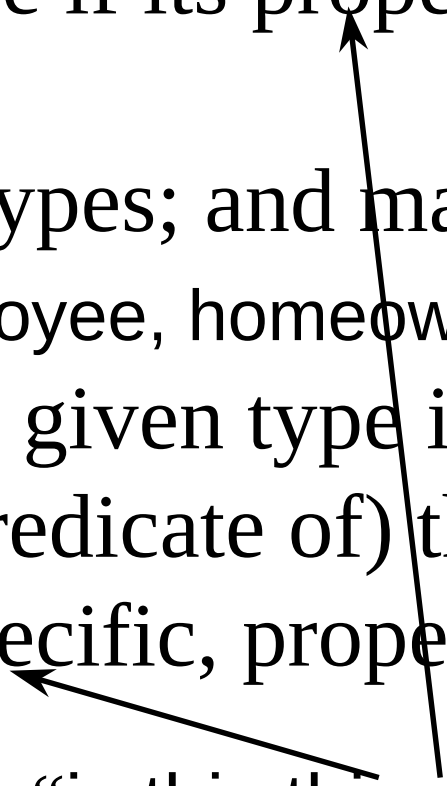
Obligation, permission, prohibition

Type and Subtype

A thing is of a particular type if its properties satisfy (the predicate of) this type.

A thing may have several types; and may acquire and lose types (person, employee, homeowner, …)

A thing is of a subtype of a given type if and only if its properties satisfy (the predicate of) the given type, and other, subtype-specific, properties.



you can ask a question “is this thing of that type?” and get a “yes” or “no” answer

--> “is this a croque monsieur?”

--> “is this a contract?”

--> “is this an underwriting prerequisite?”

Type and subtype (cont.)

- Not only objects have types; relationships have them, too!
- The type invariant determines, in part, the set of available operations

Common generic concepts (in RM-ODP) (cont.)

Role: Identifier for a behaviour, which may appear as a parameter in a template for a composite object, and which is associated with one of the component objects of the composite object.

name: “a term which, in a given naming context, refers to an entity” identifier: “an unambiguous name, in a given naming context”

naming context: “a relation between a set of names and a set of entities…”

contractual context: “the knowledge that a particular contract is in place, and thus that a particular behavior of a set of objects is required. An object may be in a number of contractual contexts simultaneously: the behavior is constrained to the intersection…”

Common generic concepts (in RM-ODP) (cont.)

Persistence: "The property that an object continues to exist across changes of contractual context or of epoch."

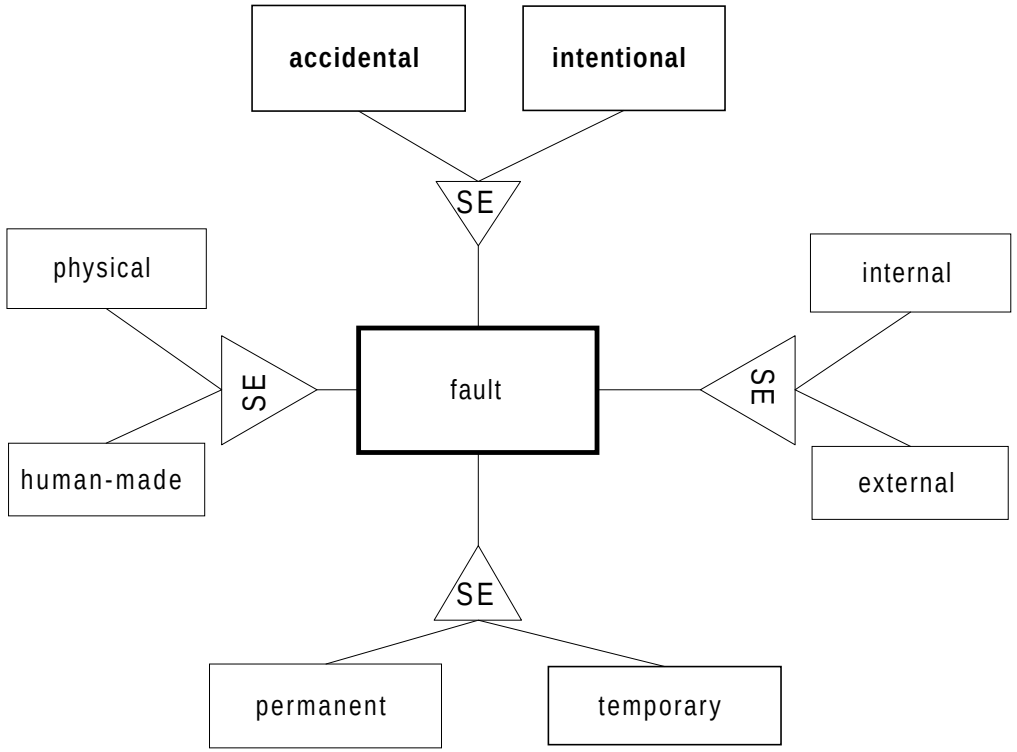
Failure: Violation of a contract. The behaviour specified in the contract is, by definition, the "correct behaviour". A failure is thus a deviation from compliance with the correct behaviour.

Error: Part of an object state which is liable to lead to failures. A manifestation of a fault in an object. Corrective action may prevent an error from causing a failure.

Fault: A situation that may cause errors to occur in an object.

Common generic concepts (in RM-ODP) (cont.)

Faults can be: accidental (that appear or are created fortuitously) or intentional (created deliberately); physical (due to some physical phenomena) or human-made (resulting from human behaviour); internal (part of an object state that may cause an error) or external (resulting from interference or interaction with the environment); permanent or temporary.



Common generic concepts (in RM-ODP Part 2)

-- continued

Semantics is precisely (semi-formally) defined

Formal definitions of some concepts and constructs provided in part 4: Architectural semantics

Viewpoints defined in Part 3 (enterprise viewpoint is in the process of standardization); all viewpoints are built upon Part 2

Basic constructs: invariants define relationships

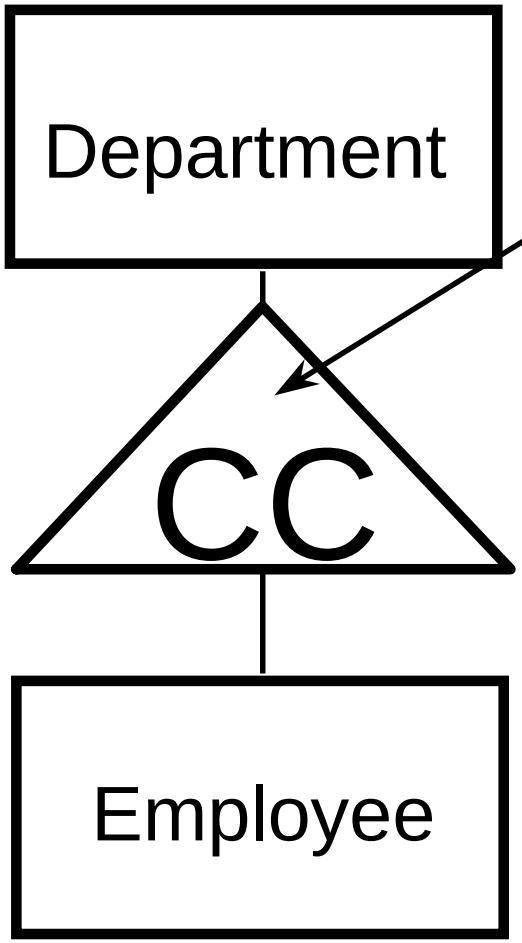
- Most invariants involve several objects
- Most invariants involve some combination of:
  - property derivations
  - existence dependencies
- Common invariants have been abstracted and documented for reuse:
  - they define generic relationships
  - international (ISO) standard: General Relationship Model

ISO General Relationship Model

... describes relationships between ... objects in an explicit, rigorous, and uniform manner, **independently** of the means of realizing or representing them

— with a strong emphasis on behavioral semantics
— and based on RM-ODP

Define “it” explicitly (and precisely)



Composition-containment

- Department can exist without Employees
- Employee has to belong to exactly one Department
- at least one property of Department is independent of its Employees
- at least one property of Department is determined by properties of Employees
- ...

```
first_level_org: containment(Department, Employee)
```

Abbreviations

So, “CC”, or “containment” is an abbreviation of:
- firstly, a relationship
- more specifically, a composition,
- even more specifically, a composition-containment

Let’s define these concepts.

Relationships

- A relationship is a collection of things.
- A relationship ***definition (its invariant)*** involves each of these things
  - and describes what remains the same *(invariant)* throughout the lifetime of the relationship

- relationship: **“a collection of … objects together with an invariant referring to the properties of the … objects”** [GRM, Clause 3.8.4]
- invariant: **“a logical predicate that must remain true during some scope; a scope might be the lifetime of a managed relationship or the execution of a relationship management operation”** [GRM, Clause 7.1.2]

Relationship Operations

- Terminate
  - Precondition: the managed relationship exists
  - Postcondition: the managed relationship does not exist; managed objects that were bound in the relationship are not bound in the relationship
- Establish, bind, unbind, query, notify [GRM, Clause 8.1]

Generic relationships

... are also defined by their invariants.

A small number of generic relationships (eg composition, reference, subtyping) occur everywhere.
Specifying them once and then reusing simplifies both understanding and implementation

Composition instantiation

A composite type **(department)** corresponds to one or more component types **(employee)**, and a composite instance corresponds to zero or more instances of each component type

At least one resultant property **(eg department's budget)** of a composite instance is dependent upon the properties of the components

At least one emergent property **(eg department's mission statement)** of the composite instance is independent of the properties of the components

...Application-specific types for the composite and the components are not equal **(obvious)**

Composition operations

- Establish
  - Signature: The class and identity of the proposed participant in the composite role and the class and identity of the proposed participant(s) in the component role to be bound in an instance of the … relationship
  - Precondition: The proposed participants are not already bound in the same instance of the … relationship class…
  - Postcondition: An instance of the … relationship class exists; the participants referred to in the signature are bound into this instance of the … relationship class
- Bind, Unbind, Terminate, … [GRM, Clause F.4]

A warm and fuzzy feeling is not a definition.

Reuse of generic relationships (elementary generic business patterns)

- Composition (and its subtypes, e.g., containment)
- Reference association
- Subtyping
- Dependency
- Symmetric relationship

Semantic properties of these are known;
they are defined once and reused for all applications
(no need to reinvent these concepts and redefine
their properties for every new application)

Collective behavior (generalized object model)

Who “owns” the *bind* operation?
Or the *hire employee* operation?
Or the composition invariant?

> These are owned by the collections of relevant objects; and messages **may** be used in their **implementation**

Distinguishing a recipient (“owner”) of an operation or of an invariant leads to overspecification (making needless choices)

Interesting Behavior

- Behavior must be described at the system level, not the object level
  - **All interesting behavior is in the relationships between objects**
  - **It is impossible to understand behavior of the system by looking only at the behavior of its parts**

Anthony Hall (Deloitte and Touche, RE'97)

References

- E W Dijkstra. *Some meditations on advanced programming*. Proceedings of the IFIP Congress 1962, North Holland, 1963, pp. 535-538.
- E W Dijkstra, *Selected Writings on Computing: A Personal Perspective*, Springer-Verlag, 1982.
- ISO/IEC JTC1/SC21, Information Technology - Open Systems Interconnection - Management Information Systems - Structure of Management Information - Part 7: General Relationship Model, ISO/IEC 10165-7, 1995.
- C A R Hoare. Mathematical models for computing science. Oxford, August 1994.
- M¡cheal Mac an Airchinnigh, Dag Belsnes and Gerard O'Regan. Formal methods & Service specification. In: *Towards a Pan-European Telecommunication Service Infrastructure* (Lecture Notes in Computer Science, Vol. 851). Ed. by H.-J.Kugler, A.Mullery, N.Niebert, Springer Verlag, 1994, pp. 563-572
- Robert N.Corley, Peter J.Shedd. *Fundamentals of Business Law*, 5th Edition. Prentice-Hall, 1990.
- Haim Kilov, Helen Mogill, Ian Simmonds. Invariants in the Trenches. In: *Object-oriented behavioral specifications* (ed. by H Kilov and W Harvey), Kluwer Academic Publishers, 1996.
- Haim Kilov, James Ross. *Information Modeling: an Object-oriented Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1994.
- David Lorge Parnas, Jan Madey, Functional Documents for Computer Systems, *Science of Computer Programming*, Volume 25, 1995, pp. 41-61.
- ISO/IEC JTC1/SC21/WG7, Open Distributed Processing - Reference Model: Part 2: Foundations (IS 10746-2 / ITU-T Recommendation X.902, February 1995).
- Haim Kilov, Ian Simmonds. Business patterns: reusable abstract constructs for business specification. In: *Implementing Systems for Supporting Management Decisions: Concepts, methods and experiences*, Edited by Patrick Humphreys et al, Chapman and Hall, 1996, pp. 225-248.
- Haim Kilov, Ian Simmonds. Business patterns and viewpoints. *Proceedings of the Workshop on Viewpoints*, ACM Symposium on Foundations of Software Engineering, San Francisco, Oct.1996.
- Haim Kilov, Ian Simmonds. How to correctly refine business specifications, and know it. *Proceedings of the Fifth OOPSLA Workshop on Specification of Behavioral Semantics*, San Jose, USA, Oct. 1996, pp. 57-69, ISSN 1089-7143.
- David Alex Lamb, Editor, *Studies of Software Design*, Lecture Notes in Computer Science, Volume 1078, Springer Verlag, 1996.
- ANSI X3H7-96-07R2. *Business rules for the Enterprise Viewpoint of RM-ODP*. (H.Kilov). 7 December 1996.