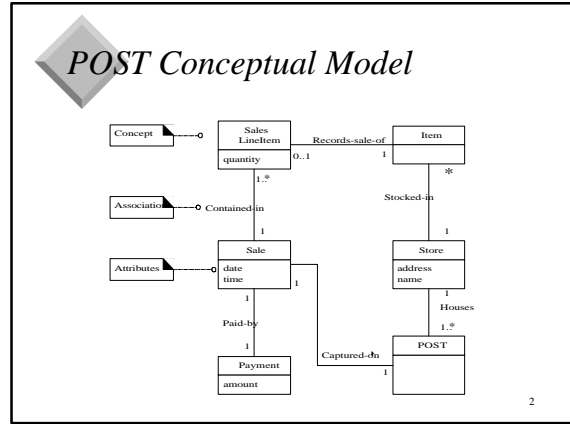


Conceptual Models

- ❖ Describe real-world concepts, not Software entities
- ❖ Elements should not be thought of as classes in C++ or Java
- ❖ Used only to help elaborate the problem domain (like a roadmap drawn on a napkin)
- ❖ Attributes are allowed, but operations are not!

1



Do's and Don'ts in Conc. Model

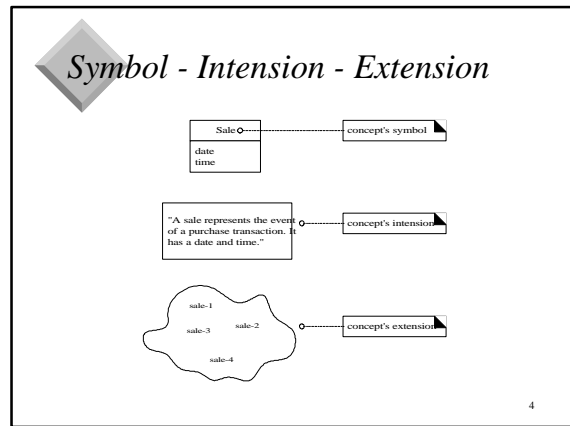
Do:

Don't:

void

void

3



Domain Concepts

- ❖ *Division by Concepts* is preferred
 - Object-oriented Analysis
- ❖ over *Division by Functions*
 - Decomposition principle
 - Structured Analysis

5

Identifying Concepts

- ❖ The “Noun Phrase” technique (Abbot, 1983)
- ❖ When applied to the POST:

6

What about Sales Receipt?

- ❖ Should it be included in the model?
 - Its common in the real world system
 - It's a sales report. Reports not explicitly stated in the use cases have little value in this model.
 - However, when a Customer wishes to return an item, it is an important object in the domain.
 - Since this develop cycle doesn't include the Return Items use case, we'll leave it out of this CM

7

Attribute or Concept

- ❖ Should destination be an attribute of Flight?
 - Or another concept altogether?
- ❖ Rule: if it can't be represented as a number or simple text string, it's a concept, not an attribute

8

POST or Cash Register?

similar concepts with different names

Either term has its merits (and drawbacks)

9

Specifications

- ❖ As artifacts in the problem domain
 - Suppose every item has an ID (UPC #), and a price. The item sells out. Someone calls for the price. Since no objects are left, no price information remains in the system.
- ❖ A Product Specification is needed
 - This begins to show the value of "patterns"

10

ProductSpecification Concept

- ❖ Keeps item info independent of the objects

11

Airline Example

12

