Handout 5

Identifying Object Attributes

[Coad-Yourdon]

**Attribute** – some data for which *each* Object in a Class has its own value.

Identify the attributes:

Identify what is the Class&Object responsible of knowing.

Ask from the single object’s prospective:

- How am I described in general?
  - in this problem domain?
  - in the context of the system’s responsibilities?

- What do I need to know?
- What states may I be in?
- What state information do I need to remember over time?

Example:
Object&Class: CheckingAccount
- belongs to owner(s)
- in a banking application:
  - has transactions (withdrawals, deposits) associated with it
- has a maintenance charge
- minimum balance requirement
- maximum number of free transactions per month
- Other?
Rules of the game:

- whole-part and other associations are attributes
- attributes have to be atomic: single value (e.g. color, status) or a tightly related grouping of data elements (e.g. address, name)
- position attributes within Class&Object which it best describes

Deferred to design – don’t worry too much during analysis about

- normalization (no redundancy, no internal structure)
- identification mechanisms
- recalculable attributes

Attributes of the Superclass are inherited by all Subclasses

- attributes are shared by all objects in the class – shouldn’t have objects that have attribute value *NOT Applicable* -- check the Gen-Spec structure if this happens (e.g. maiden name)

- reexamine Class&Objects with a single attribute – may be redundant

Instance connection attributes

Instance connections model association between Objects. Shown as a line between associated objects, with multiplicity labels on each side.

Sometimes association is complex and requires a new Class&Object to describe it (e.g. customer owns account)

Whole –part relations are attributes too!