Object design

- Analysis reduces the gap between the problem and the machine by identifying objects representing problem-specific concepts.
- During analysis the system is described in terms of external behavior such as its functionality (use case model), the application domain concepts it manipulates (object model), its behavior in terms of interactions (dynamic model), and its nonfunctional requirements.

- System design reduces the gap between the problem and the machine in two ways.
  - First, <u>system design results in a virtual machine that provides a higher level of abstraction than the machine</u>. This is done by selecting off-the-shelf components for standard services such as middleware, user interface toolkits, application frameworks, and class libraries.
  - Second, <u>system design identifies off-the-shelf components for application domain objects such as reusable class libraries of banking objects</u>. After several iterations of analysis and system design, the developers are usually left with a puzzle that has a few pieces missing. These pieces are found during object design. This includes identifying new solution objects, adjusting off-the-shelf components, and precisely specifying each subsystem interface and class. The object design model can then be partitioned into sets of classes that can be implemented by individual developers.

Object design includes four groups of activities

- *Reuse*
- *Interface specification*
- *Restructuring*
- *Optimization*

Reuse:

- Off-the-shelf components identified during system design are used to help in the realization of each subsystem.
- Class libraries and additional components are selected for basic data structures and services.
- Design patterns are selected for solving common problems and for protecting specific classes from future change.
- Often, components and design patterns need to be adapted before they can be used.
- This is done by wrapping custom objects around them or by refining them using inheritance.
- During all these activities, the developers are faced with the same buy-versus-build trade-offs they encountered during system design.

Interface specification

- During this activity, the subsystem services identified during system design are specified in terms of class interfaces, including operations, arguments, type signatures, and exceptions.
- Additional operations and objects needed to transfer data among subsystems are also identified.
- The result of service specification is a complete interface specification for each subsystem.
- The subsystem service specification is often called subsystem **API** (Application Programmer Interface).

Restructuring

- Restructuring activities manipulate the system model to increase code reuse or meet other design goals.
- Each restructuring activity can be seen as a graph transformation on subsets of a particular model.
- Typical activities include transforming N-ary associations into binary associations, implementing binary associations as references, merging two similar classes from two different subsystems into a single class, collapsing classes with no significant behavior into attributes, splitting complex classes into simpler ones, and/or rearranging classes and operations to increase the inheritance and packaging.
- During restructuring, we address design goals such as maintainability, readability, and understandability of the system model.

Optimization

- Optimization activities address performance requirements of the system model.
- This includes changing algorithms to respond to speed or memory requirements, reducing multiplicities in associations to speed up queries, adding redundant associations for efficiency, rearranging execution orders, adding derived attributes to improve the access time to objects, and opening up the architecture, that is, adding access to lower layers because of performance requirements.

- Object design is not sequential.
- Although each group of activities addresses a specific object design issue, they usually occur concurrently.
- Usually, interface specification and reuse activities occur first, yielding an object design model that is then checked against the use cases that exercise the specific subsystem.