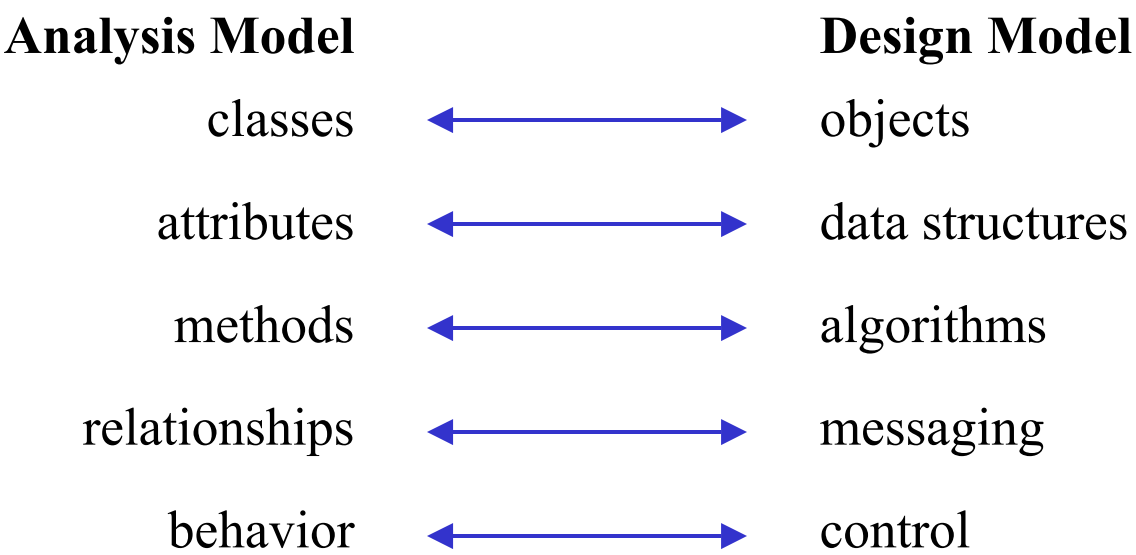
Design methods: Object-Oriented Design†

- Basic picture:



†Material from text by Budgen and from "Software Engineering (fourth edition)", by Roger Pressman.

Generic components of OO design model

Problem Domain

Subsystems responsible for specific customer requirements.

Human Interaction

Subsystems that implement user interface.

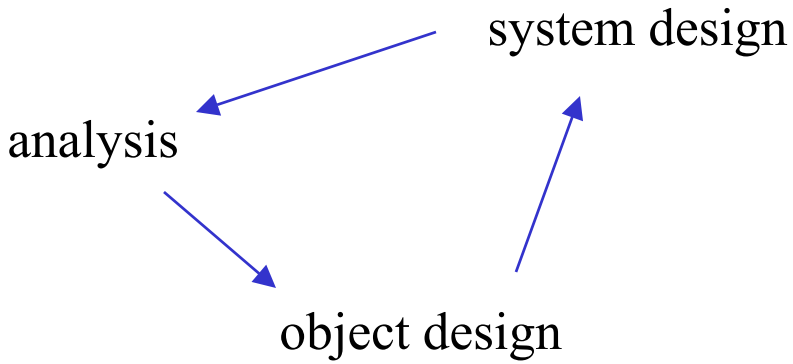
Task Management

Subsystems responsible for controlling and coordinating concurrent tasks.

Data Management

Subsystem responsible for storage and retrieval of objects.

Overall process



Two levels of granularity in design:

1. The *system design process*; and
2. The *object design process*.

System design process

- Partition analysis model into subsystems.
- Identify concurrency dictated by problem.
- Allocate subsystems to processes and tasks.
- Choose strategy for implementing data management.
- Identity global resources and the control mechanisms required to access them.
- Design an appropriate control mechanism for the system.
- Consider how boundary conditions should be handled.
- Review and consider tradeoffs.

Object design process

- Object descriptions:
  - a *protocol description*; and
  - an *implementation description*.

- Need to design algorithms and data structures (often straightforward, but not necessarily).

- Three broad categories of operations.
  1. Operations that manipulate data.
  2. Operations that perform a computation.
  3. Operations that monitor an object for the occurrence of a controlling event.

Partitioning the analysis model

How does one recognize appropriate clusters?

Basic guidelines: as subsystems are defined (and designed), they should conform to the following design criteria.

- Each subsystem should have a well-defined interface through which all communication with the rest of the system occurs.
- With the exception of a small number of communication classes, the classes within a subsystem should collaborate only with other classes within the subsystem.
- The number of subsystems should be kept small.
- Subsystems can be partitioned internally to help reduce complexity.
- Communication between subsystems is either *peer-to-peer* or *client-server*.

Concurrency and subsystem allocation

Dynamic aspects of object-behavior model provide the indication of concurrency among objects or subsystems.

If objects or subsystems must act on events asynchronously and at the same time, they are viewed as concurrent.

When concurrent, there are two options:

1. Allocate to independent processors; or
2. Allocate to the same processor and provide concurrency support through OS features.

The task management component

A basic strategy.

- Characteristics of tasks are determined by understanding how the task is initiated.
- A coordinator task and associated objects are defined.
- The coordinator and other tasks are integrated.

Also relevant: the priority and criticality of tasks must be determined.

The task management component (cont'd)

A basic task template.

Task name: the name of the object.

Description: purpose of object in narrative form.

Priority.

Services: a list of operations that are object responsibilities.

Coordinates by: the manner in which object behavior is invoked.

Communicates via: input and output data values for the task.

The data management component

There are two distinct areas of concern:

1. Management of data critical to the specific application; and
2. Creation of an infrastructure for storage and retrieval of objects.

Often, a DBMS is an appropriate choice for 2. (Relational technology is pervasive.) A DBMS can be used to resolve the following problems and issues:

- Reliability (with backup and recovery);
- Concurrency control; and
- Data independence.

For many tasks/requirements, SQL is much more convenient than other languages.

The data management component (cont'd)

The first area of concern is the specific issue of identifying attributes and operations (schema and transaction types) required to manage objects.

If an RDBMS or file system is the underlying infrastructure, then a relevant issue is "how do I store myself?"

The resource management component

A control mechanism is needed for other kinds of resources.

Examples include: disk drives, processors, communication lines, memory, …

Human-computer interface component

The HCI is almost always a critically important subsystem. Relevant analysis products include:

- use cases, and
- roles played by users (called actors).

Typically proceed by top-down design of menu/command hierarchies.

A wide variety of HCI development environments and standards already exist. Reusable classes (with appropriate attributes and operations) already exist for windows, icons, mouse operations and many other interaction functions.

Inter-subsystem communication

Once each subsystem is specified, one needs to define collaborations between subsystems. A collaboration can be viewed as a high-level transaction type or request.

One way to proceed.

1. First identify transaction types and participating subsystems. For each type, identify one or more *contracts* between pairs of subsystems.
2. If modes of interaction are complex, a high-level event-flow graph should be defined (called a *collaboration graph*).

Inter-subsystem communication (cont'd)

3. For each contract, identify the following:
   - *type* (i.e., client-server or peer-to-peer);
   - *collaborators*;
   - *components of subsystems supporting services implied by the contract*; and
   - *message format*.