Getting Started

In the beginning… there is:

Specification:

Design a library catalog system. The system must support the registration of patrons, adding and removing books, checking books in and out, satisfying queries regarding the availability of books, and determining which patron has a book.

The specification is somewhat unsatisfactory… that is typical.

Much is unspecified, some of that may be important.

Often, much is said in the specification that is unimportant.

Identifying the Types (Objects)

We must:

- identify potential types/objects from the specification
- eliminate phony candidates
- determine how the legitimate types/objects will interact
- extrapolate classes from the types/objects

This process:

- requires experience to do really well
- requires guidelines, none of which is entirely adequate
- often uses several approaches together
- should lead to too many rather than too few potential types/objects

Approaches

Abbott and Booch suggest:

- use nouns, pronouns, noun phrases to identify types/classes and objects
- singular → instance/object, plural → type/class
- not all nouns are really going to relate to types/objects

Coad and Yourdon suggest:

- identify individual or group "things" in the system/problem

Ross suggests common type/object categories:

- people
- places
- things
- organizations
- concepts
- events

Objects and the Problem Domain

What constitutes a "potential type/object" depends on the problem domain.

Discuss with a domain expert — a person who works in the domain in which the system will be used.

Try to identify types/objects from the way that the users/experts think about the system/problem.

There are usually no handy domain experts for CS projects!

Specification:

Design a library catalog system. The system must support the registration of patrons, adding and removing books, checking books in and out, satisfying queries regarding the availability of books, and determining which patron has a book.

Categories of Types/Classes

- tangible things from the problem domain
- system interfaces and devices
- agents, providing types/objects to carry out operations
- events and transactions, something done
- users and roles
- sub-systems
- external systems
- containers

Eliminate "false" Objects

An type/object should:

- be a real-world entity
- be important to the discussion of the requirements
- have a crisply defined boundary
- make sense; i.e., the attributes and behaviors should all be closely related

Danger signs:

- class name is a verb
- class is described as performing something
- class involves multiple abstractions
- class is derived from another, but has few features itself
- class has only one public responsibility/method
- class has no responsibilities/methods

Example: Library System

Looking for nouns:

Specification:

Design a **library catalog system**. The **system** must support the **registration** of **patrons**, adding and removing **books**, checking **books** in and out, satisfying **queries** regarding the availability of **books**, and determining which **patron** has a **book**.

First cut:

- library
- catalog
- system
- registration
- patron
- book
- query

Example: Library System

Looking for structures:

Specification:

Design a **library catalog system**. The **system** must support the **registration** of **patrons**, adding and removing **books**, checking **books** in and out, satisfying **queries** regarding the availability of **books**, and determining which **patron** has a **book**.

Do any of the surviving candidate classes require data structure support?

- Catalog
- Patron
- Book

Example: Library System

Delegating control:

Specification:

Design a **library catalog system**. The **system** must support the **registration** of **patrons**, adding and removing **books**, checking **books** in and out, satisfying **queries** regarding the availability of **books**, and determining which **patron** has a **book**.

What about overall control?

- The primary controller may be either procedural or an object.
- CirculationDesk
  - uses the Catalog, which contains the BookList, a list of Book objects
  - uses the PatronList
  - (somehow) provides support for each of the required actions
- However, the CirculationDesk should respond to instructions (events), not seek them out. In the implementation, we must parse an input script, or have a GUI. Either way, that's not part of the CirculationDesk, although it would interact with it.