# OO Analysis: Domain Model

Domain Model describes the problem domain (context). A Domain Model visualizes, using UML class diagram notation, noteworthy concepts, or objects. It is a kind of "visual dictionary." Not a picture of software classes.
A domain model gives a conceptual visualization of the problem, it shows:

- domain objects or conceptual classes.
- associations between conceptual classes.
- attributes of conceptual classes.

So, domain model is a visual representation of conceptual classes or real-situation objects in a domain. It is illustrated with a set of class diagrams without operations/methods. It helps us identify, relate, and visualize important information. Domain Model is a basic class diagram with conceptual classes in the problem domain. It represents real-world concepts, not software components.

To create a domain model, usually following activities are performed:

- Concepts Extraction
- Noun Analysis
- Object/Class Identification
- Class → Name, Attributes
- Relationships between classes
- Basic Class/Object Diagram (Static, Structural View)

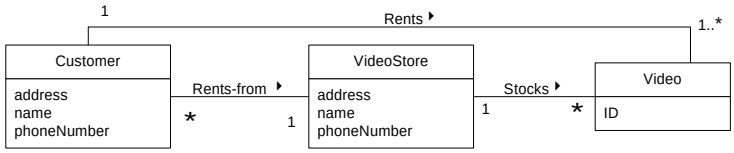
We have already discussed Object/Class Identification, and Relationships between classes in lecture notes.
Now we create a domain model of a simple example.

"A video rental store, give videos to its customers. Video store stocks videos"

From the above system description we identify following classes:

Customers VideoStore Video

Attributes of classes and relationships/associations between these classes is depicted in the form of a domain model as shown below:




Using another example of library system, we create domain model by performing most the activities of domain model.

The library contains books and journals. It may have several copies of a given book. Some of the books are for short term loans only. All other books may be borrowed by any library member for three weeks. Members of the library can normally borrow up to six items at a time, but faculty members may borrow up to 12 items at one time. Faculty members may borrow journals

*Borrowing:* The system must keep track of when books and journals are borrowed and returned, enforcing the rules described above.

By performing noun analysis, we identify following candidate classes:

| | |
|---|---|
| Library | the name of the system |
| Book | |
| Journal | |
| Copy | |
| ShortTermLoan | event |
| LibraryMember | |
| Week | measure |
| MemberOfLibrary | repeat |
| Item | book or journal |
| Time | abstract term |
| MemberOfStaff | |
| System | general term |
| Rule | general term |

Next we discard those which are 'obviously' not good candidate classes for any one of the following reasons:

library, because it is outside the scope of the system.
short term loan, because a loan (short term or otherwise) is really an event.
member of the library, which is redundant: it means the same as library member.
week, because it is a measure of time, not a thing.
item, because it is vague: when we clarify it we see that it means book or journal.
time, because it is outside the scope of the system.
system, rule (not part of the domain).

After that we have first-cut list of following probable classes:

- Book
- Journal
- Copy (of book)
- Library member
- Faculty member

Next we identify relationship between classes, and for that we identify and name important real-world relationships or associations between classes. To clarify our understanding of the domain, by describing our objects in terms of how they work together.

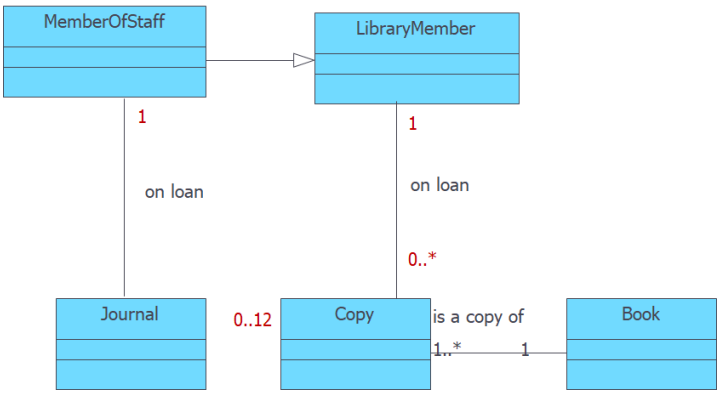
Relationship between Classes

| | | |
|---|---|---|
| Book | is an | Item |
| Journal | is an | Item |
| Copy | is a copy of a | Book |
| LibraryMember | | |
| Item | | |
| MemberOfStaff | is a | LibraryMember |

After identification of classes and relationships, we can easily see the following operations of the classes.

| | | |
|---|---|---|
| LibraryMember | borrows | Copy |
| LibraryMember | returns | Copy |
| MemberOfStaff | borrows | Journal |
| MemberOfStaff | returns | Journal |

By using the information, we have gathered so far, in the form of classes, relationships between classes, and operations being performed by classes the domain model is depicted as shown below:




Now we describe the process of domain model creation for an information modeling system for a school.

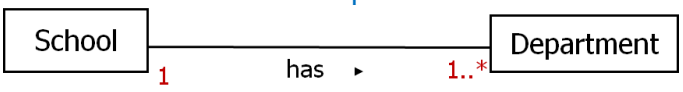
School has one or more Departments. Department offers one or more Subjects.
A particular subject will be offered by only one department. Department has instructors and instructors can work for one or more departments. Student can enroll in up-to 5 subjects in a School. Instructors can teach up-to 3 subjects. The same subject can be taught by different instructors. Students can be enrolled in more than one school.

First, we identify classes, and following classes are easily identified:

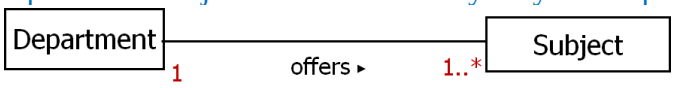
School Departments Subjects
Instructor Student

Now, we look into system description and focusing on classes and operations being performed by classes we can easily identify associations/relationships between classes. We create domain in an incremental way as described below:

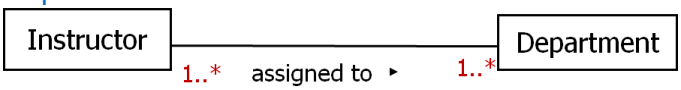
School has one or more Departments.



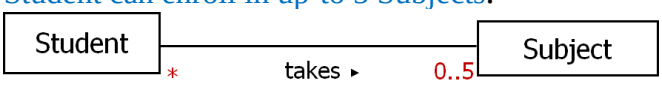

Department offers one or more Subjects.
A particular subject will be offered by only one department.



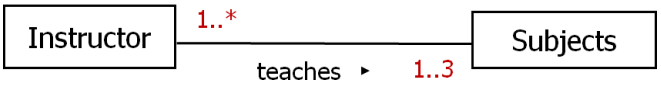

Department has Instructors and instructors can work for one or more departments.



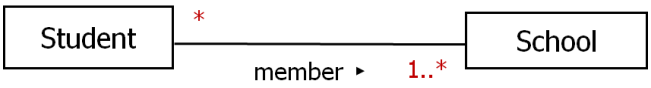

Student can enroll in up-to 5 Subjects.



Instructors can teach up to 3 subjects.
The same subject can be taught by different instructors.




Students can be enrolled in more than one school.




We have illustrated all the relationships between five classes, by combining all these we have following domain model: