Chapter 8

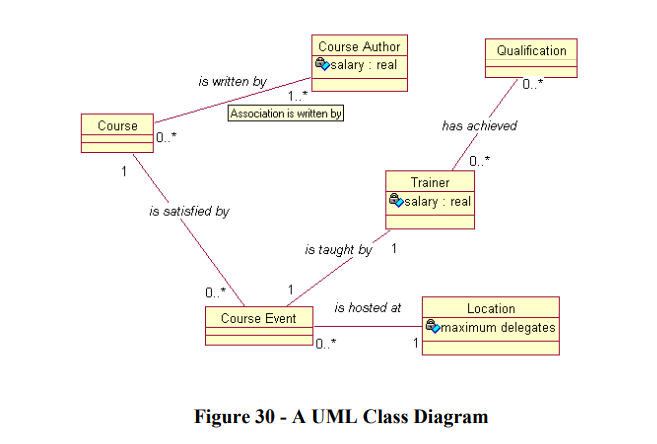
Conceptual Modelling

Conceptual Modelling (sometimes called Domain Modelling) is the activity of finding out which concepts are important to our system.

Aim:

* + - to understand the problem further
    - develop a better awareness of our customer’s business.

UML does no support Domain Modelling🡺we are going to use is the class diagram.



-We are not yet interested in the system design(since still analysing)

🡺 We will make sketchy class diagram(Analysis Class Diagram), and will not contain any design decisions.

Should completely distinguish Analysis Class Diagram from the Design Class Diagram by calling the Analysis Diagram the Conceptual Model.

On the conceptual model, we aim to capture all of the concepts or ideas that the customer recognizes

For example, some good examples of concepts would be:

• Lift in a lift control system

• Order in a home shopping system

• Footballer in a football transfers system (or a PlayStation football game!).

Bad example:

* CustomerDetailsForm - the window that asks for details of the new customer in a shopping system

• DbArchiveTable - the database table holding a list of all old orders

Why is bag example? because they are *focussing on design* - the solution, and not the problem. the conceptual model will *change* into *a full design class diagram* in the *construction phase*.

Explain: In the DbArchiveTable example, we are already tying ourselves down to a relational database solution. What if it turns out later that it is more efficient, cheaper, and perfectly acceptable to use a simple text file?

**Tip**: If the customer doesnít understand the concept, it probably isn’t a concept!

Finding Concepts

-similar approach to finding Use Cases, with as many interested stakeholders as possible.

Brainstorm suggestions (phiên thảo luận ý kiến) for concepts, capture all the suggestions.

When complete, work as a group to discuss and justify each suggestion.

Tip: customer must understand the concept, and discard any that donít apply to the problem, and discard any that are touching on design.

Extracting Concepts From Requirements (Extract :trích xuất)

Craig Larman (ref [2]) suggests the following candidate concepts from the requirements:

• Physical or tangible objects

• Places

• Transactions

• Roles of People (eg Customer, Sales Clerk)

• Containers for other Concepts

• Other Systems external to the system (eg Remote Database)

• Abstract Nouns (eg Thirst)

• Organisations

• Event (eg Emergency)

• Rules/Policies

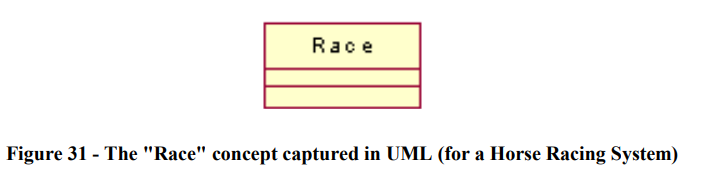
• Records/Logs

Note: *You must have input from the customer.*

1st: gathering concepts in a mechanical manner is poor practise.

2nd:not depending too much to extracting noun phrases from documents.

**The Conceptual Model in the UML**

****

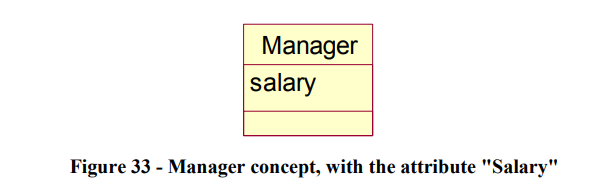
Name of concept: capitalized, at the top of the box

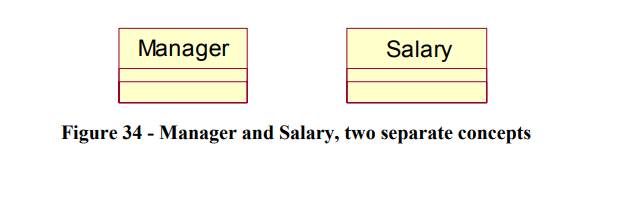
Two box below: attributes and behaviour.

Tip: *we needn’t worry about behaviour for now. We can do it in the design stage of construction.*

**Finding Attributes**

-Best way to find is brainstorming session with the stakeholders.





-One agrue that the concept “Manager” might have might named “salary”.Other argue that “Salary” is also a concept. So, should we make promote it from an attribute to a concept?

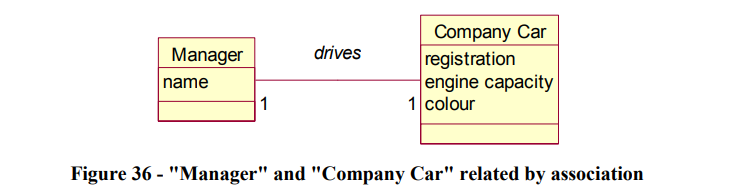
Tip: *if in doubt, make it another concept.*

• Single valued strings or numbers are usually attributes

• If a property of a concept cannot do anything, it might be an attribute

**Associations**

In any non-trivial system, at least some of the concepts are going to have some kind of conceptual relationship with other concepts.

****

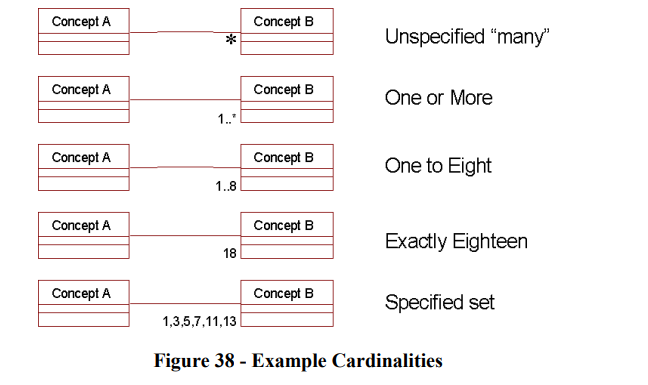
Note:

1st: the association has a descriptive name.

2nd: there are numbers at each end of the association which describe the cardinality of the association, and tell us how many instances of each concept are allowed.

**Possible Cardinalities**

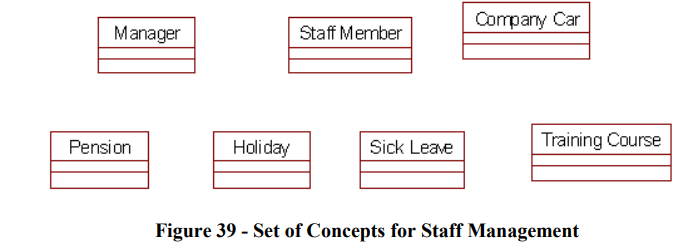
Basically, there are no restrictions on the cardinalities you are able to specify.

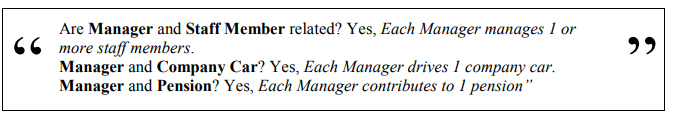


The difference between \*(mean “many”) and 1..\* (mean “ one-to-many):

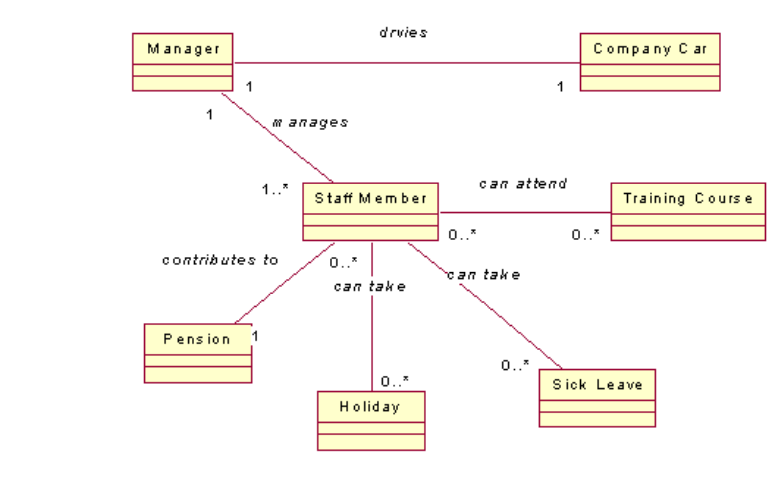
The former is a vague "many", meaning that perhaps any number of concepts are allowed, or maybe we haven't made the decision yet.

**Building the Complete Model**

****

****

The best way to proceed is to “fix” one concept, say ìManagerî and consider every other concept in turn. Ask yourself “are these two concepts related?”, and if so, immediately decide on the name of the association, and the cardinality …



Note:

* Should not leave off the association name until later since this is making extra work for yourself.
* associations are less important than attributes🡺 Any missing associations can do
* in design but missing attributes is not.
* concentrate on concepts and attributes, and try to fix the most obvious associations.
* the diagram should make sense to the customer when you "read back" the diagram in English