

Lecture 10: Modelling and Translation Example

Text problem to database solution, via UML

Assignment 1 last year described a business problem and asked for a database design and queries to solve it.

Today we work through the process of producing a complete solution.

See Assignment 1 from last year.

<http://cs.anu.edu.au/students/comp2400.2007/assignment1>.

- it is based on a real business operating in Canberra (where Greg worked before discovering Australia Post)
- Greg also has the benefit of having seen *many* attempted solutions, some of which were very good
- Your first assignment will be similar, and is coming soon

Read the Assignment!

Many problems resulted from people not reading what was asked.

Problem statements, and other relevant documents are a good starting point for modelling.

- nouns tell you the classes, eg product, order (unless there is only one instance, eg warehouse)
- underlining the nouns can be a good way to start
- same thing will be described in different ways
- choose good names (short, obvious, descriptive)
- look for properties of these things (attributes), and relationships between them (associations)

Keep it simple, solve the problem

You don't need to model everything mentioned in the document, you just need to solve the problem!

- models can easily grow out of control
- get clear on the purpose of the model, and leave everything else out!
- (however, some allowance for future needs can be prudent)
- the specific requirements statements tell you what you *must* be able to produce
- you can mentally “query” or navigate a UML class diagram, and see what's missing
- do we need to model customers here?
- no!

Decisions Decisions

- How should we model envelopes and boxes?
- a Packaging class?
- an attribute of an Order class?
- kinds of Product?

Diagram is not enough

- A UML diagram is of limited use by itself
- it needs supporting text, describing what the model elements represent
- eg. are packing materials included as products?

Object Diagram - Test Case

- try to exercise each possibility in each query
- eg. product out of stock, product with plenty in stock

Derived Attributes and Views

This problem naturally suggests the UML notion of *derived attribute*.

eg. the weight of an order

Database *views* are the obvious way to implement these.

Moral: read the manual, find the features that suit the job.

Object Diagram - Test Case

- try to exercise each possibility in each query
- eg. product out of stock, product with plenty in stock

Presenting the Solution

Guys in suits don't read code.

The proposal should be presented as a solution to a business problem, not as a lot of technical mumbo-jumbo.

Respect the readers time, tell them what they are reading and what it is for.