

Design Methodology

Dr John May
(j.may@bristol.ac.uk)
Safety Systems Research
Centre, UoB

1

Main Contents of Unit

- Background
 - what is design?
 - re-visit concepts of object orientation
- OO Design with UML (Unified Modeling Language)
- Functional Design with SSADM
- What unit is *not*
 - OO programming – other units
 - OO programming technology (.net, Java Beans, EJB, Corba...)

2

Course resources

- *Possible* book for OOD part of unit:
 - "Using UML" by Stevens, Pooley, revised edition, Addison-Wesley
- *Possible* book for SSADM part of unit:
 - "SSADM Models & Methods, Version 4 - Revised Edition" Skidmore, Mills, Formen, revised edition, NCC Publications 1994

3

Course resources (cont.)

- For OOD
 - TogetherSoft ControlCentre – being installed on Faculty lab machines, also I should be able to get you licenses for your own PCs (free)
 - ArgoUML – already on PCs in faculty lab, and you can download it free for your own PCs
- For SSADM – use any general drawing tools
- Unit resources web page
<http://www.cs.bris.ac.uk/Teaching/Resources/COMS30203/>

4

Self-learning

- You will need to read a fair amount of material outside of the lecture notes
 - on this unit it is possible to pass using just the lectures, but to do well you will need to use books and/or web (some resources provided to get you started, but probably need to do your own research on web too)
 - time is made available to do this - resist temptation to do nothing!

5

Assessment

- Coursework is 2 components, both individual:
 - 1. construct a design using UML - c. 65%
 - (hand in around middle of March - TBD)
 - 2. construct a design using SSADM - c. 35%
 - (deadline early May - TBD)

6

Unit objectives: I

- Software modelling
 - ▮ Model OO programs, using UML
 - ▮ Model functional programs, using SSADM
- UML and SSADM are just modelling notations
 - ▮ they describe software designs, and they describe bad designs as well as they describe good ones
 - ▮ they do not *perform* 'design'

7

Modelling?

- Any way to describe a system
 - usually an abstraction (does not describe all details of the system, just those you are interested in)
 - may describe static and/or dynamic (time-varying evolution) features of system
 - purpose is often to make the system easier to comprehend
 - sometimes used to understand system behaviour before it has been built: static (e.g. finite element analysis of structures), dynamic (simulation, or analytical e.g. electrical circuit models, or for software - formal methods)
- Usually different in nature from real system
 - 'paper' or 'computer model' instead of 'bricks and mortar'
 - easier to explore alternative *designs*, but more...
 - different perspectives help us comprehend a system

8

E.g. Modelling of buildings

- Use blueprints (standard notation: paper/electronic diagrams)
- Modelling is mainly static rather than dynamic

9

Design?

- In general: deciding on system components and the relationships they have with each other (?my attempted defn)
- Software: deciding on system components and inter-component communications
 - ▮ design can be partially included in the requirements if a buyer wants the software to solve the problem in a particular way, or re-use components
- Design without build - requires modelling

10

E.g. Design of buildings

- A *Design* is a layout of rooms and other spaces, the connections between them (doors, hatches, stairs, lifts, windows), and the physical infrastructure to support the rooms and the services in them
- ...mainly static modelling
 - ▮ but a key point in the study of architecture is that static design strongly influences the way people can live and interact in a building (i.e. key dynamic properties of the system)

11

Unit objectives: II

- Once we can model OO programs, we will learn OO *Design Patterns* using the modelling notation
 - ▮ DPs are a newly developing field within software engineering
 - ▮ they describe designs that
 - ▮ 1. are regarded as good design practice
 - ▮ 2. can be used in many situations ('*design reuse*')

12

Next few lectures

■ Next 2 - introduction to OO concepts (for newcomers to them)

- *people familiar with OOP language may find these 2 lectures boring and prefer to just reference the notes*
- they are mainly aimed at people who have done C or some other functional language, but no OOP
- students with little programming experience will find this unit hard to say the least (please see me afterwards)

■ Then approx. 3-4 video lectures on UML and basic OOD

- by one of the main players in field of OOD

13

Lectures (cont.)

■ Then I introduce some examples, and a little more UML notation

- | there is **lots** of UML notation, but we will not need much of it
- | that doesn't mean the design issues we will study are easy!

■ After that - key design patterns

■ Then we move away from OO to SSADM

14