

(11224) INTRODUCTION TO SOFTWARE ENGINEERING

Lecture 8: Structured Design in UML Class Diagram

Shereen Fouad

Structured design

- Approach to design that:
 - starts with a problem description,
 - breaks it up into sub- problems,
 - then tackles each sub-problem **similarly**
until the problems are small enough that the solutions to each sub-problem is obvious and can be written down immediately.
- We are concerned with structured design in UML class diagram

Structured design

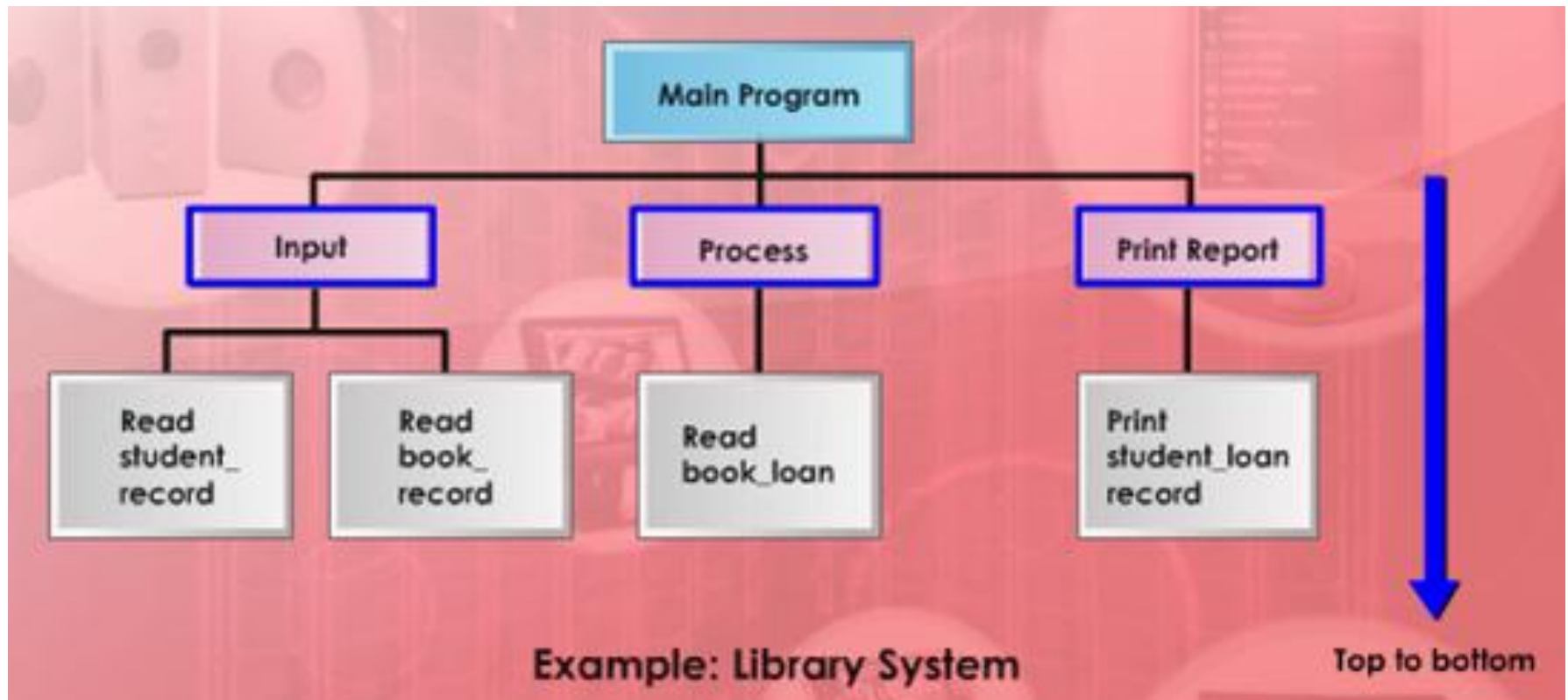
- Top-down structural design
- Bottom up design
- Inside-out structural design
- Mixed structural design



Top-down structural design

- It starts with a good global view of the required system, and builds a top level view of the design of the system before proceeding to lower level detailed views
- It proceeds step wise by applying purely top-down operations on elements of the diagram until all the requirements are adequately represented in the diagram.

Top-down structural design



Why Top-down ?








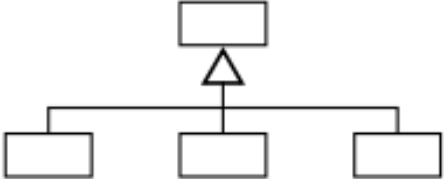






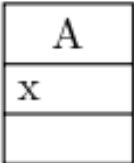
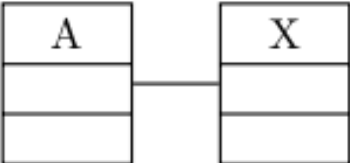
Why Top-down ?

- This is a very disciplined and well-organised approach,
- Typically leads to clean, well structured designs **if all the essential requirements are known**
- Well understood when the design process starts.
- This type of thinking allow a big group of people to work together.

W.r.t UML class diagram

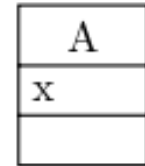
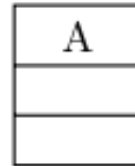
- First get the overall structure right (the main set of classes, generalisation hierarchies and simple associations),
- If the design is settled, refine them into the full set of classes with more details about the relationships,
- Finally fill in the last details about associations, attributes and operations.

Top-Down UML Class Diagram Structural Operations

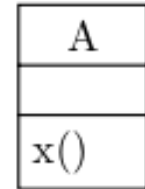
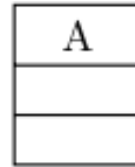
Operation	Starting Structure	Resulting Structure
TD1: Replace a class with two or more classes connected by an association		
TD2: Replace a class with a class with a recursive association		
TD3: Replace a class with an inheritance hierarchy		
TD4: Replace a class with two classes connected by a dependency		
TD5: Replace an association with a pair of associations		
TD6: Replace an association with a class associated to the classes at the ends of the original association		
TD7: Replace an attribute in a class with an association to a new class		

Top-Down UML Class Diagram Structural Operations

TD8: Add an attribute to a class



TD9: Add an operation to a class



TD10: Refine an association by adding directionality, cardinality constraint, or promoting it to a composition or an aggregation

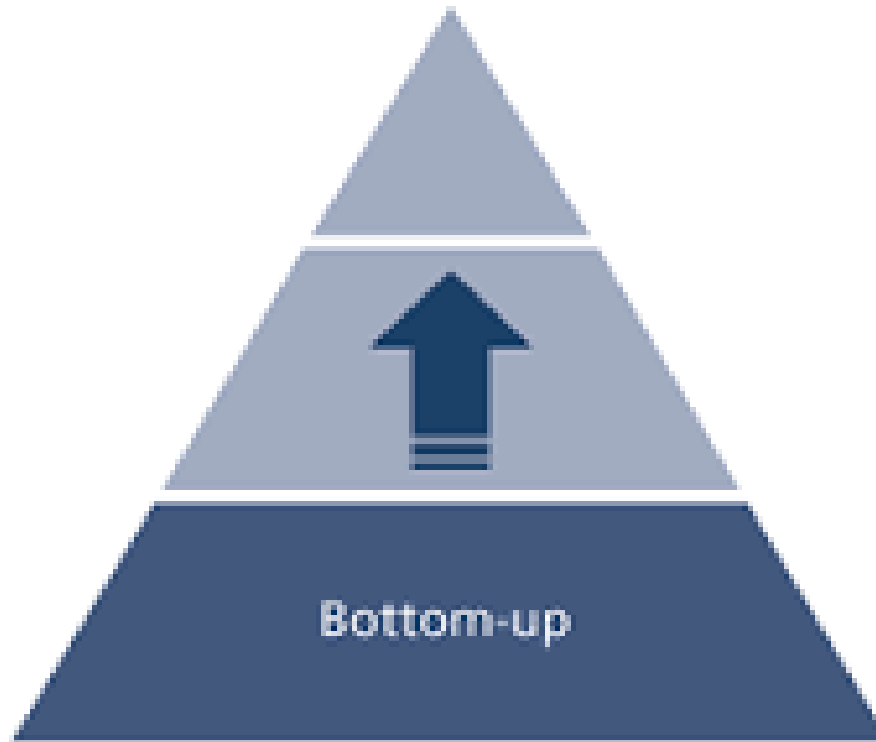


One of:



⋮

Bottom up design











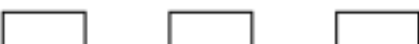
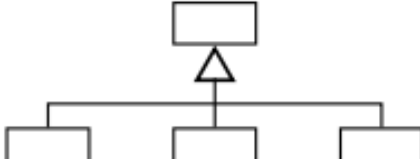
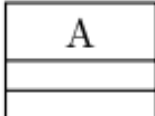
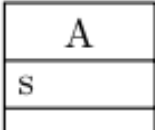
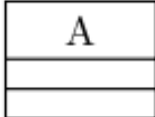
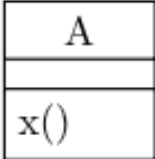
Bottom up design

- Starts by searching through the requirements to extract individual concepts,
- Individual concepts are then added to the diagram as free-standing attributes or operations
- Then bottom-up operations are applied to the diagram to implement local design decisions by
 - collecting related attributes and operations into classes, or
 - combining classes into associated classes and generalisation hierarchies.

Bottom-up design example



Building blocks are an example of bottom-up design because the parts are first created and then assembled without regard to how the parts will work in the assembly.

Operation	Starting Structure	Resulting Structure
BU1: Introduce an attribute		a
BU2: Introduce an operation		x()
BU3: Introduce a class		
BU4: Introduce an association, composition or aggregation between one or more classes (includes introducing recursive or n-ary associations)		One of:    
BU5: Introduce a dependency between a pair of classes		
BU6: Introduce a super class for a set of existing classes		
BU7: Move an existing attribute into a class	 s	
BU8: Move an existing operation into a class	 x()	

Why would anyone choose the Bottom-up approach?



Why would anyone choose the Bottom-up approach?

- This approach can be helpful when the requirements are poorly understood or poorly specified,
- One does not need the clear overview that top-down design needs in order to get started.
- This approach can be particularly useful as a requirements capture strategy

Disadvantage?



Disadvantage?

- One can get lost in the huge amount of detail that bottom-up design involves
- One only gets any real high level overview of the design when the design is nearly finished.
- Poses the risk that a major problem is discovered near the end of the process that cannot be solved locally but requires a major restructuring of the design or throwing away and redoing large parts of the design.

Which is better?



Which is better?

- It's sometimes good to alternate and mix between them.

Can I mix between Bottom-up and Top-down designs? How?



Can I mix between Bottom-up and Top-down designs? How?

- First use bottom-up design to try to build an initial design.
- In doing so, one identifies and fills in many details in the requirements that were previously missing or poorly specified.
- These details are recorded and added to the requirements.
- Finally, one can restart the design using a top-down approach but now with a much better understanding of the requirements.

Quiz !!

- **Product design is mainly ?**
 - a. Top-down approach
 - b. Bottom-up approach
 - c. None of the mentioned
- **Which of the following is not an area of concern in the design model?**
 - a. Architecture
 - b. Data design
 - c. Interfaces design
 - d. Project scope
- **In system design, we do the following?**
 - a. Hardware design after software.
 - b. Software design after hardware.
 - c. Parallel hardware and software design.
 - d. No hardware design is needed.

What about testing?



What about testing?

- With Top-Down it's harder to test early because parts needed may not have been designed yet
- With Bottom-Up, you may end up needing things different from how you built them

Inside-out structural design

- In large or complex designs and which have the kinds of issues that make top-down design less suitable,
- One can get lost in the huge amount of detail that bottom-up design involves.
- In such cases, the inside-out approach can help!

Inside-out structural design

1. Pick out a small number of concepts that appear central or particularly important.
2. Extract only attributes and operations related to those concepts and start working on the design of those.
3. As parts of the design develops, the design is expanded to bring in parts of the requirements that are closely related to the ones already dealt with.

Solving a puzzle

- The work proceeds in the Inside-out like solving a puzzle by
 - first find a corner,
 - then add pieces that connect to the corner,
 - then adding pieces that connect to pieces previously added,
 - and so on until the entire puzzle is solved.



Why?

- This approach has the same problems as bottom-up design, but rather than building many detailed design parts over the whole design, it works only in one place and grows its design from there.

Mixed structural design

- This approach is suitable for very large complex designs where at least a very high-level overview of the issues are clear, and especially when the design problem is too large for a single person to carry out alone.

How?

- We start with a top-down design,
- but only proceed far enough to partition the design problem, typically into major sub-systems.
- The result is a framework or skeleton design that shows how these sub-systems are related.
- Each sub-system is then designed independently,
- The sub-system designs can use top-down or bottom up approaches, as appropriate, but, in the end, all the sub-system designs need to be merged into one unified design.
- This is where the framework design comes into play.