Chapter 4 (part 1)

Designing Architecture

We discuss..

How the principal design decisions are made?

Overview

Design activities & approaches

Architecture Design Stages and strategies

Architectural Conception

- Abstraction
- Separation of concern; etc.

Past experiences in Design

- Domain specific software architecture
- Architectural pattern
- Architectural style

Designing Software Architecture

For many serious engineers and building architects, design is viewed as something that cannot be taught as a method.

Do you believe in this?

In fact:
As human beings we all have the ability to design.

Design, as a set of activities, may be taught, studied and analyzed.

Architecture Design Stages

The Design Process may be summed into 4 “stages”:

Feasibility : identifying potential design concepts

Preliminary design: selecting and developing the “best” design concept

Detail design: further defining and refining the design concept (providing technical description of the concept)

Planning: evaluating and altering (or tuning) the design concept to fit the requirements of software development, production, consumption, and retirement

Just like everything else, this is not all sequential - most likely iterative

Some Concerns over the Design “Process”

- **Start-up problem*:*** Identifying potential design concepts (the first step is very important & may be the key stumbling block).

- **People problem:** For a very large software system, there may be a need for more than one architect or chief designer - increasing the chance of not agreeing on any design concept.

- **Complexity problem**: for a system that is composed of many heterogeneous components or parts, the complexity of finding an overall system becomes very large and sometimes prohibitive.

Design Strategies

One may wonder around a while in the beginning to get started:

1. Looking and understanding at key requirements
2. mapping functional requirements to major components
3. relating the components to produce the overall system
4. Analyze the relationships in terms of non-functional required properties (e.g. performance, security, reliability, etc.)

As we proceed, we will revisit, refine, and modify the original ideas.

Design activities are <u>not always performed sequentially</u>:

- In cyclical mode:
- In parallel mode
- In incremental mode
- In adaptive mode

Design Strategies (Cont.)

Cyclical mode: As problems or infeasible approaches are identified in stages 2 to 4 of the standard model, the process reverts to an earlier stage.

Parallel mode: After stage 1 of the standard model, independent alternatives are explored in parallel; at suitable times selection is made between the identified viable alternatives

Adaptive mode: The design strategy to follow in the next stage of the design activity is decided at the end of a given stage, based upon insights gained during that stage.

Incremental mode: Design at each stage of development is treated as a task of incrementally improving whatever design or previous product exists after a preceding stage.

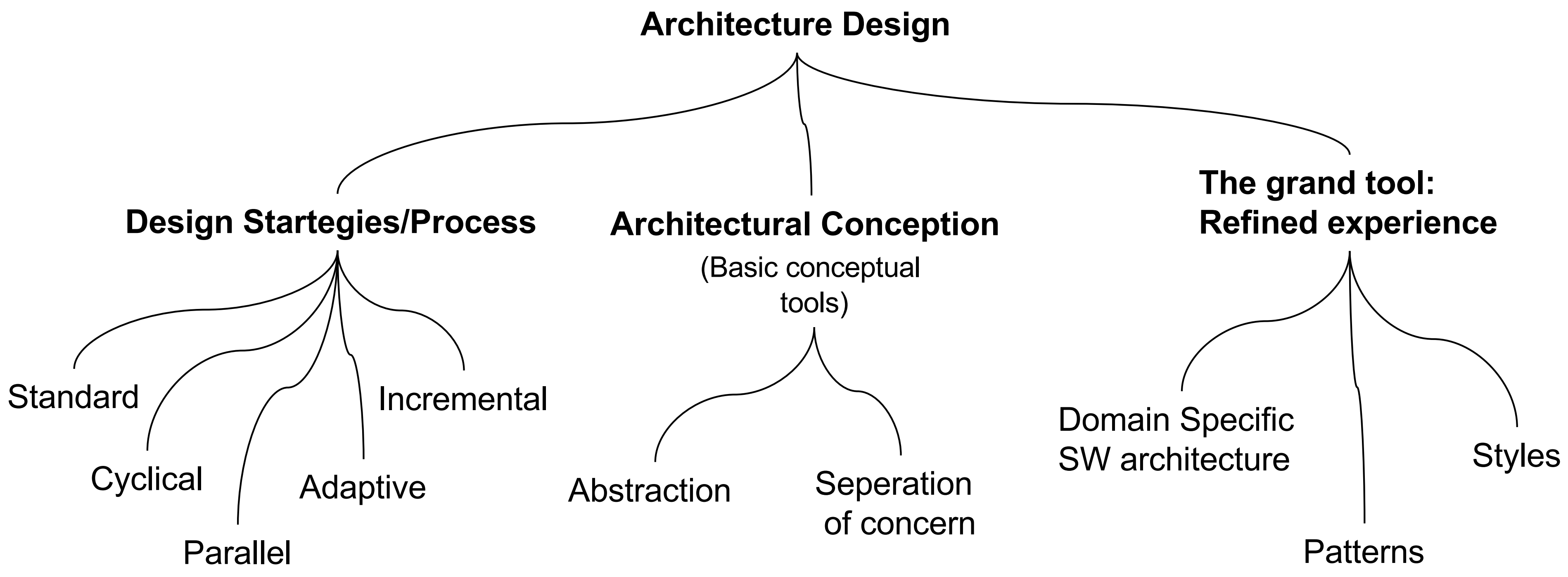
Critical First Step of the Design

What if the standard and alternative approaches don’t work?

Bsides the alternative strategies the question remains:

How to identify a candidate set of pheasible concepts?

The Big Picture



Architectural Conception

The standard approach does not tell, however, is ***how*** to identify that set (or one) of viable arrangements..

Some “Basic” Design Approach/Tool:

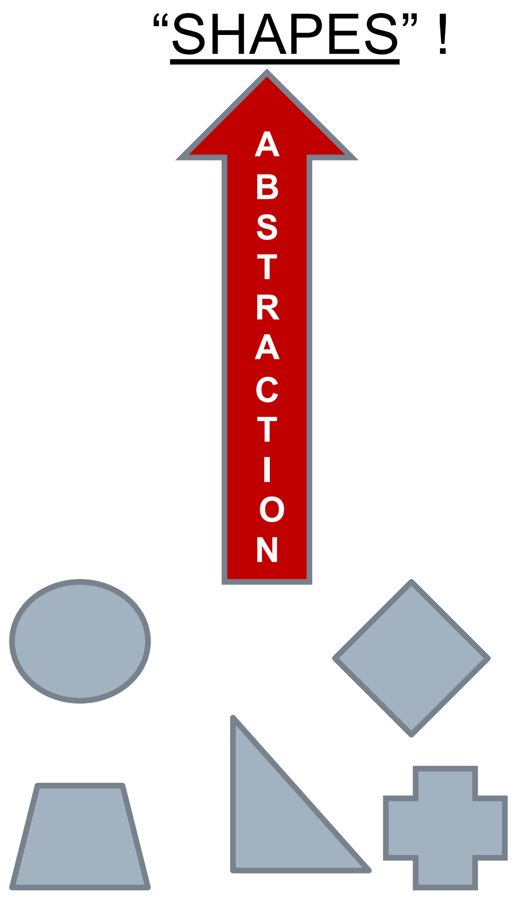
- Abstraction
- Separation of Concern
- Decomposition/Composition (Tsui terminology)

These concepts are not totally separate

Abstraction

- Abstraction may be defined in many ways:
  - “The selection of a set of basic concepts to represent a more complex whole” --- from your textbook
  - The simplification and ignoring of the details to emphasize what is considered as important.
  - The generalization from the details and the specifics

*As it relates to design, however, abstraction is usually employed as a tool to be used when moving **downward***



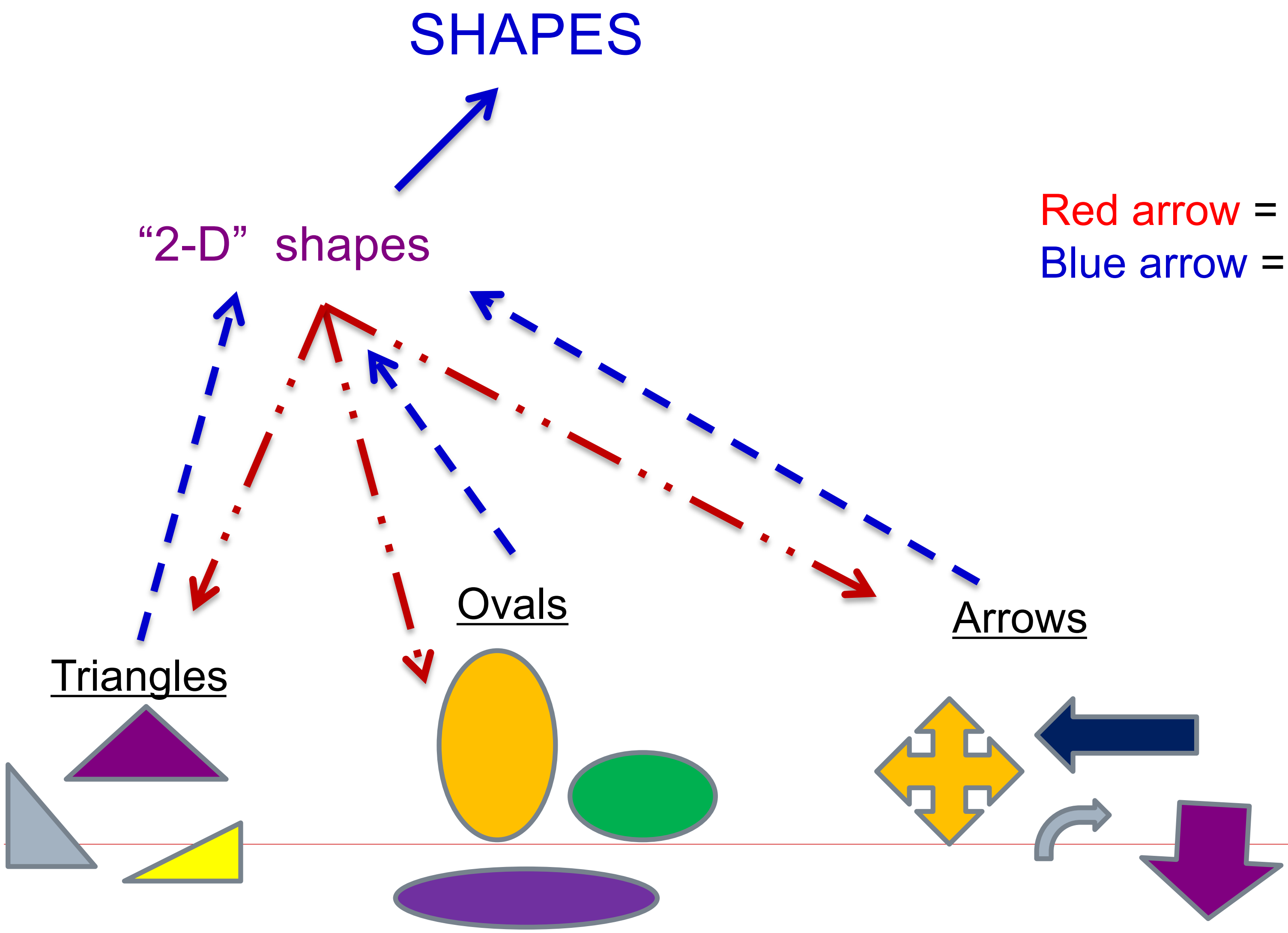
More on Abstraction

We often start by "matching" a problem component to some abstract model/solution:

- Major requirement (problem domain)
- Major solution (modeling "abstraction" or "machine")

Application Problem domain	**Simple Abstract "machine"**
word processing	Structured document & layout
process control	finite state machine
financial & accounting	spreadsheet & database & transactions
scientific computing web pages	matrices and algorithms HTML based document
etc.	etc.

Levels of Abstraction
Decomposition & Composition



Red arrow = decomposition
Blue arrow = composition

Approaches to Levels of Abstraction

- Work at a level below the “whole” component or application.
  - we are looking at solutions to sub-components and then compose & generalize the solutions to apply to the whole
- work at a level above the “whole” or desired application
  - we are looking at a more general solution and applying only the specific parts of the general solution to the problem at hand and fine tune to specialize the solution .

Separation of Concern

Separation of Concern is the subdivision or decomposing of a problem/solution into “independent” parts.

Separation of system’s structure into components:

- Functional components and sub-components
- Connectors that “interconnect” components

Most of the time *total independence is not possible*, but we want:

- Each individual component to have strong cohesion
- The components to have loose coupling

Recall that cohesion and coupling are important software properties.

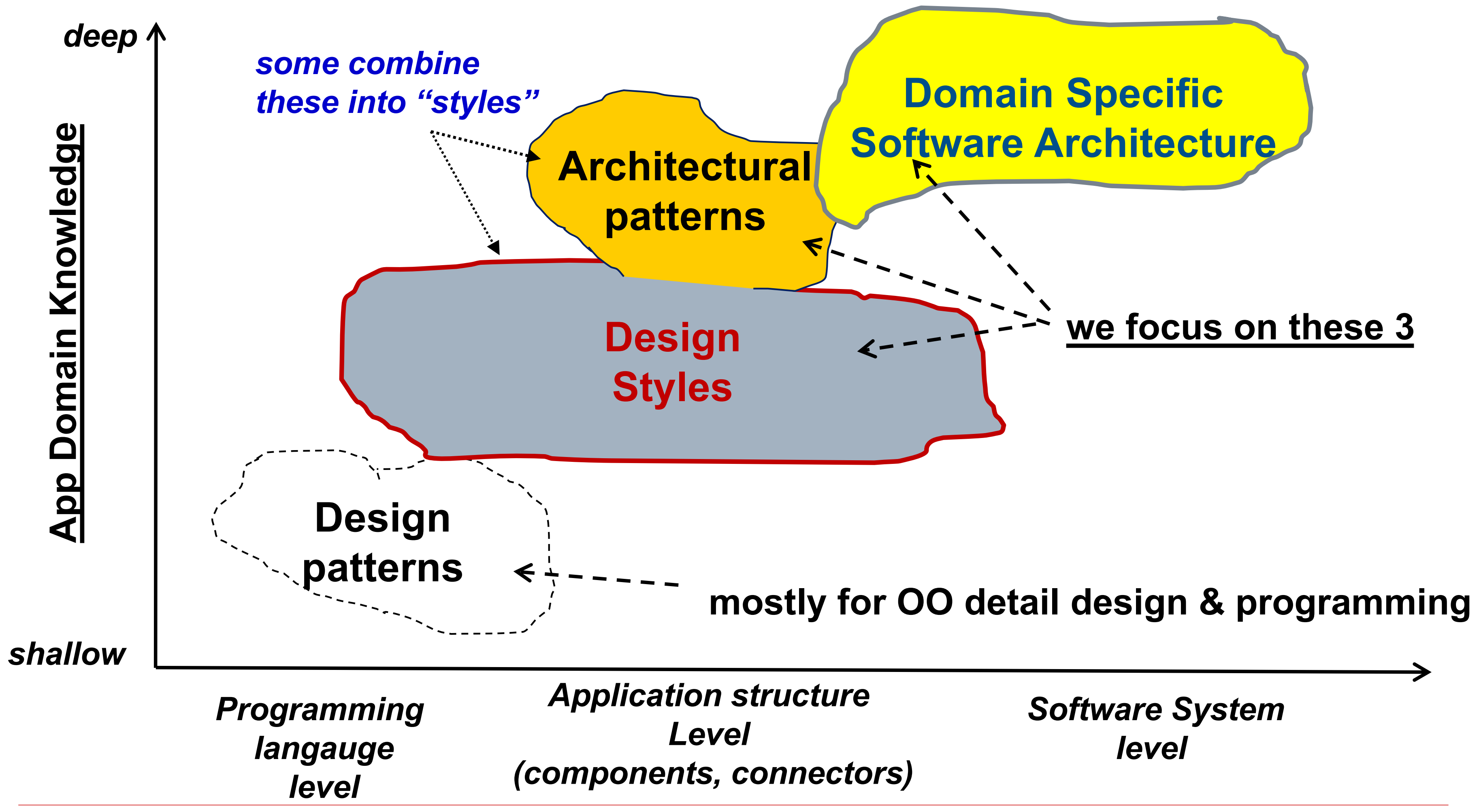
Any Other Help in Designing Architecture?

Just as in other fields, past experiences provide fertile ground for potential solutions:

- to satisfy a specific goal
- within some particular application context

In software architecture we may also use these past experiences, expressed in the form of architectural “**styles**” and “**patterns**”

A Broad View of Styles & Patterns

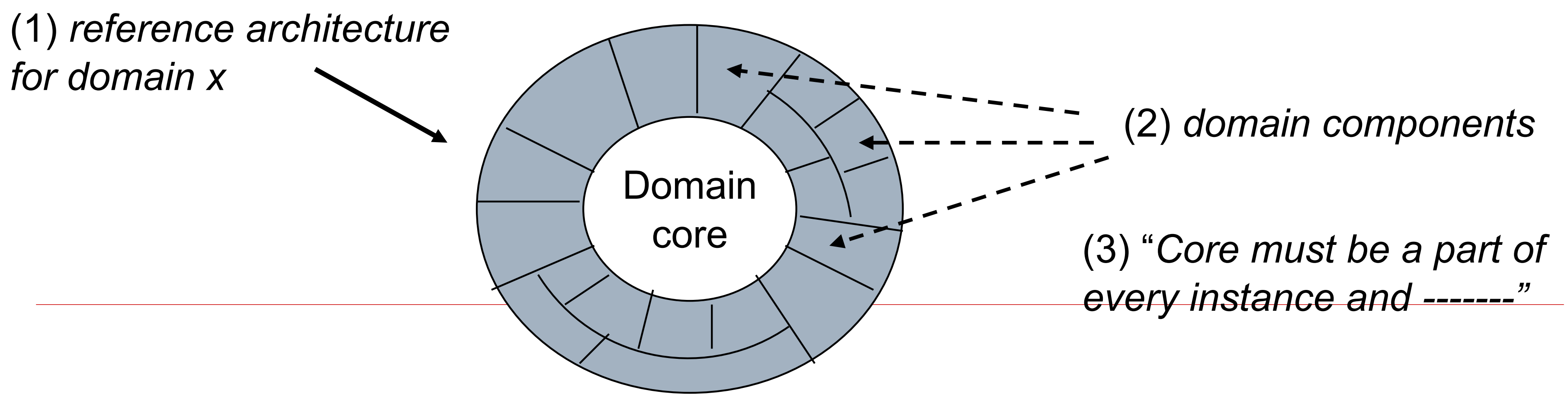

deep
shallow
App Domain Knowledge
some combine these into "styles"
Architectural patterns
Domain Specific Software Architecture
Design Styles
we focus on these 3
Design patterns
mostly for OO detail design & programming
Programming langauge level
Application structure Level (components, connectors)
Software System level

Scope of Applicability of knowledege

Domain Specific Software Architecture (DSSA)

Domain Specific Software Architecture contains a substantial amount of information based on past experiences within the specific application domain. It's a combination of:

1. A reference architecture for a specific domain
2. A library of software components containing reusable chunks of domain expertise
3. A method of choosing and configuring components to work within an instance of the reference architecture.


(1) *reference architecture for domain x*
Domain core
(2) *domain components*
(3) "*Core must be a part of every instance and -------*"

Try during the class yourself

How would you approach “abstracting” the description & design of a system for the following?

- Online classes
- In class lectures
- Part time students
- Adult-students
- Fulltime students
- Full time lecturers
- Part time lecturers