

# CSC444 - Software Engineering I

Week 4-2

## Design

- Characteristics
  - A problem solving activity
  - Creative
  - Requires trial and error
  - No correct solution: good or bad solutions
  - No clear cut start or end

## ... cont'd

- Complexity necessitate abstraction
  - decompose into parts (modules) of lower complexity
  - many different decompositions
  - functional, data, ... decompositions
- Create a technical specification
  - various notations
  - used for communication

## Evaluating a design

- Completeness
  - does it capture all the requirements in the SRS?
- Easy to understand
  - modularity:
    - high degrees of cohesion
    - low degrees of coupling
  - complexity:
    - size, McCabe, Halstead

## Comparison

- Functional abstraction
  - Easy to grasp
  - Difficult to change data representation
  - Difficult to change algorithms or functionalities
- Data abstraction
  - Not as easy to identify data
  - Easy to change data representation
  - Easy to change algorithms, difficult for functionalities

## Software architecture

- Natural evolution of design abstractions
  - design patterns
  - architectural patterns
- Abstract, very high level, box and arrow diagrams
- Effective means of communication
- Define a system in terms of its computational components and their interactions
  - components: client, server, database, filter
  - interactions: function call, shared variables, protocol

... cont'd

- Captures design decisions
- Improves communication
  - division of work

## Architectural styles

- Architecture is a formal arrangement of architectural element, e.g. your house, that house, the house down the road
- An architectural style abstracts from the specifics of an architecture, e.g. bungalow, semi-detached, two-story
- A style describes a certain codification of elements and their arrangement.

## Describing style

- **Problem**
  - a description of the type of problem it addresses
- **Context**
  - imposing requirements, e.g. Unix and pipe and filter
- **Solution**
  - components and connectors, control structure,
- **Variants**
  - variations, specialization
- **Examples**

## Pipes and filters

- **Problem:** sequential transformations on data
- **Context:** uses OS to exchange data, error handling is difficult
- **Solution:**
  - System model: continuous data flow between components
  - Components: filters that perform local processing
  - Connectors: data streams in ASCII
  - Control structure: each component has its own thread
- **Variants:** filters consume all their input before producing any output

