**What’s a “Component”?**


  "A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties."

  - Purpose: to be composed (collaborated) with other components (units), likely by non-creator of the component ("third parties")
  - Responsibility: "contractually" "interfaces"
  - Properties: self-contained ("can be deployed independently")
  - Attributes / dimensions:

**Experiment Principles**


  **Experiment objective**

  **Theory**

  - Cause construct
  - Effect construct

  **Observation**

  - Treatment
  - Outcome

  **Independent variable**
  **Dependent variable**

  **Experiment operation**
Level of Control


Independent variables:
- Objects (UNIX vs. OO)
- Subjects (undergrad vs. grad)
- Tasks (WC vs. LS)
- Methods (top-down vs. bottom-up)

Treatment

Process

Independent variables with fixed levels

Architectural Styles

- Design and construction of buildings for human use
  - Classify architecture in terms of form, techniques, materials, time period, region, culture, etc.
  - Emphasize characteristic features of design
  - Leading to a terminology such as Gothic "style"

Software Architectural Styles

- A program (computing system): standard terminology
  - Why: early architectural decision is hard to change
  - Structure: component
  - Relationship: connector
  - Behavior: externally visible properties

Documenting Software Architectures

- Philippe Kruchten's 4+1 view model [IEEE SW, 1995]
  - Logical view, e.g., Class Diagram, Sequence Diagram
  - Development view, e.g., Component Diagram, Package Diagram
  - Process view, e.g., Activity Diagram, Statechart
  - Physical view, e.g., Deployment Diagram
  - Scenarios, e.g., Use Case Diagram

- Other views
  - Module, Functional, Use Action, Data Flow, Control Flow...

- A generic architecture of the chosen domain
  - Course project progress
Summary

→ Design & Architecture Reuse
  ☐ Components
  → Made to be composed (collaborated) with other components & (re-)used by others
  ☐ Design Patterns
  → GoF book, "Design Patterns: Elements of Reusable Object-Oriented Software"
  ☐ Software Architecture
  → Characterize a family of styles/systems

→ To-do:
  ☐ Project
    → Stay focused
    → Share your domain/topic with us next Monday (informally)
  ☐ Read "libraries" paper