The Acyclic Dependency Principle

No cycle in the component dependency graph

- A cycle in the component dependency graph
  - No way to decide the order of building components
  - Very hard to isolate components, unit-test and release
- Break the cycle
  - Apply the Dependency Inversion Principle (DIP)
  - Create a new abstract component
- Component structure cannot be designed from the top down
  - Different from functional decomposition (top-down design)

The Stable Dependency Principle

Measure the stability of a component

- Fan-in: incoming dependencies
  - the # of classes outside the components that depend on the classes inside
- Fan-out: outgoing dependencies
  - the # of classes inside the components that depend on the classes outside
- I (Instability) = Fan-out / (Fan-in + Fan-out)
  - I = 0: responsible and independent; most stable and very hard to change
    - e.g., abstract component (all interfaces or abstract classes)
  - I = 1: irresponsible and dependent; no reason not to change
    - e.g., component with main function