Reusability

Goals of Reusability

- Expected benefits
  - » **Timeliness – less software to develop, hence develop faster**
  - » **Decreased maintenance effort – leave evolution of reused software to others**
  - » **Reliability – the more software components are reused the more reliable they become through error correction**
  - » **Efficiency – the best algorithms and the best data structures are used**
  - » **Consistency – have more regular, coherent design**
  - » **Investment – preserves know-how and inventions of the best developers**

Reuse Path Principle
Be a reuse customer before you try to be a reuse producer

What Should We Reuse

- People – the developers themselves
- Designs & Specifications
- Design Patterns
- Program source text
- Abstracted modules

Non-technical Issues

- Not invented here
- Economics of procurement
- Software companies and their strategies
- Accessing components
- Formats for component distribution

Technical Problems

- Change and constancy
- Reuse–Redo dilemma

Requirements on Module Structures

- Type variation
  - » **Same algorithms work on different data types**
- Routine grouping
  - » **Which routines should be grouped as a coherent package**
- Implementation variation
  - » **Different data structures require different implementations**
- Representation independence
  - » **Be able to use a component without knowing how it is implemented**
- Factoring out common behaviours
  - » **Extract the general algorithms and leave the details to be filled in later**