Components: How Far Can We Go?

Noah Mendelsohn
Lotus Development Corp.
October 8, 1997
Objects and Components

OBJECT:
A thing

COMPONENT:
A thing designed to be used as part of other things
Consider...

- What makes a good component?
- How much software can we/should we componentize?
- Which things shouldn't be components?
Some Observations about Mechanical and Software Components
Special Purpose vs. General Purpose

- Most real-world components are special purpose (look around!)
- Special purpose parts can still be components...why:
  - Generic tools & assembly techniques
  - Repair, replacement, versioning
  - Multiple instances, one application
  - Structure most software as components
General Purpose Components

- General purpose parts exceptionally valuable.

- Examples:
  - Utility items (screws, bolts, wires, UI widgets, DB access parts, etc.)
  - Systems of components (pipes & fittings, electrical conduit, etc.)

- Standardization & specification

- Try to create general purpose components
The interface is bigger than you think

- Explicit API: what you think you're doing
- Implicit API: everything else that breaks if you get it wrong

  ▪ Error Handling
  ▪ Memory Usage,
  ▪ ....I won't close the file
  ▪ Asynchrony
  ▪ etc.

- The answer: carefully specify explicit and implicit API
Interchangeability vs. Refinement

- **Boeing 757**
  - Cabin: almost all parts are special purpose
  - Under the skin: wires, screws, clamps, etc.

- Seamless user interfaces demand refined componentry. Therefore...

- ...end user components less interchangeable than hidden components
Raw materials vs. components

► Raw materials:
  ▪ Cloth
  ▪ Wood (finished or unfinished)
  ▪ Leather
  ▪ Metal
  ▪ etc.

► Software "raw materials":
  ▪ Tailorable components
  ▪ Filters
  ▪ Delegates
How big is the "nerve bundle"?
How big is the "nerve bundle"?

- Spell checker UI component has complex relationship to WordPro application
  - UI Integration
  - Asynchrony
  - Etc.
- Dictionary has simple interface: easy to replace
- Conclusion: look for the narrow "nerve bundles"
There are (should be) more components than you think

- Functional decomposition
  - Spell check UI vs. Dictionary
  - Example: HTML Rendering vs. Web Browsing
- Model/view separation
- Business object/business logic
- Where appropriate, subdivide your componentry
How far can we go?

- Should all software be componentry?

- What granularity?
  - Every class a component?
  - Packaging
  - Registration
  - Instantiation overhead
  - Component vs. object

- Package as componentry in cases where reuse and generic mgmt. justifies the cost