Reuse

(Assume have source code, not a commercial product)

- Ariane 5, Therac-25, British ATC, ...
- Expectation:
  - Significantly lower development costs and time. Amortize costs among all users or uses.
- Assumptions:
  - Will be reused enough to recoup extra costs
  - Can easily and cheaply integrate components into a new environment (interoperability).

Reuse: Empirical Data

- High reuse in some limited environments, not widespread however.
- NASA Goddard Study
- Garlan, Allen, Ockerbloom:
  - Performance problems (from large size and complexity) Complexity frequently inappropriate for tasks performed.
  - Trouble fitting components together.

    In some cases, took significant reengineering to make the interoperate properly.
  - Maintaining synthesized system difficult in absence of low-level understanding.

Reuse: Empirical Data (2)

- Siemens (hardware ASICs) reuse study:
  - Time to build a reusable component can be 120-150% of time needed to develop component for single use (excludes documentation).
  - For reusuable component, needed to develop new documentation -- took one to two times the effort of designing the component.
  - Overhead to develop a reusable component (design + doc) recaptured after fifth use.
  - Frequency of reuse increases with degree of comprehensibility.
  - Habitability even more important: measure of how "at home" a potential user of reusable component feels. Highly subjective but effect even greater than that of comprehensibility.

Reuse: Technical Issues

- Configuration control problems

  May change continuously (any developer may be able to check out and change).

- Unneeded functionality (interoperability and performance issues)

  May need to write software utilities (restrictive wrappers) to restrict functionality.

- Much of savings may be offset by need for more testing (Weyuker)
- Debugging may be significantly more difficult (Weyuker)
- Longevity -- does ppotential for reuse decrease over time?
- Reuse designs rather than finished products?

Reuse: Other

- Management issues (e.g., reward structures)
- Platforms and reuse at Xerox (hopefully, one of the Xerox students can tell us about this).
  - One organization within Xerox reports they use half dozen different software platforms to build half dozen different products.
  - Achieve approximately 80-90% reuse
- Other comments or experiences?

COTS

- Main difference from reuse is lack of source code
- Potential advantages:
  - Reduce front-end acquisition or development costs.

    Amortize costs over large number of users.

    Compensate for lack of expertise (Shelley Hayes)
  - Allow for more rapid infusion of technology
- But new risk drivers
  - Loss of market control (less control leads to higher risks)
  - High speed market

    Must deal with rapid obsolescence (shortened lifetimes)

    New versions or releases brought to market frequently.
  - For government, shift from "buyer’s market" to "seller’s market"

COTS: Management Issues

- Lower development costs offset by higher lifetime costs?
  - "Sustainment" costs substantial -- need to be planned and managed.
- What if vendor goes out of business or stops producing and will not maintain old versions?
  - Even if escrow agreements, hard to maintain software you did not write and must hire developers expert in that code.
- Dependency on vendor. Can charge anything or make other demands (e.g., Microsoft case findings of fact).
- Higher speed of change requires greater strategic flexibility.
  - Requires flexible and proaction system evolution management.

COTS: Technical Issues

- Functionality provided may not remain what you need over time.

- Few parts in a software system truly independent.
  - May need wrappers and patches as substitutes for real source-code-based maintenance.
  - Differences (e.g., timing) may be introduced in new products

- What happens when support from COTS vendor ceases?

  Can user change requests be satisfied?

- May be Trojan horses or security flaws in COTS software and almost certainly will not know until too late.

COTS: Technical Issues (2)

- Must be accepted "as is" and may not satisfy user requirements. Compromises may be required.

- May be difficult or impossible to certify (safety).
  - Need to defend yourself from mistakes in supplier’s code.
  - Developed to commercial not government or safety standards.

- Requires continuous lifetime system engineering effort.
  - Identify and integrate product obsolescence information with technology trends and new user requirements.

Your experiences and comments: