Architectural Implications of Common Operator Interfaces

Richard N. Taylor
Nenad Medvidovic
Peyman Oreizy

Information and Computer Science
University of California, Irvine 92697-3425
+1-714-824-6429
http://www.ics.uci.edu/~taylor/

© 1998, Richard N. Taylor
COIs are a Good Thing

- Reduced operator training costs
- Reduced operator error rates
- Reduced maintenance costs
- Higher reliability software

But how much are you willing to give up in return?
COIs and Platform Dependence

• What’s the relationship between
  » the application and the user interface code?
    · Calls entangled in application code?
    · Structure of application code determined by the UI software?
  » the user interface code and the toolkit(s)
  » the toolkits and the operating system/platform?
    · Is only a single platform allowed?
Two Visions of the Meaning of COIs

• Microsoft’s approach
• Galaxy
• Same problem with both: a lifetime commitment to a proprietary platform
How UI Software Determines Application Structure

Typical assumption of only a single thread of control

“Application” code executed as the result of a callback from the UI event monitor loop
Evolution Issues

• Changes to existing screens
• Changes to system functionality
• Additions of new screens
• Dynamic evolution
Our Priorities

• Focus of ground system design should be at the architecture level
• Architectures should be chosen with the full range of domain characteristics and evolution issues in mind
  » Concurrency: telemetry processing; user requests
  » Physical distribution
  » Heterogeneous components and platforms
  » Event-based
  » Frequent evolution

• Implementation technologies should not constrain the architecture
• Therefore: the technologies for providing COIs should not determine system architecture; rather architecture and goals should determine the UI technologies
Implicit Invocation

One key mechanism for achieving independence

• Many realizations and variations of the idea
• Smalltalk’s Model-View-Controller
• Softbench/Tooltalk
• Java AWT 1.1 listener model
• C2 Architectural Style
The C2 Architectural Style

• A component- and message-based architectural style
• C2 architectures are networks of concurrent components hooked together by connectors

- no component-to-component links
- specific topology rule
- all communication by exchanging messages
- substrate independence
The Cargo Router Example