Using CORBA Messaging, Real-Time Event Service and ORB Concurrency Models in a Real-Time Embedded System

Bruce Trask
Contact Systems
50 Miry Brook Rd
Danbury, CT 06810
btrask@contactsystems.com
Surface Mount Technology (SMT) Automated Assembly Equipment

Machines that combine high speed and precise positioning and placement of devices onto printed circuit boards
Under the hood:
A Closer Look
The Cart Subsystem
Concurrent Tasks

- Motion Control
- Vision
- Update
- Feeder Actuation
- GUI
SMT Assembly Equipment Industry Dynamics

• Increased Global Competition
• Increased Performance Requirements from Customers
• Integrated Manufacturing Environment
• Shorter Time-to-Market Windows
Reasons for “Going Distributed”

• Separation of concerns
• Increased performance requirements
• Modularity requirements
• Third party software platform requirements
CORBA Features Used:

- Two-way Synchronous Method Invocations
- Naming Service
- CORBA Messaging (AMI and Client-Side Timeouts)
- Real-Time Event Service
- COSEvent Service
- Interoperable functionality with a Java ORB
- Advanced use of the POA
- ORB-Controlled Concurrency Model
Two-way Synchronous Method Invocations

• The problem we are trying to solve
• Simplification of programming for non-networking programmers
• object->operation() model key for fast development – very well received by programmers new to distributed systems
• Selective Ignorance with Object References – pointers on steroids
• Keep effort on the problem domain of picking and placing parts, not on the solution domain.
• Program the machine as if it were one system.
• CORBA implementation meets the real-time constraints of out problem domain.
Naming Service

• The problem we are trying to solve
• Primary bootstrap mechanism for all parts of the machine.
• Used in conjunction with DHCP server
CORBA Messaging (AMI)

- The problem domain – maps directly to “fire and forget” messaging model. Vision System, Feeder System, Cart System
- Need response but not right now
- Decoupled invocation from execution simplifies application programming. Obviates the need for the application to supply the mechanism. Simpler compared with application level solutions.
- Works well in conjunction with concurrency models on the server side – Thread-Pool Reactor
- Callback model obviates polling.

Performance

- Choice of AMI strategies maps to our problem domain
  - Servant-per-AMI-call strategy (i.e. use a different reply handler for each request)
  - Activation-per-AMI strategy (i.e. use the POA dynamic activation mechanisms to distinguish between all requests)
  - Server differentiated-reply strategy (e.g. return some kind of request ID from the server)
CORBA Messaging (QoS Framework)

• The problem domain – maps directly
• Parts of the machine become disconnected regularly and routinely in the course of running the machine.
• Once again, pushed the responsibility onto the ORB infrastructure and thus off of the application code. Simpler faster development
• Localize timeouts only to the areas that require it – ORB, Thread and Object level policies.
• More control over distributed system idiosyncrasies
Real-Time Event Service

• The problem domain – smart distributed observer pattern needed between Cart, Controller, Host Systems
• More control over event notification – using subscription and filtering
• Certain performance-critical systems cannot be bothered with all events
• Handles ill-behaved suppliers and consumers

• Benefits
  - More Efficient Event Management
  - Less Network Traffic
  - Simpler Application Programming
  - Decreased Consumer Load
Use of C++ ORB with Java ORB

- Allows powerful, responsive user interfaces for embedded systems with no keyboard or display
- Embedded system microweb server serves back Java applet which thereafter communicates with server via CORBA calls.
- Interoperable with the main C++ ORB framework of the machine.
- Aid in development/testing
- Used by tech support
- Used by manufacturing personnel for monitoring.
- COSEvent Service
Portable Object Adapter

- The problem domain – Cart example
- Use of servant activators simplifies application programming while economizing resources
- Hierarchy of POAs
- Reference-counted servants
ORB Concurrency Model

• The Problem Domain – The cart as an example
• Thread-Pool Reactor
• Simplification of programming at the application layer – borrow multiple threads from the ORB infrastructure upcalls. Simpler compared with alternatives without it.
• In combination with client-side AMI, ORB concurrency model on the server side gives us a complete working-out-of-the-box framework that solves a large portion of our programming concerns.
Conclusion

• Benefits of intuitive mapping onto the object->operation() paradigm.
• Fast Development – use of CORBA services
• CORBA Messaging directly maps to problem domain
• Problem at hand is one of application-problem-domain programming combined with integration of ORB enabled subsystems. Much easier than reinventing it all.
• CORBA meets our real-time and performance requirements while providing many other benefits
• CORBA Middleware vital for our real-time embedded application.