The Component Interaction Domain: Modeling Event-Driven and Demand-Driven Applications

Xiaojun Liu
Yang Zhao

5th Biennial Ptolemy Miniconference
Berkeley, CA, May 9, 2003

Outline

- Interaction between Software Components
- Software Framework Examples
  - The CORBA Event Service
  - The Click Modular Router
- The Component Interaction (CI) Domain
- Demonstration
- Current Status and Future Work
Aspects of Component Interaction

• Making sense of the information exchanged
  - E.g. a pure event notification, or an array of floating-point numbers
  - Managed by the type system
• The communication protocol
  - E.g. rendezvous, or read from/write to a FIFO queue
  - Managed by a model of computation

Aspects of Component Interaction

• Initiation
  - Which participant initiates the interaction?
  - What components can initiate interaction?
• Control flow
  - How do interacting components obtain the execution context?
• Timing...
The CORBA Event Service

- Support asynchronous, decoupled communication between objects

```
- PushConsumer
  +push(data: in any)
  +disconnect_push_consumer()

- PushSupplier
  +disconnect_push_supplier()

- PullConsumer
  +disconnect_pull_consumer()
  +pull(): any
  +try_pull(has_event: out boolean): any

- ProxyPushConsumer
  +connect_push_supplier(push_supplier: PushSupplier)

- ProxyPullSupplier
  +connect_pull_consumer(pull_consumer: PullConsumer)

- ProxyPullConsumer
  +connect_pull_supplier(pull_supplier: PullSupplier)

- ProxyPushSupplier
  +connect_push_consumer(push_consumer: PushConsumer)

- ConsumerAdmin
  +obtain_push_supplier(): ProxyPushSupplier
  +obtain_pull_supplier(): ProxyPullSupplier

- SupplierAdmin
  +obtain_push_consumer(): ProxyPushConsumer
  +obtain_pull_consumer(): ProxyPullConsumer

- EventChannel
  +for_consumers(): ConsumerAdmin
  +for_suppliers(): SupplierAdmin
```

The Push Model

- The supplier initiates the transfer of data
- The consumer reacts to supplied data
- The event channel provides the execution context for the consumer
The Pull Model

- The consumer initiates the transfer of data
- The supplier reacts to demand of data
- The event channel provides the execution context for the supplier

Mixed Models

- Push supplier and pull consumer
  - The event channel acts as an event queue

- Pull supplier and push consumer
  - The event channel must act as an active mediator for any interaction to take place
The Click Modular Router

- Elements are C++ objects that interact by making method calls
- Packets are passed as method argument or return value
- Flexible, configurable, and highly efficient routers have been built with this software architecture

The Component Interaction Domain

- Push/pull interaction
- Active/passive actors
- Multi-threaded execution

Push/Pull Interaction

- Push model
  - Event-driven applications

- Pull model
  - Demand-driven applications
Active Actors

• Active actors initiate all computation in the model
  - Source actors with push output
  - Sink actors with pull input
  - Actors with pull input and push output

Passive Actors

• Passive actors react to pushed event or pull demand
Multi-Threaded Execution

- Each active actor has its own execution thread
- Passive actors share the same execution thread as the director

A Router Model in the CI Domain

This model illustrates queue management using push and pull style communication in distributed systems. The "Distributor" actor will scale packets to the lone receivers. It monitors the state of the queues to help it make decisions.
**CI vs. the Click Router Architecture**

- Click is a specialized, highly efficient, runtime software architecture based on object-oriented programming. Router elements interact by making method calls. The control flow is fixed once the router is built.
- The CI domain provides a more general model architecture based on actor-oriented programming. Actor interaction is mediated by the director, so is more decoupled. The director (or code generator) can partition a model and choose a different analysis/scheduling strategy for each part.

**Current Status and Future Work**

- **Current status**
  - A preliminary implementation is included in the current release
- **Future work**
  - Design the infrastructure for building actors with mixed push/pull input ports and output ports
  - Partitioning of CI models for scheduling and allocating execution threads
  - Explore the relation to dynamic dataflow that uses both data-driven and demand-driven execution strategies