Concurrency Seminar 2004

An Extensible Binding Framework for Component-Based Middleware

By Nikos Parleventas, Geoff Coulson, and Gordon S. Blair

Presented by Erdal Nci

Unpacking the title (1)

- "An Extensible Binding Framework for Component-Based Middleware"
  - process of associating or interconnecting different objects
  - a binding type (BT) is a service description that supports a particular pattern of interaction between application components
    - e.g. RMI, publish/subscribe, message passing and eventing, A/V streaming, ...

Unpacking the title (2)

- "An Extensible Binding Framework for Component-Based Middleware"
  - components
  - reusable deployment units of software
  - component framework (CF)
    - architecture
    - collection of rules and interfaces that govern the interaction of a set of components plugged into it
    - contracts
    - reusable software architecture with runtime plugins
    - environment of well defined architectural properties and invariants for plugins
Unpacking the title (3)

A binding framework for component-based middleware is therefore a framework for plug-in binding types.

Overview

1. Motivation
   2. Architecture
   3. Conceptual Binding Model
   4. How to add a new BT?
      - BT specification
      - The Binding Contract
   5. Conclusion

Motivation

- Support
  - current middleware platforms support only a small, predefined, set of fundamental binding types
  - e.g. in CORBA: RMI, media streams, event handling
  - other possibilities: groups, comm, shared spaces, ...

- Integration
  - different APIs are a lack of integration
    - leads to missed opportunities for design and code reuse
    - increases the cognitive load on middleware users
    - leads to problems in realizing globally-coordinated QoS across binding types
Overview

1. Motivation
2. Architecture
   3. Conceptual Binding Model
   4. How to add a new BT?
      ➢ BT specification
      ➢ The Binding Contract
3. Conclusion

Architecture

Overview

1. Motivation
2. Architecture
3. Conceptual Binding Model
   4. How to add a new BT?
      ➢ BT specification
      ➢ The Binding Contract
3. Conclusion
Conceptual Binding Model (1)
- bindings between participants
- responsible for binding establishment
- binder verifies that participants conform to participant roles, defined in the BT specification
- binder invokes appropriate operations on participant components and establishes binding
- return BindingCtl for binding control & management

Conceptual Binding Model (2)
- remote participants are represented as reps (remote participant representatives)
- generator creates a ref (value that can be passed around)
- ref is transferred to binder's side and passed to a resolver
- resolver creates rep using ref, can be passed to binder
- APU = anonymous participant user if binding initiator is not explicitly represented

Conceptual Binding Model (3)
- StdGenerator
  - special generator implementation
  - provided by the framework by default
  - used to marshal arbitrary component references
  - e.g., pointers to component instances
  - necessary because component model presumes o-o interfaces -> component references may need to be passed as arguments
- StdResolver
  - dispatch given ref to the appropriate resolver (based on the BT identifier from the ref), invoke resolver and return resulting rep
Overview

1. Motivation
2. Architecture
3. Conceptual Binding Model

4. How to add a new BT?
   - BT specification
   - The Binding Contract

5. Conclusion
Publish/Subscribe Example (1)

Specifying Binding Types (1)
- BT as a set of four `collaborations`
  - Binding Participation
    - describes interaction among binding participants
    - includes definition of participant roles
    - 3 roles: publisher, subscriber, channel
    - publisher invokes an operation on event interface
      - corresponding operation gets invoked on all subscribers
    - synchronous communication
    - at read-time semantics
  - IRef Generation and Resolution
    - describes the process of managing irefs
    - Channel: IRef Generate (event_interface)
    - Subscriber: IRef Generate (subscriber_interface)
    - use Std Resolver to get irf

Specifying Binding Types (2)
- BT as a set of four `collaborations`
  - Binding Establishment
    - describes sequence of actions to set-up a binding
    - Publisher: pub Bind (rep)
    - Subscriber: bindingCtl Bind (rep, subscriber)
  - Binding Control and Management
    - describes process of managing an already-established binding (monitoring, controlling, adding/removing participants, ...)
    - via bindingCtl or pub
Publish/Subscribe Example (3)

- Implementation of the channel?
  - based on simple RMI BT
    - channel has manager component for subscribers
  - multi-party protocol
    - channel represented as multicast address
  - enable control of QoS characteristics of event delivery
    - e.g. via binding CLI or additional generator interfaces

Overview

1. Motivation
2. Architecture
3. Conceptual Binding Model
4. How to add a new BT?
   - BT specification
     - The Binding Contract
5. Conclusion
The BT Contract (1)
- BT implementations
  - Binde\,Provider
  - Generator\,Provider
- supplies implementations for the binder and resolver roles of the BT
- supplies the generator roles
- can be deployed and replaced independently

The BT Contract (2)
- operations for lifecycle management
  (initialisation and termination)
- BT state
  - READY
    - new\,and
  - ACTIVE
    - has\,reference
  - can only be terminated when in READY state
- changes state autonomously
- has to notify Binding\,FF about state changes

The BT Contract (3)
- Binding\,FF implementation
  1. access point for BTs
  2. manage configurations of BT components
  3. provide BT components with access to other BTs and low-level services
The BT Contract (4)
- IDT Access:
  - Get registered IDT
  - Get registered IDT
  - Get service ID from IDT
  - invokes registry
- Registry:
  - persistent repository
  - maps from BT
  - binder to
  - BinderProvider and
  - GeneratorProvider
  - invokes Installer if not found
- Installer:
  - downloads and installs
  - components corresponding to
  - globally unique BT
  - identifier

The BT Contract (5)
- use IDTImplementation to initializeterminate
  - components
- offer
  - IDTImplementationUse
  - for notifications
- lifecycle management policies with plugable unloader components
  - track changes and
decide when to
  - remove them
  - allow clients to
dynamically add,
remove, replace and
  - retrieve BinderProvider,
GeneratorProvider and
  - unloader components

The BT Contract (6)
- IDT Services defines an
  - operation GetService() which
  - receives as an
  - input a service identifier
  - and an interface type
  - and return an interface
  - of the requested type
  - used to provide BT
  - components with access
to other BT's and low
  - level services

interface GetService(service_id, interface_type)
Overview

1. Motivation
2. Architecture
3. Conceptual Binding Model
4. How to add a new BT?
   ➢ BT specification
   ➢ The Binding Contract

5. Conclusion

Conclusion

• Interesting approach, but some aspects are still open:
  ➢ Where to get the refs from?
  ➢ Who does specify the format of refs?
  ➢ Who assigns the BTIds?
  ➢ Who manages the BTIds?
  ➢ What about QoS negotiations?
  ➢ Protecting from "bad" BTs?

Finish

Thank you!