Concurrence Seminar 2004

An Extensible Binding Framework for Component-Based Middleware

By Nikos Parkeonias, Geoff Couten, and Gordon S. Blair

Presented by Erdal Naş

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, AV 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 inventories 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, comms, 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
1. Motivation

2. Architecture

3. Conceptual Binding Model

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

5. Conclusion

---

1. Motivation

2. Architecture

3. Conceptual Binding Model

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

5. Conclusion

---

- 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

---

- 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 BindingOpt for binding control & management

---

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

Architectures

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
      > Publisher, subscriber, channel
    – binding properties such as... (interface)
  – Inref Generation and Resolution
    • describes the process of managing inrefs
      > Channel: inref generate (event, interface)
      > Subscriber: inref generate (subscriber, interface)
      > use stdin method to get reps

Specifying Binding Types (2)

• BT as a set of four "collaborations"
  – Binding Establishment
    • describes sequence of actions to set-up a binding
      > Publisher: appSend (app)
      > Subscriber: bindingCtl (app, subscriber)
  – Binding Control and Management
    • describes process of managing an already-established binding (monitoring, controlling, adding/removing participants, ...)
      > via bindingCtl or app
**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
  - Binder 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 (initialization and termination)
- BT state
  - READY
    - not used
  - ACTIVE
    - has reference counting
    - can only be terminated when in READY state
    - changes states automatically
    - has to notify BindingCF about state changes

**The BT Contract (3)**

- Binding CF 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)

- IBT Access:
  - Get BusInfo from BT
  - Get BusInfo via API
  - Get BusInfo via DUT
- Invokes registry
- Registry:
  - persistent repository
  - maps from BT
  - Binder to
  - Binder
- Invokes Installer
  - Installer:
  - download and install
  - components corresponding to
  - globally unique BT
  - Binder

The BT Contract (5)

- Use BT implementation
  - to initialize terminate
  - components
- Offer
  - IBT implementation
  - for initializing
- Lifecycle management policies
  - with plugable
  - unloader components
- Track change and
  - decide when to
  - remove them

The BT Contract (6)

- IBTService defines
  - an operation (GetService())
  - which receives an
  - input service identifier
  - and an interface type
  - and returns an interface
  - of the requested type
- Used to provide BT
  - components with access
  - to other BT's and
  - internal 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 do the refs come from?
  - Who does specify the format of refs?
  - Who assigns the BTIDs?
  - Who manages the BTIDs?
  - What about QoS negotiations?
  - Protections from "bad" BTs?

Finish

Thank you!