Overview

- Collaborative Object Modeling and architectural design mETHod (COMET)
  - Object Oriented Analysis and Design Method
  - Uses UML (Unified Modeling Language) notation
    - Standard approach for describing a software design
  - COMET = UML + Method
- Provides steps and guidelines for
  - Software Modeling and Design
  - From Use Case Models to Software Architecture
Steps in Using COMET/UML

1 Develop Software Requirements Model
2 Develop Software Analysis Model
3 Develop Software Design Model
   – Design Overall Software Architecture (Chapter 12, 13)
   – Design Distributed Applications (Chapter 12, 13, 15)
   – Structure Subsystems into Concurrent Tasks (Chapter 18)
   – Design Information Hiding Classes (Chapter 14)
   – Develop Detailed Software Design
Architectural Design of Distributed Applications

• Distributed processing environment
  – Multiple computers communicating over network
• Typical applications
  – Distributed real-time data collection
  – Distributed real-time control
  – Client / server applications
• COMET/UML for Distributed Applications
  – Addresses structuring application into distributed subsystems
Figure 15.1 Basic Client / Server configuration

Figure 15.4 Distributed processing configuration
Active and Passive Objects

- Objects may be active or passive
- **Active object**
  - Concurrent task or component
  - Has thread of control
- **Passive object**
  - a.k.a. Information Hiding Object
  - Has no thread of control
  - Operations of passive object are executed by task

UML notation for messages

- Simple message
  - No decision yet made about message type
- Asynchronous loosely coupled message communication
  - UML 1.3
  - UML 1.4 and in UML 2.0
- Synchronous tightly coupled message communication with reply
  - c1) Option 1:
  - c2) Option 2:

Copyright 2011 H. Gomaa
Characteristics of Distributed Applications

• Structure of component-based distributed application
  – Consists of one or more component-based subsystems
  – Each subsystem designed as a distributed component
  – Execute on multiple nodes in distributed configuration

• Structure of component-based subsystem
  – Consists of one or more objects
  – Objects all execute on same node

• Communication between component-based subsystems
  – Message communication

Steps in Designing Distributed Applications

• System Decomposition
  – Decompose system into distributed subsystems
    • Each subsystem designed as configurable component
    • Component
      – Active self-contained object with a well-defined interface, capable of being used in different applications from that for which it was originally defined
      – Define message communication interfaces

• Subsystem Decomposition
  – Structure subsystem into active objects (tasks) and passive objects

• System Configuration
  – Define component subsystem instances of target system
  – Map to hardware configuration
Define Subsystem Interfaces

- Message Communication between distributed subsystems
  - Loosely coupled (asynchronous) message communication
  - Multiple Client / Server message communication
    - Tightly coupled (synchronous) message communication
  - Remote Procedure Call
  - Group Message Communication
    - Broadcast message communication
    - Multicast message communication
  - Object Broker Architecture
  - Negotiation among Software Agents
  - Transaction Processing
Asynchronous Message Communication
(Loosely Coupled)
• Producer sends message and continues
• Consumer receives message
  • Suspended if no message is present
  • Activated when message arrives
• Message queue may build up at Consumer

![Diagram of Asynchronous Message Communication]

Figure 12.8 Asynchronous message communication

Synchronous Message Communication
with Reply (Tightly Coupled)
• Service
  – Responds to message requests from several clients
• Client
  – Sends message to Service and Waits for response
• Remote Procedure Call
  – Client makes RPC to service on different node
  – Communication details hidden from client & service
• Remote method invocation (RMI)
  – Client object sends message to service object

![Diagram of Synchronous Message Communication with Reply]

Figure 15.7 Synchronous message communication with reply
Group Message Communication

- One-to-many message communication
  - Same message sent to several recipients
- Broadcast message communication
  - Message sent to all recipients
- Multicast message communication
  - Same message sent to all members of group
- Subscription/Notification communication
  - Client subscribes to group
  - Receives messages sent to all members of group
  - Sender sends message to group
    - Does not need to know recipients

Figure 17.10 Example of subscription/notification (message multicast) communication
Distributed Objects and Object broker

- Clients and Servers designed as distributed objects
- Object Broker
  - Mediates interactions between clients and servers
  - Frees client from having to maintain information
    - Where particular service provided
    - How to obtain service
- Servers register Services & Location with Broker
- Clients request information from Broker about Servers
- Broker provides different services

Object Broker Architecture

- White pages
  - Client knows name of service but not location
  - Forwarding design
    - Broker forwards client request to Server
    - Broker forwards Server response to Client
  - Handle-driven design
    - Broker returns handle (remote reference) to Client
    - Client uses handle to communicate with Server
Figure 16.1 Service registration with Broker

- Services registers service information with Broker
  - Service name
  - Service interface
  - Service description
  - Location
- Service re-registers after moving to different location

Figure 16.2: Broker Forwarding (White pages) pattern

- Broker Forwarding Pattern
  - Broker forwards client request to Service
  - Broker forwards Service response to Client
**Figure 16.3: Broker Handle (White pages) pattern**

- Broker Handle Pattern
  - Broker returns handle (remote reference) to Client
  - Client uses handle to communicate with Service

**Steps in Using COMET/UML**

1. Develop Software Requirements Model
2. Develop Software Analysis Model
3. **Develop Software Design Model**
   - Design Overall Software Architecture (Chapter 12, 13)
   - **Design Distributed Applications (Chapter 12,13,15)**
   - Structure Subsystems into Concurrent Tasks (Chapter 18)
   - Design Information Hiding Classes (Chapter 14)
   - Develop Detailed Software Design