Reflective Middleware

INF5360 – 11.03.2008
Tommy Gudmundsen
tommygu@ifi.uio.no
Papers presented

• Grace, P., Blair, G.S., Samual, S., "ReMMoC: A Reflective Middleware to Support Mobile Client Interoperability"
Outline

• Introduction
• ReMMoC
  – Reflective Middleware for Mobile Computing
• Break
• OpenCOM v2
  – Experimental language-independent component-based systems-building technology
ReMMoC : Reflective Middleware for Mobile Computing

• The problem and a sample scenario
• The contribution of the paper
• Component Model
• The design and implementation of ReMMoC
• The ReMMoC programming Model
• The ReMMoC API
• (Evaluation and related work)
• Summary and discussion
The problem

- Heterogeneous service availability
  - Service Discovery
    - SLP, UPnP
  - Service Middleware
    - RMI, P/S, SOAP

- Interoperability from mobile clients
A sample scenario
The paper contribution

• Presents ReMMoC
  – Documents the design and implementation
• A reflective middleware platform for mobile computing
  – Built using reflection and component technology
• WSDL programming model
Component Model (1)

• OpenCOM v1
  – Lightweight, efficient and reflective component model
  – Based on a subset of Microsoft’s COM
  – Provides a runtime / kernel that manages:
    • Creation and deletion of components
    • Acts upon requests to connect and disconnect components
Component Model (2)

• Key concepts
  – Interfaces
    • Unit of service provision
  – Receptacles
    • Unit of service requirement
  – Connections
    • Bindings between interfaces and receptacles
  – System graph
    • Introspection and adaption → Reflection
OpenCOM Component Frameworks (1)

• Component Framework
  – A collection of rules and contracts that govern the interaction of a set of components.
  – Is itself an OpenCOM component
  – Can be composed, replaced, and interconnected like other components.
OpenCOM Component Frameworks (2)
ReMMoC Design & Implementation

- Supports mobile application development and overcomes the heterogeneous property of the mobile environment
- Configurable and reconfigurable
- A set of component frameworks
  - Binding
  - Service discovery
The ReMMoC Platform
The Binding and Service Discovery Component Frameworks

- New implementation protocols can be plugged-into the CFs
- Major and minor reconfigurations
- Dependency
  - SD on DiscoverDiscoveryProtocol
  - Both on Accept (compositions)
The ReMMoC Programming Model

- **Web Services Architecture Roles**
  - Service Requestor (SR)
  - Service Provider (SP)
  - Discovery agency (Find the WSDL)

- **Web Service Description Language WSDL**
  - Abstract description of service
  - Service may be implemented on any middleware.
Mapping WSDL to MW Paradigms
The ReMMoC API

Event-driven

• Operations to:
  – Locate a service
  – Lookup and invoke abstract WSDL operation
  – Invoke operations on known services
  – Create/host service provider operations

• API calls are mapped to the correct MW binding
Evaluation

• Minimize memory footprint
• Only a minimum number of components
  – Reflection
• Reflective VS non-reflective
  – Reflective consumes more memory
• Configurations less than 100Kbytes
  – Suitable for mobile devices
Related Work

- Asynchronous Mobile Middleware
- Adaptive Middleware
Summary and discussion

- Cascading levels of heterogeneity
  - SD + binding
- Mobile client interoperability
  - WS + Reflective Middleware
- Lightweight wrt complexity?
- Much is up to the developer?
- Security and resource management
Break?
OpenCOM version 2

- The problem
- The contribution of the paper
- Requirements
- Component Frameworks
- Reflective meta-models
- Programming Model
- OpenCOM Programmer Roles
- Summary and discussion
The problem

• Component technology are not to often targeted towards systems-building, but rather application-building

• When component technology is target towards systems-building, is often narrowly-targeted.
  – Application domain
  – Hardware/OS deployment environment
The paper contribution

• A proposal for a general-purpose component-based systems-building technology.
Requirements

- Wide applicability
- Policy independence
- Support for runtime reconfiguration
- Selective transparency of deployment-environment specific features.
- Separation of concerns
- High performance
OpenCOM v2

• A generic runtime component model as the foundation with:
  – Component frameworks
  – Reflective meta-models

• Based on OpenCOM v1
  – Used to build middleware only
  – Deployed in standard OS environments only
Component Frameworks

- Composite components built in terms of the underlying component model
- Accepts plug-in components that add to or modify the composite behavior
- Addresses a particular functional domain
- Architectural patterns rather than distinct mechanisms
Reflective meta-models

• Represents an aspect of the target system
• Enable inspection and adaptation of the represented aspect
• Separation of concerns
  – System building
  – System adaptation
OpenCOM Programming Model(1)

• Capsule
  – Unit of scope and management
• Caplet (sub-scope within a capsule)
  – Root (the original capsule environment)
  – Slave
• Binder
• Loader
(The core)
OpenCOM Programming Model (2)
OpenCOM Programming Model (3)

• Caplet CF
  – Caplet management / hosting.
  – Java in one C++ in another.

• Loader and Binder CFs
  – Selective transparency

• Custom CF
  – Policy embodied, meta-model aware.
OpenCOM Programmer Roles

- **systems programmer role** (base level OpenCOM programming)
  - components and CFs

- **meta-systems programmer role** (meta-level OpenCOM programming)
  - architecture
  - interception
  - interface
  - resources
  - ... reflective meta-models

- **OpenCOM deployer**
  - caplet, loader, binder plug-ins

- **deployment programmer role** (programming in the native deployment environment)
  - per-deployment env.
  - meta-level plug-ins
Summary and discussion

- General architecture for building systems software.
- The users will have a lot of work on their hands…