Motivation

- Common server activities include:
  - Service (re)configuration and run-time control
  - Daemonization and comm. endpoint initialization
  - I/O port demultiplexing and dispatching
  - Process and thread creation
- Conventional server designs are overly static, i.e.:
  - Must modify, recompile, and relink existing code
  - Must terminate and restart running processes
- The Service Configurator pattern increases server extensibility by dynamic configuring network services

Original SunSoft IIOP Reference Implementation

- Limitations with SunSoft IIOP
  - Not a complete ORB
  - Inefficient TypeCode interpreter
  - “One-size fits all” design
  - Functionality was entirely static
  * i.e., all enhancements require changing the ORB source code

Example: The ACE ORB (TAO)

- TAO Overview
  - High-performance, real-time ORB
    * Telecom and avionics focus
  - Leverages the ACE framework
    * Runs on VxWorks, POSIX, and Win32
  - Small memory footprint
  - Highly configurable

Service Configurator

A Pattern for Dynamically Configuring Network Services

Prashant Jain and Douglas C. Schmidt
pjain@cs.wustl.edu and schmidt@cs.wustl.edu

Washington University, St. Louis
June 19, 1997

This work is sponsored by Siemens MED
Key Strategies and Patterns in TAO

- **Key ORB Strategies**
  - Concurrency strategy → e.g., Thread-per-Request, Thread-per-Connection
  - Demultiplexing strategy → e.g., Dynamic Hashing, Perfect hashing, Active Demultiplexing
  - Dispatching strategy → e.g., Rate Monotonic, Earliest Deadline First

- **Key ORB Patterns**
  - Service Configurator
  - Strategy
  - Abstract Factory
  - Reactor
  - Active Object

Overview of the Service Configurator Pattern

- **Intent**
  - Decouples the behavior of services from the point in time at which service implementations are configured into an application or system.

- **Forces resolved**
  - How to defer the selection of a particular type, or a particular implementation, of a service until very late in the design cycle
  - How to build complete applications by composing multiple independently developed services
  - How to optimize, reconfigure, and control the behavior of the service at run-time
Using the Service Configurator Pattern in TAO

Concluding Remarks

Beneefits of patterns, in general
- Document common traps and pitfalls
- Guide design choices
- Preserve crucial design information
- Facilitate design reuse

Beneefits of Service Configurator pattern
- Increases flexibility and extensibility of networking apps.
- Centralizes administration and control

URls

Conftime Configuration

Run-time Conftime

For each Service do
    Service Configuration
    Service Processing
    Service Termination

Benefits of patterns, in general

Using the Service Configurator Pattern in TAO

Concluding Remarks

Beneefits of patterns, in general
- Document common traps and pitfalls
- Guide design choices
- Preserve crucial design information
- Facilitate design reuse

Beneefits of Service Configurator pattern
- Increases flexibility and extensibility of networking apps.
- Centralizes administration and control

URls

Conftime Configuration

Run-time Conftime

For each Service do
    Service Configuration
    Service Processing
    Service Termination

Benefits of patterns, in general

Using the Service Configurator Pattern in TAO

Concluding Remarks

Beneefits of patterns, in general
- Document common traps and pitfalls
- Guide design choices
- Preserve crucial design information
- Facilitate design reuse

Beneefits of Service Configurator pattern
- Increases flexibility and extensibility of networking apps.
- Centralizes administration and control

URls

Conftime Configuration

Run-time Conftime

For each Service do
    Service Configuration
    Service Processing
    Service Termination