Large-Scale CORBA-Distributed Software Framework for NIF Controls

Robert W. Carey
Lawrence Livermore National Laboratory

November 2001
Key topics

• Layered Architecture
• Leveling and Dependency Management
• Framework Abstractions
• Distribution and Object-Oriented Design Patterns
• Distribution Complexity
• Persistence Layer
• Process Management
A common framework is used to build all software applications.

**Server**
- Integration Services
  - System manager
  - Device hierarchy
  - Access control
- Database
  - History
  - Shots
  - Configuration

**Workstation**
- Supervisory Console
  - Operator Controls
  - Status Display
  - Event Log

**Software Distribution Bus (exists on network)**
- CORBA (Object Request Broker)

**Front End Processor**
- Device Control
- Status Monitor
  - Controller
  - Interface Driver

300 front-end processors interface to NIF equipment

Software objects representing control points “plug in” to the software distribution bus.
ICCS Distributed Component Architecture

- Logical Control Unit (LCU) objects in the Supervisors control hardware through device objects in the Front-End Processors (FEP)
- A Framework Layer (FWL) resides in each distributed process

Diagram:

- Framework Server
- GUI
- Supervisor
- FEP
- Device
- CORBA
- POWER AMP
- MAIN AMPLIFIER
- Preamp
- Fiber
- Master oscillator
Key topics

- Layered Architecture
- Leveling and Dependency Management
- Framework Abstractions
- Distribution and Object-Oriented Design Patterns
- Distribution Complexity
- Persistence Layer
- Process Management
Frameworks are constructed in layers to permit retargeting

<table>
<thead>
<tr>
<th>Framework Templates Layer</th>
<th>Support layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract classes for control systems</td>
<td>COTS and components</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Framework Services</td>
<td>Supervisor applications,</td>
</tr>
<tr>
<td></td>
<td>Front End Processors,</td>
</tr>
<tr>
<td></td>
<td>Database servers,</td>
</tr>
<tr>
<td></td>
<td>etc</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>NIF Building blocks</td>
<td>Devices, Shot phases,</td>
</tr>
<tr>
<td></td>
<td>etc</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>ICCS Programs</td>
<td>Configuration Server,</td>
</tr>
<tr>
<td>Client and server mainlines</td>
<td>System Manager,</td>
</tr>
<tr>
<td></td>
<td>GUI’s,</td>
</tr>
<tr>
<td></td>
<td>etc</td>
</tr>
</tbody>
</table>
ICCS is divided into subsystems to partition activity and ensure performance.
Key topics

- Layered Architecture
- Leveling and Dependency Management
- Framework Abstractions
- Distribution and Object-Oriented Design Patterns
- Distribution Complexity
- Persistence Layer
- Process Management
Supervisors and FEPs are built on a common set of frameworks.

This common architecture enhances reliability and improves maintainability.
The frameworks provide architecture templates and utility services

- Framework templates define the architecture for each type of process in ICCS:
  - Supervisory Shot Control Processes
  - Supervisory Status and Control Processes
  - Front End Processors
  - Graphical User Interfaces

- Frameworks also provide utility services:
  - Configuration
  - Messaging (Events, Alerts, Logging)
  - Status Monitoring
  - Reservation
  - Archiving
  - Sequence Control
All application processes are constructed the same.
They all startup and shutdown using the same protocol.
Content of application processes is completely data driven.
Service distribution is encapsulated by Framework Service APIs.
Key topics

- Layered Architecture
- Leveling and Dependency Management
- Framework Abstractions
- Distribution and Object-Oriented Design Patterns
- Distribution Complexity
- Persistence Layer
- Process Management
O-O design patterns extended for distribution

- **Observer (Publish/Subscribe)** is used extensively by the framework
  - Alerts, Events, Process State, Status Propagation
  - Publishing is de-coupled for each subscriber
  - Failure on publish cancels subscription

- **Object Factory**
  - process content is data driven
  - Persistence brokers serve process data via CORBA

- **Model-View-Controller Architecture**
  - Thin GUI layer presents the appearance (View)
  - Model defines system semantics
  - CORBA bridges Java and Ada language environments
Key topics

- Layered Architecture
- Leveling and Dependency Management
- Framework Abstractions
- Distribution and Object-Oriented Design Patterns
- Distribution Complexity
- Persistence Layer
- Process Management
ICCS employs a component-based communication architecture

- 300 CORBA IDL classes (150 device, 50 framework and 100 supervisor)
- Policies define interface de-coupling mechanisms and common exception pattern
- Framework provides connection management
  - Detection
  - Notification
  - Recovery
- Timed Invocation is a requirement
- Formal testing exposes deadlocks

Goal: Fault Resilience – degraded operation in the presence of server failure and recovery upon server restoration
Test suite characterizes detailed effects of CORBA failure modes

- Failures under different socket conditions:
  - Server fails before/after initial client connection
  - Client fails after server connection

- Failures during request processing:
  - Server fails during processing
  - Client fails during a request
  - Client registers callback; client fails, restarts, and re-registers. Server attempts client call-back
  - Client sends request to server. Server hangs

Failure modes are characterized for both CORBA and TCP/IP
Process failure detection

• The System Manager Framework monitors heartbeats from all the processes to detect process failures

• The System Manager GUI informs operators of process failures
Failure notification

- The connection management in each process receives process state updates of interest and maps this state to object references.
- The connection management layer notifies client objects of server object failures.

Supervisor

1: Connect()
2: Get_Ref()
3: Update_Map()
4: Subscribe_To_Process()
5: Heartbeat not delivered
6: Update()

Name Server

System Manager Server

Map
Object => Process

CML

FEP

The National Ignition Facility
Failure Recovery

- Connection objects manage object references and contain reconnection strategy

```
6: Process Update()
   Map Object Ref => Process

3: Update Map()
   Name Server

5: Subscribe()
   Config_Local

1: Connect()
   Connection

8: Get Ref()
   FEP device

2: Get Ref()
   Supervisor

7: Process State
   System Manager

4: Ref/Process Map

9: Get Ref State()

10: Ref State

12: device command
```
Key topics

• Layered Architecture
• Leveling and Dependency Management
• Framework Abstractions
• Distribution and Object-Oriented Design Patterns
• Distribution Complexity
• Persistence Layer
• Process Management
Persistence Brokers insulate applications from DBMS

Supported persistence mechanisms
- Oracle8i
- HDF
- XML
Key topics

- Layered Architecture
- Leveling and Dependency Management
- Framework Abstractions
- Distribution and Object-Oriented Design Patterns
- Distribution Complexity
- Persistence Layer
- Process/Network Management
ICCS process/network management leverages SNMP technology

- Each ICCS host contains a SNMP sub-agent
  - CPU utilization
  - File System Statistics
  - Memory utilization
  - Task states
- HP OpenView Network Node Manager collects statistics for:
  - ICCS Processes
  - Network switches
  - Network Hosts
  - File Systems
ICCS Framework is positioned for scaling in ‘04

- **Successes**
  - ✔ Layering/Leveling
  - ✔ Framework abstractions implemented
  - ✔ Patterns and distribution
  - ✔ Persistence Layer
  - ✔ Iterative process gives us practice
  - ✔ Formal testing is our savior

- **Summary of Challenges**
  - Manage distribution complexity
  - Systematic connection management in a “fine-grained” CORBA implementation
  - TCP/IP transport configuration
  - Configuration data management