Towards a Pattern Language for NEST Middleware

Venkita Subramonian & Chris Gill, Washington University, St.Louis
David Sharp, The Boeing Company, St. Louis

{venkita,cdgill}@cs.wustl.edu
David.Sharp@MW.Boeing.com

Research supported by DARPA (NEST), Boeing OEP contract
What is NEST?

- Networked Embedded Software Technology
- DRE system with 100 to 100,000 networked nodes
- Resource constrained hardware components
- Requires fine-grain fusion of hardware and software components
- Applications in advanced avionics and space systems, weapon systems, wireless devices
NEST Services

- Predictable and dependable behavior despite local failures
- Real-time Coordination Services
  - Fault tolerance
  - Data exchange
  - Synchronization
  - Self-stabilizing protocols
  - Replication
- Automated synthesis of services
An Open Experimental Platform for NEST

Active Damage Interrogation

Structure with Embedded or Bonded Piezoelectric Transducers

Acoustic Waves (kHz Range)

Actuator Excitation

Information Processing System

Sensor Measurements
Why Middleware for NEST?

- Service reuse across NEST applications
- Flexible
  - Can be customized to a particular NEST application/execution context
  - Can exist across various levels of scale
- Address Design Forces in NEST
  - Distribution of control
  - Resource Management
  - Fault Tolerance
  - Time Synchronization
  - Heterogeneous processing
  - Dynamic Reconfiguration
A Pattern Map for NEST Middleware

Blackboard
- Knowledge source implementation
- Internal structuring

Broker
- Communication/concurrency
- Request dispatching

Layers
- Inter-layer communication

Pipes & Filters
- Location dynamics
- Scalability

Lookup
- Service lookup
- Performance optimization
- Resource optimization

Locate & Track
- Scalability

Lazy Acquisition
- Resource optimization

Pick & Verify

Evictor

Leasing

POSA2 patterns
- Communication/concurrency

Recoverable Distributor
- Fault tolerance/data sharing
- Decoupled data updates/fault notification
- Fault tolerance/data consistency protocols

Strategy

Distributed Observer

Distributed Notifier
Towards a NEST Middleware Pattern Language

- Lookup Service
- NEST Client
  - Locate & Track
  - Pick & Verify
  - Evictor
  - Leasing
  - Lazy Acquisition
  - Broker
  - POSA2 patterns

- Resource Locator
- Recoverable Distributor
- Distributed Observer
- Distributed Notifier
- Fault Tolerance
- NEST Service
- Communication /Concurrency

Resource Management
A stake in the ground

- ACE/TAO
  - Pattern rich middleware frameworks
  - Capture some inherent structure of the NEST domain

- Feature reduction/extension from ACE/TAO
  - Call tracing mechanisms
  - minimumCORBA and single thread options

- OEP Application domain
  - Understand the OEP application
  - Candidate list of middleware services
Conclusions and Future work

- What part of CORBA might we need in a NEST environment – *nanoCORBA*??
- Customized subsetting/extension of ACE and TAO for NEST
- Leverage approaches for subsetting/extension
  - Generic/Aspect-Oriented/Generative Programming
  - Automated custom generation that leverages the evolution of the baseline