ACE Overview
Douglas C. Schmidt

Motivation: the Distributed RT Communication Software Crisis

Symptoms
- Hardware gets smaller, faster, cheaper
- Software gets larger, slower, more expensive

Culprits
- Accidental and inherent complexity

Solutions
- Frameworks, components, and patterns

Techniques for improving software quality and productivity

APPLICATION APPLICATION SPECIFIC LOGIC USER INTERFACE

MATH DATA BASE

NETWORKING USER INTERFACE DATABASE EVENT LOOP

EVENT LOOP CALL BACKS INVOKES ADTS

FRAMEWORK ARCHITECTURE

PROVEN SOLUTIONS
- Components: self-contained, pluggable
- Frameworks: reusable, semi-complete
- Patterns: problem/solution pairs in a context

Developing Distributed RT Systems
Using OS System-Hiding Frameworks

Sponsors
NSF, DARPA, ARD, BBN, Boeing, Critical, Commerce, COTS, EXOS, Flexia, IBM, MIT, Sprint, Toshiba, USENIX

AC, Oracle, OSF, OINX, PicoTeam, SAIL, Software, SGI, Sun, Siemens, 7T, Hughes, Kodak, Korn, Lockheed, Lear, Lincoln, Motorola, NCR, NCR, NRC, NDS, Nominok, NT
c, Cabletron, DEC, CPM, Eng. Dept.

Douglas C. Schmidt
Associate Professor
University of California, Irvine
schmidt@uci.edu
www.ece.uci.edu/~schmidt

Source
949-824-1901

Developing Distributed RT Systems
Using OS System-Hiding Frameworks

Techniques for improving software quality and productivity

APPLICATION APPLICATION SPECIFIC LOGIC USER INTERFACE

MATH DATA BASE

NETWORKING USER INTERFACE DATABASE EVENT LOOP

EVENT LOOP CALL BACKS INVOKES ADTS

FRAMEWORK ARCHITECTURE

PROVEN SOLUTIONS
- Components: self-contained, pluggable
- Frameworks: reusable, semi-complete
- Patterns: problem/solution pairs in a context

Developing Distributed RT Systems
Using OS System-Hiding Frameworks

Sponsors
NSF, DARPA, ARD, BBN, Boeing, Critical, Commerce, COTS, EXOS, Flexia, IBM, MIT, Sprint, Toshiba, USENIX

AC, Oracle, OSF, OINX, PicoTeam, SAIL, Software, SGI, Sun, Siemens, 7T, Hughes, Kodak, Korn, Lockheed, Lear, Lincoln, Motorola, NCR, NCR, NRC, NDS, Nominok, NT
c, Cabletron, DEC, CPM, Eng. Dept.

Douglas C. Schmidt
Associate Professor
University of California, Irvine
schmidt@uci.edu
www.ece.uci.edu/~schmidt

Source
949-824-1901
**Roadmap to Levels of Middleware Abstraction**

- **Observations**
  - Historically, apps built directly atop OS
  - Today, more and more apps built atop middleware
  - Middleware has several layers
- **General R&D challenges**
  - Performance optimizations
  - Quality of Service (QoS)
  - Software architecture & patterns

---

**Why We Need Communication Middleware**

- **System call-level programming is wrong abstraction for application developers**
  - Too low-level → error codes, endless reinvention
  - Error-prone → HANDLEs lack type-safety, thread cancellation woes
  - Mechanisms do not scale → RTOS TSS
  - Steep learning curve → Win32 Named Pipes
  - Non-portable → socket bugs
  - Inefficient → i.e., tedious for humans
- **GUI frameworks are inadequate for communication software**
  - Inefficient → excessive use of virtual methods
  - Lack of features → minimal threading and synchronization mechanisms, no network services

---

**The ADAPTIVE Communication Environment (ACE)**

- **ACE Overview**
  - A concurrent OO networking framework
  - Available in C++ and Java
  - Ported to VxWorks, POSIX, and Win32
- **Related work**
  - x-Kernel
  - SysV STREAMS

http://www.cs.wustl.edu/~schmidt/ACE.html
ACE Overview

- Multimedia services
- Protocol management
- Real-time Alaska
- Concurrency
- Databases
- Real-time services
- Distributed

• Domains

ACE Statistics

- 20,000 lines of C++
- Over 300,000 lines of C++
- Large user community
- Extensive community support
- Large user community
- Supported on: UNIX, Windows, MVS, and embedded platforms
- Currently used by dozens of companies
- Failures rarely result from unknown science, but from unknown engineering
- Patterned design choices
- Facilitated design reuse
- Preserved crucial design information
- Guided design choices
- Observation
- Benefits of patterns
- Observation

Patterns for Communication Middleware

- Event Patterns
- Concurrency Patterns
- External Polymorphism
- Wrapper Facade
- Connector Acceptor
- Thread Pool
- Thread-per Session
- Thread-per Request
- Asynchronous Completion Token
- Thread Specific Storage

- Active Object
- Half-Sync/Half-Async
- Leader/Followers
- Service Configurator Object
- Lifetime Manager
- Reactor
- Proactor
- Double Checked Locking
- Thread-Safe Interface
- Scoped Locking
- Strategized Locking
- Initialization Patterns
- Synchronization Patterns

Applications

- Specifications
- Patterns for Communication
- Middleware

Domains

- Real-time avionics
- Distributed interactive simulations
- Satellite communication
- Network management
- Medical imaging
- Multimedias services
- Protocol management
- Real-time Alaska
- Concurrency
- Databases
- Real-time services
- Distributed

Use-cases for ACE and TAO

Dozens of companies

- Boeing
- Cisco
- Ericsson
- Kodak
- Lockheed
- Lucent
- Motorola
- Nokia
- Nortel
- Raytheon
- SAIC
- Siemens
- StorTek

Supported on: UNIX, Windows, MVS, and embedded platforms

• Patterned design choices
• Facilitated design reuse
• Preserved crucial design information
• Guided design choices

Observation

- Benefits of patterns
- Observation

Patterns for Communication Middleware

- Event Patterns
- Concurrency Patterns
- External Polymorphism
- Wrapper Facade
- Connector Acceptor
- Thread Pool
- Thread-per Session
- Thread-per Request
- Asynchronous Completion Token
- Thread Specific Storage

- Active Object
- Half-Sync/Half-Async
- Leader/Followers
- Service Configurator Object
- Lifetime Manager
- Reactor
- Proactor
- Double Checked Locking
- Thread-Safe Interface
- Scoped Locking
- Strategized Locking
- Initialization Patterns
- Synchronization Patterns

Applications

- Specifications
- Patterns for Communication
- Middleware

Domains

- Real-time avionics
- Distributed interactive simulations
- Satellite communication
- Network management
- Medical imaging
- Multimedias services
- Protocol management
- Real-time Alaska
- Concurrency
- Databases
- Real-time services
- Distributed

ACE Statistics

- 20,000 lines of C++
- Over 300,000 lines of C++
- Large user community
- Extensive community support
- Supported on: UNIX, Windows, MVS, and embedded platforms

• Patterned design choices
• Facilitated design reuse
• Preserved crucial design information
• Guided design choices

Observation

- Benefits of patterns
- Observation

Patterns for Communication Middleware

- Event Patterns
- Concurrency Patterns
- External Polymorphism
- Wrapper Facade
- Connector Acceptor
- Thread Pool
- Thread-per Session
- Thread-per Request
- Asynchronous Completion Token
- Thread Specific Storage

- Active Object
- Half-Sync/Half-Async
- Leader/Followers
- Service Configurator Object
- Lifetime Manager
- Reactor
- Proactor
- Double Checked Locking
- Thread-Safe Interface
- Scoped Locking
- Strategized Locking
- Initialization Patterns
- Synchronization Patterns

Applications

- Specifications
- Patterns for Communication
- Middleware

Domains

- Real-time avionics
- Distributed interactive simulations
- Satellite communication
- Network management
- Medical imaging
- Multimedias services
- Protocol management
- Real-time Alaska
- Concurrency
- Databases
- Real-time services
- Distributed
Applying ACE to Satellite Communication Systems

Domain Challeges

- Long latency satellite links
- High reliability
- Prioritization

Domain Challenges

- Support platform upgrades
- Reusable components
- COTS and open systems
- Periodic & aperiodic
- Deterministic & stochastic

Applying ACE to Real-time Avionics

1: Sensors generate data

- FLIR, GPS, IFF

2: Sensor proxies demarshal data & pass to event channel

3: Push events

4: Pull data

HUD, Nav, Airframe, WTS

Applying ACE to Distributed Interactive Simulations

Agent Architecture

Network Operations Center

Agent

Agent

Agent

Network

Agent

Agent

Agent

Server

Server

Server

Agent

Agent

Agent

Agent

Audio, Video

HSM, Archive

Playback

Interactive Audio, Video

Hardware, Embedded

MIB, TA0

Agent

Agent

Target

Device

Command

Data

Local Area Network

Ground Station

Gateway

WIDE AREA NETWORK

SATELLITE TRACKING STATION

PEERS

STATUS INFO

COMMANDS

BULK DATA TRANSFER

PEERS

GATEWAY

WIDE AREA NETWORK

SATELLITE TRACKING STATION

PEERS

STATUS INFO

COMMANDS

BULK DATA TRANSFER

PEERS

GATEWAY
Applying ACE to Network Management

Session Router Module

Presentation Module

Event Filter Module

Event Analysis Module

Reactor

Domain Challenges

- Low latency
- Multi-platform
- Family of related services

ACE Overview

Lessons Learned Building ACE

- Be patient
- Good components, frameworks, and software architecture take time to develop
- The best components come from solving real problems (i.e., don't build components in isolation)
- Keep feedback loops tight to avoid runaway reuse
- Produce reusable components by generalizing from working applications

Concluding Remarks

Developers of real-time communication software confront recurring challenges that are largely application-dependent.

- Service initialization and distribution
- Error handling
- Control
- Event demultiplexing
- Concurrency control
- Synchronization
- Scheduling

Developers of real-time communication software confront recurring challenges that are largely application-dependent. Successful developers resolve these challenges by applying appropriate design patterns to create communication frameworks. Application frameworks are an effective way to achieve broad reuse of existing components. Open source development models are particularly attractive. However, reuse models require a different mindset than building components in isolation. reusable components...
All source code for ACE is freely available at www.cs.wustl.edu/schmidt/ACE.html.

Obtaining ACE

- www.riverace.com
- ACEannounce@wustl.edu
- ACEusers@wustl.edu
- ACEannounce-request@wustl.edu
- ACEusers-request@wustl.edu

Mailing Lists
- ace-users@cs.wustl.edu
- ace-users-request@cs.wustl.edu
- ace-announce@cs.wustl.edu
- ace-announce-request@cs.wustl.edu

Newsgroup
- comp.soft.sys.ace

Commercial Support
- www.riverace.com

UC Irvine