Developing Distributed RT Systems Using OS System-Hiding Frameworks

Douglas C. Schmidt

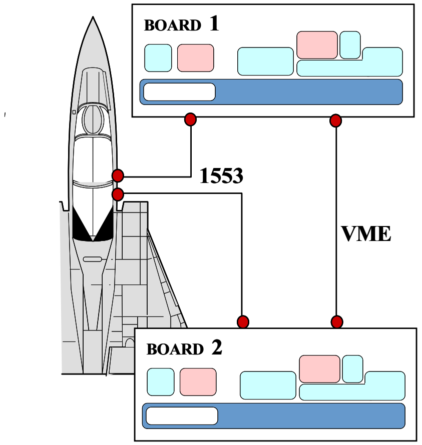
Associate Professor
schmidt@uci.edu
www.ece.uci.edu/∼schmidt/

Elec. & Comp. Eng. Dept.
University of California, Irvine
(949) 824-1901

Sponsors

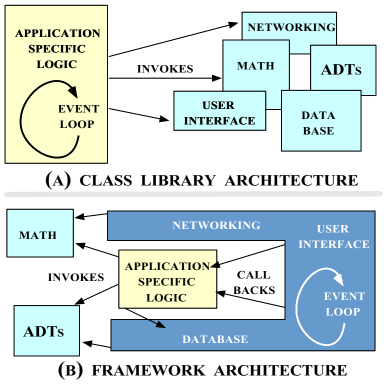
NSF, DARPA, ATD, BBN, Boeing, Cisco, Comverse, GDIS, Experian, Global MT, Hughes, Kodak, Krones, Lockheed, Lucent, Microsoft, Mitre, Motorola, Nokia, Nortel, OCI, Oresis, OTI, Raytheon, SAIC, Siemens SCR, Siemens MED, Siemens ZT, Sprint, Telcordia, USENIX

Motivation: the Distributed RT Communication Software Crisis



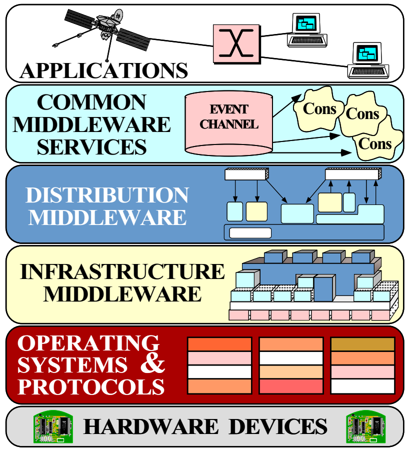
- **Symptoms**
  - Hardware gets smaller, faster, cheaper
  - Software gets larger, slower, more expensive
- **Culprits**
  - *Accidental* and *inherent* complexity
- **Solutions**
  - Frameworks, components, and patterns

Techniques for Improving Software Quality and Productivity



- **Proven solutions**
  - *Components*
    - ∗ Self-contained, "pluggable" ADTs
  - *Frameworks*
    - ∗ Reusable, "semi-complete" applications
  - *Patterns*
    - ∗ Problem/solution pairs in a context
  - *Architecture*
    - ∗ Families of related patterns and components

Roadmap to Levels of Middleware Abstraction

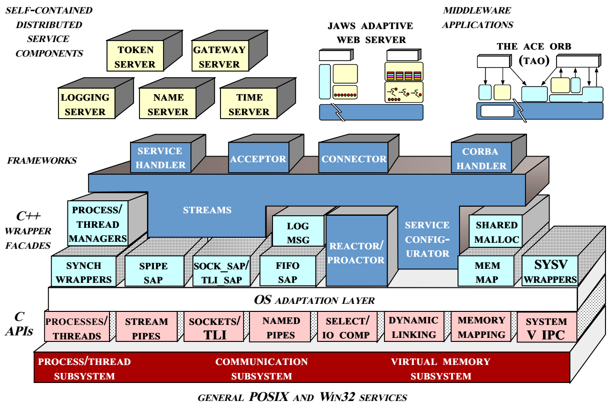


- **Observations**
  - Historically, apps built directly atop OS
  - Today, more and more apps built atop *middleware*
  - Middleware has several layers
- **General R&D challenges**
  - Performance optimizations
  - Quality of Service (QoS)
  - Software architecture & patterns

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)



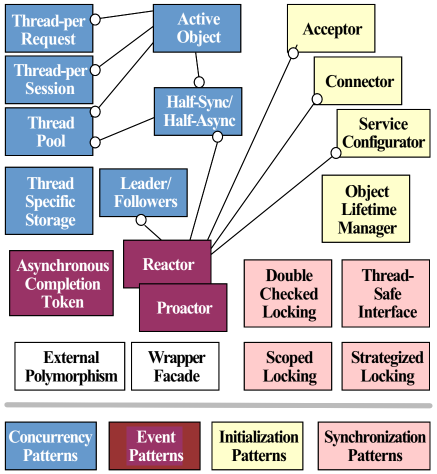
http://www.cs.wustl.edu/∼schmidt/ACE.html

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

ACE Statistics

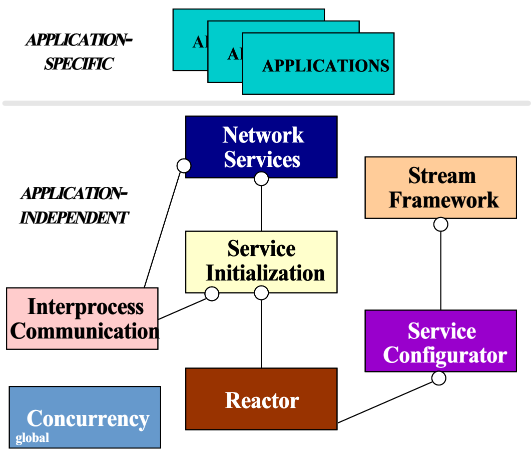
- ACE contain > 200,000 lines of C++
  - Over 30 person-years of effort
- Ported to UNIX, Win32, MVS, and embedded platforms
  - *e.g.*, VxWorks, LynxOS, Chorus, pSoS, QNX
- Large user community
  - www.cs.wustl.edu/∼schmidt/ACE-users.html
- Currently used by dozens of companies
  - Boeing, Cisco, Ericsson, Kodak, Lockheed, Lucent, Motorola, Nokia, Nortel, Raytheon, SAIC, Siemens, StorTek, etc.
- Supported commercially
  - www.riverace.com

Patterns for Communication Middleware



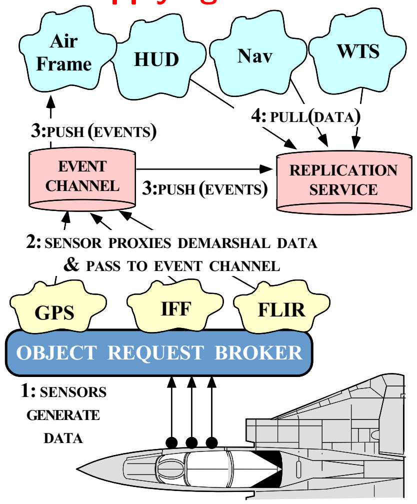
- **Observation**
  - *Failures rarely result from unknown scientific principles, but from failing to apply proven engineering practices and patterns*
- **Benefits of Patterns**
  - Facilitate design reuse
  - Preserve crucial design information
  - Guide design choices

Use-cases for ACE and TAO



- **Domains**
  - Real-time avionics
  - Distributed interactive simulations
  - Satellite communication
  - Network management
  - Medical imaging
  - Multimedia services

Applying ACE to Real-time Avionics

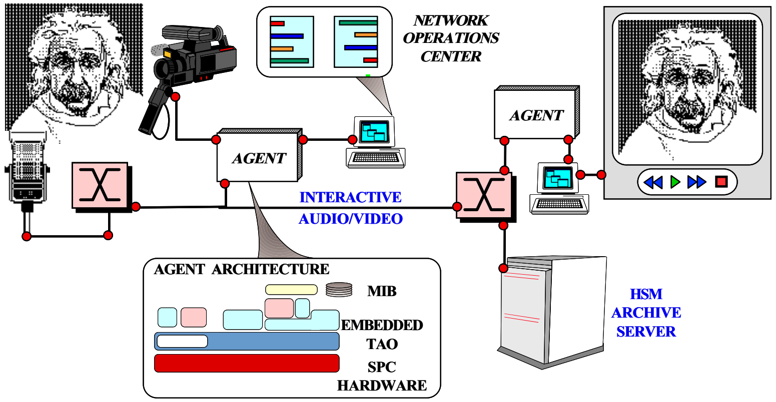


Domain Challenges

- Deterministic & statistical real-time deadlines
- Periodic & aperiodic processing
- COTS and open systems
- Reusable components
- Support platform upgrades

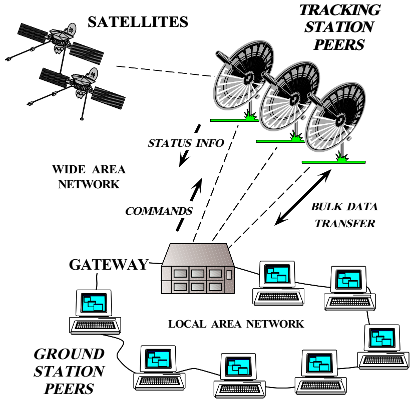
www.cs.wustl.edu/~schmidt/TAO-boeing.html

Applying ACE to Distributed Interactive Simulations



www.cs.wustl.edu/~schmidt/Words99.ps.gz

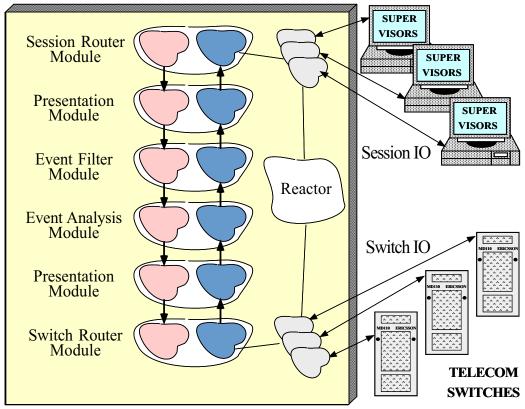
Applying ACE to Satellite Communication Systems



- **Domain Challenges**
  - Long latency satellite links
  - High reliability
  - Prioritization

www.cs.wustl.edu/~schmidt/TAPOS-95.ps.gz

Applying ACE to Network Management



www.cs.wustl.edu/~schmidt/DSEJ-94.ps.gz

- **Domain Challenges**
  - Low latency
  - Multi-platform
  - Family of related services

Lessons Learned Building ACE

- **Be patient**
  - Good components, frameworks, and software architectures take time to develop
- **Reuse-in-the-large works best when:**
  1. The marketplace is competitive
  2. The domain is complex
  3. Skilled middleware developers
  4. Supportive corporate culture
  5. "Reuse magnets" exist
  6. Open source development models
- **The best components come from solving real problems**
  - Keep feedback loops tight to avoid "runaway" reuse efforts
- **Produce reusable components by generalizing from working applications**
  - *i.e.*, don't build components in isolation

Concluding Remarks

- Developers of real-time communication software confront recurring challenges that are largely application-independent
  - *e.g.*, service initialization and distribution, error handling, flow control, event demultiplexing, concurrency control, synchronization, scheduling
- Programming directly to the underlying OS APIs is tedious, error-prone, and non-portable
- 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 software

Obtaining ACE

- All source code for ACE is freely available
  - www.cs.wustl.edu/~schmidt/ACE.html
- 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