

# **Middleware in COTS Command and Control Architectures**

**GSAW 98 - COTS and Legacy Software  
Integration Break-out Session**

**February 26, 1998**



# Background

---

- **Emerging trend towards COTS and GOTS leverage**
- **Integration challenges are inevitable**
  - COTS products
  - Legacy applications
- **Flexibility and adaptability are key**
  - Emerging but immature technologies and standards
    - JAVA, Object Databases, CORBA
  - Similar and sometimes conflicting technologies
    - CORBA, COM, MOM
    - C/C++/J++/VB
  - Heterogeneous networks
  - Scalability

# Open Architecture Philosophy

---

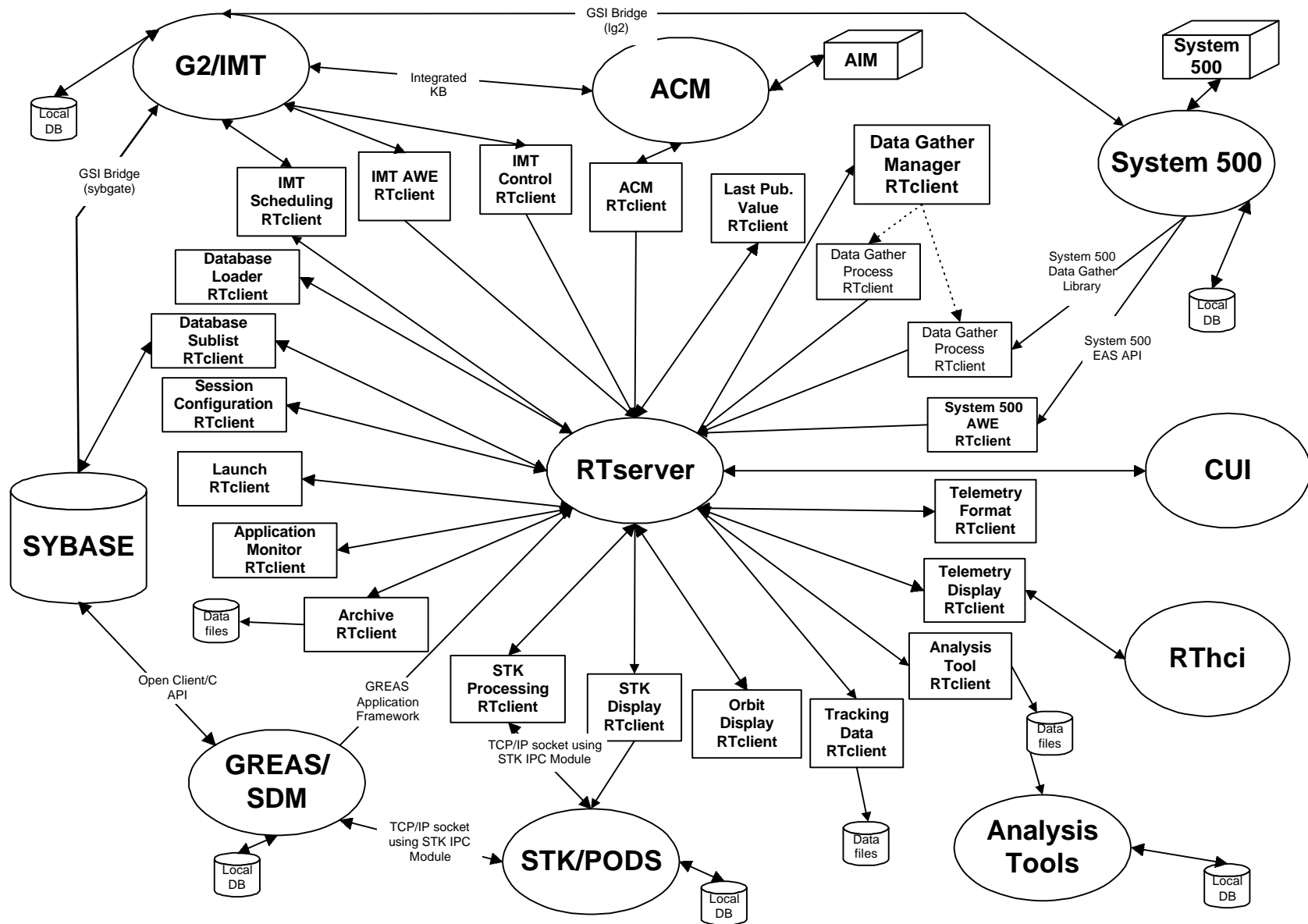
- **Reduce cost and complexity by relying on native features of GOTS and COTS products.**
- **Reduce lifecycle costs**
  - Rely on vendor/supplier maintenance support for COTS products.
  - Adherence to applicable (and sensible) design and implementation standards
    - POSIX, ANSI, IEEE, etc.
- **Provide abstractions for replaceable components**
  - Small, specialized, loosely coupled components
- **Model the architecture on events, not the HMI**
  - Message- and event-driven processing

# RSC Architecture Overview

---

- **RDT&E Support Complex, Kirtland AFB, NM**
- **COTS-based real-time architecture**
- **Integration of “horizontal” and “vertical” applications**
  - Horizontal: Sybase, GREAS, Mathematica, PVWAVE
  - Vertical: STK, OASYS, PODS, IMT, AIM, ACM, System 500
- **Commercial message-oriented middleware provides the “glue”**
  - Message-based, publish-subscribe, event-driven paradigm
  - Standardized, extensible message passing protocol eases integration of COTS and legacy products
  - Messaging bus for all system communication: status, control, telemetry data

# Middleware Interfaces



# Scalability and Adaptability

---

- **Scalability**
  - 1 to N workstations
  - 1 to N “strings” or “instances”
- **Portability**
  - Standards conformance.
  - COTS products drive compatibility
- **Individual components easily replaceable**
  - Top-down architecture
  - Functionally decomposed into small, specialized units
  - Design lends itself to OO migration

# CORBA and MOM

---

- **“So, what’s your CORBA migration path?”**
- **Separate but comparable technology**
  - CORBA relies on synchronous transactions
  - MOM is asynchronous
    - Important to the event-driven paradigm - applications need to generate and respond to multiple different events in a timely manner
  - High speed message passing is **CRITICAL** to real-time TT&C
  - Current implementations of CORBA have insufficient performance characteristics to fulfill high data-rate mission requirements
- **No asynchronous messaging standard for CORBA**
  - Talarian is working with the OMG to define an asynchronous messaging standard for CORBA (see earlier philosophy)

# The Bottom Line

---

- **This architecture exists and flies satellites today; it is not just on paper**
- **There are less than 200K LOC (custom) in this system**
- **Total investment in deploying this architecture to date is less than \$10M**
- **It exists today, and it will still be new tomorrow**