



JPL

# COTS-based OO-Component Approach for



## Software Inter-operability and Reuse



### (Software Systems Engineering Methodology

**GSFC-SEW25, 11/29/00**

L. Hall, Principle Author & Technical Group Supervisor  
ReUse/OO-Component Team  
(C. Hung, C. Hwang, A. Oyake, J. Yin)

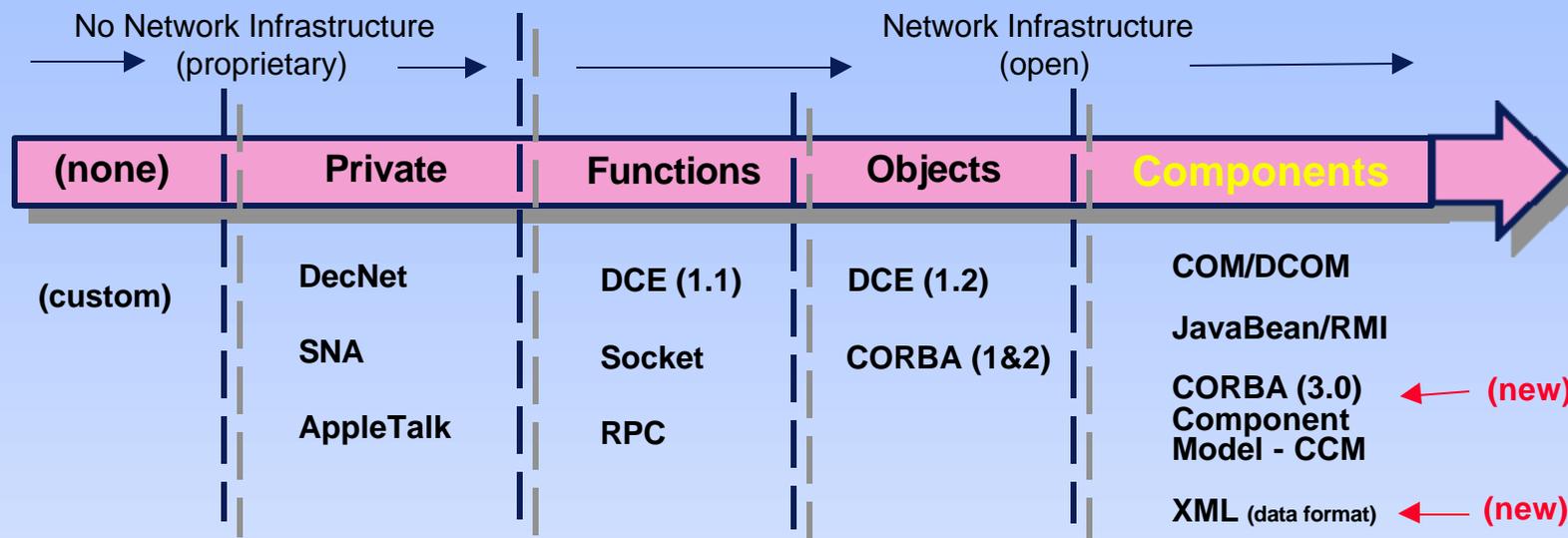


*DC&SE Group  
LH(369)-12/04/2000*



# Distributed Computing Technology Evolution

## \*\*\* S/W ReUse - Components \*\*\*

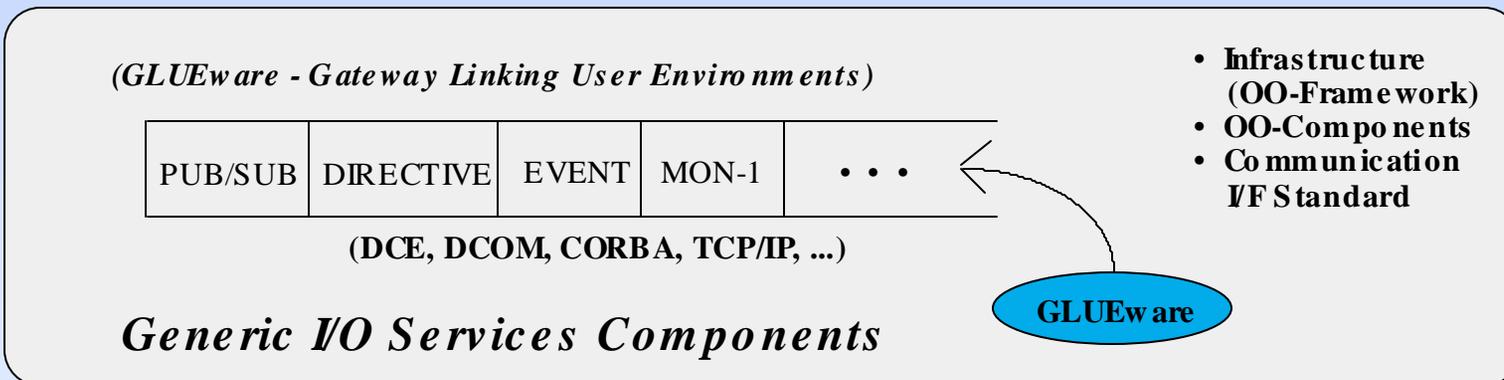
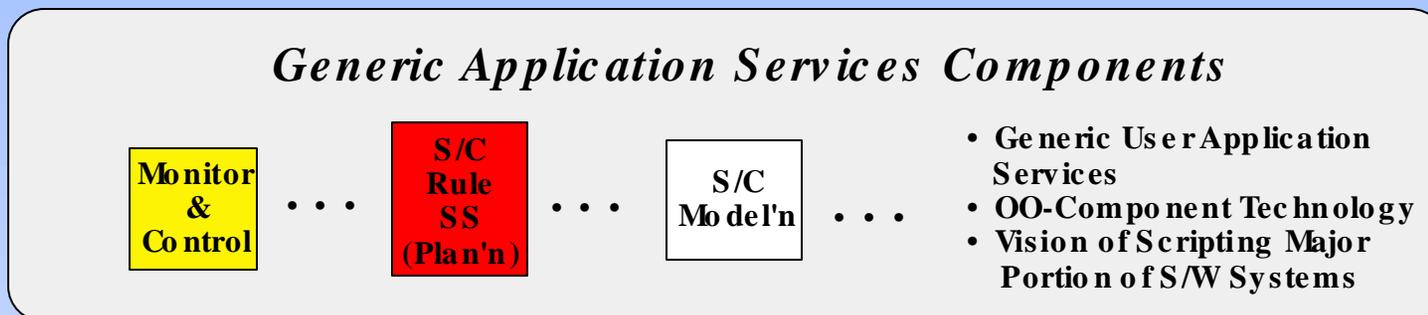


- **Distributed Computing allows modern software structure to occur across distributed networks in an increasingly flexible and effective manner.**
- **Software Component Technology allows distributed application pieces to flexibly be reused, inter-operate, and evolve over time.**
- **OO-Component Approach to S/W SE provides reusable infrastructure and generic s/w components for common services across subsystems (includes communications and application services)**
- **Benefits (= Cost Savings) - Interoperability (old/new/diff.), Extensibility (design patterns), Easy Reuse, Easy Assembly (additions after delivery), Runtime Flexibility (static/dynamic/swap, Adaptability, Enforce OO Design (standard), Development Flexibility (independent environments, different implementations of services)**





# Potential ReUsable Generic Communication & Application Component-based S/W Services



- Tiered design architecture for flexible change or growth
- Separate application-specific code from support services
- Well-defined API's with wrapped configurable services





## Background & Objectives

JPL

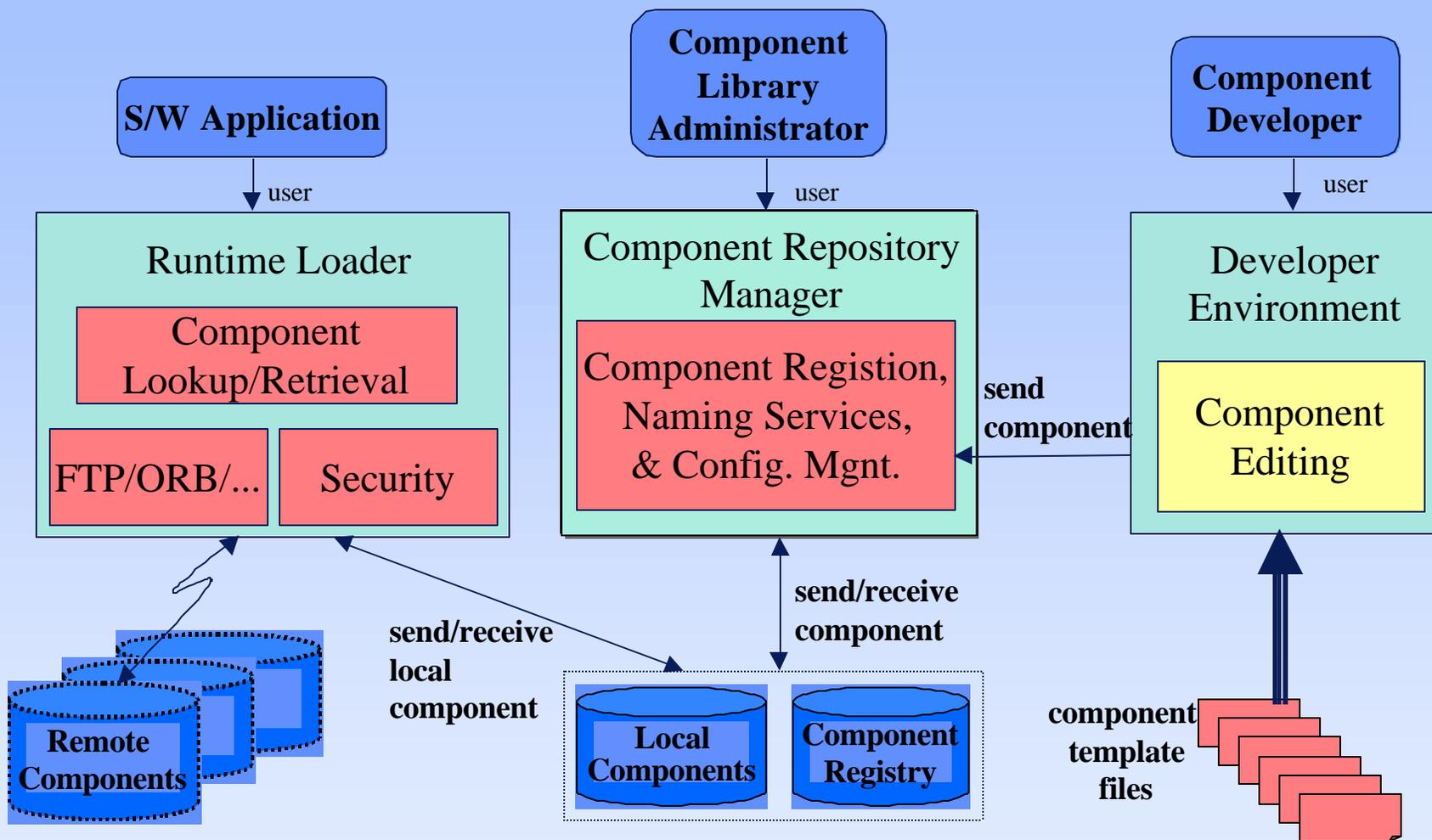
- **Background:**
  - Performed same application prototypes based on 2 open standards [used same tiered architecture & OO-component approach w/C++, UNIX]
    - 1) Microsoft's COM - I/F definition to wrap DCE-based custom services
    - > 2) COTS-based CORBA - replaced custom services w/COTS components
- **Objectives:**
  - S/W engineering methodology or framework to allow an architectural roadmap which promotes flexible selection and reuse of system elements
  - Reusable common S/W services and supporting infrastructure
  - New technology risk mitigation effort to reduce traditional development problems and gain benefits of forward-thinking solutions
- **How:**
  - Object-oriented techniques (design patterns, inheritance, wrapping, ...)
  - S/W component structuring concepts based on open standards for reusable , reconfigurable common services
  - Max use of proven COTS services, yet no inter-mingling inside application-specific code





# Component Deployment & Management Strategy or Roadmap [Backup]

JPL



- Elements Recommended to Support Component Implementation Approach
- Build upon best element of different standards to create a robust, generic framework





## Prototyped Application & Results

JPL

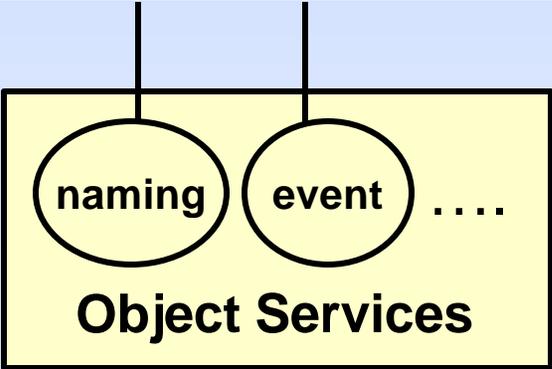
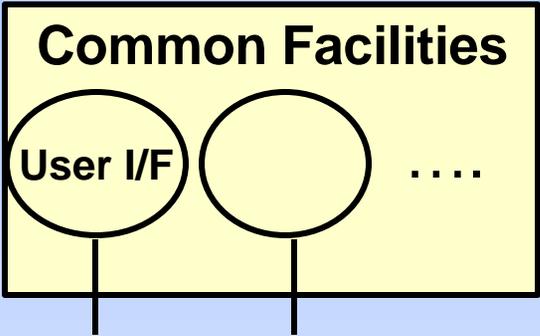
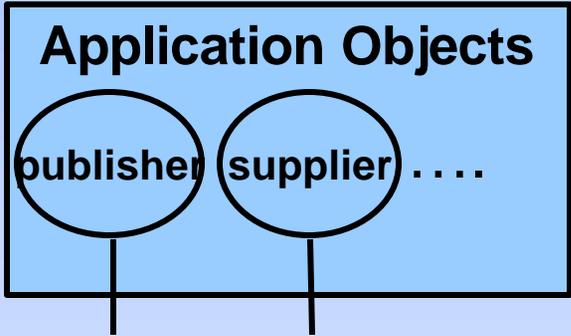
- Prototyped Publish/Subscribe and Naming Services within NASA Deep Space Network's Monitor & Control Infrastructure Services (MCIS)
- Distributed computing architecture based on Common Object Request Broker Architecture (CORBA) open standards for communication infrastructure support
- COTS implementation of CORBA services provided via Adaptive Communication Environment - The ACE ORB (ACE-TAO)
  - TAO's Naming Service
  - TAO's R/T Event Service [or Notification Service]
- Build COTS-based OO-component M&C Pub/Sub services by wrapping CORBA ACE-TAO services to allow structured code that adheres to layered architecture and allow flexible configuration/change, future additions, cleaner API's and development environment, ...
- Exercise typical Publish/Subscribe operational scenarios between distributed DSN Subsystems to demo meeting actual requirements
- Document summary of lessons learned through prototyping experience



DC&SE Group  
LH(369)-12/04/2000

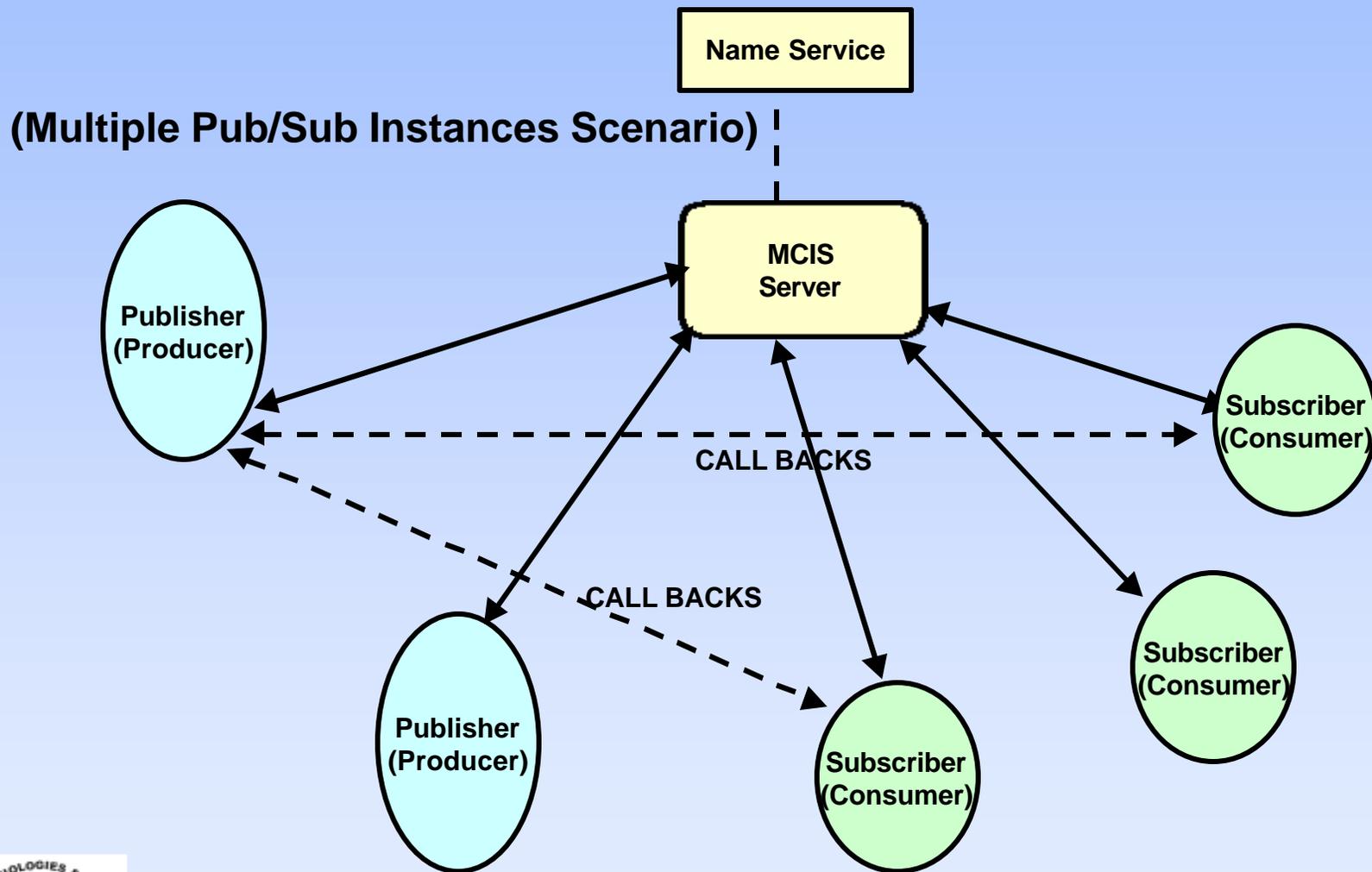


**CORBA Overview**  
OMG Reference Model Architecture  
[Backup]





# DSN MCIS Pub/Sub Architecture Summary

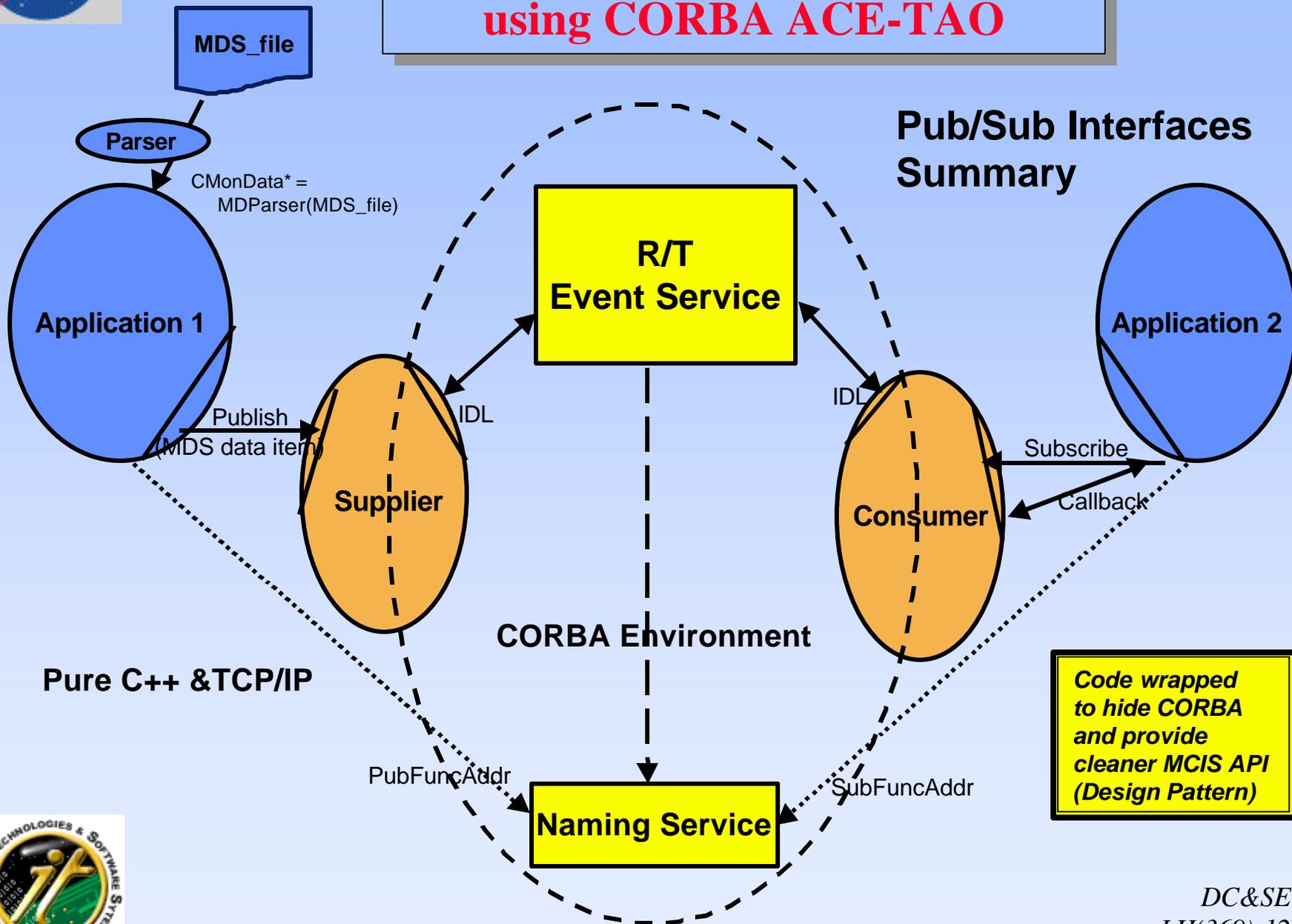


MCIS - Monitor & Control Infrastructure Services



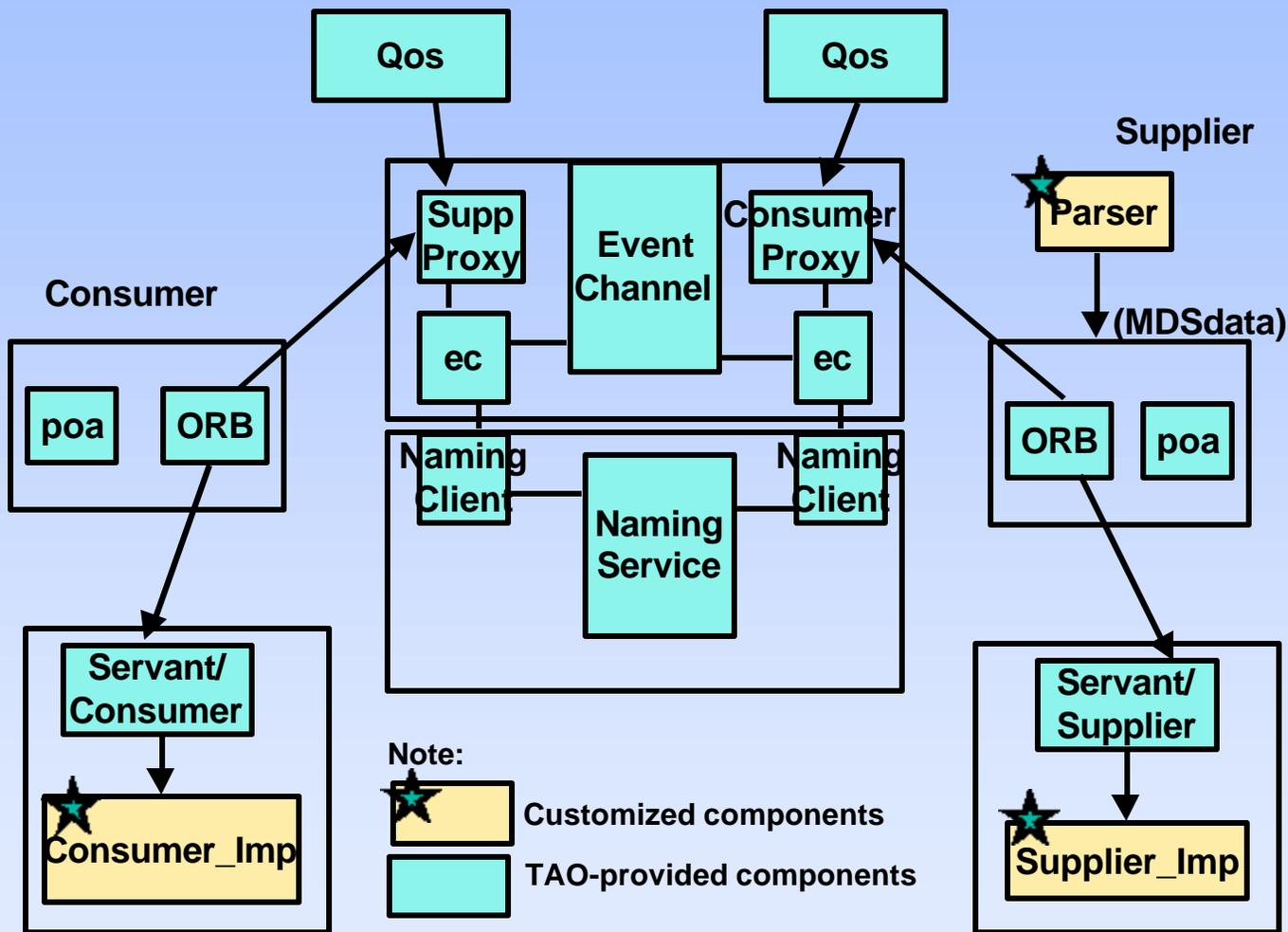
# M&C Pub/Sub Common Services using CORBA ACE-TAO

JPL





# CORBA vs. Custom Components used for MCIS Implementation

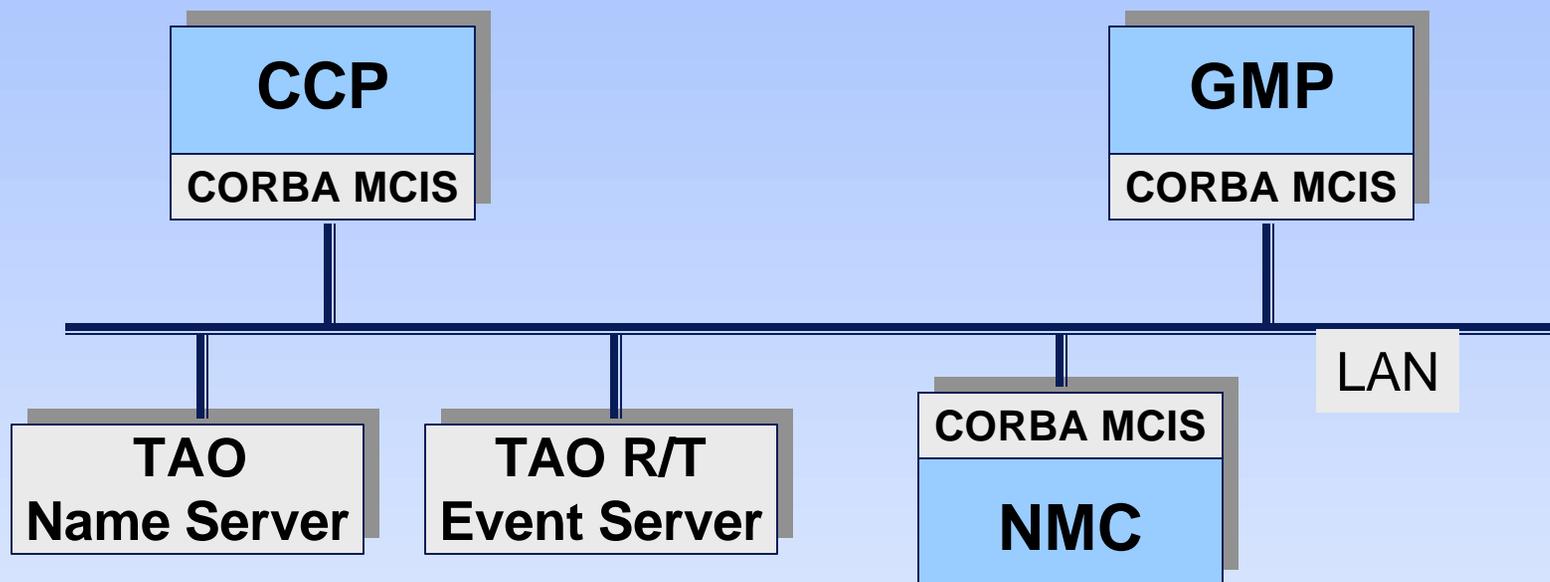


- In-house developed code < 1% of COTS (unstripped)  
[- 5% code compared to in-house socket dev.]





# CORBA MCIS Demonstration Test Configuration [Backup]



**NMC** - Network Monitor and Controller Subsystem

**GMP** - GCF Monitor Processor Subsystem

**CCP** - Command Control Processor Subsystem

 - TAO services that run on the same or different hosts

 - applications that run on the same or different hosts





# Lessons Learned From CORBA

JPL

## PROS of CORBA

- CORBA is a **platform independent** distributed computing solution. It is a standard - <http://www.omg.org>
- CORBA IDL (Interface Definition Language) defines a language independent specification, as well as language mappings for most of the major languages.
- CORBA provides many tools to solve very many disparate system engineering problems.
- There are many ORB implementations - Compared to DCE, where there has only been one (Transarc) vendor.
- CORBA allows you 100% leverage of your legacy systems, so old code can be encapsulated under IDL interfaces.

## CONS of CORBA

- There is a very steep learning curve with CORBA. It requires strong understanding C++ and IDL.
- Requires understanding distributed computing paradigms; e.g. pub/sub remote procedure calls, etc.
- CORBA assumes you are familiar with OOP principles such as polymorphism, abstraction, etc.
- Not all ORBs are created equal! - Many ORBs do not support the **latest** CORBA versions (versions 2.3 - 3.0).
- Commercial ORBs tend to be expensive > \$10,000..





## Lessons Learned from CORBA

JPL

### PROS of CORBA (cont.)

- CORBA allows integration of other Component Models such as EJB and COM.
- CORBA provides many standard services out of the box. Traditionally at JPL, tools and services have been developed to support distributed computing applications.
- CORBA rapidly accelerates the development time for complex applications. E.g. this demo took less than 2 months to code.
- CORBA provides an Interoperable Naming Service (INS) which automates the object name mapping.
- CORBA provides an Event or Notification Service - Note under DCE, JPL created customized solutions for event handling.

### CONS of CORBA (cont.)

- **Steep learning curve.**
- Configuration of systems may be challenging (language compiler versions, OS versions, exception handling, etc).
- Configuration of the ORB is non-trivial (specifically configuring services), due to non-standard implementation across COTS.
- Requires an understanding of dynamic memory issues and dynamic types (`_var` types, `type any`).
- Full implementation of services is not guaranteed in all COTS.





# Lessons Learned from ACE-TAO

## Pros of ACE-TAO

- The ACE-TAO ORB is **open source** so debugging is simplified by the scrutiny of many users.
- ACE-TAO provides very good debug-level messages, so developers can easily identify problems.
- ACE-TAO implements many of the CORBA Services and even extends them (real times services, etc).
- There is a large user base in the scientific and business community, and lots of user feedback, and a very active mailing list.

## Cons of ACE-TAO

- System requirements are large and involve sys-admin assistance (compilers, OS upgrades, etc).
- Large disk footprint > **2 Gigs !!** It forced team to obtain a new disk drive.
- ACE-TAO make-file rules to build ACE-TAO components can be complicated.
- Though ACE-TAO supports many CORBA services, it may not fully implement them, or may provide a non-standard service.





## Lessons Learned from ACT-TAO

### Pros of ACE-TAO (cont.)

- Lots of **direct support** from the DOC team at UCI and Washington University.
- In developing our applications, the performance was comparable to applications developed using DCE (in our small testbed). I.e., no noticeable performance degradation.
- The **Name Service** is easily configurable and provides a novel way for clients and servers to “discover” each other.
- The **Event Service** is also very easy to configure.

### Cons of ACE-TAO (cont.)

- Limited documentation - The documentation assumes knowledge of CORBA.
- The Event Service is light-weight. Recently received full Notification implementation is a plus.
- Both the Event and Name Service may leave behind ghosts when killed (i.e., dead processes showing up as active), which can interfere with a new Service.





JPL

## CASE 1. - Developing applications using Design Patterns

### Benefits of Patterns

CASUtionsOO FrameworkBe(ACECASunderlyionsframework for TAO)atterns



*DC&SE Group*  
*LH(369)-12/04/2000-*



## Summary of Conclusions

JPL

- Challenges of developing efficient, robust, extensible concurrent applications can be difficult. Distributed computing addresses not easily solved complex topics that are less problematic for non-concurrent, stand alone applications.
- Should avoid creating in-house infrastructure support services for distributed computing due to its large scope and complexity. Should instead concentrate on application-specific developments.
- Best approach to distributed computing system development is by adopting open standards for inter-operability and to avoid continuous reinvention.
  - Open standards like CORBA address all areas of complexity that arise in interconnected systems.
- Recommend taking advantage of OO-component structuring techniques coupled with a modular architecture and COTS to achieve reuse (requires good systems engineering assessment and design).

**OBJECTIVE:** Provide advanced technical solutions via s/w prototypes and supporting framework, tasks concentrate on their application-specific domain solution(s)



DC&SE Group  
LH(369)-12/04/2000-