

# **An Object-Oriented Model for Developing a Reference Architecture of the Air Force Satellite Control Network**

**Sergio J. Alvarado, Ph.D.**

**Intelligent Systems Section  
Information Technology Department  
Computer Systems Division  
The Aerospace Corporation**

# Outline

- Objectives, Motivation, and Background
- System Development Cycle
- Object Models, Dynamic Models, and Functional Models
- System Analysis Using Object-Oriented Techniques
- System Design Using Object-Oriented and Distributed Computing Techniques
- System Implementation Using Commercial-Off-The-Shelf Products
- Conclusions

## Background

- Programs for telemetry processing, tracking, and commanding (TT&C) within the Space and Missile System Center (SMC) have been impacted by declines in the defense budget, the need to upgrade existing ground control systems, and an increase in the number of operational satellites.
- These constraints have resulted not only in increased workloads for ground-control operators, but also in a decrease in the workforce and a decrease in SMC's capability to develop customized ground control systems.
- To handle these problems, programs such as the AFSCN are replacing their legacy mainframe systems with open-architecture systems using object-oriented technologies, distributed-computing technologies, and COTS products.

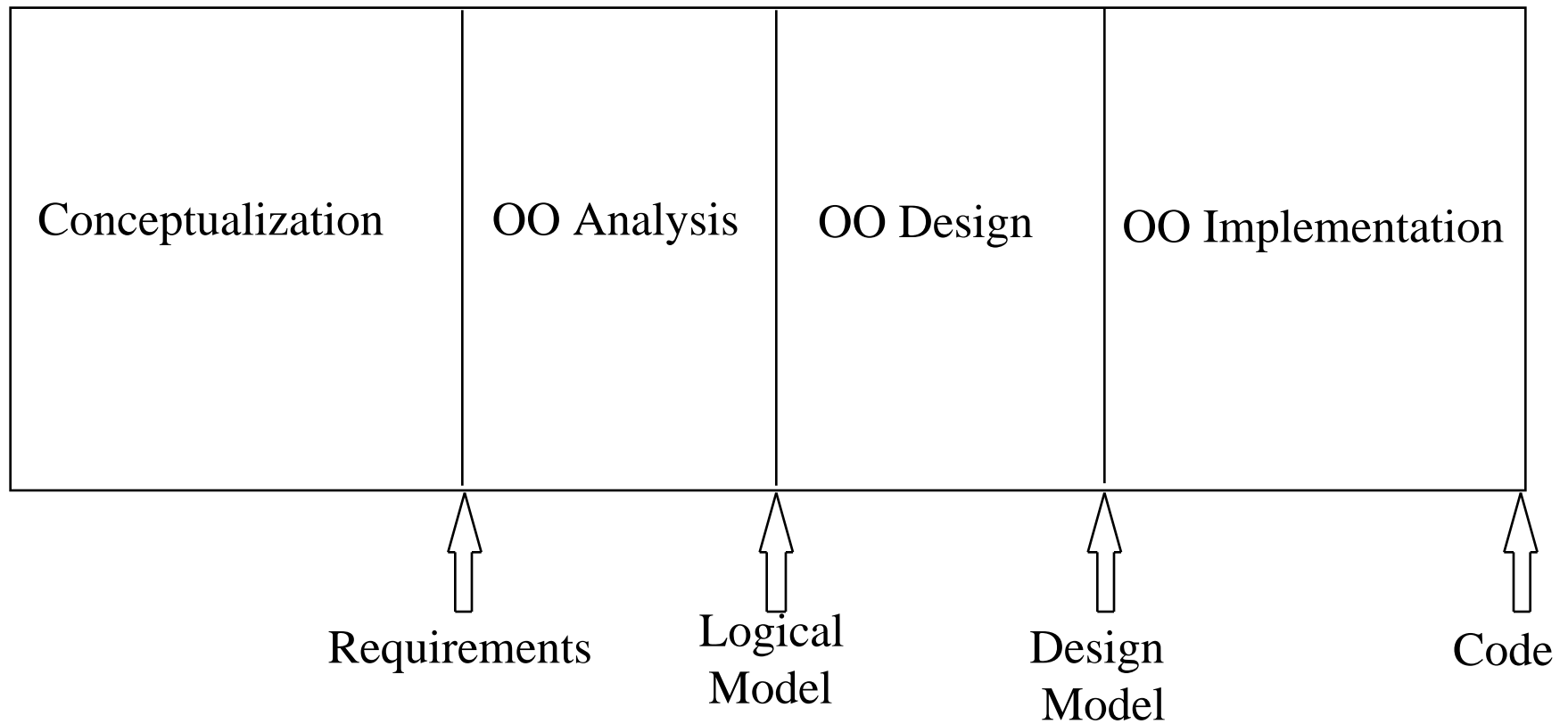
## Objectives

- Object models, dynamic models, and functional models of satellite-support activities are being built using the Object Modeling Technique (OMT) for analyzing the architecture's requirements and specifying the architecture's basic building blocks.
- Requirement analysis involves mapping components of the OMT models to requirements and specifications for AFSCN segments .
- Specification of the architecture's components involves mapping the OMT models to an existing functionally-based reference architecture of the AFSCN in order to characterize architectural components in terms of data and processes organized by classes of objects for satellite-support activities.
- Evaluating completeness of AFSCN requirements and enhancing the level of detail of the functionally-based reference architecture of the AFSCN.

## Related Previous Work

- Object-Oriented Analysis for Command and Control Segment Upgrade.
- Reference Architecture for the SSCS (core TT&C).
- Reference Architecture for the RMS.
- Object-Oriented Analysis of Milstar Mission Control Segment.
- Functional Decomposition of the AFSCN Ground Segment.
- Initial Development of the SSCS FRD.

# Object-Oriented System Development

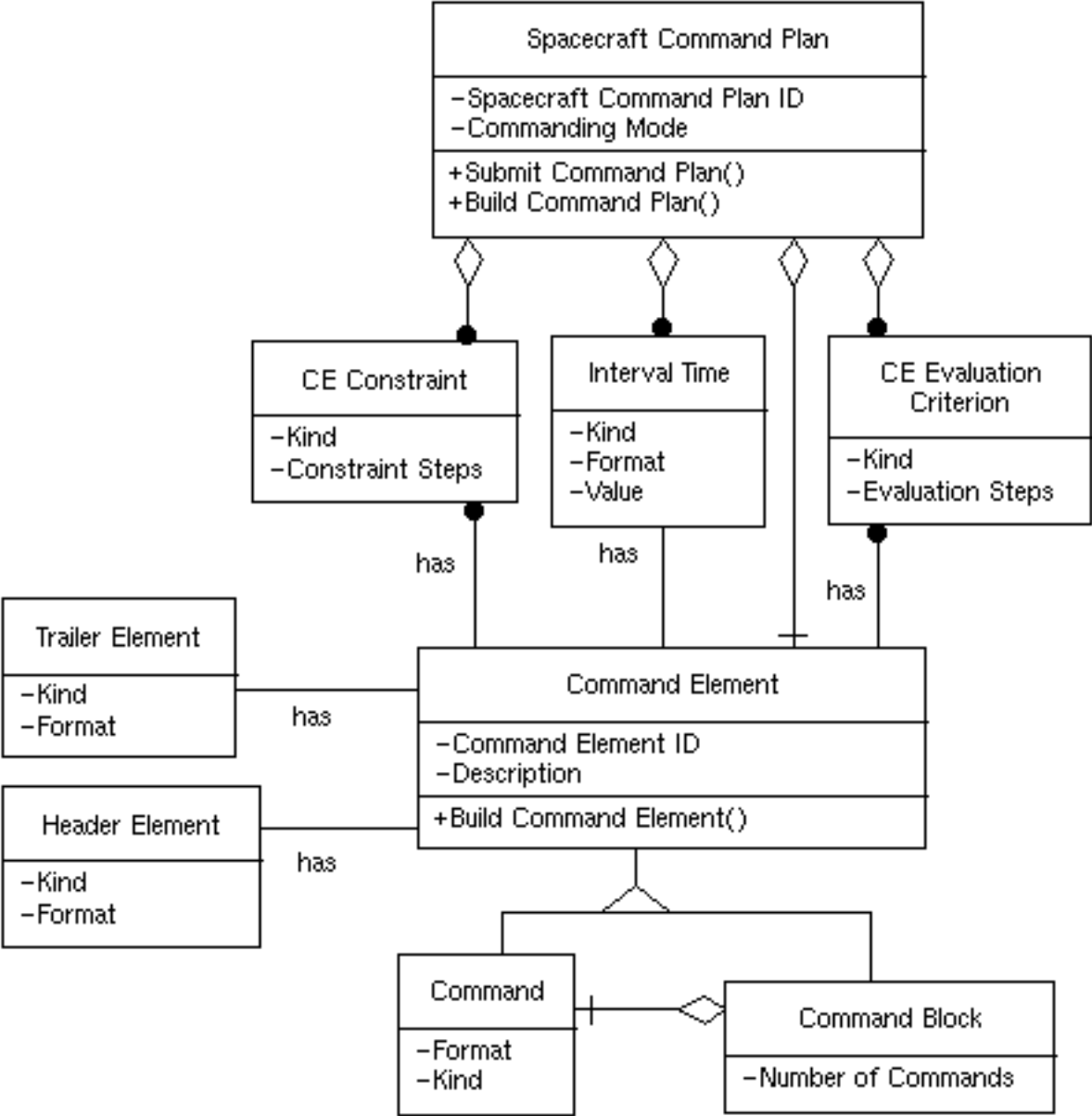


# Analysis Document

**Analysis Document = Problem Statement +  
Object Model +  
Dynamic Model +  
Functional Model**

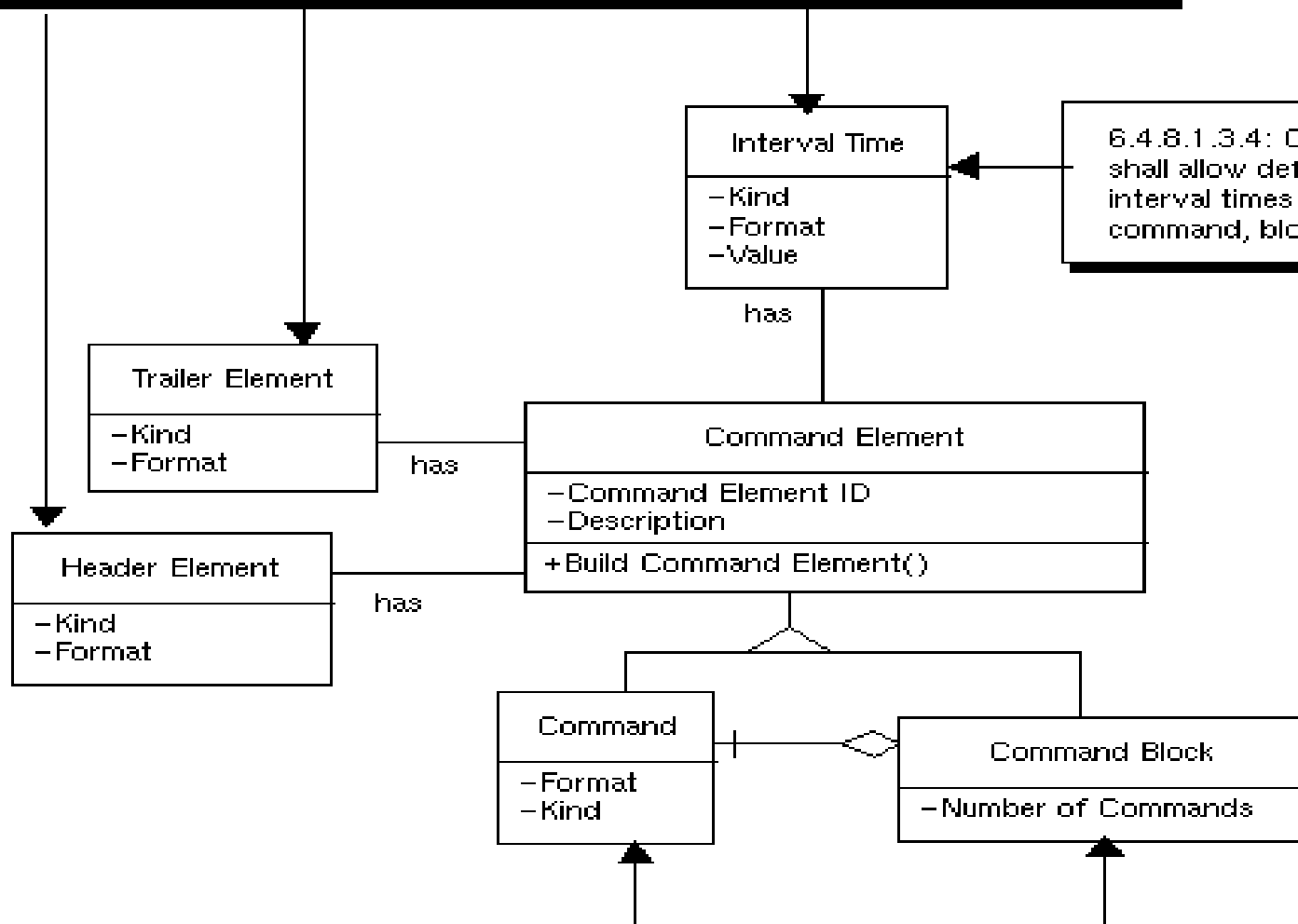
- Build an object model (object diagrams and data dictionary).
- Develop a dynamic model (state diagrams and event-flow diagrams).
- Construct a functional model (data-flow diagrams).
- Verify, iterate, and refine the three models.

# Inheritance and Associations



# Example of Requirement Mapping

6.4.8.1.3.1: Core TT&C shall format data streams ... with: (a) the command headers ...; (b) one or more command data records with interval between each ...; and (c) command trailers ...



6.4.8.1.2: Core TT&C shall allow selection of the following command types: (a) single; (b) block ...

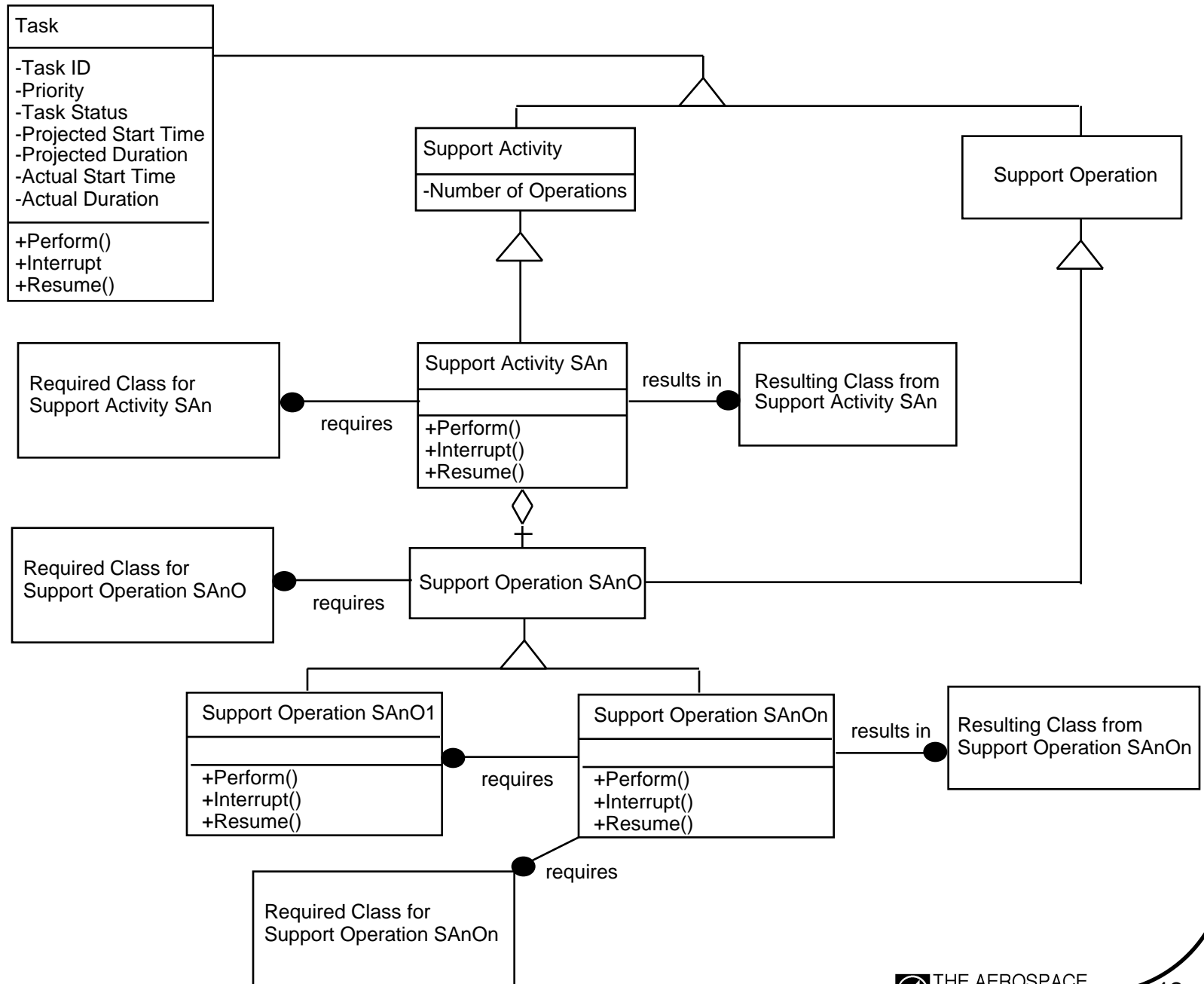


# Design Document

**Design Document =      Basic System Architecture +  
High-Level Strategy Decisions +  
Detailed Object Model +  
Detailed Dynamic Model +  
Detailed Functional Model**

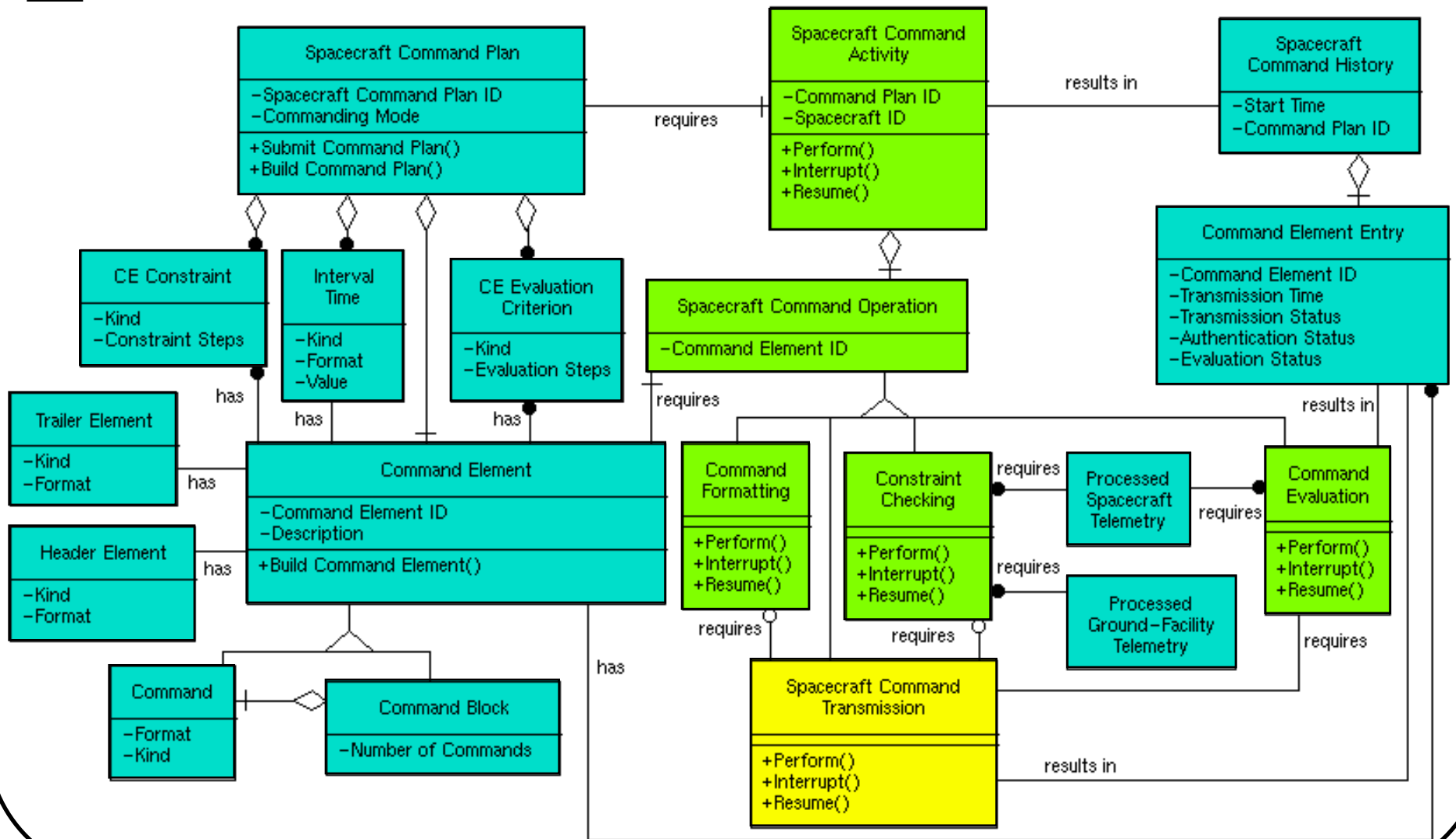
- Organize the system into subsystems.
- Identify concurrency issues.
- Allocate subsystems to processors and tasks.
- Choose the basic strategy for implementing data stores in terms of data structures, files, and databases.
- Obtain operations for the object model from the other models.
- Design algorithms to implement operations.
- Determine the exact representation of classes and associations.
- Package classes and associations into modules.

# Design Pattern for Support Activities



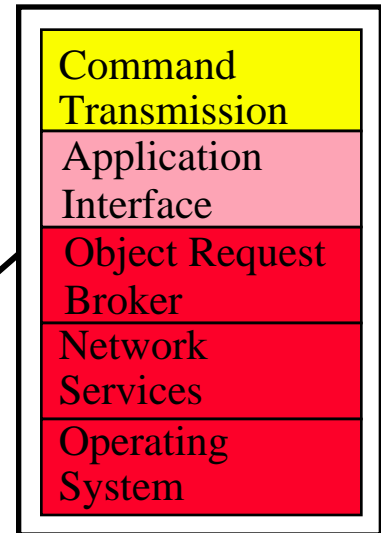
# Composite Classes for Spacecraft Command Activity

- Command Repository**
- Command Engine**
- Command Transmission**

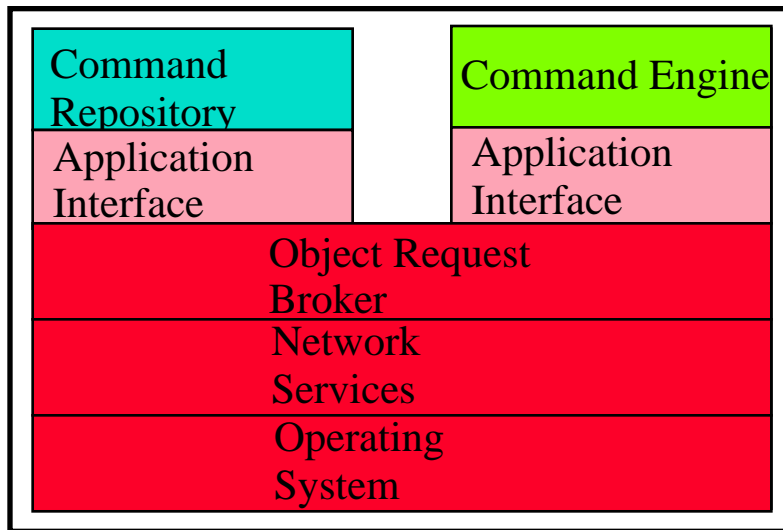


# Client/Server Architecture

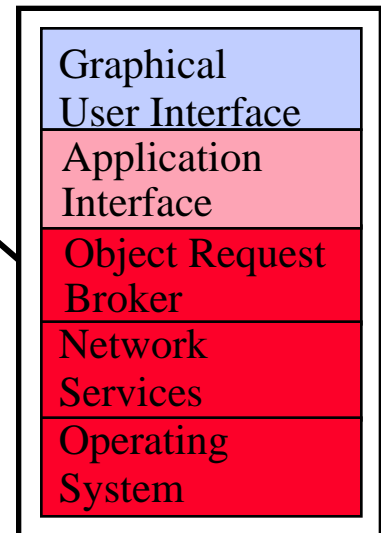
Command Front-End Host



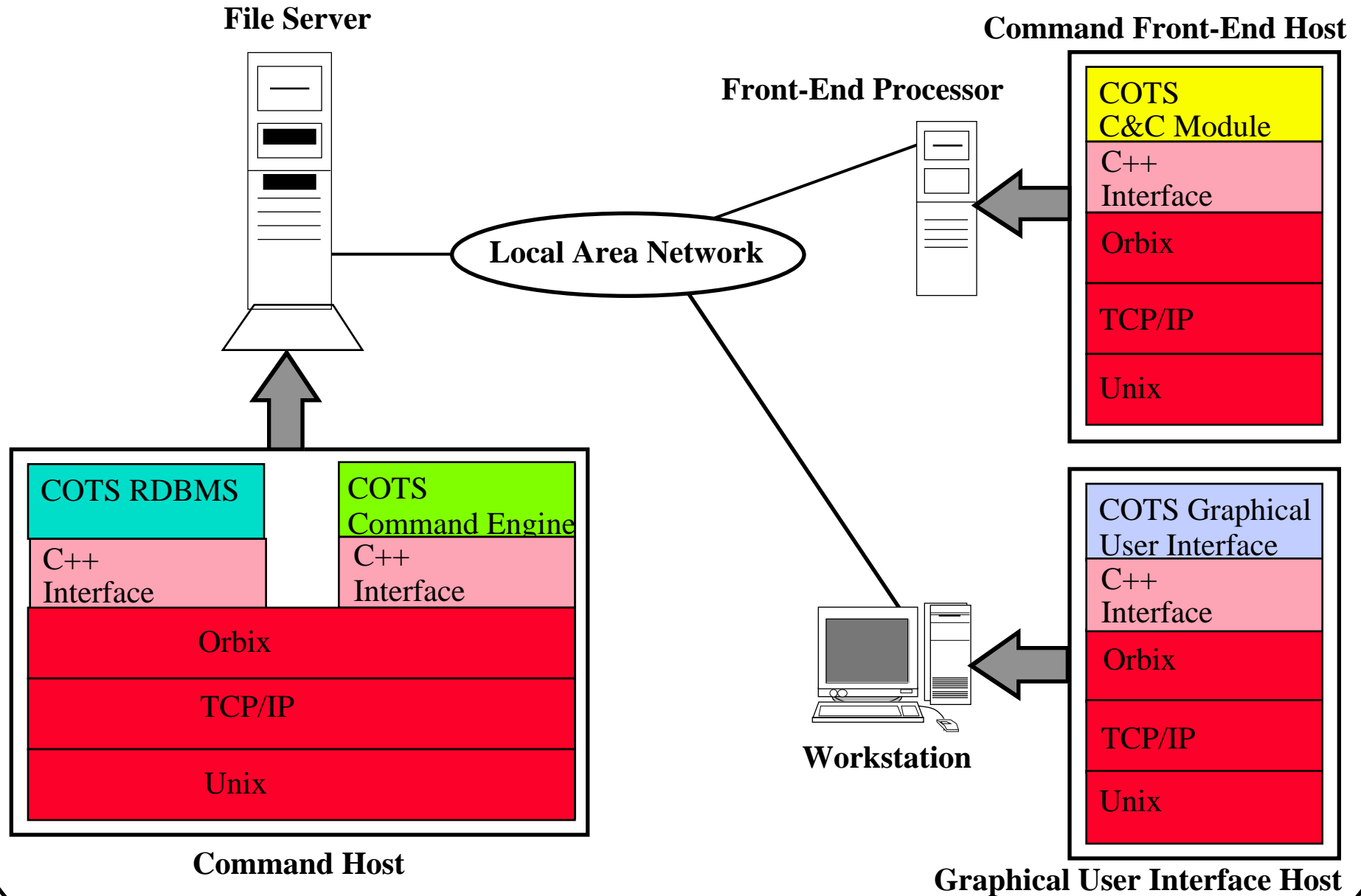
Command Host



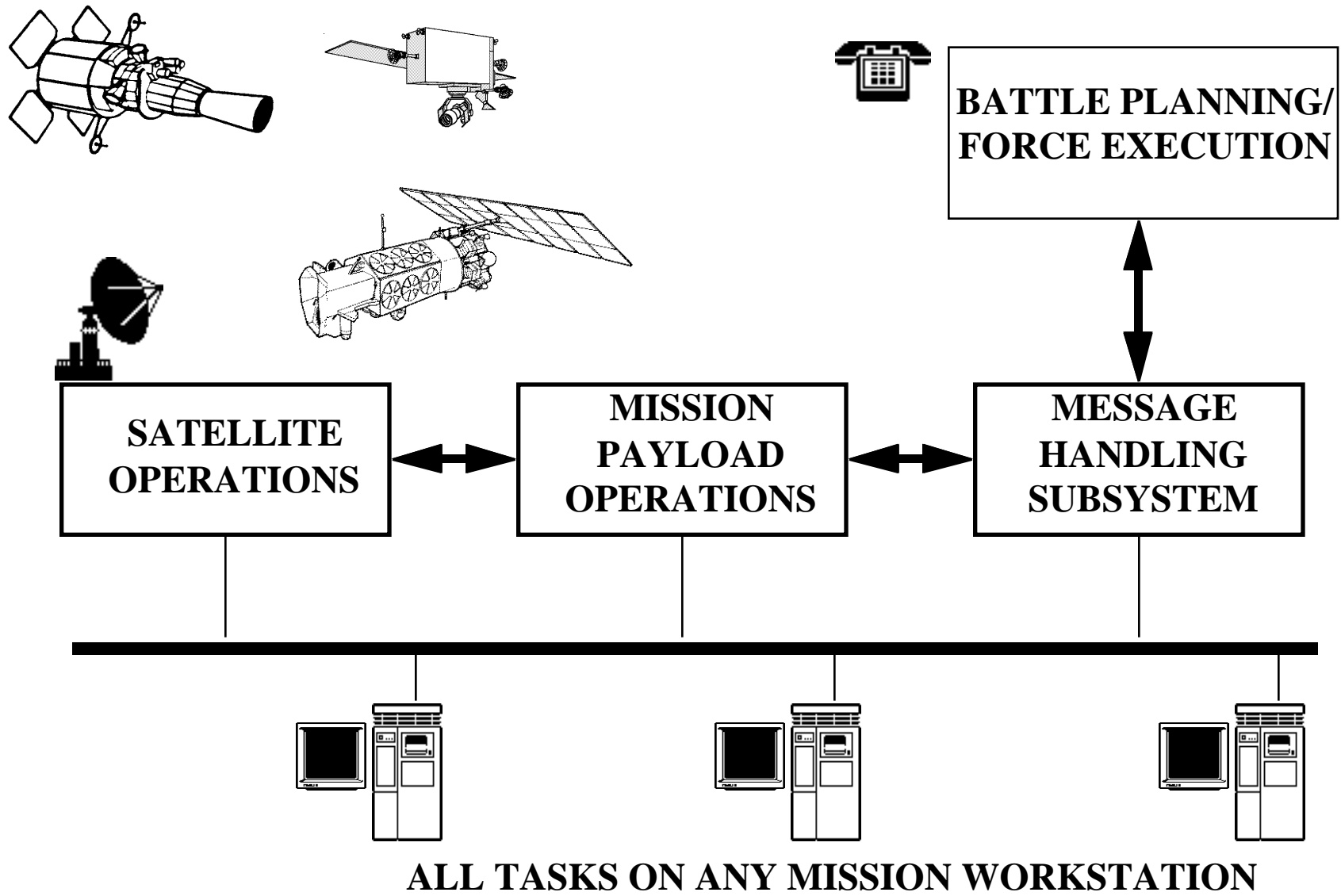
Graphical User Interface Host



# Example of a COTS-Based System Implementation



# Multimission Ground Processing Testbed: Laboratory for Testing Emerging Information Technologies



# Multimission Ground Processing Testbed Technologies

- Distributed Open System and OO Technology:
  - Workstation-based.
  - Fiber Distributed Data Interface (FDDI) Local Area Network (LAN).
  - Architecture for assessing portability and reusability of OO components.
- Human Computer Interface (HCI):
  - Multi-satellite TT&C operator displays.
  - Commonality/uniformity across multiple programs.
- Middleware for COTS integration:
  - Generic interface definitions.
  - Middleware layer for COTS product communication.
- Expert systems:
  - Anomaly diagnosis, and command verification and validation.

## COTS Products in the Multimission Ground Processing Testbed

• <b>Telemetry Server</b>	<b>Loral 550</b>
• <b>Middleware</b>	<b>RT SmartSockets</b>
• <b>HCI</b>	<b>SAMMI, Dataviews, TeleUse</b>
• <b>Decision Support</b>	<b>G2, Nexpert Object</b>
• <b>Orbit Determination</b>	<b>STK, PC SOAP</b>
• <b>Facility Control</b>	<b>MAS</b>
• <b>Database</b>	<b>SyBase, Oracle</b>
• <b>Commanding</b>	<b>Spacecraft Command Lang</b>
• <b>Scheduling</b>	<b>GREAS, ROSE</b>
• <b>External Communications</b>	<b>Message Handling System</b>



**Current**



**Evaluated**

## Conclusions

- The challenges posed by replacing legacy mainframe systems with open-architecture systems can be met with object-oriented technologies, distributed-computing technologies, and COTS products.
- A system development approach involving object-oriented technologies, distributed-computing technologies, and COTS products encourages system developers to think in terms of the application domain through the life cycle of the system.
- The essence of the development is the identification and organization of application-domain concepts.
- The payoff comes from addressing front-end conceptual issues, rather than back-end implementation issues.