

# **An Evaluation of Object-Oriented Architecture Models for Satellite Ground Systems**

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

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

# Outline

- Background and Objectives.
- Example of an Architecture Model for a Satellite Commanding System.
- Example of COTS Product Dependencies in Architecture Models.
- Problems of COTS Product Dependencies in Architecture Models:
  - Lack of Requirement Traceability.
  - Lack of Interface Definition.
  - Lack of Risk Mitigation.
  - Lack of Documentation.
- Avoiding COTS Product Dependencies in Architecture Models.
- Conclusions.

## Background

- Legacy mainframe systems for mission control, telemetry processing, tracking, and commanding within the Space and Missile Systems Center are being replaced by open-architecture systems using object-oriented (OO) technologies, distributed computing technologies, and commercial-off-the shelf (COTS) products.
- Developing these ground systems is increasingly dependent upon architecture models that provide concurrent multiple views of the systems and serve as the basis for the systems' design and implementation.
- Because of schedule deadlines and/or lack of perceived benefits of using architecture models, the development process often becomes one of making COTS products work together as opposed to one of using architecture models to provide the structure for COTS product integration.
- As a result, these ground systems may pose short-term and long-term development risks for developers and customers.

## Objectives of This Work

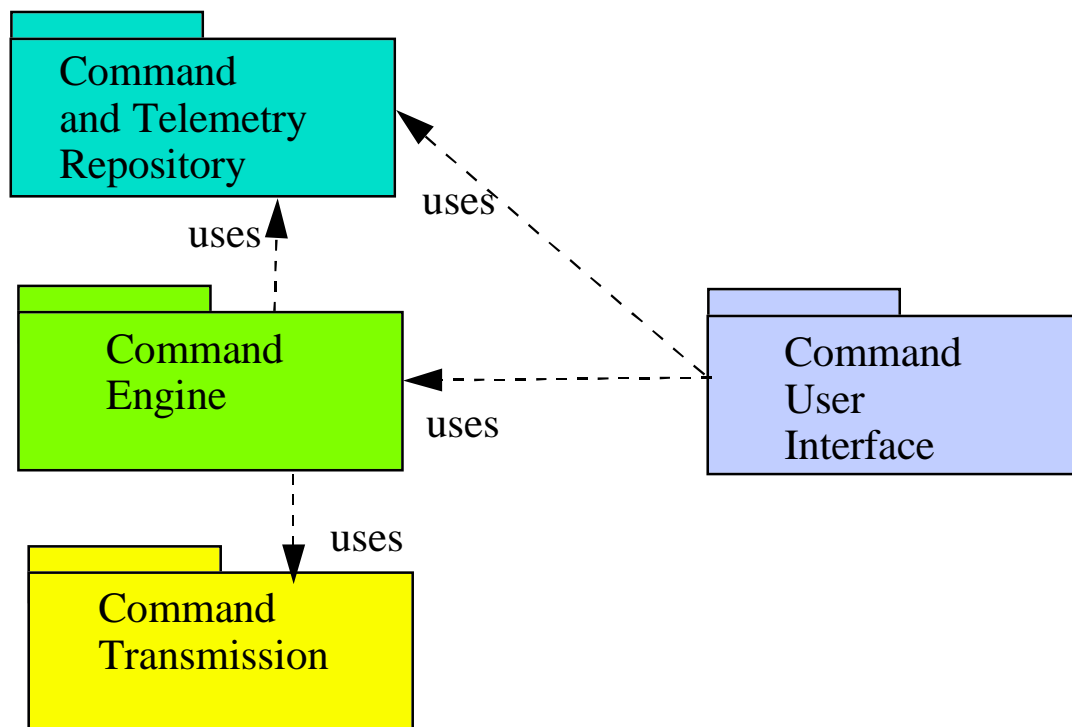
- To use an existing functionally-based reference architecture for satellite telemetry processing, tracking, and commanding (TT&C) as a basis for identifying the data and processes underlying the classes of objects for satellite-support activities.
- To use these classes of objects to build an OO TT&C model with multiple architecture views, including logical organization, dynamic behavior, process decomposition, software organization, and physical realization.
- To use the OO TT&C model to analyze technical issues in software acquisition of OO satellite ground systems, with emphasis on replicating and categorizing the problems of COTS product dependencies in architecture models of satellite ground stations.

## Related Previous Work

- Alvarado, S. J. (1997). An Object-Oriented Model for Developing a Reference Architecture of the Air Force Satellite Control Network. *Proceedings of the Ground System Architecture Workshop, GSAW97*. The Aerospace Corporation. El Segundo, CA.
- Lockheed Martin Advanced Concepts Center and Rational Software Corporation (1996). *Succeeding with the Booch and OMT Methods: A Practical Approach*. Menlo Park, CA: Addison-Wesley.
- Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Software Architecture*. Reading, MA: Addison-Wesley.

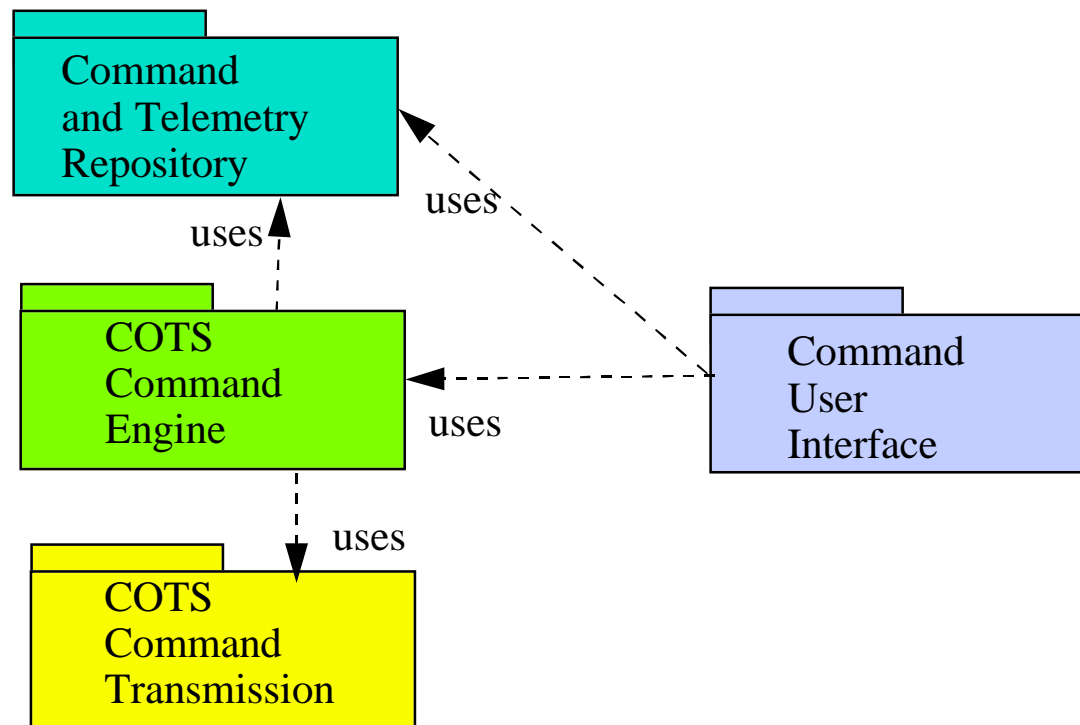
## Example of an Architecture Model: Class Category Diagram for a Satellite Commanding System

- Class categories (or subsystems) of a commanding system relate to one another via a “uses” (or client-server) relationship from the using class category (or client) to the used class category (or server).



## Example of COTS Product Dependencies in a Class Category Diagram for a Satellite Commanding System

- COTS products for the command engine and command transmission are included in the class-category diagram, but their logic organization and dynamic behavior are not further represented using class diagrams and message trace diagrams.



## Problems of COTS Product Dependencies in Architecture Models: Lack of Requirement Traceability

- Consider the following requirement:

**Requirement 6.4.8.1.6:** TT&C system shall provide the capability to verify preconditions for command activities, to validate command sequences, to authenticate commands, to verify correct command transmission, and to verify correct command execution.

- Detailed requirements such as 6.4.8.1.6 cannot be traced down within the high-level representation provided by COTS dependent class category diagrams.
- As a result, it is not possible to verify that all system requirements are captured and represented correctly.
- Thus, consistency checking and validation of the architecture model cannot be completely performed within COTS dependent class category diagrams.

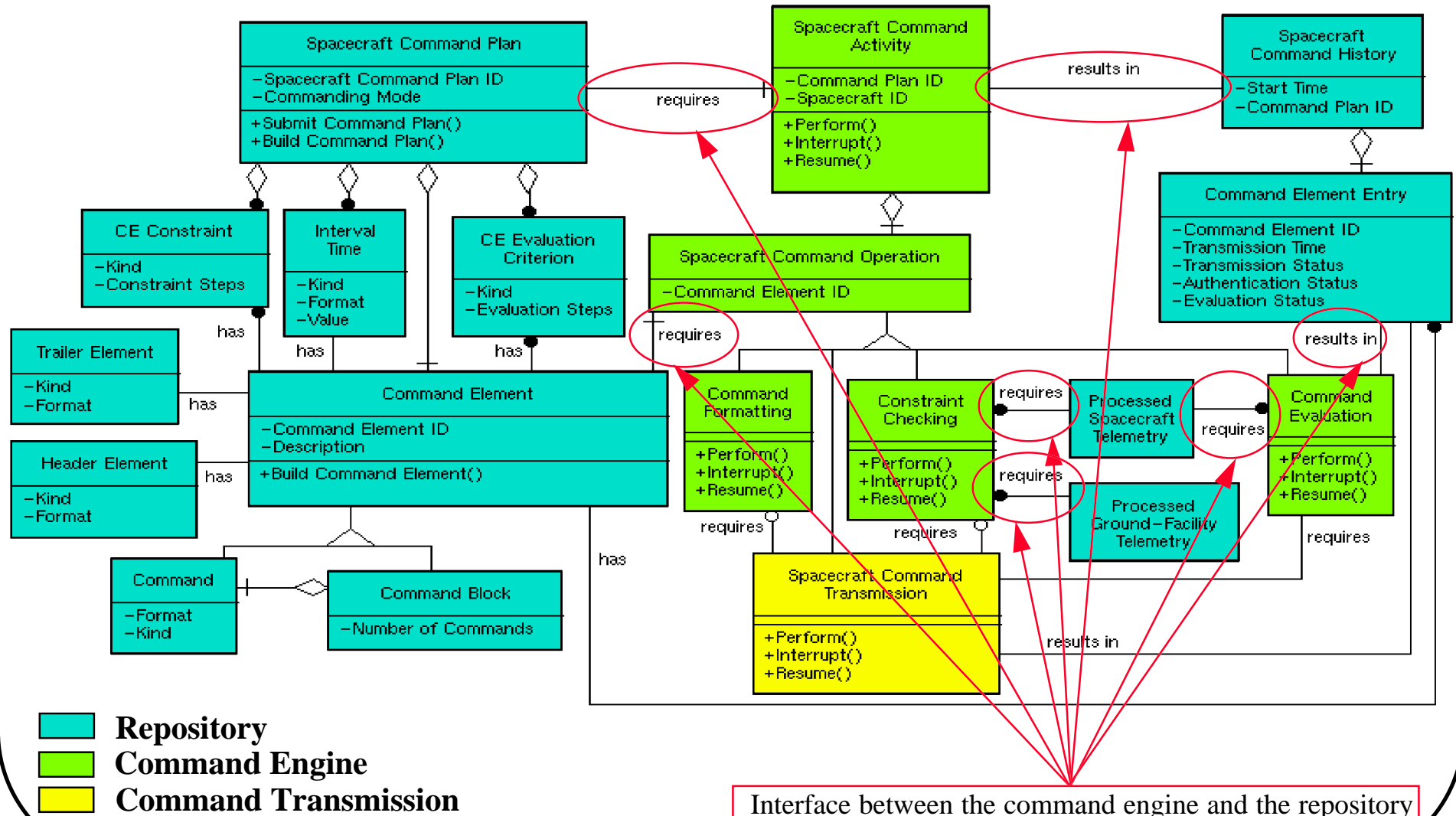


## **Problems of COTS Product Dependencies in Architecture Models: Lack of Interface Definitions**

- Consider the relationship between the command engine and the repository. This relationship indicates that the command engine uses services from the repository.
- This high level relationship does not appropriately capture how the command engine and the repository interact with one another, i.e., it does not represent what components of the command engine may request the services and what components of the repository may provide such services.
- In general, subsystem interfaces cannot be completely represented within the high-level representation provided by COTS dependent class category diagrams.

# Example of Interface Representation With Class Diagrams

- Representing subsystem interfaces requires determining the classes that provide and use services.



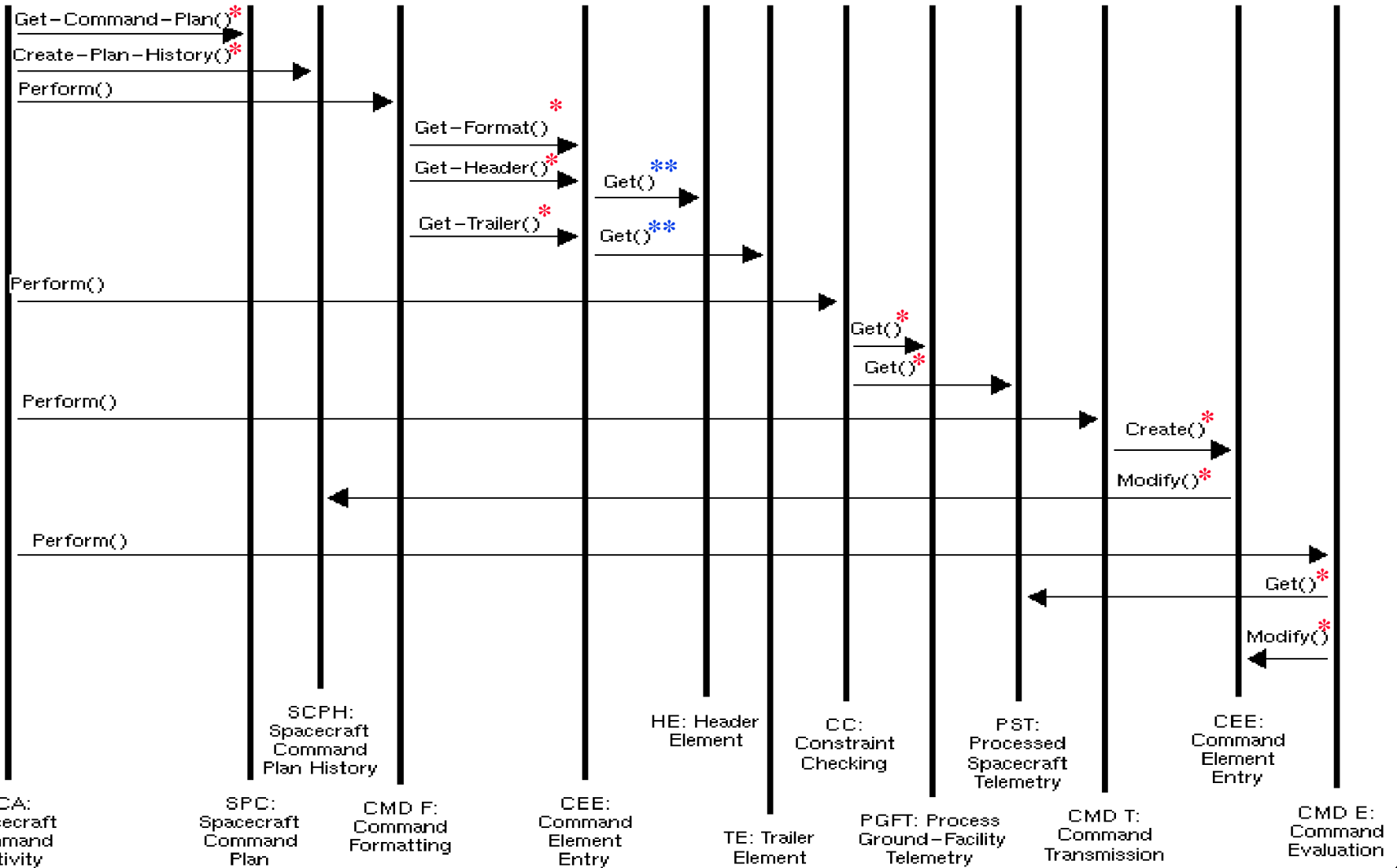
Interface between the command engine and the repository

## **Problems of COTS Product Dependencies in Architecture Models: Lack of Risk Mitigation**

- Consider the repository class category. The services provided by the repository must be accessed to format commands, check command constraints, transmit commands, and evaluate command execution.
- If these services are not provided in a timely manner, the repository may become a system bottleneck and pose development risks.
- The high-level representation provided by COTS dependent class category diagrams does not contain sufficient information to fully identify whether class category relationships may become system bottlenecks.
- As a result, COTS dependent class category diagrams cannot be used to identify which subsystems and/or subsystem relationships may pose development risks.

# Example of Using Message Trace Diagrams to Identify Risks

- Representing the services provided by a subsystem is required to identify whether the subsystem may be a development risk.



\* Requests to the repository

\*\*Requests within the repository

## **Problems of COTS Product Dependencies in Architecture Models: Lack of Documentation**

- The high-level representation provided by COTS dependent class category diagrams does not contain sufficient information to fully document the logical organization and the dynamic behavior of the system.
- System documentation produced using such diagrams is dependent on and constrained by the COTS product documentation.
- As a result, COTS dependent documentation needs to be modified when COTS products evolve or are replaced during the design, implementation, evolution, and/or maintenance of the system.

## Developing Documentation of System Architecture Models

- Documentation should include diagrams that represent logical organization (e.g., class diagrams), dynamic behavior (e.g., message trace diagrams), process decomposition, software organization, and physical realization of the system.
- All diagrams should be free from COTS product dependencies and have written explanations of the information depicted in them.
- Documentation should show how the classes of objects in the architecture model can be related to or interfaced with the commercial classes in the class libraries of the selected COTS products.



## Conclusions

- Architecture models that provide concurrent multiple views should be the basis for the design and implementation of satellite ground systems.
- Architecture models should represent system requirements at an abstract level.
- Architecture models should not be dependent on the specific COTS products selected for the system implementation.
- Dependence on COTS product information reduces the long-term value of architecture models.