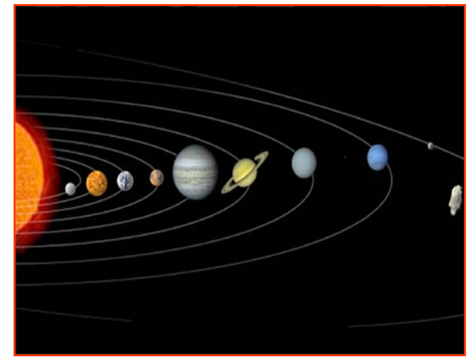
Architecture-Centric Evolution and Evaluation (ACE2)

Reference Architecture for Space Data Systems

30 March 2004

Peter Shames, NASA/JPL

Space Data System
Several Architectural Viewpoints



Enterprise — Business Concerns; Organizational perspective

Connectivity — Physical Concerns; Node & Link perspective

Functional — Computational Concerns; Functional composition

Information — Data Concerns; Relationships and transformations

Communications — Protocol Concerns; Communications stack perspective

Derived from: RM-ODP

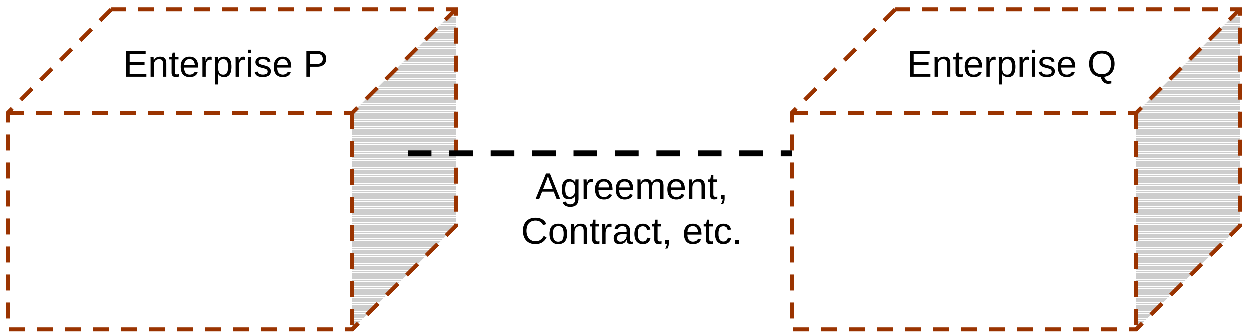
Technical Approach

- Develop a methodology for describing systems, and systems of systems from several viewpoints
  - Initial focus was CCSDS, but it is more generally applicable to space data systems
  - Derived from Reference Model of Open Distributed processing (RM-ODP), which is ISO 10746
  - Adapted to meet requirements and constraints of space data systems
- Define the needed viewpoints for space data system architecture description
  - Does not specifically include all elements of RM-ODP engineering and technology views, assume use of RM-ODP for these
  - Does not encompass all aspects of Space Systems, i.e. power, propulsion, thermal, structure, does not preclude them either
- Define a representational methodology
  - Applicable throughout design & development lifecycle
  - Capture architecture & design artifacts in a machinable form, able to support analysis and even simulation of performance
  - Validate methodology by applying it to several existing CCSDS reference models and existing systems
- Identify relevant existing commercial methodologies
  - Evaluate UML 2.0 and SysML, now in progress
  - Explore applicability of methodology & tools

High Level RASDS Methodology / Tool Requirements

- Meta-model and model language that is independent of specific tool environments and implementations
  - Models can be exchanged and imported into other tool suites
- Tool suite with a graphical interface that enables creation, manipulation, display, archiving, and versioning of meta-models, component and connector type templates, and instance models
- Support development of machine readable, portable architecture meta-model for RASDS
- Support development of instance models for specific space systems deployments
- Provide a framework that supports coarse grained simulation of behavior and performance characteristics of instance models

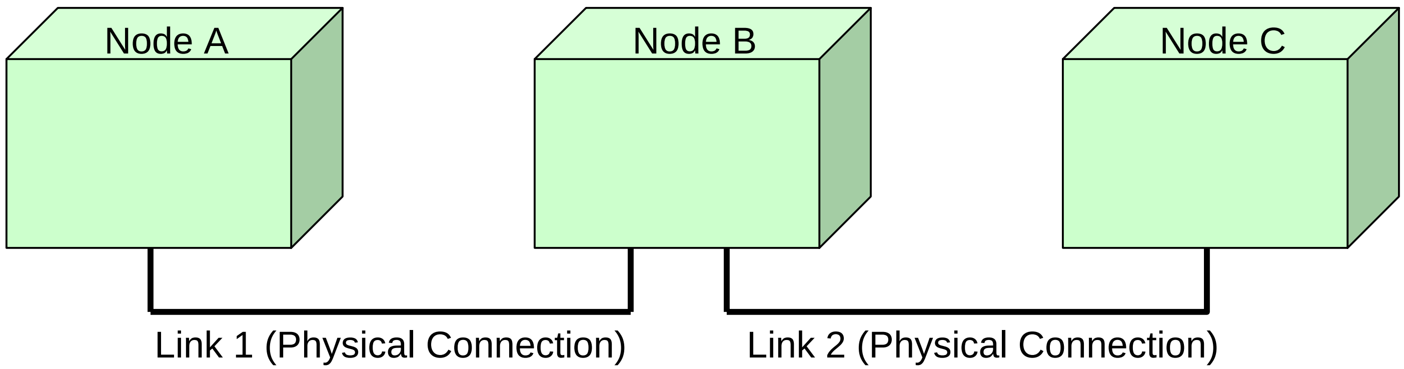
Enterprise View (Enterprise Objects)



Enterprise Objects:
- Organizations
- Facilities

Enterprise Concerns:
- Objectives
- Roles
- Policies
- Activities
- Configuration
- Contracts
- Lifecycle / Phases

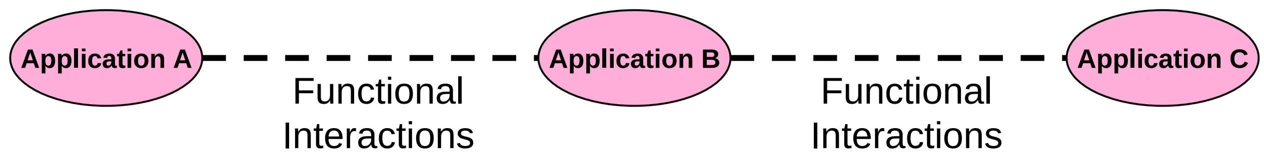
Connectivity View (Nodes and Links)



Connectivity Objects :
- Physical Nodes
- Physical Links
- (Physical behavior)

Connectivity Concerns:
- Distribution
- Communication
- Physical Environment
- Behaviors
- Constraints
- Configuration

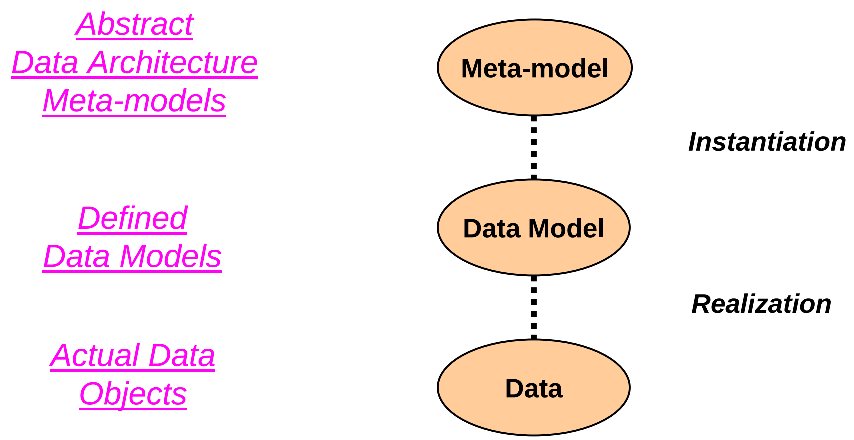
Functional View (Functional Objects)



Functional Objects:
- Functional Elements
- Related Implementations
- Information Flows

Functional Concerns:
- Behaviors
- Interactions
- Interfaces
- Constraints

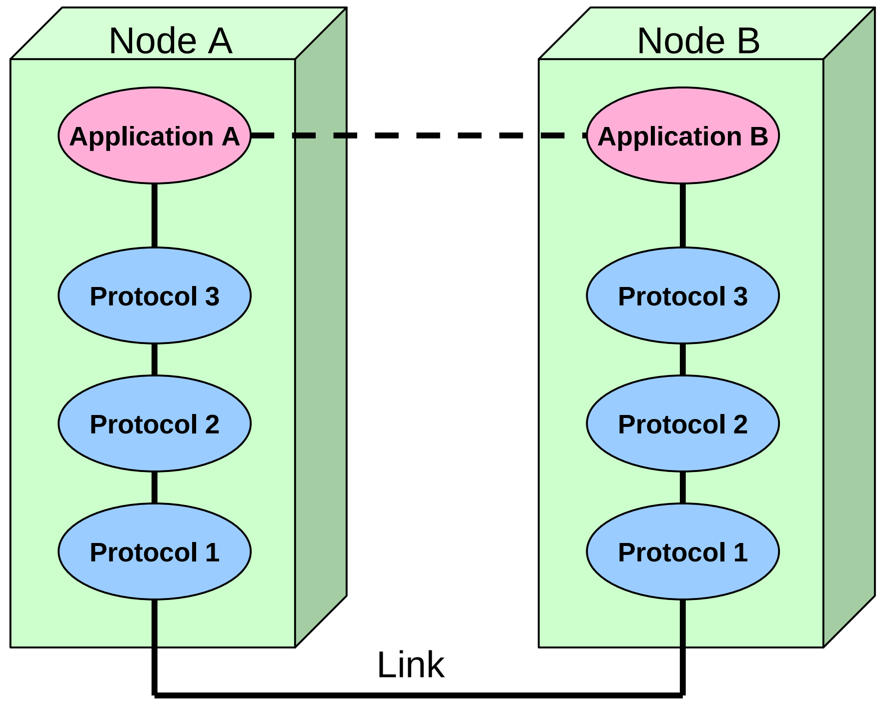
Information View (Information Objects)



Information Objects:
- Information models & objects
- Information Infrastructure (specialized functions)

Information Concerns:
- Structure
- Semantics
- Relationships
- Permanence
- Rules

Communication View (Shown w/ Nodes, Links, Functional Objects and Communications Objects)



Communications Objects:
- Protocol Objects (specialized functions)
- Service Interfaces

Communications Concerns:
- Standards
  - Interfaces
  - Protocols
- Technology
- Interoperability
- Suitability

ACE2 Baseline Topics

1. Architecture as a Basis for Understandability
2. Architecture as a Basis for Assessing Maintainability
3. Architecture as a Basis for Assessing Extensibility
4. Architecture as a Basis for Assessing Executabilty

Architecture as a Basis for Understandability

"Software architectures should provide views of the software system with levels of granularity appropriate for each stakeholder (i.e., acquirer, overseer, developer, tester, and operator) so that they have insight into new system functionality resulting from changing requirements or specifying new ones."

- RASDS is intended to provide an architectural view of end to end data systems, including hardware and software.
  - Provides insight into functionality and relationship among elements so that complexity may be managed
  - Formal representation (using SysML) is expected to provide means to analyze effects of new or changed requirements
  - It intentionally does not address implementation details, but these may be naturally elaborated based upon the existing views
  - Primarily intended for use with acquirer, overseer, system engineer and developer, additional views and details required for operator and tester

Architecture as a Basis for Assessing Maintainability

"Software architectures should link system requirements to detailed system implementation so that stakeholders can assess the degree of system change and the impact on cost and development schedule that may result from maintainability requirements regarding upgrades, changes, and integration of COTS product used in the system implementation."

- RASDS provides the means to represent software and hardware elements as they will be deployed, thus supporting allocation of functionality, design trades, deployment trades, and analysis of impact of requirements changes
- RASDS does not explicitly address requirements traceability, though the expected adoption of SysML as a formal representation does provide this functionality
- Since RASDS is intended to address architectures, not implementations, it does not directly address maintainability or COTS
- COTS products are implementation artifacts, but the RASDS provides guidance on how to describe their functionality, effects, and interfaces
  - Suitable modeling of functionality and interfaces may prove very useful in early identification of model clashes

Architecture as a Basis for Assessing Extensibility

"Software architectures should link system requirements to detailed system implementation so that stakeholders can assess the degree of system change and the impact on cost and development schedule that may result from new requirements on increased system size, complexity, system environments, services, and interoperability."

- RASDS provides the means to describe and reason about system and component size, complexity, performance, and operating environments
- It is specifically intended to address interoperability issues and addresses service and protocol interfaces as a primary means of achieving this
- While RASDS does not directly address requirements traceability down to implementation details, is is expected that the SysML formalisms and tools will provide this functionality
- We intend to be able to assess end to end system performance via coarse grained simulation of behavior based upon the RASDS models of the system, primarily using the Connectivity and Functional Views of the modeled system.

Architecture as a Basis for Assessing Executabilty

"The level of granularity of the software architecture should support the development of executable models that enable stakeholders to measure the impacts of new requirements on system performance and reliability."

- Using the Connectivity and Functional Views (and in the Communications view where needed) is it possible to model system behavior at a coarse level of granularity
  - This permits assessment of alternative allocations of functionality and performance trade studies
  - It also supports analysis of different protocol approaches to dealing with complex communications environments and highly mobile elements
- Using SysML to realize RASDS models will permit specification of behavior and analysis of performance
  - It will also support model elaboration and refinement to provide the needed levels of granularity
- Initial studies of formal methods of describing and simulating behavior of RASDS models, using xADL, are expected to yield early insights into the utility of this approach

BACKUP SLIDES

Formal Method Evaluation

- Studied UML 2.0, SysML, xADL
- Unified Modeling Language (UML 2.0)
  - Too focused on software systems
  - Includes elements that are not needed for RASDS
  - Some commercial tool support now
- System Modeling Language (SysML)
  - Has most of the required features
  - Needs some extensions for RASDS viewpoints and details
  - Commercial tools support expected 2005
- xADL
  - Extensible approach that can accommodate RASDS
  - xADL needs to be customized, not interoperable w/ XMI
  - Tool support from UCI and USC, academic quality

SysML Background

- Informal partnership of modeling tool users, vendors, etc.
  - Organized in May 2003 to respond to UML for Systems Engineering RFP
  - Includes many aerospace companies and major UML tool vendors
- Charter
  - The SysML Partners are collaborating to define a modeling language for systems engineering applications, called Systems Modeling Language™ (SysML™). SysML will customize UML 2 to support the specification, analysis, design, verification and validation of complex systems that may include hardware, software, data, personnel, procedures, and facilities.

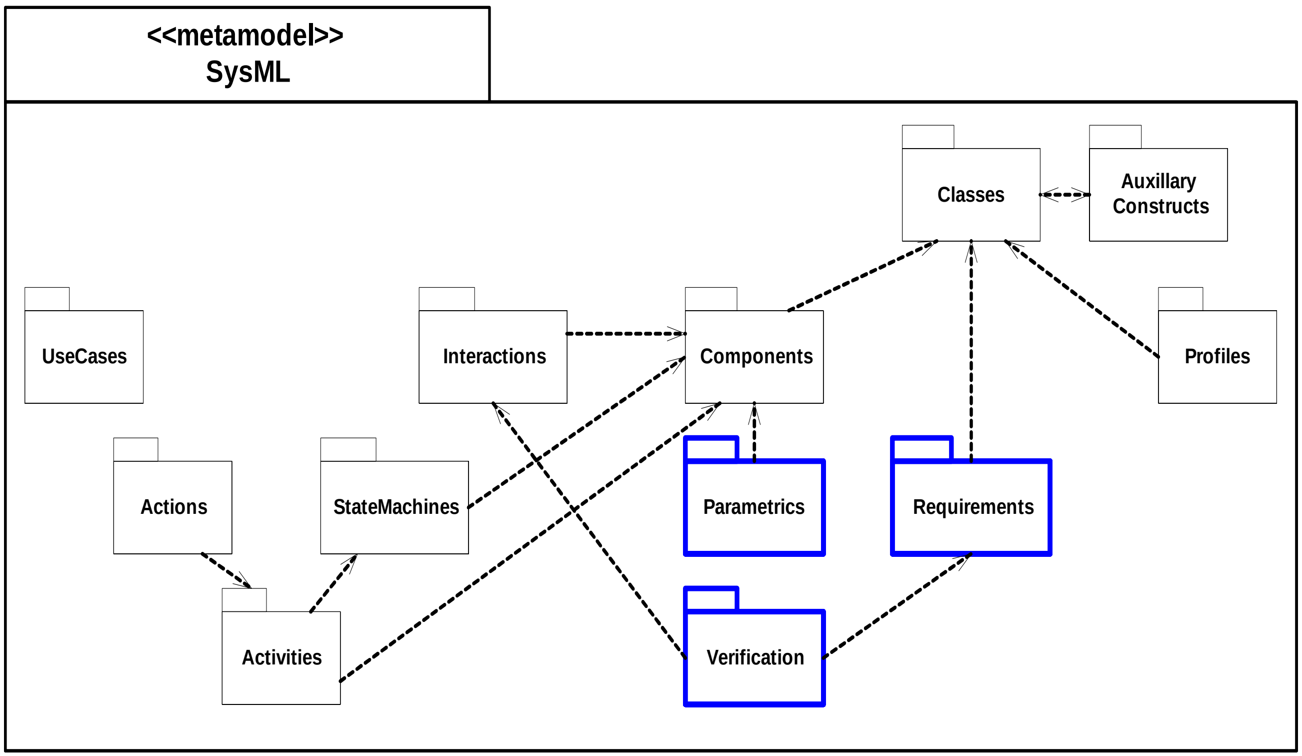
Source: SysML Partners

SysML Motivation

- Systems Engineers need a standard language for analyzing, specifying, designing, verifying and validating systems
- Many different modeling techniques
  - Behavior diagrams, IDEF0, N2 charts, …
- Lack broad based standard that supports general purpose systems modeling needs
  - satisfies broad set of modeling requirements (behavior, structure, performance, …)
  - integrates with other disciplines (SW, HW, ..)
  - scalable
  - adaptable to different SE domains
  - supported by multiple tools

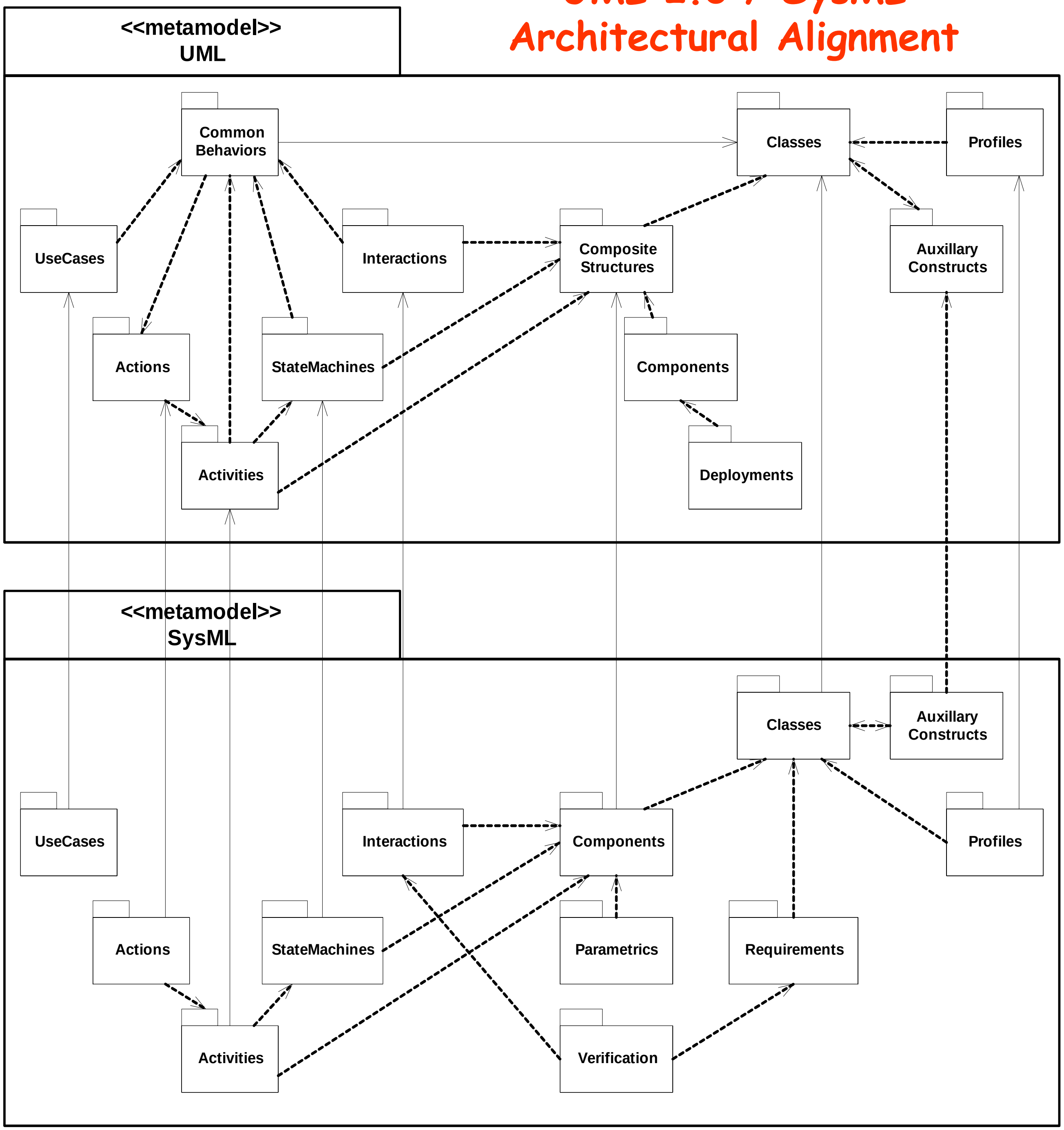
Source: SysML Partners

SysML Language Architecture



Source: SysML Partners

UML 2.0 / SysML Architectural Alignment



Source: SysML Partners

SysML Analysis

- **Analyzed requirements in UML for Systems Engineering RFP and SysML Draft Response (January 25, 2004)**
- **Initial analysis indicates that SysML meets or exceeds the requirements for RASDS, with some specific exceptions:**
  - Need clarification of how SysML can support the following:
    - Policies and agreements in the Enterprise View
    - Detailed communication protocol definitions in the Communications View
  - The ability to explicitly relate model elements between model viewpoints is partially addressed by SysML, but must be augmented by RASDS methodology specific relationships and constraints.
  - The behavior and executability aspects of SysML are outside current RASDS scope, but are expected to prove useful. Requirements and parametric diagrams are not currently required for RASDS, but are likely to be useful in the long run.
  - SysML is expected to be adopted by the OMG in late 2004 with tool support anticipated to follow.

Mapping RASDS into SysML

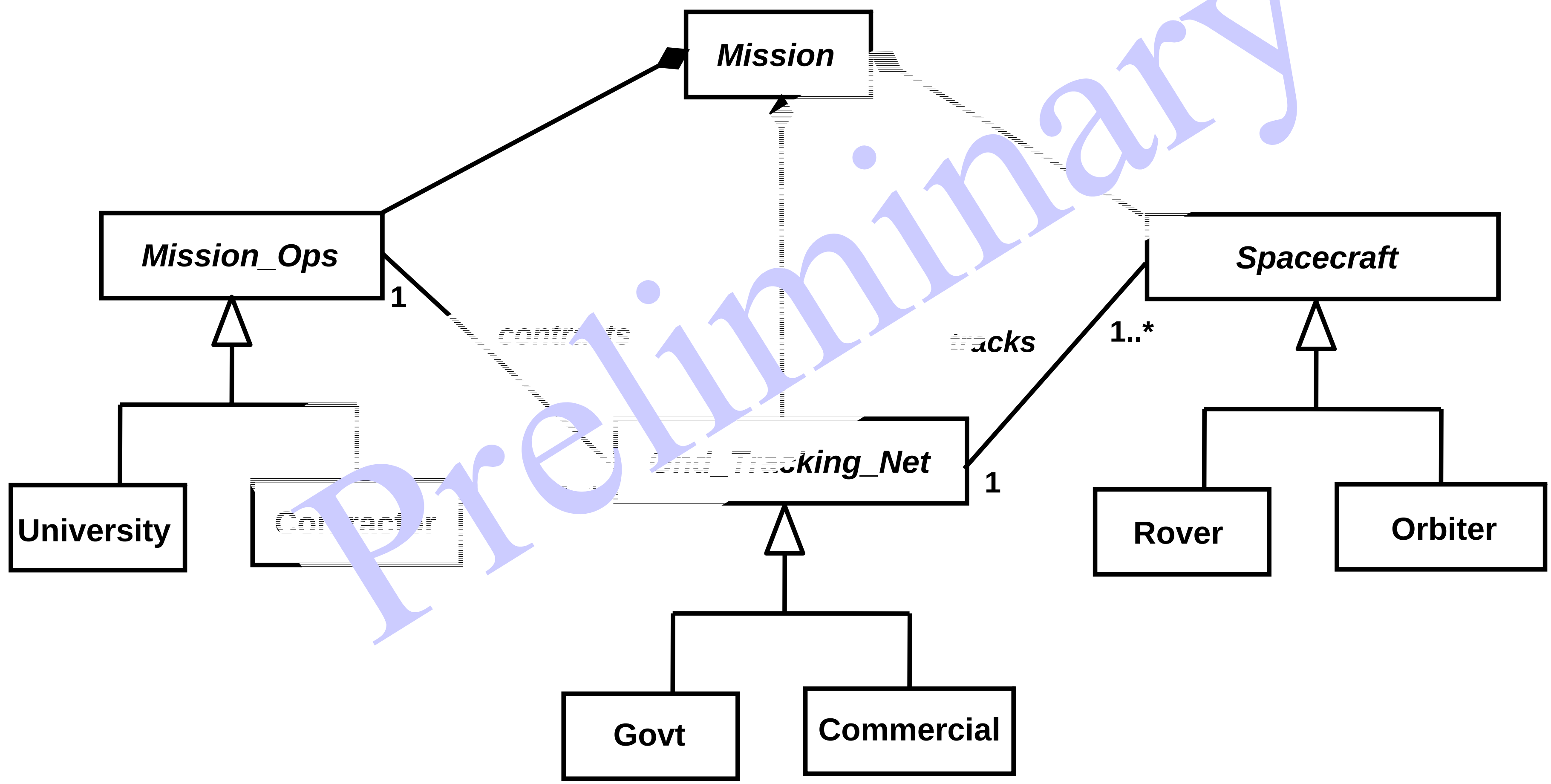
- No simple one for one mapping
- RASDS uses Viewpoints to expose different concern of a single system
- SysML uses specific diagrams to capture system structure, behavior, parameters and requirements
- Several SysML diagrams, focused on different object classes, may be applied to any given RASDS Viewpoint
- Extended SysML Views may be used to define the relationships between Viewpoints and Diagrams
- SysML will support more accurate fine grained modeling of behavior than was expected of RASDS

Mapping RASDS into SysML

- Enterprise
  - Organizational component & collaboration diagrams
  - Use case, interaction overview diagrams & constraints
- Connectivity
  - Physical component, composition, collaboration & class diagrams
  - Parametric diagram
- Functional
  - Functional component, collaboration & class diagrams
  - Activity, sequence, parametric & timing diagrams
- Informational
  - Information composition, class & parametric diagrams
- Communication
  - Protocol component & collaboration diagrams
  - State machine, sequence, activity & timing diagrams

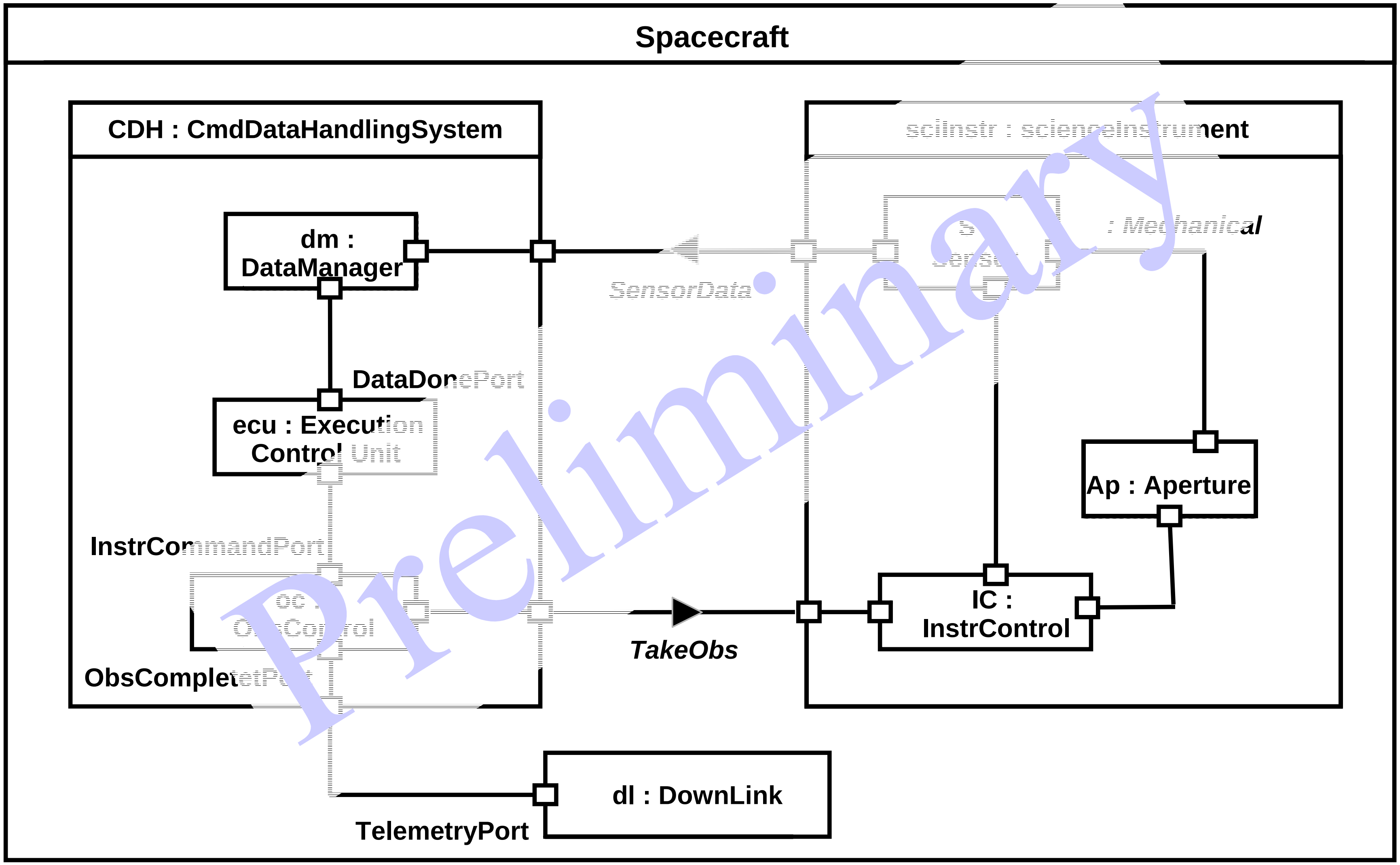
Enterprise View Using SysML Class Diagram

- Organizational structure & agreements:



Derived from: SysML Partners

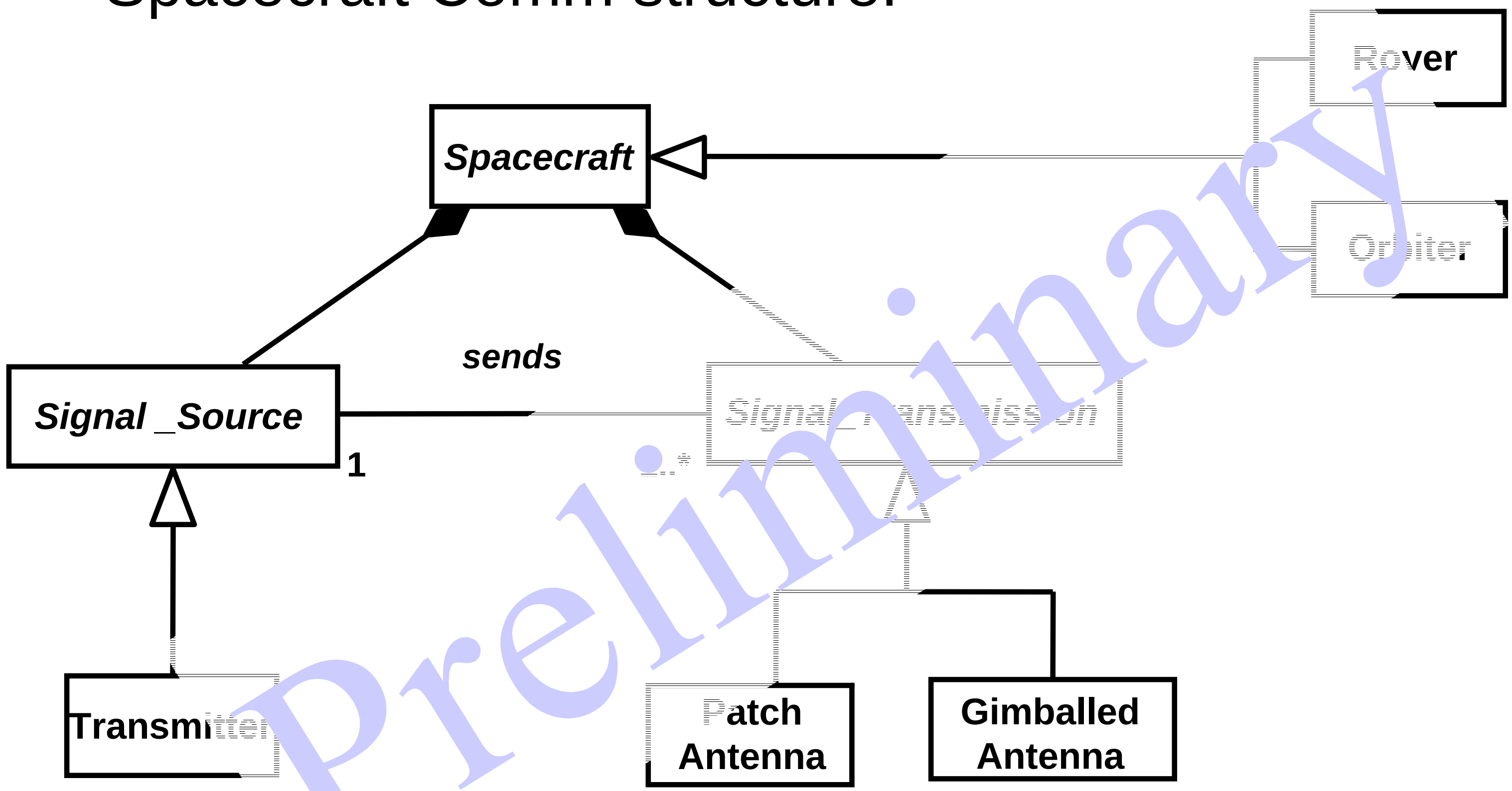
Connectivity View
(Nodes & Links) Using SysML Components



Derived from: SysML Partners

Connectivity View (Composition) Using SysML Classes

- Spacecraft Comm structure:

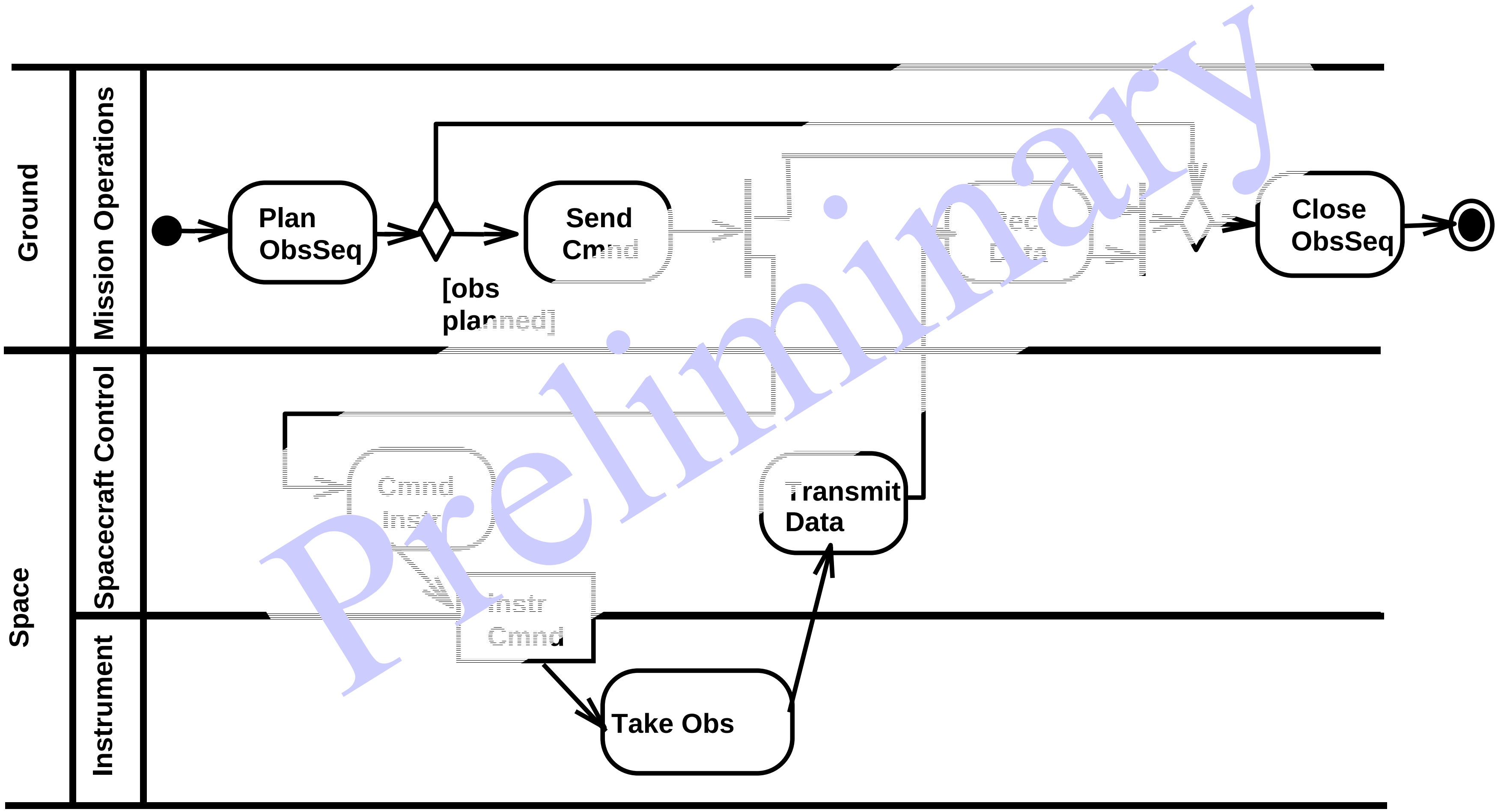


Global structure inherited by each kind of Spacecraft …

… and constrained for each kind

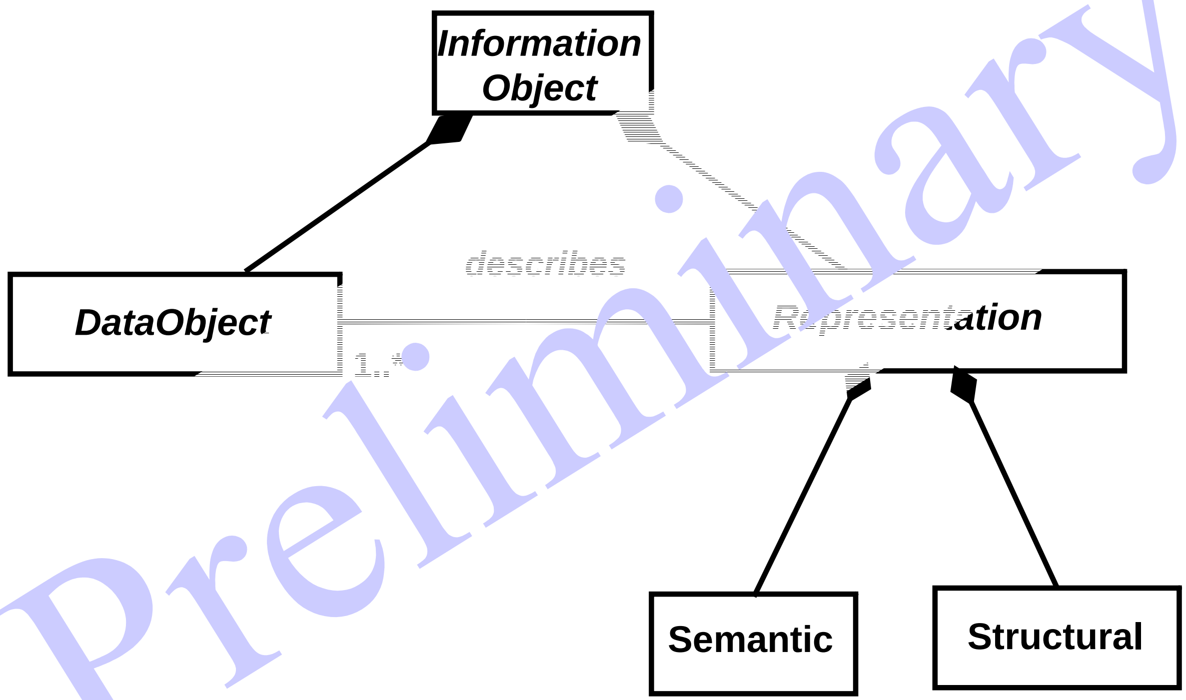
Functional View Using SysML Activity Diagram

- Showing component allocations (optional)



Informational View Using SysML Class Diagram

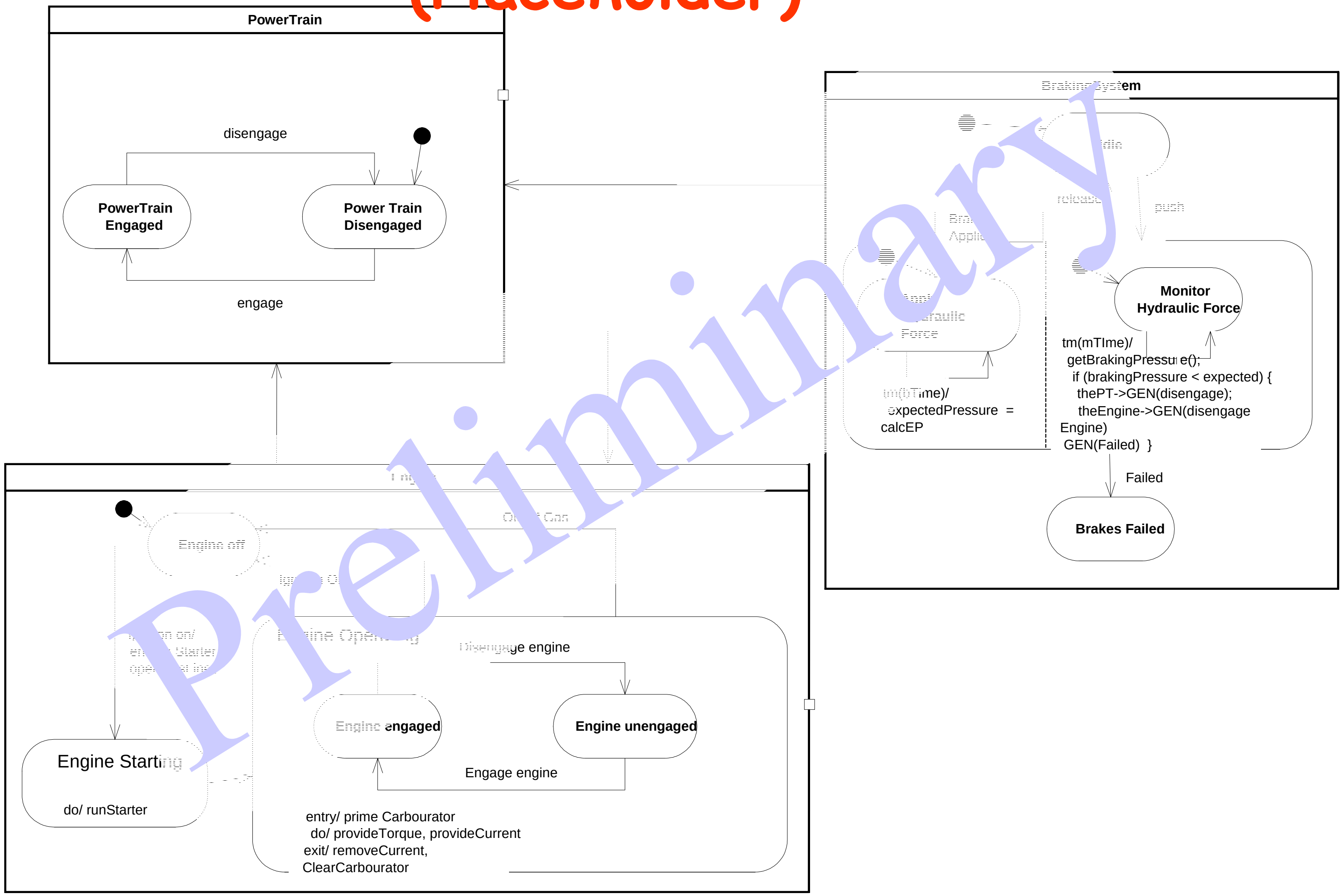
- Reusable, refinable information structure:



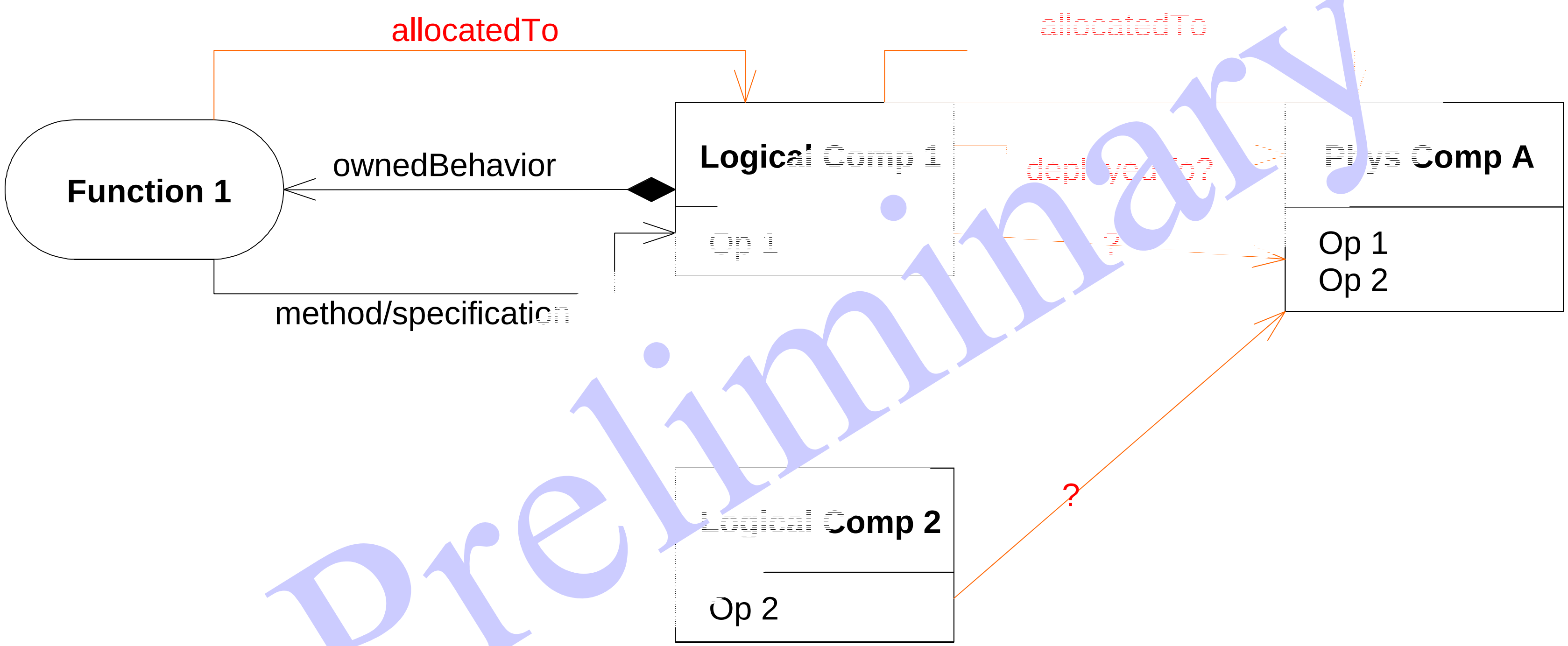
Global representation inherited by each kind of Information Object

Derived from: SysML Partners

Communication View Using SysML State Machine Diagram (Placeholder)

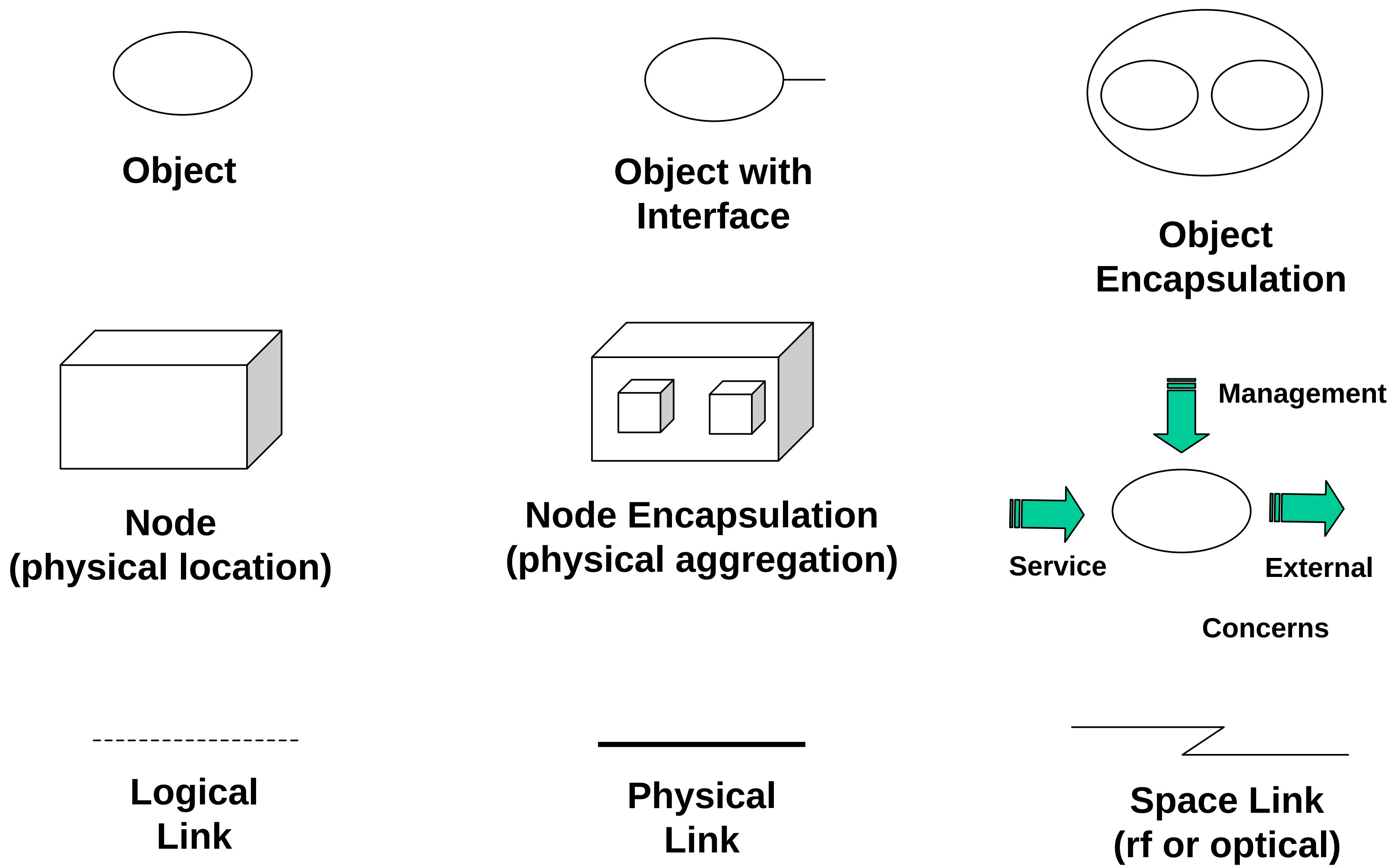


Functional - Logical - Physical Allocation: Viewpoint Relationships



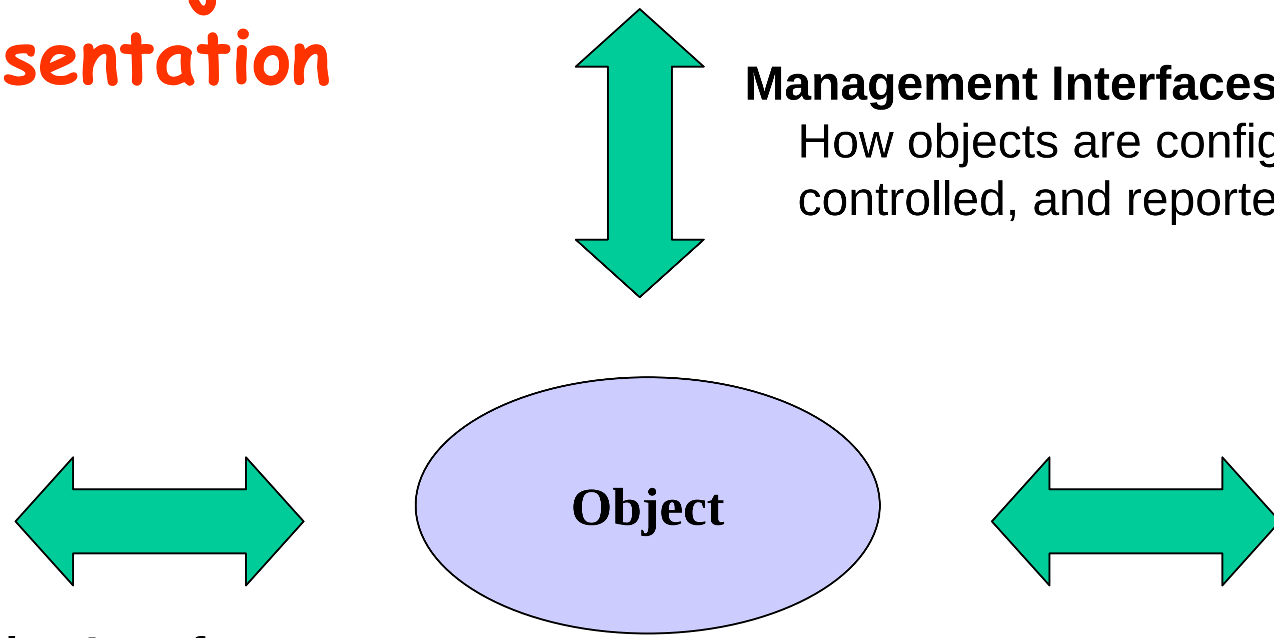
Derived from: SysML Partners

Space Data System Architectural Notation



Unified Object Representation

Management Interfaces:
How objects are configured
controlled, and reported upon



Service Interfaces:
How services are
requested & supplied

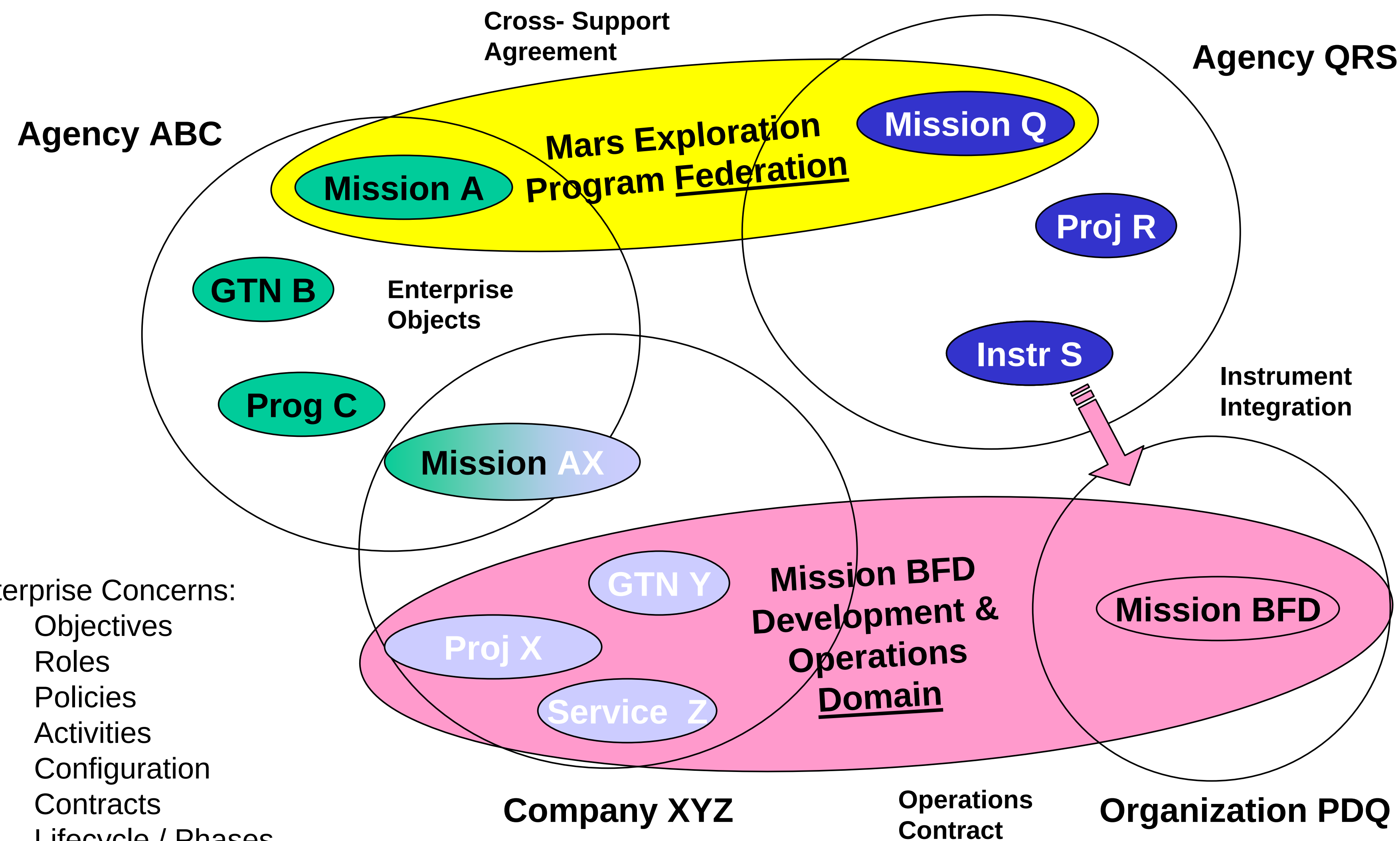
Core Functions
What the object
does

External Interfaces:
How external elements
are controlled

Concerns:
Issues
Resources
Policies

Enterprise View

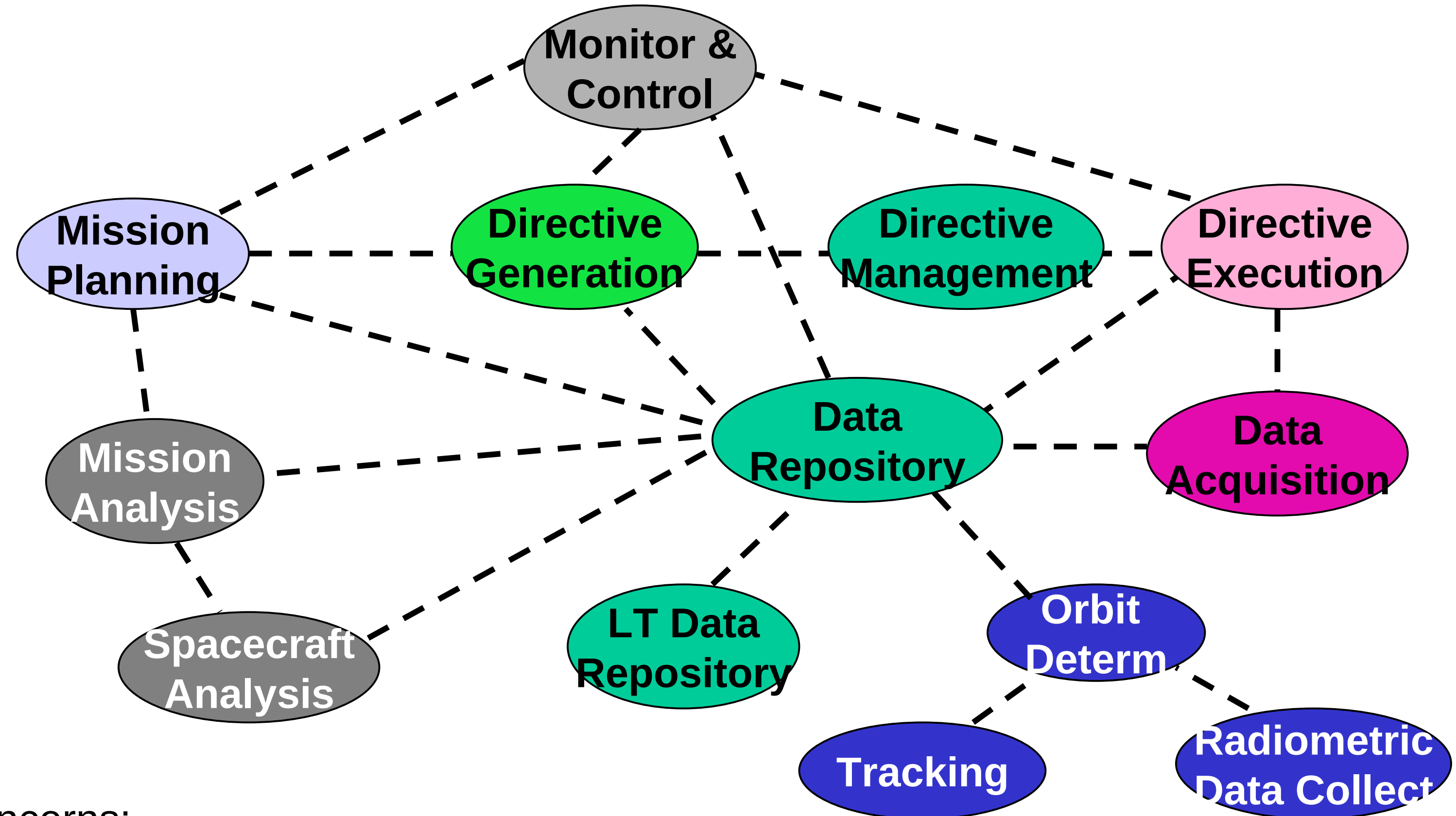
Federated Enterprises with Enterprise Objects



Enterprise Concerns:
- Objectives
- Roles
- Policies
- Activities
- Configuration
- Contracts
- Lifecycle / Phases

Functional View

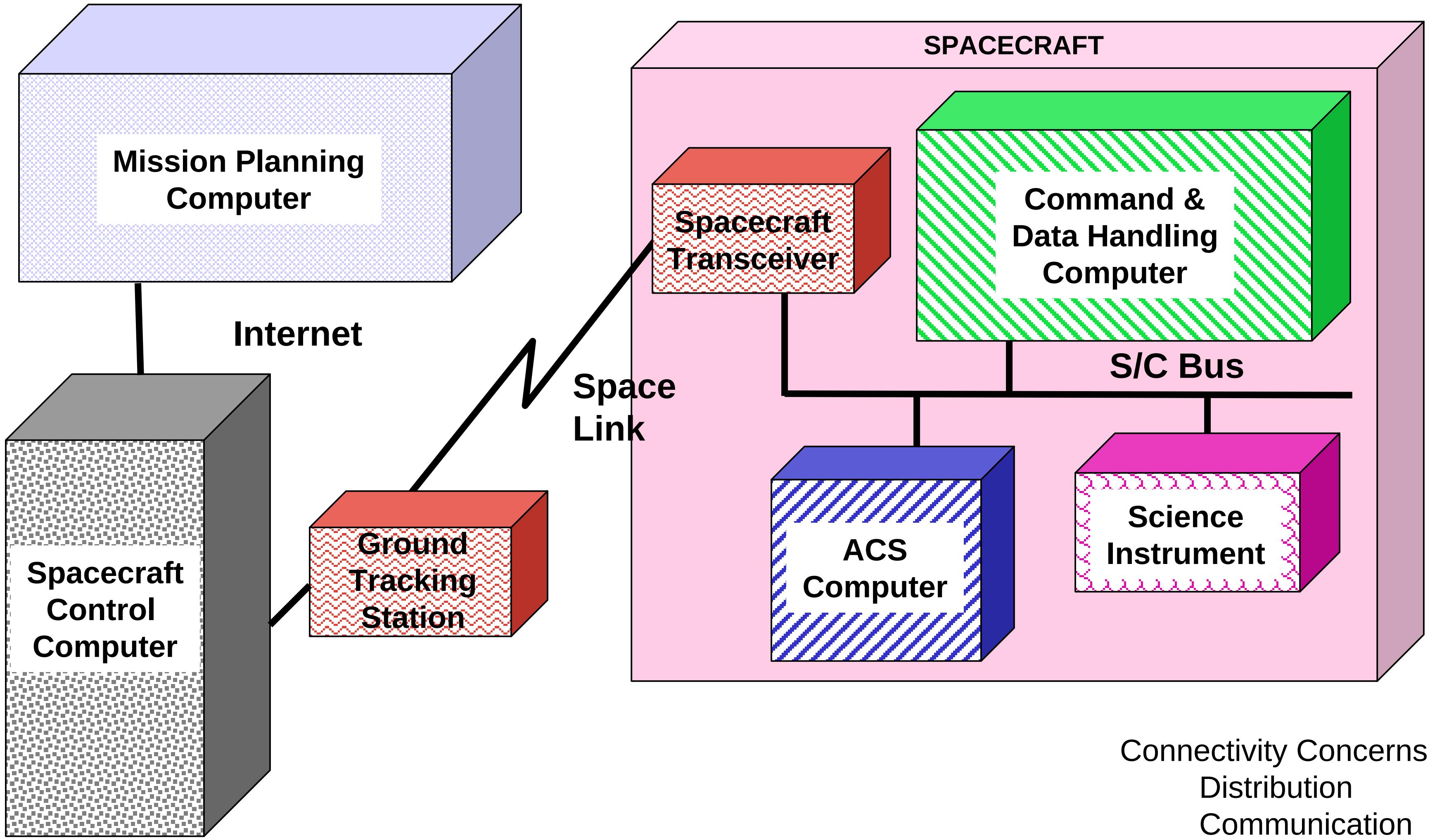
Example Functional Objects & Interactions



Functional Concerns:

- Behaviors
- Interactions
- Interfaces
- Constraints

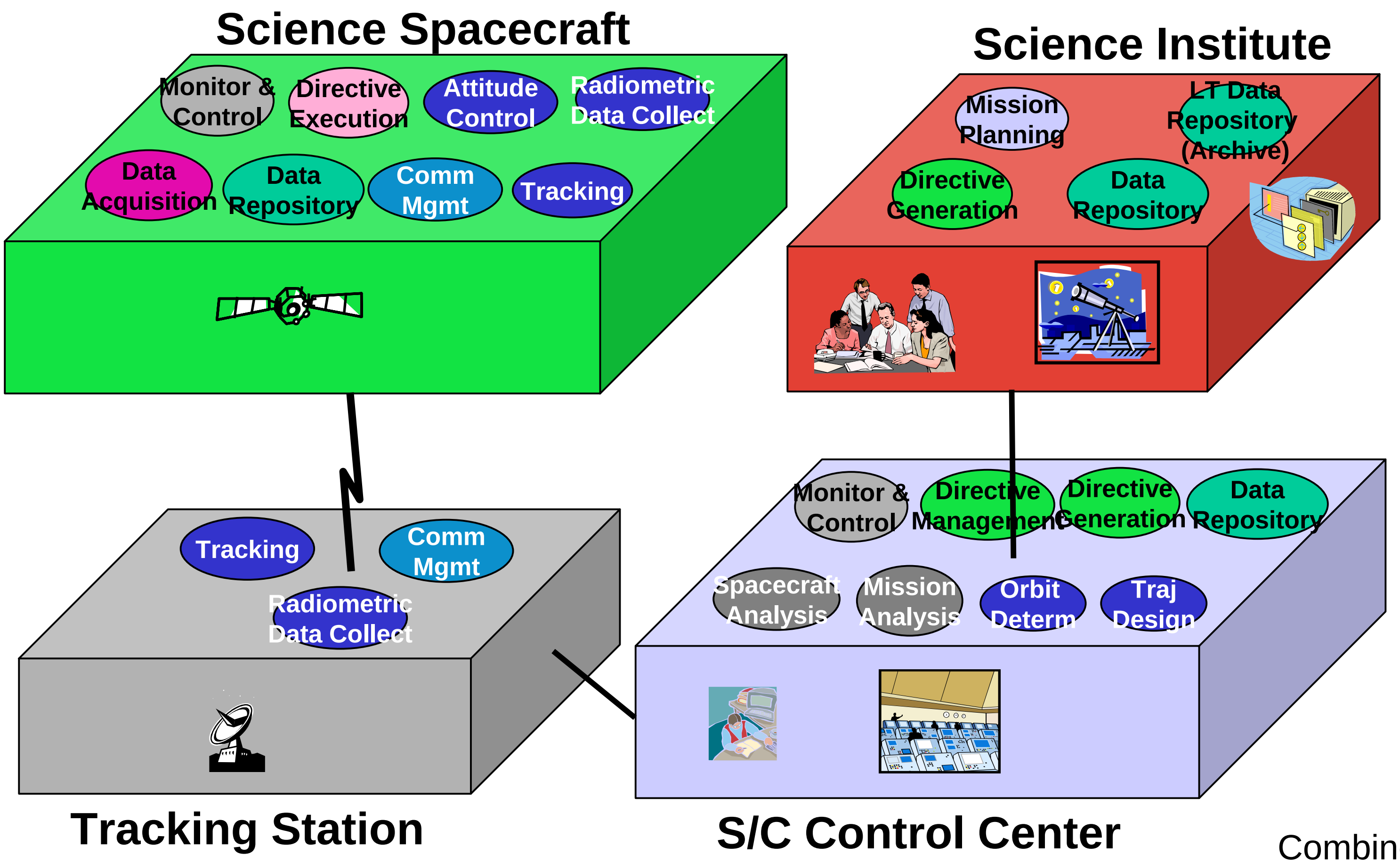
Connectivity View
Nodes & Links



Connectivity Concerns:
- Distribution
- Communication
- Physical Environment
- Behaviors
- Constraints
- Configuration

Connectivity & Functional View

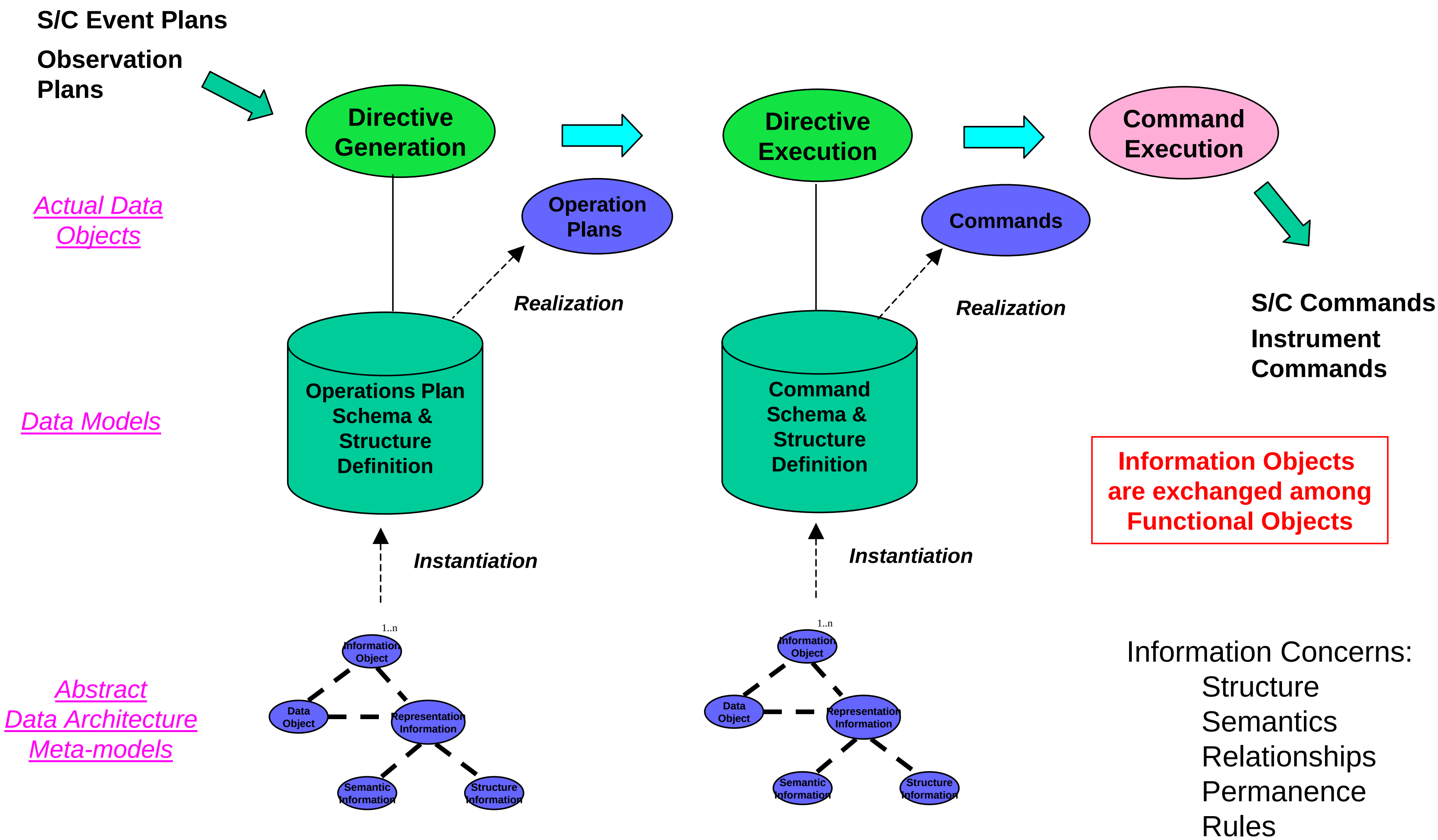
Mapping Functions to Nodes



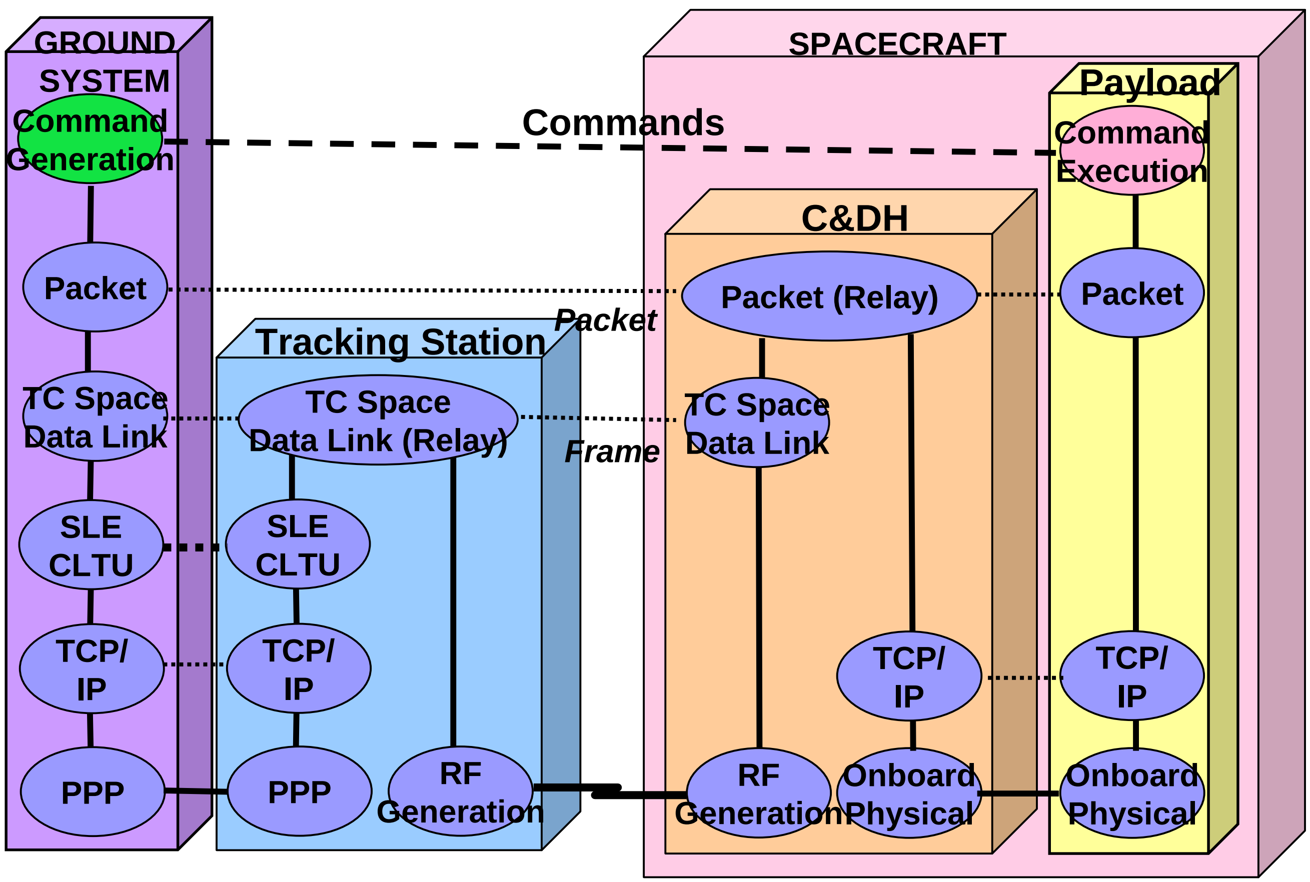
Combined View:
- End to End Behavior
- Performance
- Throughput
- Trade studies

Information Objects

Relationship to Functional View

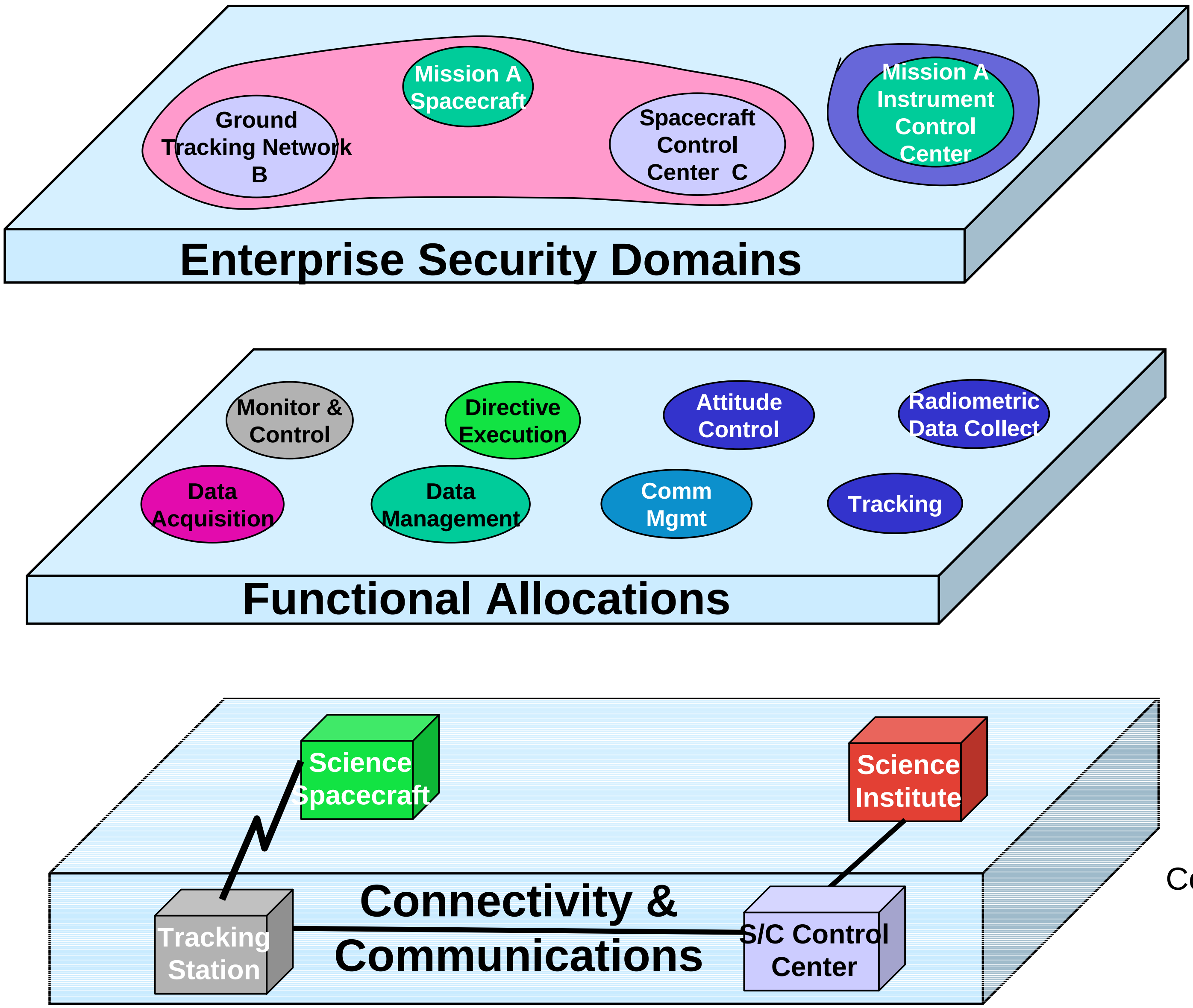


Communications Viewpoint Protocol Objects
End-To-End Command Processing



Communications Concerns:
- Standards
  - Interfaces
  - Protocols
- Technology
- Interoperability
- Suitability

Security Analyses
Multiple Viewpoints & Relationships



Trust relationships
Policies
Privacy / proprietary issues

Access control
Authentication

Firewalls
Encryption
Boundary access points

Combined View:
- Relationships
- Allocations
- Performance
- Trade studies

Next Steps

- Validate SysML modeling approach
  - Complete analysis of RASDS to SysML mapping
  - Validate with SysML Partners
  - Seek concurrence with CCSDS SAWG community

IFF agreed, then:

- Adopt an agreed RASDS formalism
  - Select specific formal methods from SysML for <u>describing RASDS architectures and systems</u>
  - Agree to final common representation and methods

- Generate baseline RASDS approach
  - Develop agreed SysML meta-models for Viewpoints
  - Define extensible library of component instances

Acknowledgements

- This task was carried out as part of the program of work of Consultative Committee for Space Data Systems (CCSDS).
- It was performed by the Architecture Working group (AWG), chaired by Takahiro Yamada, ISAS
- Other AWG members who actively participated are listed below:
  - Fred Brosi, NASA/GST
  - Dan Crichton, NASA/JPL
  - Adrian Hooke, NASA/JPL
  - Steve Hughes, NASA/JPL
  - Niklas Lindman, ESA/ESOC
  - Nestor Peccia, ESA/ESOC
  - Lou Reich, NASA/CSC
  - Don Sawyer, NASA/GSFC
  - Peter Shames, NASA/JPL
  - Anthony Walsh, ESA/Vega