CCSDS
Architecture Working Group

Tools for the Reference Architecture for Space Data Systems

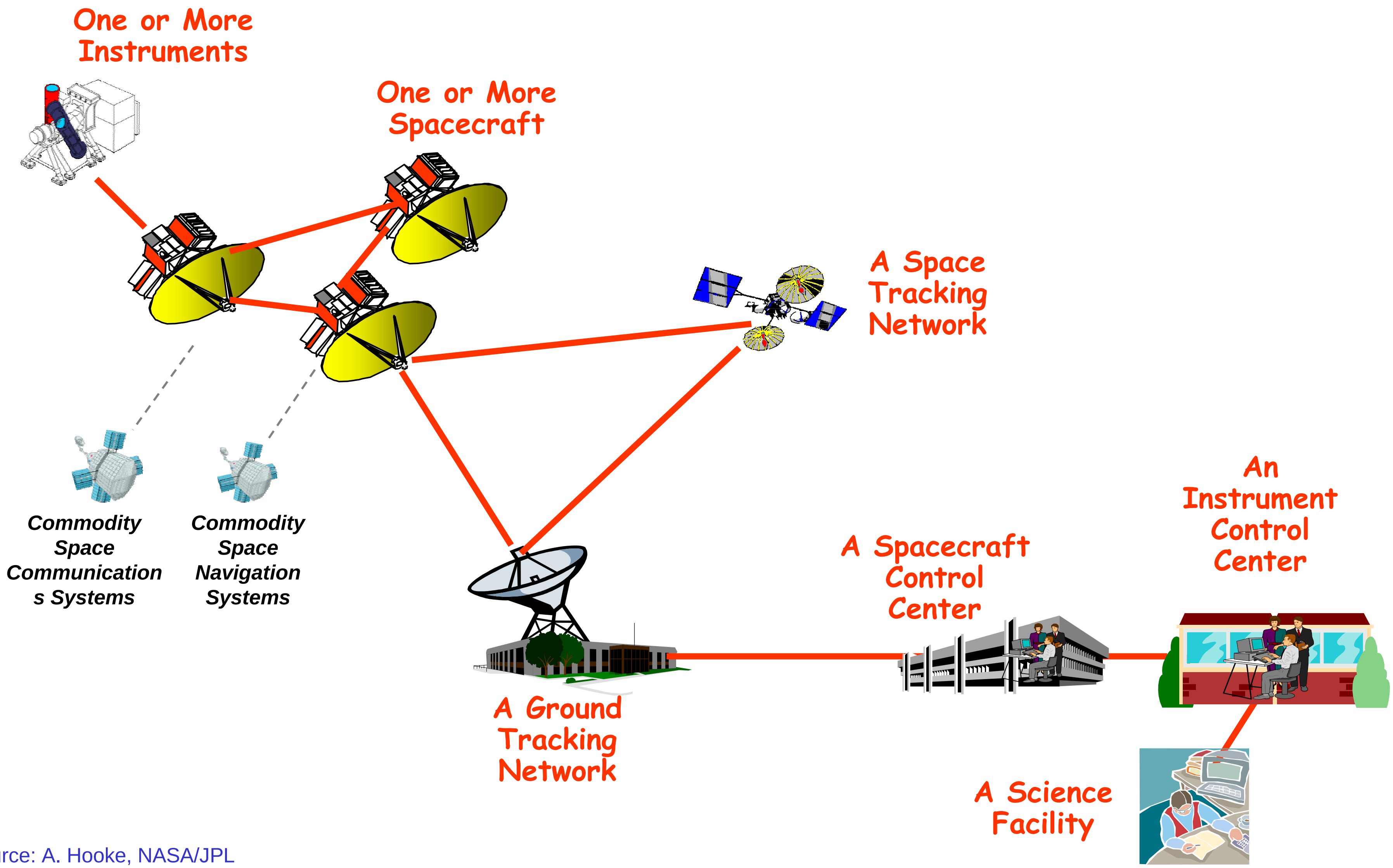
30 March 2004

Peter Shames, NASA/JPL, Takahiro Yamada, JAXA/ISAS
Niklas Lindman, ESA/ESOC

Agenda

- Introduce the Reference Architecture for Space Data Systems (RASDS)
- Define the RASDS requirements on formal methodologies & tools
- Describe the current work to identify a suite of tools that can be used to define, manipulate, store, manage, and analyze these models
  - UML 2.0
  - SysML
  - xADL / Archstudio

A Physical View of a Space Data System



Source: A. Hooke, NASA/JPL

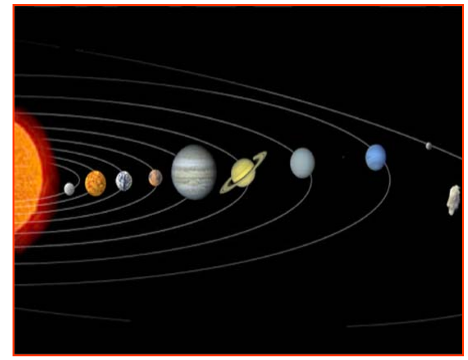
Reference Architecture Purpose

- Establish an overall CCSDS approach to architecting and to developing domain specific architectures
- Define common language and representation so that challenges, requirements, and solutions in the area of space data systems can be readily communicated
- Provide a kit of architect's tools that domain experts will use to construct many different complex space system architectures
- Facilitate development of standards in a consistent way so that any standard can be used with other appropriate standards in a system
- Present the standards developed by CCSDS in a systematic way so that their functionality, applicability, and interoperability may be clearly understood

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

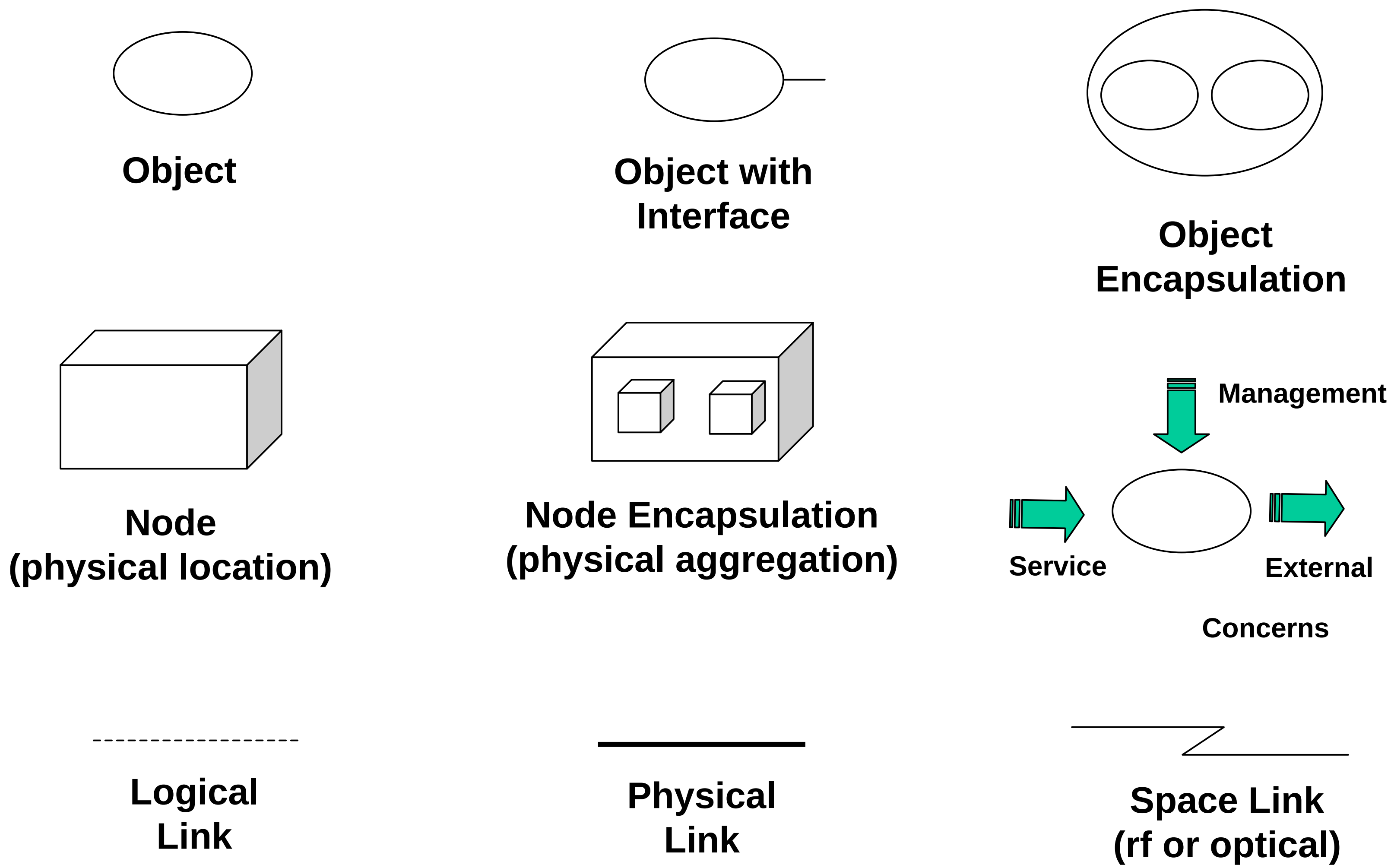
Space Data System
Several Architectural Viewpoints



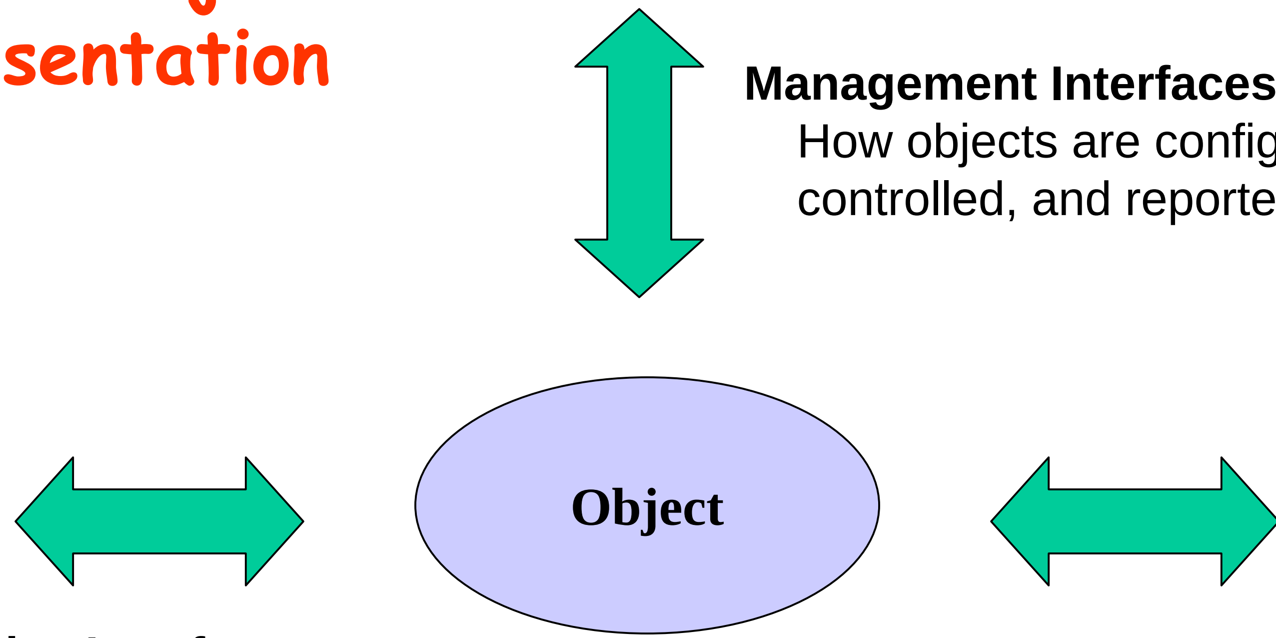
Enterprise	**Business Concerns** **Organizational perspective**
Connectivity	**Physical Concerns** **Node & Link perspective**
Functional	**Computational Concerns** **Functional composition**
Information	**Data Concerns** **Relationships and transformations**
Communications	**Protocol Concerns** **Communications stack perspective**

Derived from: RM-ODP

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

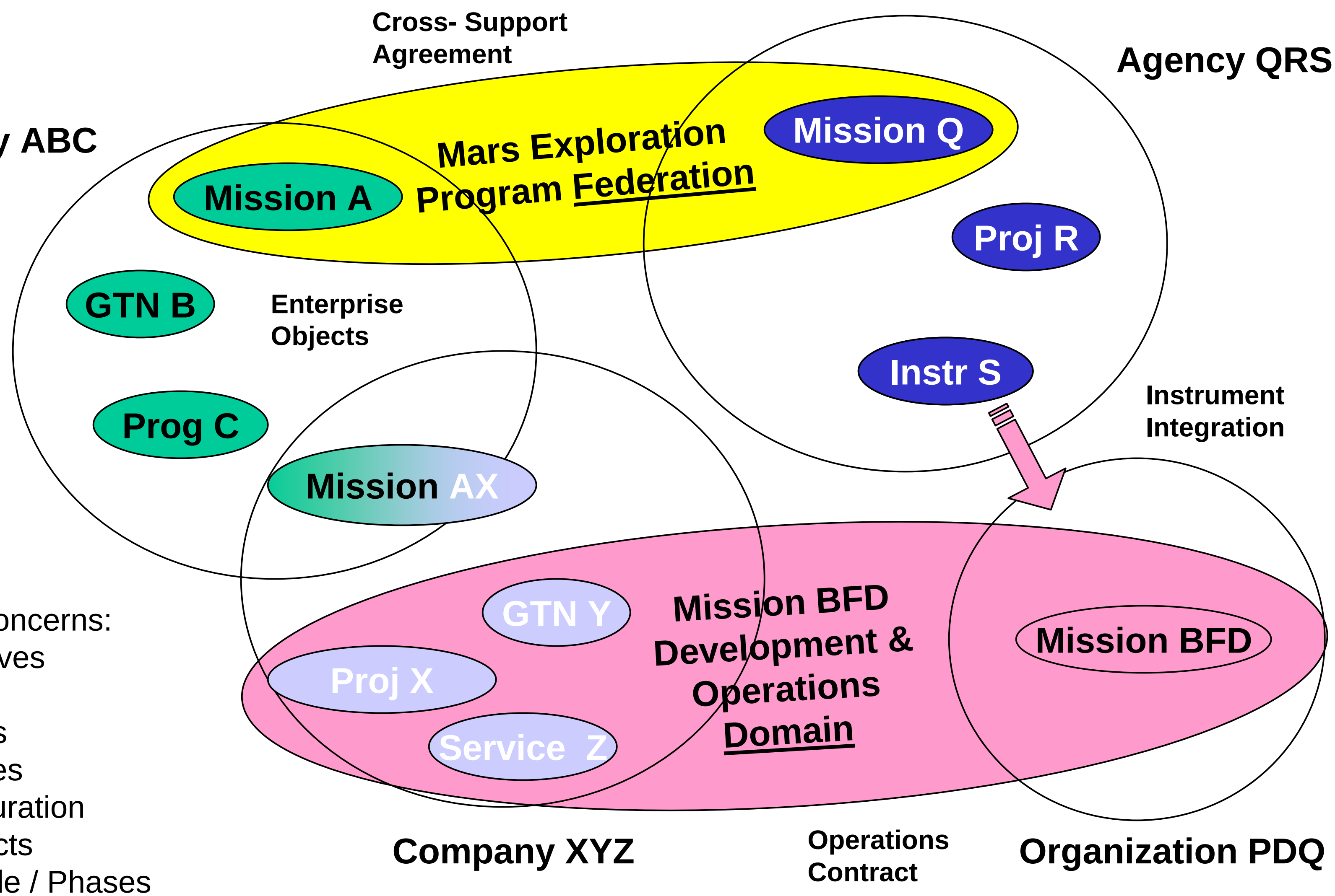
Core Functions
What the object does

External Interfaces:
How external elements are controlled

Concerns:
Issues
Resources
Policies

Enterprise View

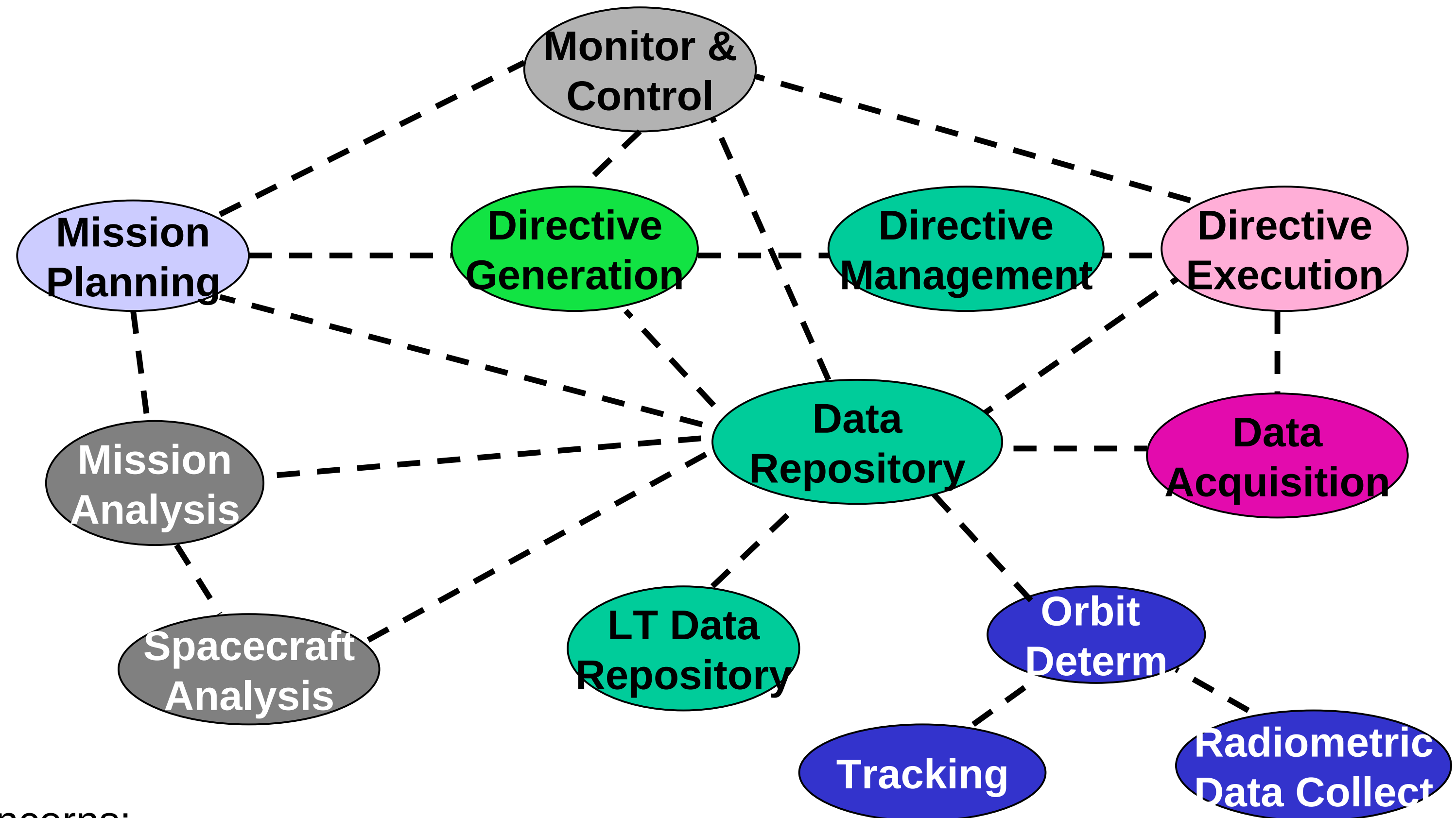
Federated Enterprises with Enterprise Objects



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

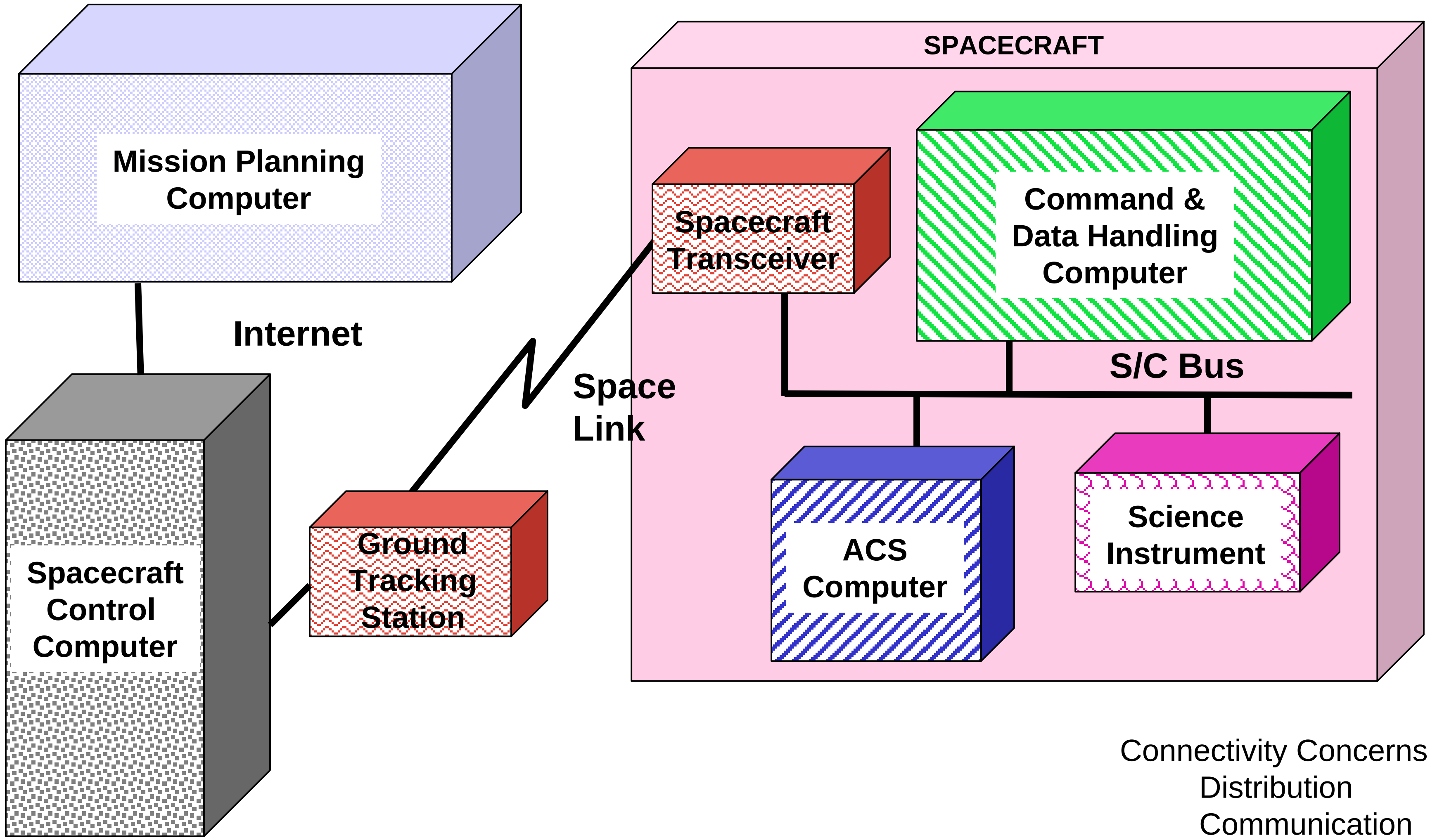
Functional View

Example Functional Objects & Interactions



Functional Concerns:
- Behaviors
- Interactions
- Interfaces
- Constraints

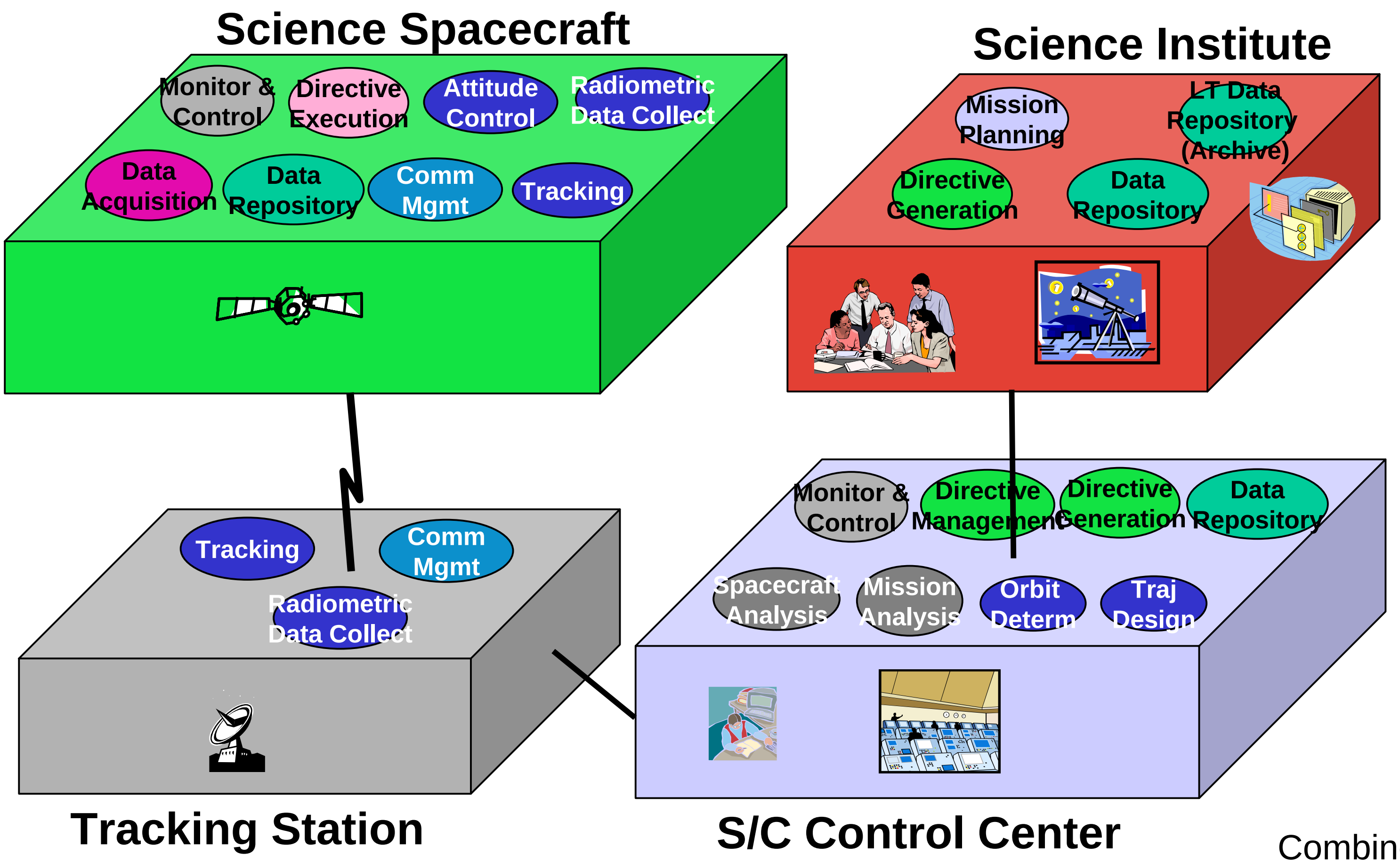
Connectivity View
Nodes & Links



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

Connectivity & Functional View

Mapping Functions to Nodes

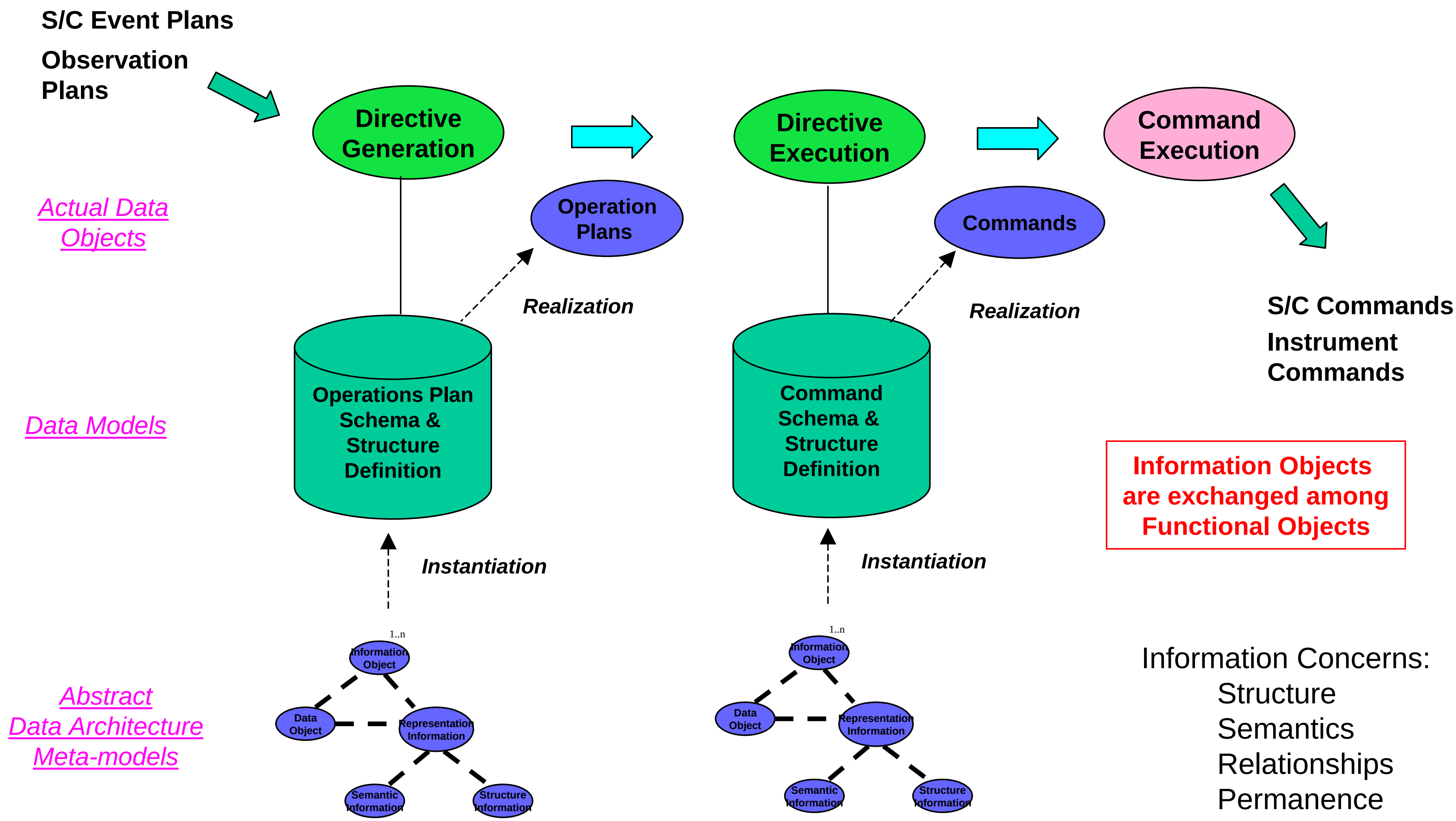


Combined View:

- End to End Behavior
- Performance
- Throughput
- Trade studies

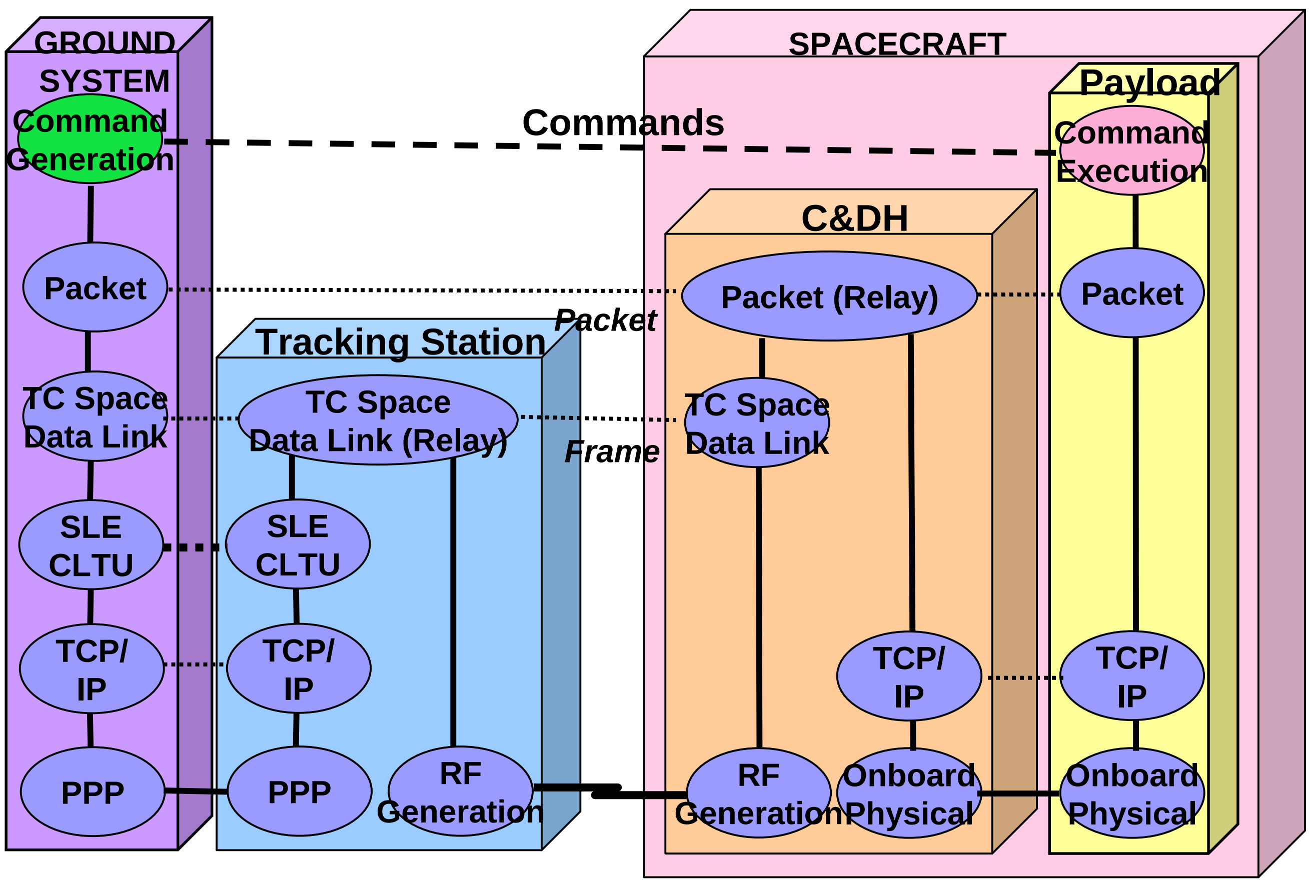
Information Objects

Relationship to Functional View



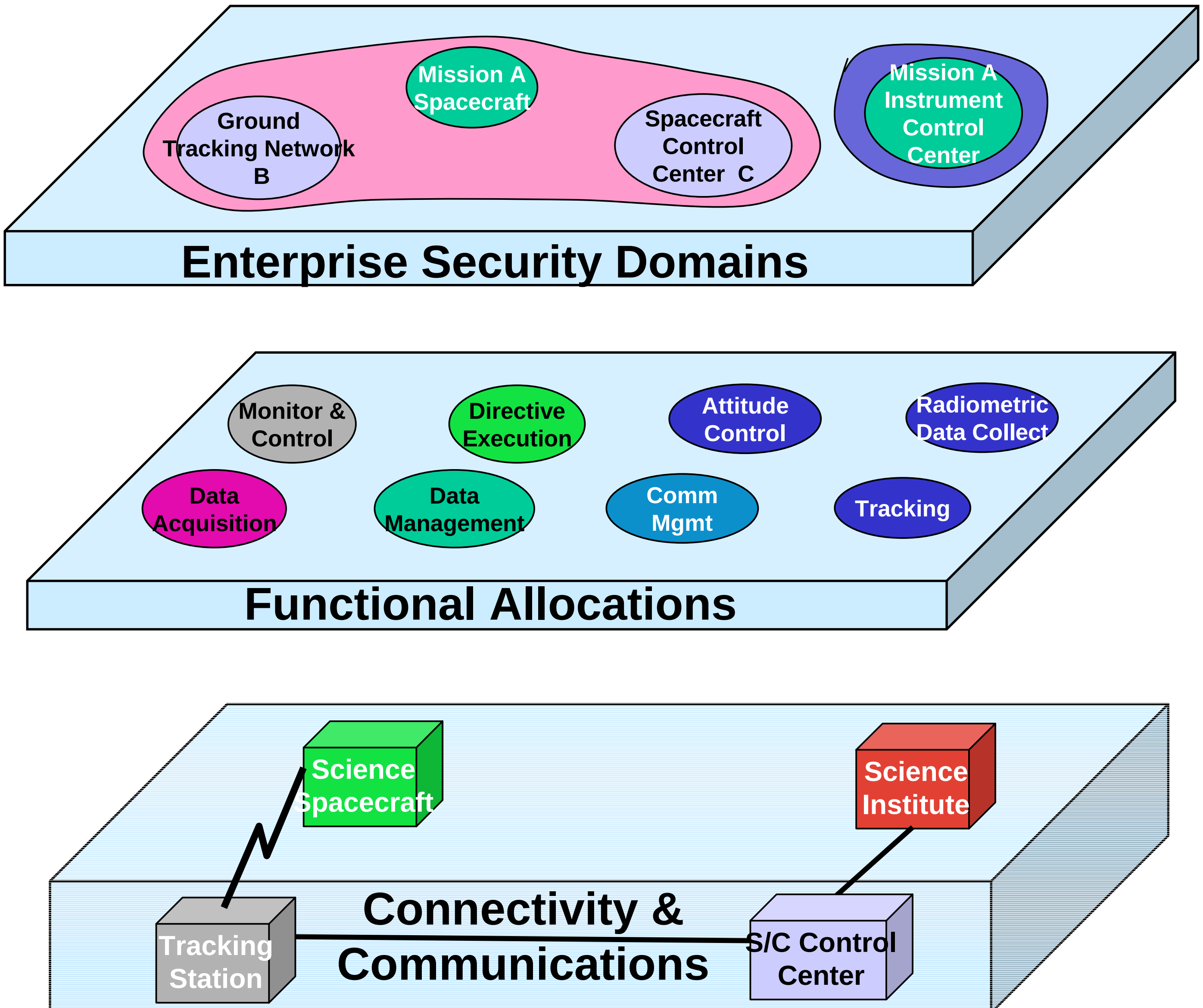
Information Concerns:
- Structure
- Semantics
- Relationships
- Permanence
- Rules

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

High Level RASDS Methodology / Tool Requirements

- Meta-model and model language that is independent of specific tool environments and implementations
  - Models may 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 instance models

RASDS Requirements on Tools / Environment

1. **Support Architectural Modeling** – provide means for developing, validating, extending, and sharing RASDS compliant models

2. **Flexibility** – allow multiple approaches to be explored at the same time

3. **Model Integrity** – provide means for ensuring model integrity by checking relationships across views and updating them automatically (or flagging problems)

4. **Model Validation** – provide means for validating model completeness and well formedness

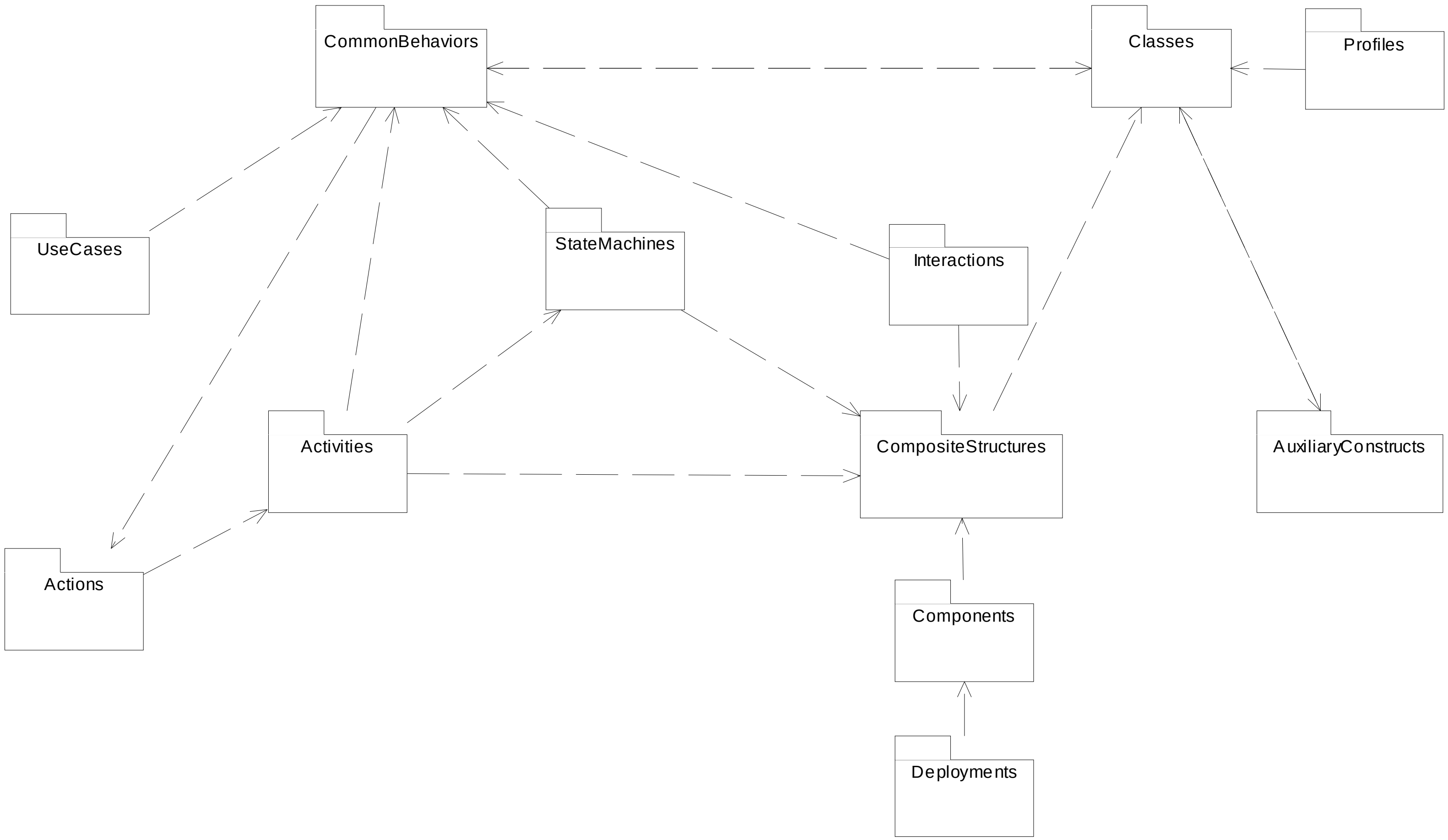
5. **Relative Ease of Use** – exhibit good ergonomics, be easy to learn and use, and provide other ease of use features like contextual help

6. **Repository / Model Sharing** – provide means for storing complete models, model elements, fragments, and templates, and for sharing these across a working group. Facilitate re-use and sharing

UML 2.0 Background

- UML 2.0 is the next generation of the Unified Modeling Language, developed by OMG
- UML is already de facto standard within software engineering community
- UML is mature and extensible, and can be adapted to support other requirements
- UML tools and training are widely available
- OMG standardization process supports UML customization for specific domains (e.g., systems engineering)

UML 2 Superstructure Architecture



Source: OMG ptc/03-08-02 (UML 2.0 Superstructure Final Adopted specification)

UML 2.0 Analysis

- **Initial analysis of UML, it fails to meet a number of the requirements for RASDS:**
  - UML is primarily intended to model software architectures
    - Limited support for hardware and physical connectivity
    - Limited applicability to system architectures
  - UML 2 provides many of the features that RASDS requires without modification
    - Composite Structure diagrams
    - Activity diagrams
  - UML 2 provides many features that are not needed for RASDS
  - UML 2 is just now being finalized, few tool suites support it at this time and there is not yet an XMI interchange spec for it

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.

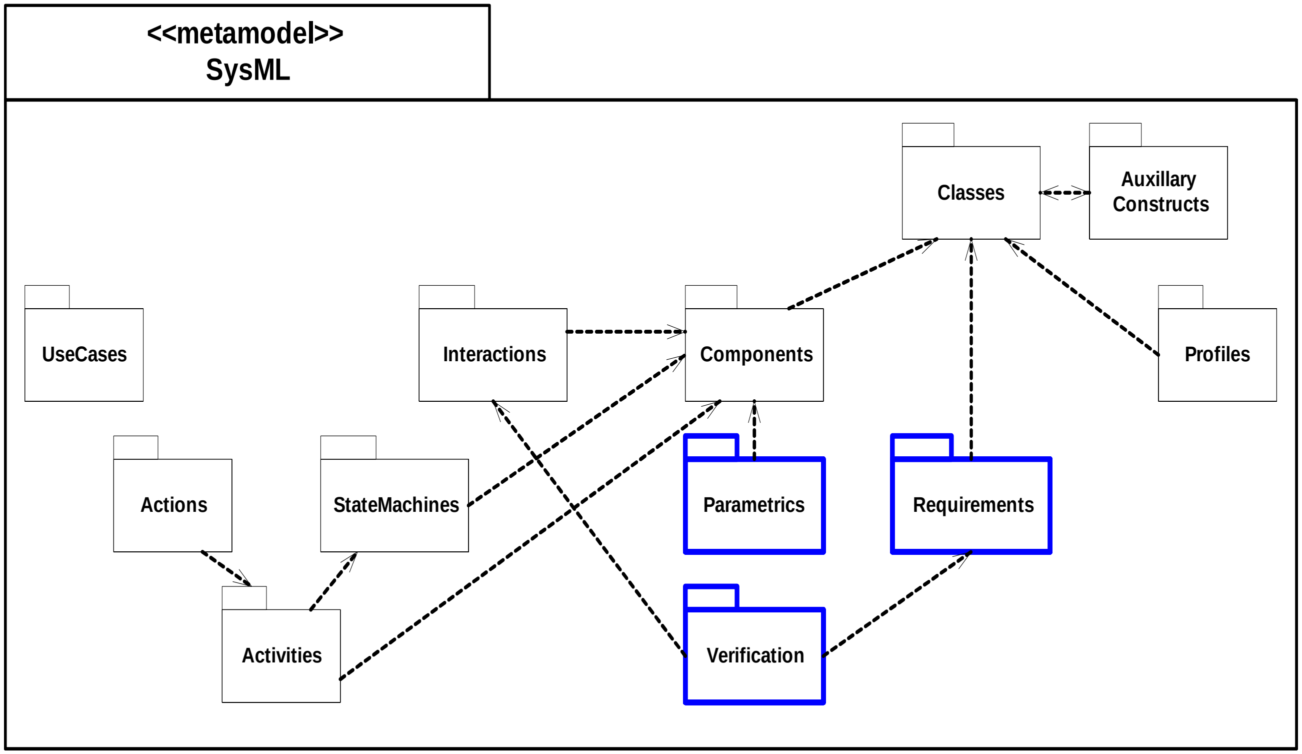
Source: SysML Partners

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

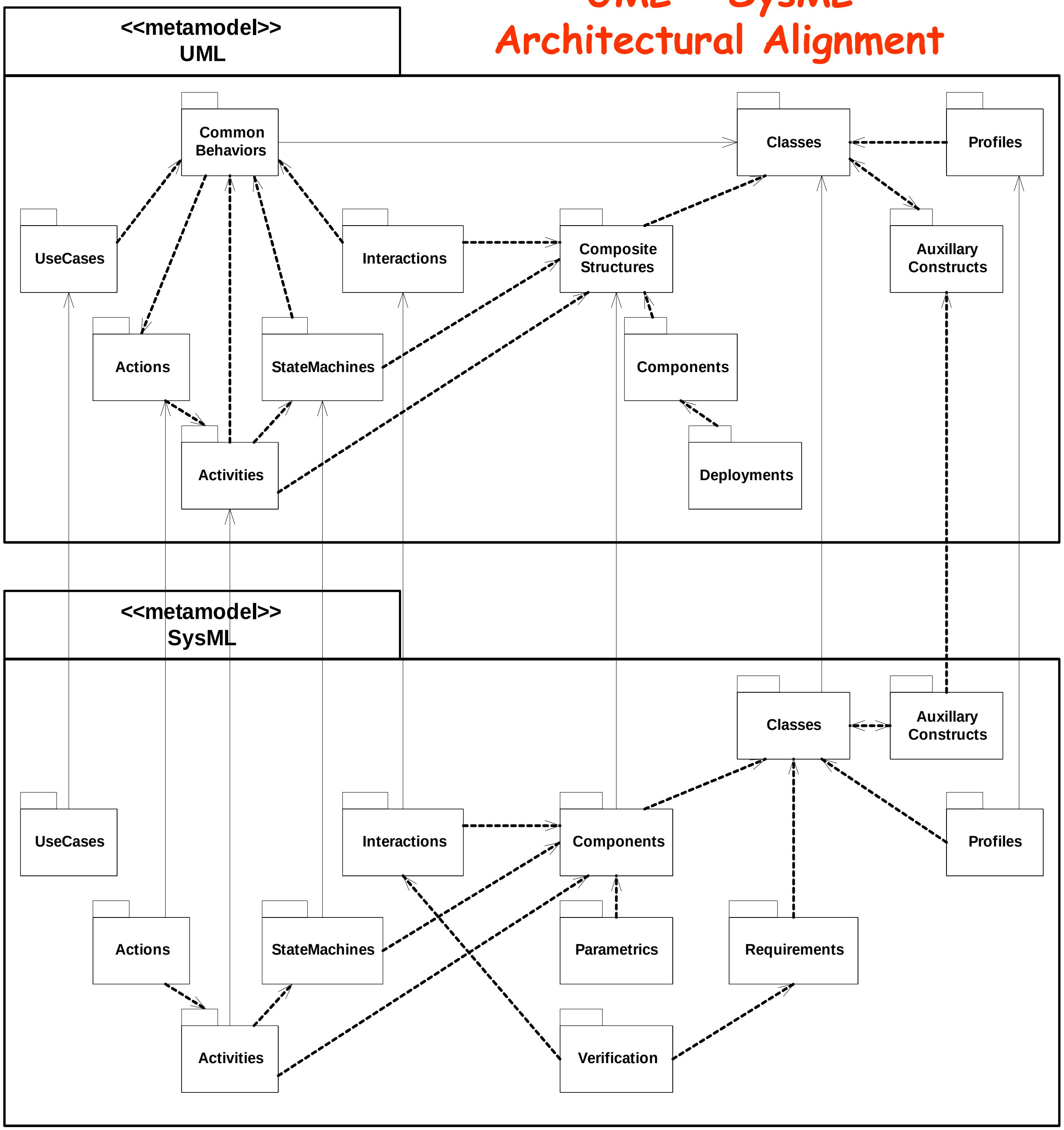
Source: SysML Partners

SysML Language Architecture



Source: SysML Partners

UML - SysML Architectural Alignment



Source: SysML Partners

SysML Extensions to UML 2

- Composite Structure Diagram
  - integrate and extend Information Items and Information Flows to include physical flows
  - include Deployment relationship
  - distinguish clearly between Component and Collaboration diagram
- Activity Diagram
  - extensions for continuous flow modeling
  - extensions to support disabling control and control operators.
  - accommodate needs of Extended Functional Flow Block Diagrams (EFFBDs)

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.

xADL Study Background

- UCI / USC Architecture Framework
  - XML architecture description language (xADL)
  - Extensible suite of tools that are driven by XML architecture specifications (Archstudio)
    - Architecture models re-configure tools / environment
  - Framework for handling CM, versioning, and product line architectures (MAE)
  - Environment for modeling system performance based upon architectural models (DeSi)

xADL Study Approach

- Develop a RASDS meta-model & instance models
  - XML, xADL, syntax driven tools
  - Independent of specific tool environments and implementations
- Provide a tool suite with a graphical interface
  - Archstudio and xADL w/ MAE as s/w arch model environment
  - Creation, manipulation, display, archiving, and versioning of meta-models
  - component and connector type templates, and instance models
- Develop specific instance models for a set of selected space systems deployments (MER / ODY / DSN end-2-end)
- Provide framework for coarse grained simulation of behavior and performance characterization
  - Model link performance, physical geometry, and protocol behavior
  - DeSi for connectivity model & simulation w/ xADL arch description

xADL Study Analysis

- Tools are academic, but widely used and well supported
  - Extensible nature should suit our immediate needs
- Use of XML based xADL provides some portability, but no great interoperability
  - Not XMI compliant
  - Some xADL constructs not mapped in UML / XMI
- We expect study to help validate RASDS
  - Model completeness
  - Model viewpoints and validation
  - Ability to simulate behavior from architectural models
- Should inform future work with SysML and/or UML

Next Steps

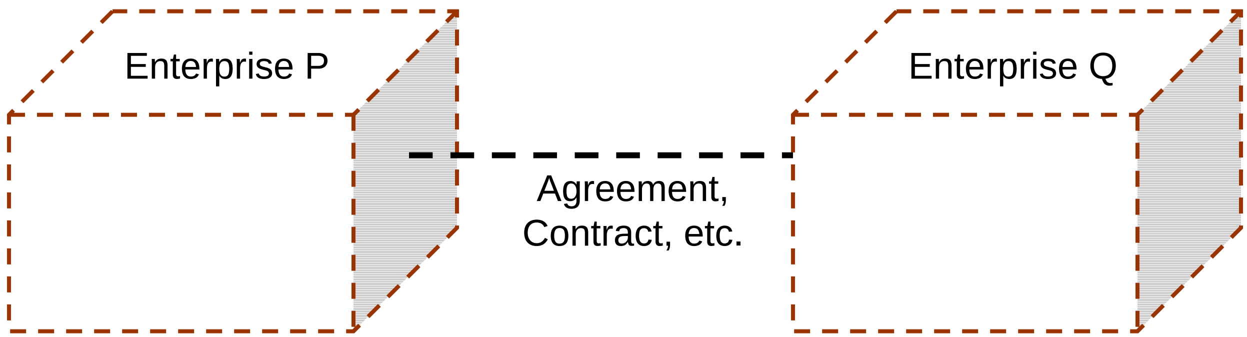
- Validate RASDS modeling approach
  - Use with various standards and mission architectural design activities
  - Complete xADL study and analysis task
- Adopt an agreed RASDS formalism
  - Select a formal method for <u>describing architectures and systems</u> (i.e., SysML, UML, or some XML interchange)
  - Agree to final common language and representation
- Generate baseline architecture and system elements
  - Develop agreed meta-model
  - Define extensible library of component instances
  - Establish extensible set of specific architectures

Acknowledgements

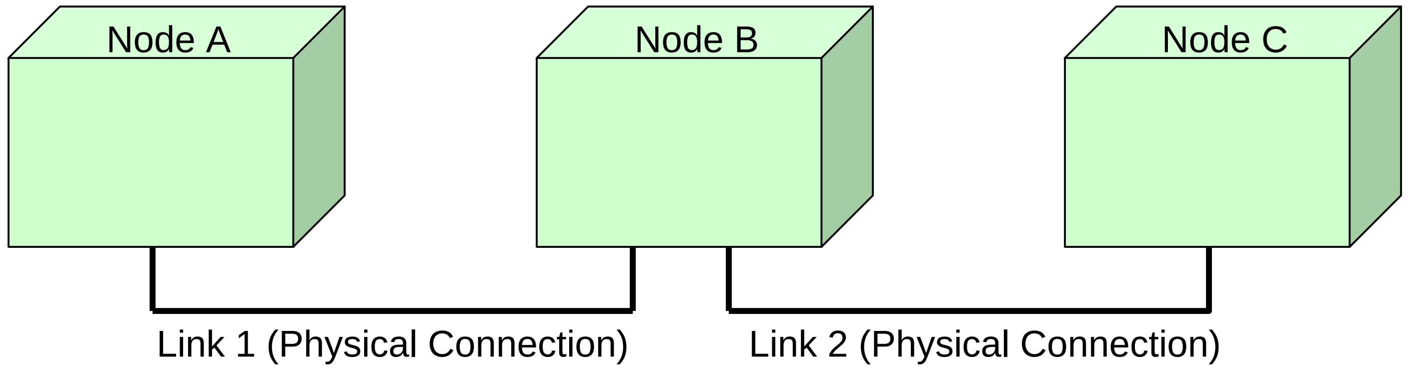
- This task was carried out as part of the program of work of Consultative Committee for Space Data Systems (CCSDS).
- It was performed by the Architecture Working group (AWG), chaired by Takahiro Yamada, ISAS
- Other AWG members who actively participated are listed below:
  - Fred Brosi, NASA/GST
  - 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

BACKUP SLIDES

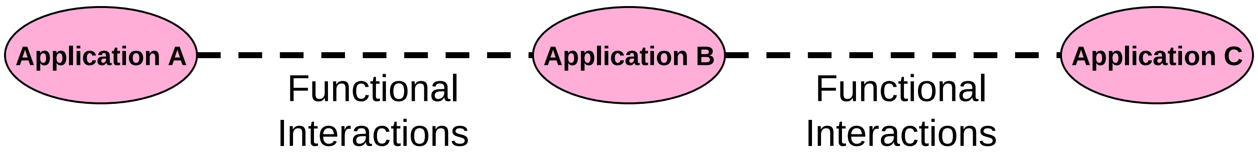
Enterprise View (Enterprise Objects)



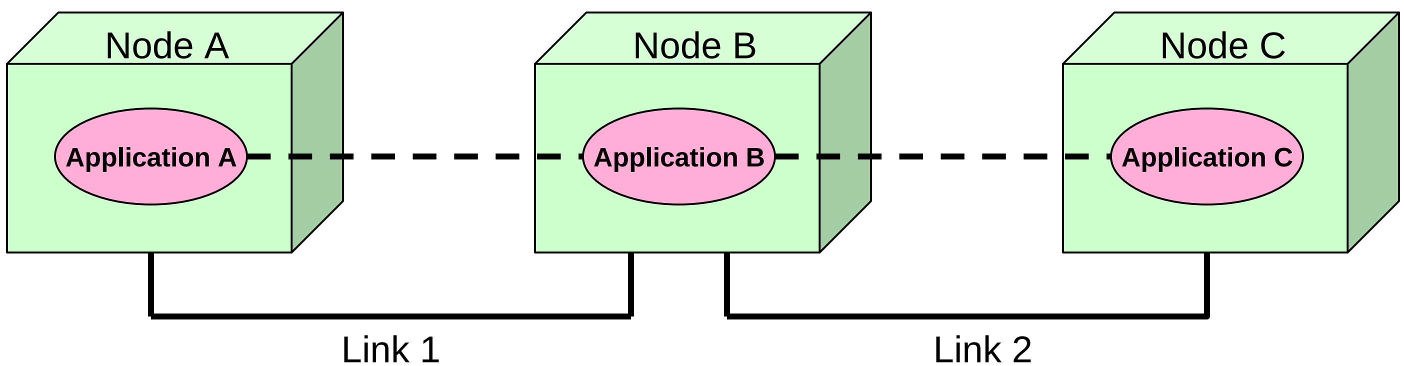
Connectivity View (Nodes and Links)



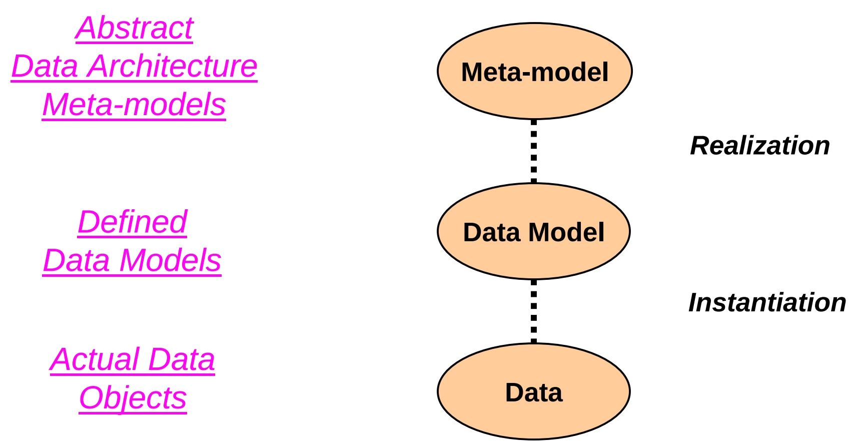
Functional View (Functional Objects)



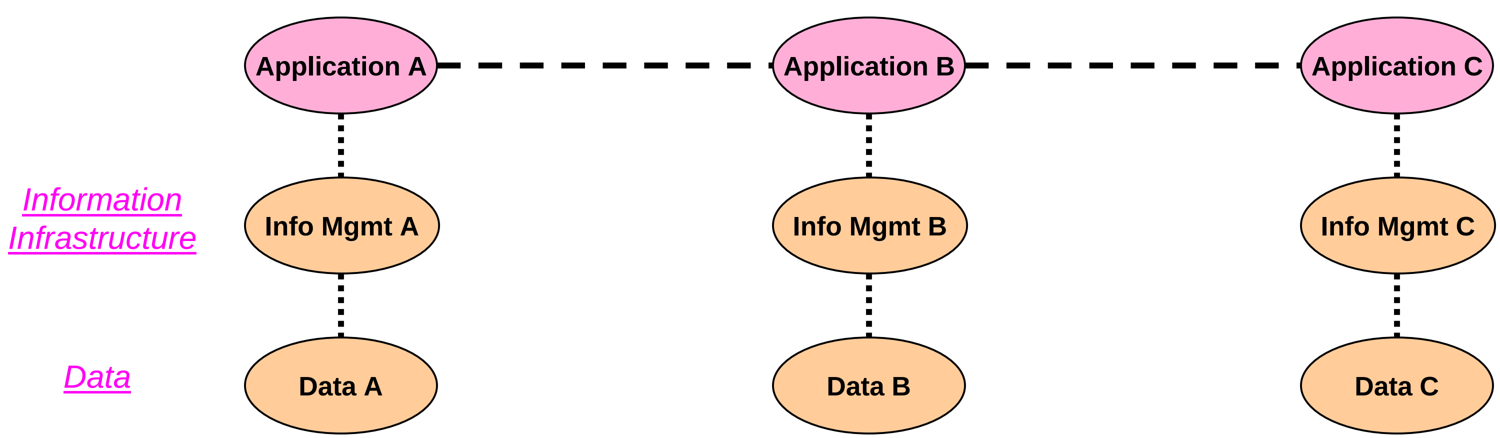
Connectivity+Functional View (Nodes, Links and Functional Objects)



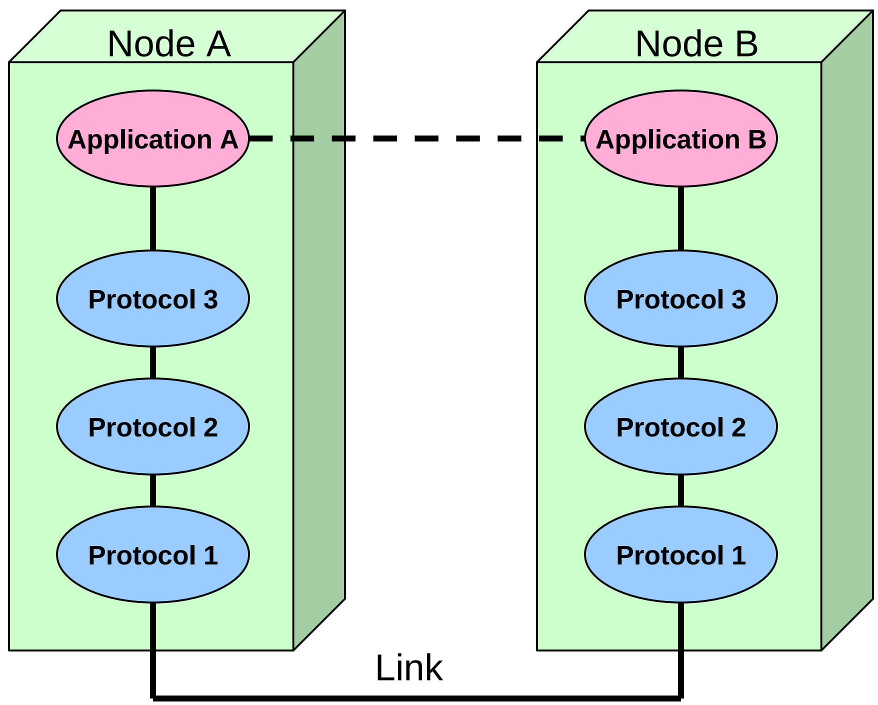
Information View (Information Objects)



Functional+Information View (Functional Objects and Information Objects)



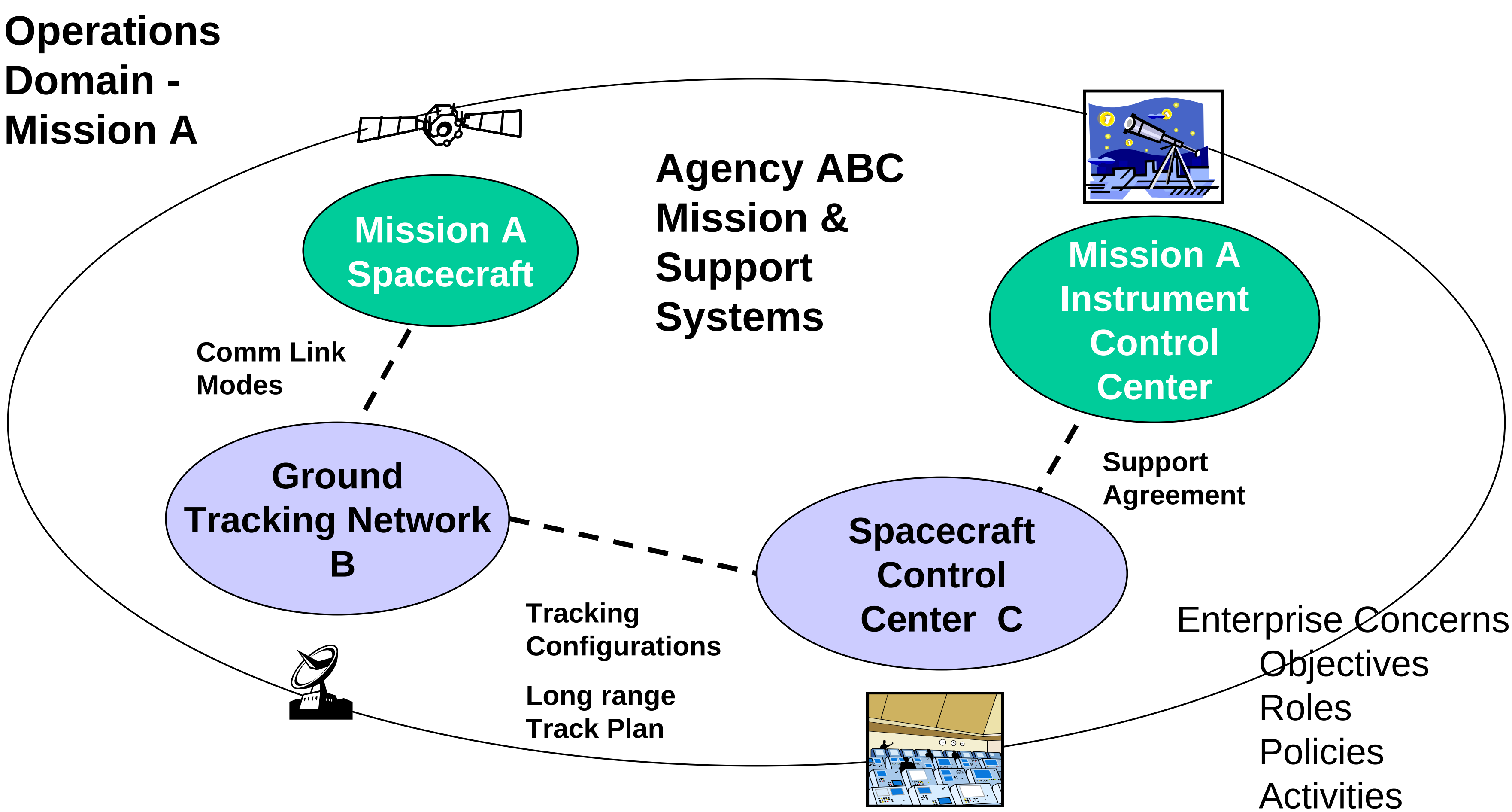
Connectivity+Functional+Communication View (Nodes, Links, Functional Objects and Communications Objects)



Enterprise View

Single Agency Mission Domain & Enterprise Objects

Operations Planning Phase



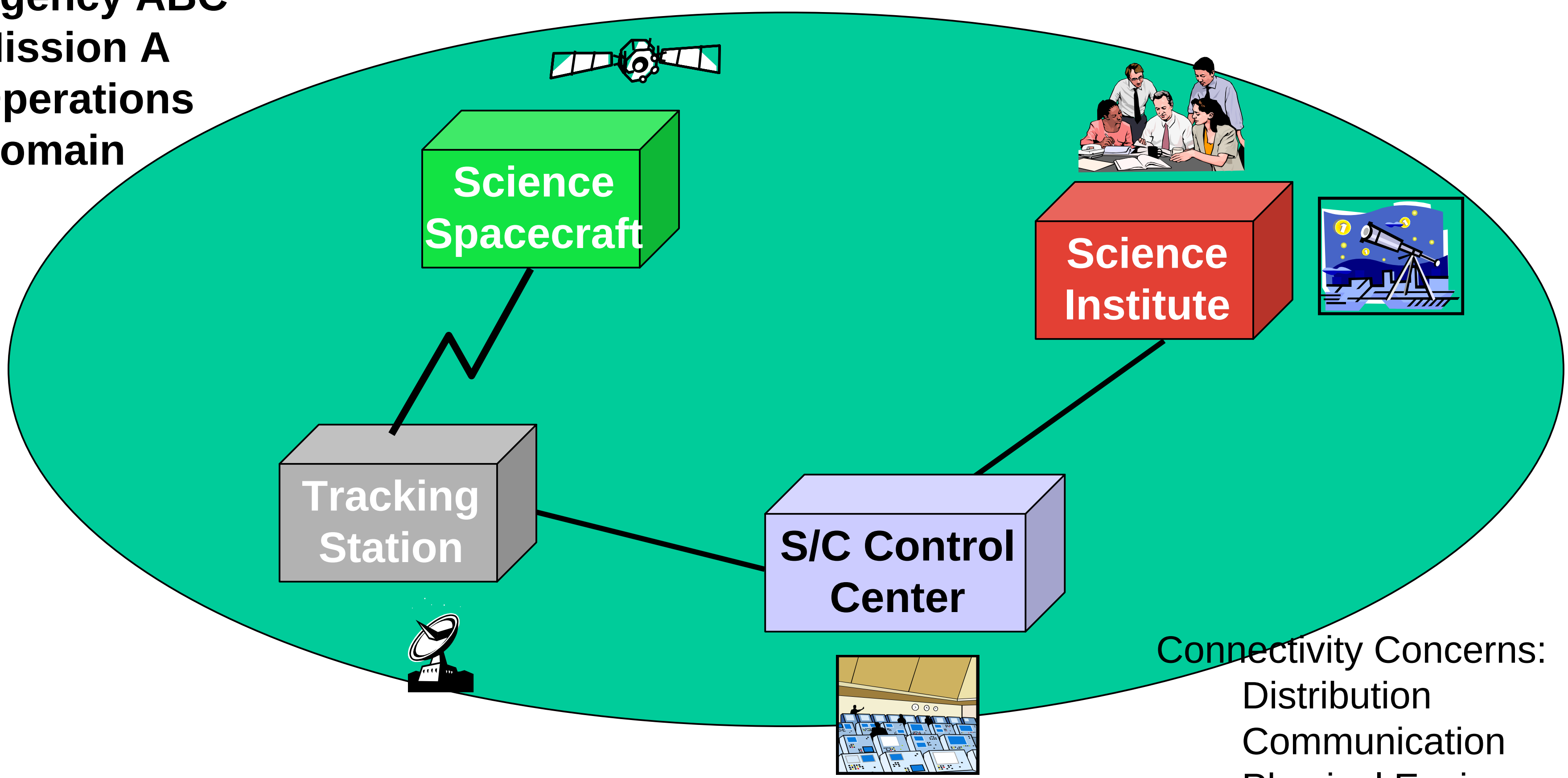
Enterprise Concerns:

- Objectives
- Roles
- Policies
- Activities
- Lifecycle / Phases
- Configuration
- Contracts

Connectivity View

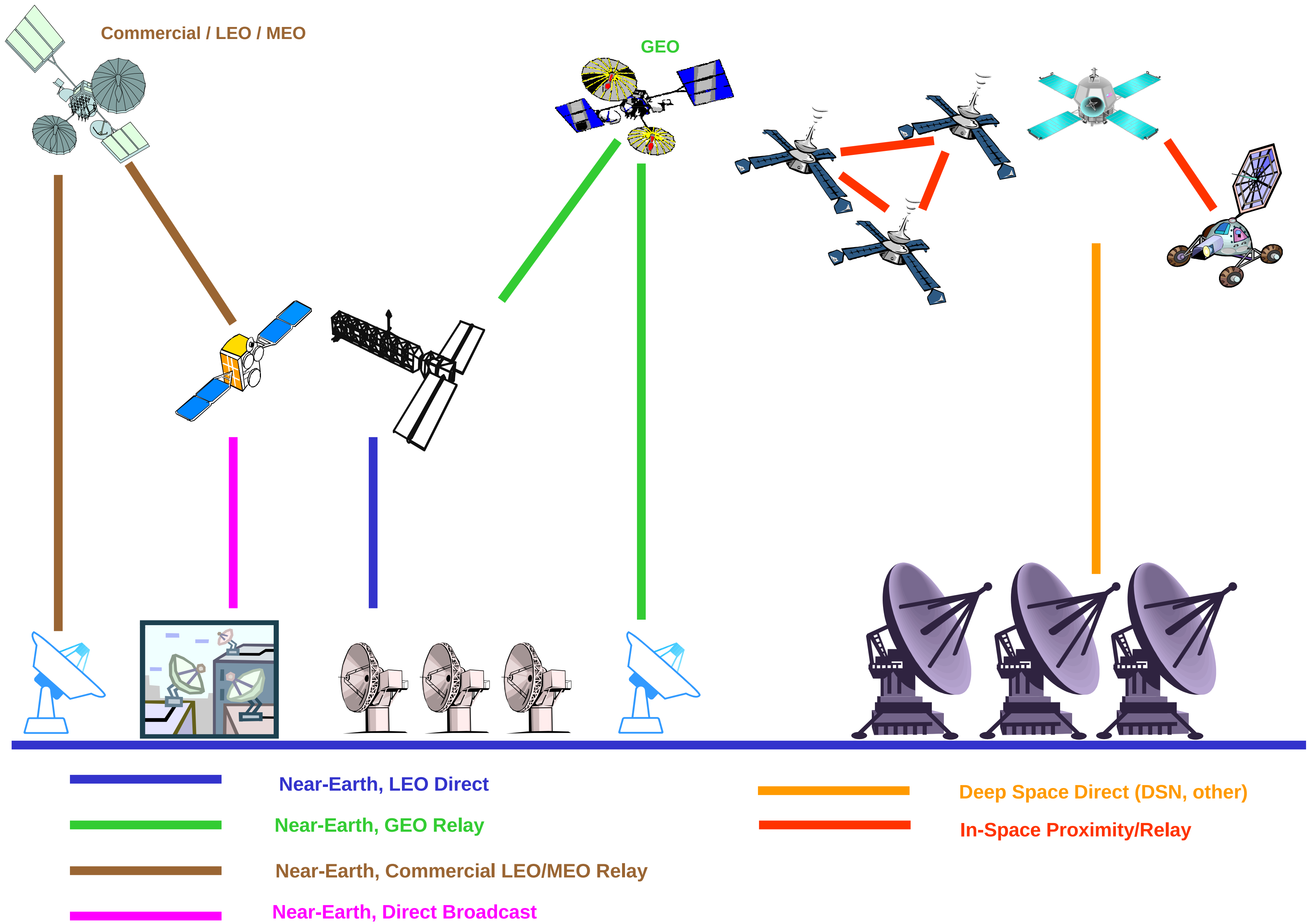
Single Agency Mission Domain & Nodes

**Agency ABC
Mission A
Operations
Domain**



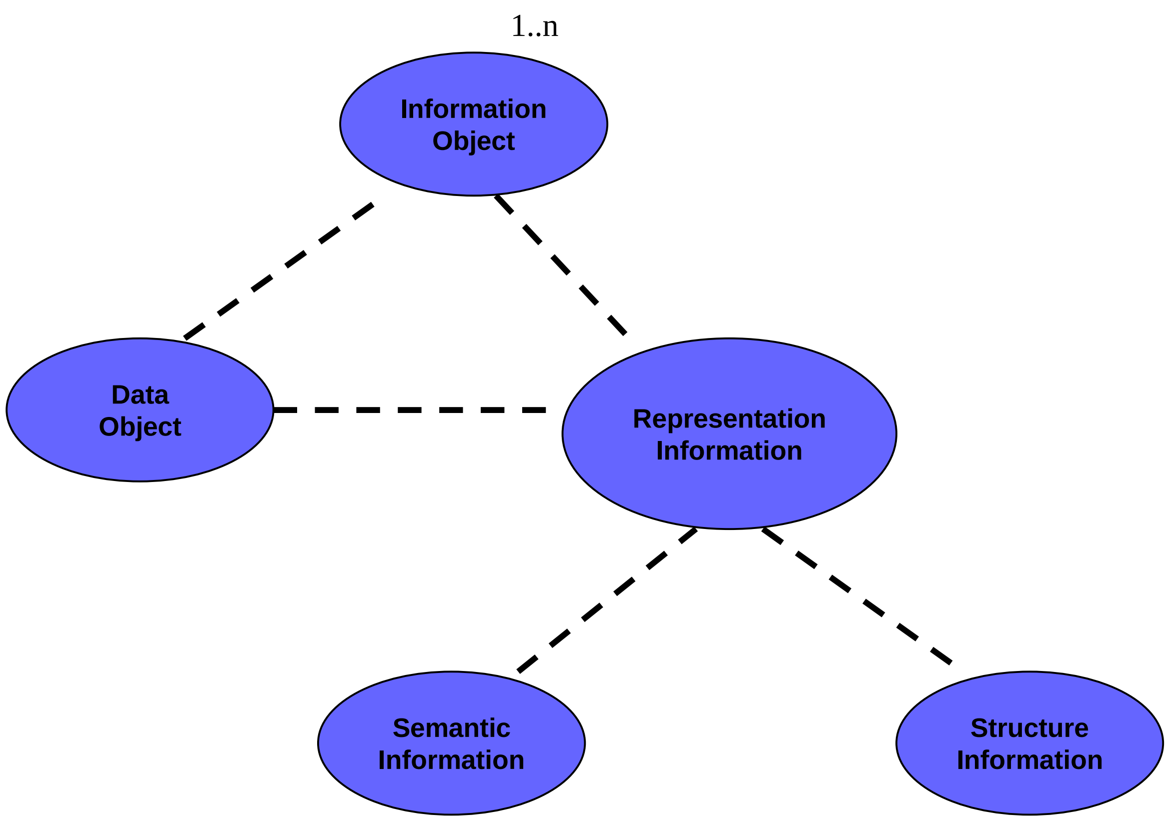
Connectivity Concerns:
- Distribution
- Communication
- Physical Environment
- Behaviors
- Constraints
- Configuration

Connector Properties: Types of Space Links



Source: A. Hooke, NASA/JPL

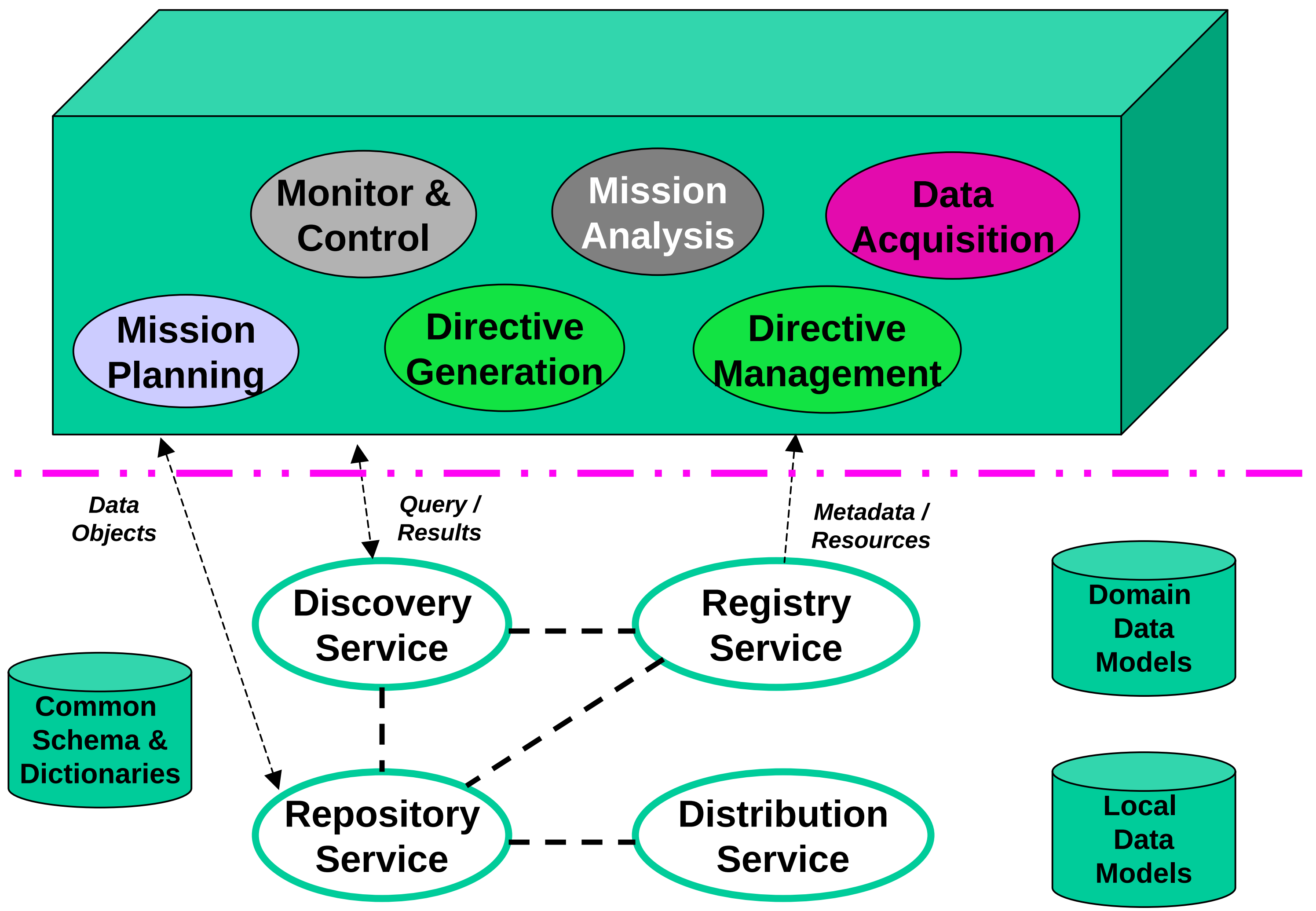
Information Object Basic Relationships



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

Space Information Management Functional Architecture

Representative Functional Objects

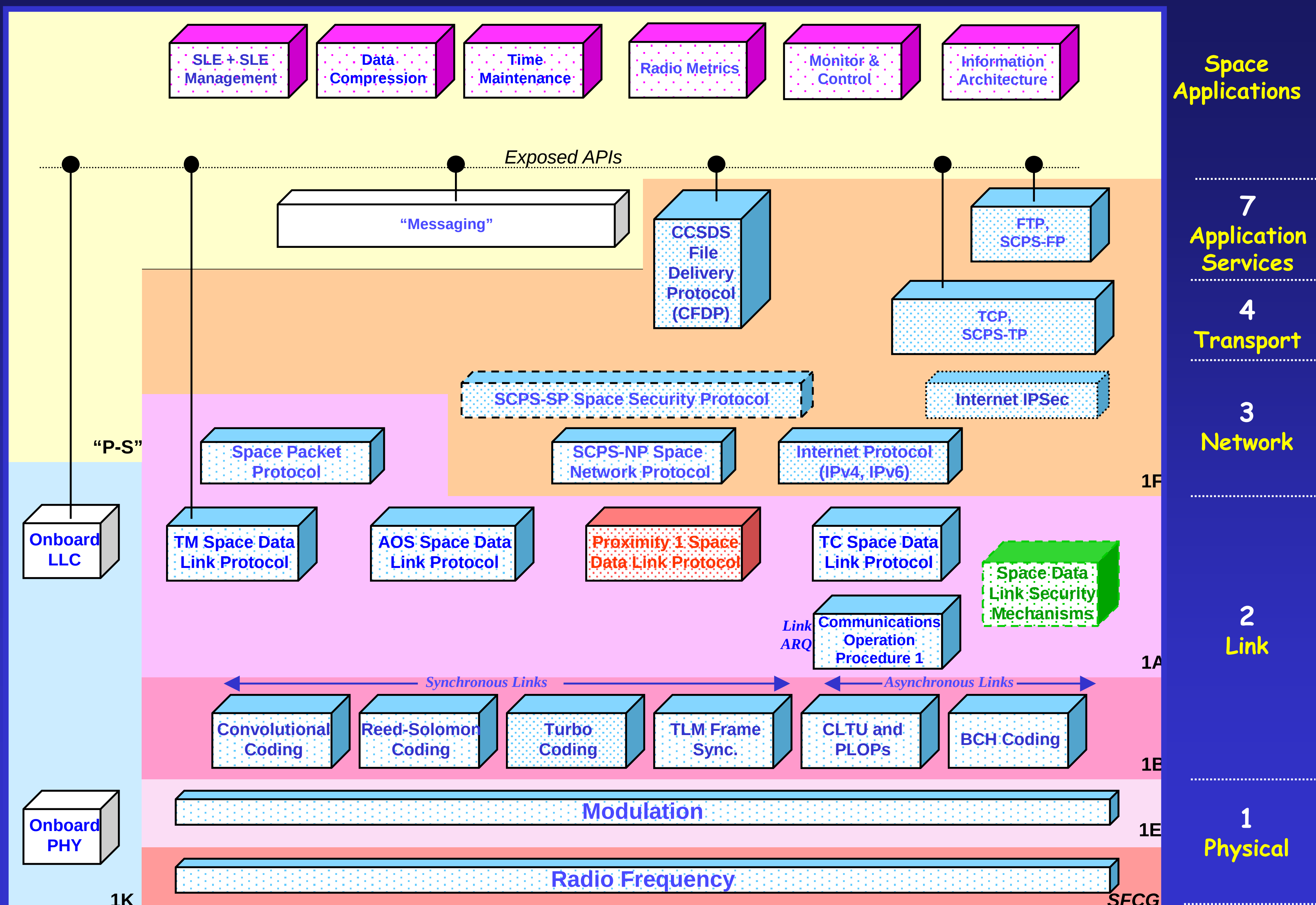


Information Management Functional Objects

RASDS Requirements, contd

- **Other Considerations:**
  - Selected set of mission / space systems deployments must be developed and agreed
  - Mission lifecycle views, concept, design, development, launch, operation
  - Architectural model lifecycle, abstract to concrete, relationship to design
  - Extracting "suitable for framing" viewpoints for different audiences from models
  - Development of prototypes of various architecture elements and approaches
  - Explore means to do trade space evaluation driven by architecture models

CCSDS Space Communications Standards



Source: Adrian Hooke, JPL