REFERENCE ARCHITECTURE FOR SPACE DATA SYSTEMS

Peter Shames

Jet Propulsion Laboratory, CIT
Pasadena, CA
peter.shames@jpl.nasa.gov

Takahiro Yamada

Institute of Space and Astronautical Science
Sagamihara, Japan
tyamada@pub.isas.ac.jp

ABSTRACT

This paper introduces the Reference Architecture for Space Data Systems (RASDS) that is being developed by CCSDS and shows how it can be used to reduce the cost of development of space data systems. RASDS uses five Views to describe architectures of space data systems. These Views are derived from the viewpoints of the Reference Model of Open Distributed Processing (RM-ODP), but they are slightly modified from the RM-ODP viewpoints so that they can better represent the concerns of space data systems.

1. INTRODUCTION

Interoperability of space data systems is of great concern to Space Agencies because sharing or reusing interoperable resources among multiple projects and multiple Agencies can reduce the cost of developing and operating space data systems. However, an on-going problem is that each space data system often has a different architecture and therefore the elements of one system cannot be easily used by other systems. Moreover, the method of describing the architecture is usually different from system to system and it is sometimes difficult to even describe the problems associated with interoperability among systems. Standard interfaces and protocols, and standard architectures, are ways of providing interoperability and reuse, and reducing costs.

To cope with this situation, the Consultative Committee for Space Data Systems (CCSDS) [1] has developed various architectures to describe space data systems. CCSDS is an international, consensus based, space system standards organization which has as members NASA, ESA, ISAS, and all the other major space agencies. Recognizing that there are already different architectures of space data systems, the approach taken by the CCSDS Systems Architecture Working Group (SAWG) was to generate a reference architecture that can be used as a framework to generate various architectures in a coherent way. This reference architecture is known as the Reference Architecture for Space Data Systems (RASDS). Using this reference architecture, architectures of different space data systems can be described in a standard way so that the commonality and differences among the systems can be easily understood.

Space agencies are designing increasingly complex missions, many of which require some level of multi-agency interoperability (spacecraft to ground system, instrument to spacecraft, spacecraft to spacecraft) or interdependency among systems (lander to orbiting relay). We have a methodology for describing these complex missions and expect to adapt a formal methodology for modeling their behavior based upon these formalized descriptions. We anticipate cost savings from using a common, highly capable approach to describing these complex systems, from using common tools and model based engineering to design these systems, from re-use of models and architectures, and for being able to share architectures and engineering models among agency partners. The major savings will come from the ability to reuse components and provide cross support among these missions.

2. OVERVIEW OF THE REFERENCE ARCHITECTURE

Space data systems are complex entities, which may be viewed from various aspects. In order to generate the architecture of a space data system in a manageable way, RASDS uses multiple Views to present the architecture of a space data system, each view focusing on one aspect of the system. The Views used by RASDS are derived from the viewpoints defined in the Reference Model of Open Distributed Processing (RM-ODP) [2] but they are slightly modified from the RM-ODP viewpoints so that they can better represent the concerns of space data systems. The views used in RASDS range from the organizational to the physical component and from abstract representation to concrete implementations, they include: Enterprise, Connectivity, Functional, Information, and Communications.

Each View is an abstraction that uses a selected set of architectural concepts and structuring rules, in order to focus on particular aspects within a space data system. Each of the Views describes the space data system in question as a set of Objects, the interactions among them,

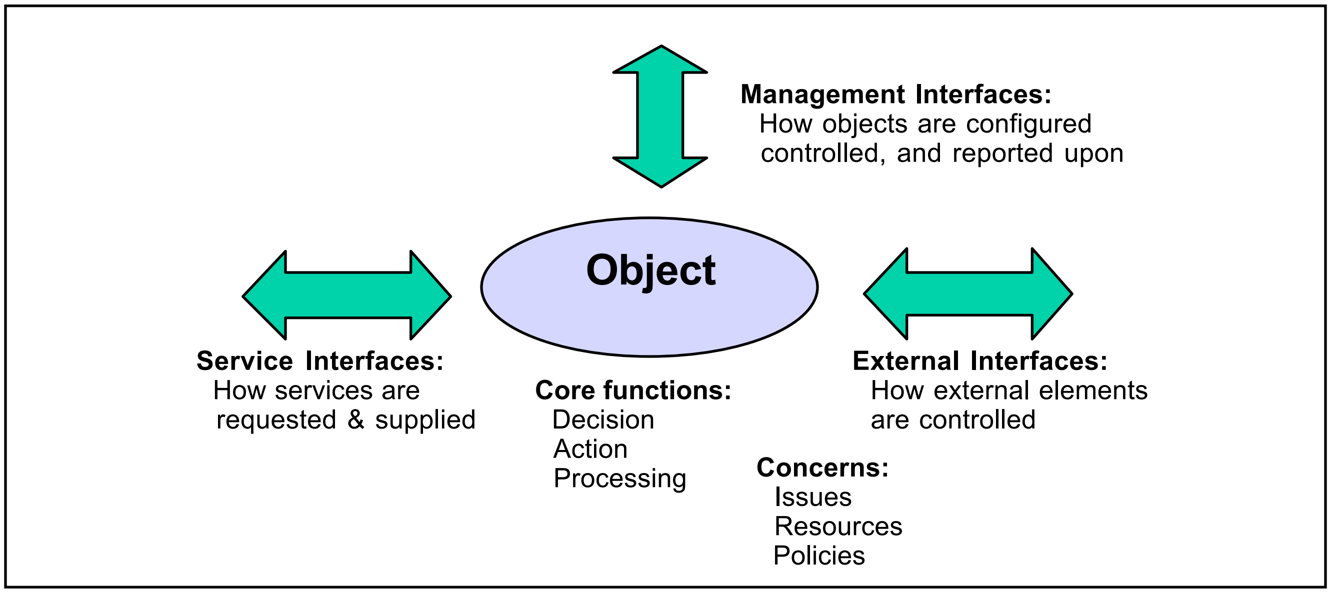

Figure 1. Representation of Objects

and the concerns that must be addressed for that viewpoint. An Object is an abstract model of an entity in the system.

As shown in Figure 1, each Object is described with its core functions and its interfaces with other Objects. Also, a set of concerns is associated with each Object.

RASDS uses the five Views that are explained in the subsequent sections to describe the architecture of space data systems. The user may decide not to use all of these five Views to describe a particular system if the system can be characterized with less than five Views. The user may also choose to combine Views using the basic concepts defined in RASDS if it is impossible to capture all the important aspects of the system with a single pre-defined View. Examples of this are shown in the text.

3. ENTERPRISE VIEW

The motivation for the Enterprise View is that we often have complex organizational relationships involving spacecraft, instruments, ground systems, scientists, staff, and contractors that are distributed among multiple organizations (space agencies, science institutes, companies, etc). The Enterprise View is used to address these aspects of space data systems and the relevant concerns that arise, i.e. polices, contracts, agreements, organizational interfaces and, from a security perspective, trust relationships.

The Enterprise View describes the organizations involved in a space data system and the relationships and interactions among them. The Enterprise View is depicted as a set of Enterprise Objects and interactions among them, where each Object is an abstract model of an

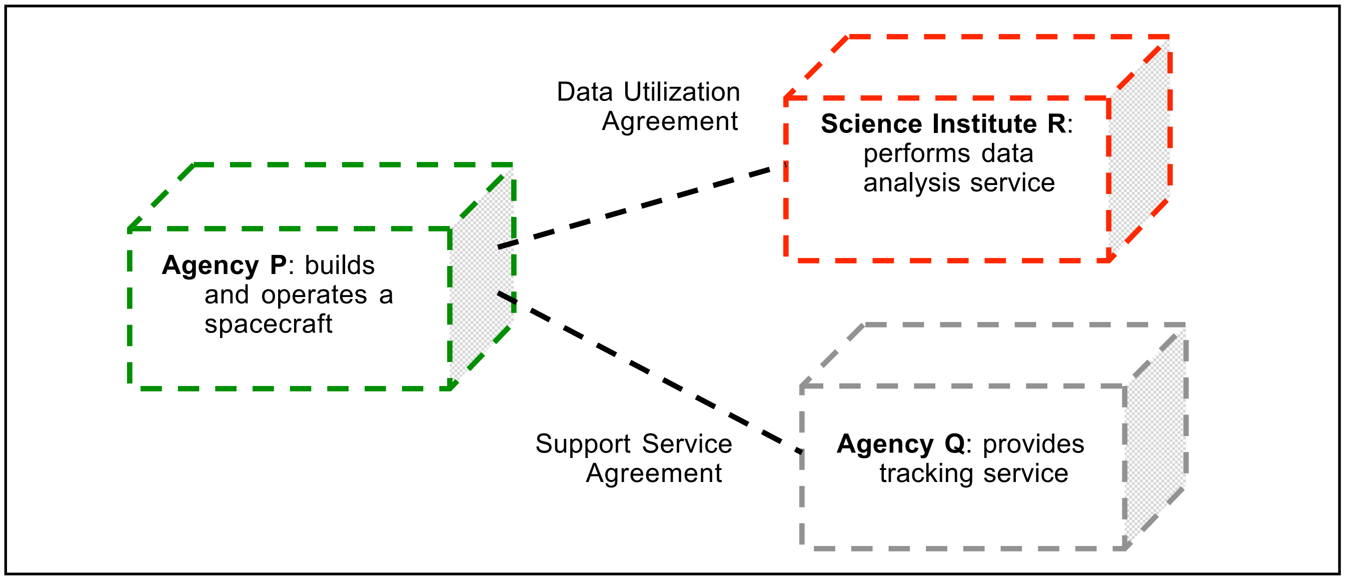
Data Utilization Agreement
Science Institute R: performs data analysis service
Agency P: builds and operates a spacecraft
Support Service Agreement
Agency Q: provides tracking service

Figure 2. Example of Enterprise View (Mission A)

organization or facility involved in a space data system. An Enterprise Object represents an independent Enterprise (such as a space agency, a government institute, a university, or a private company) or an element belonging to an Enterprise (such as a tracking network, a control center, a science center, or a research group). An Enterprise Object may be composed of other Enterprise Objects. A group of Enterprise Objects that plays some role in a space data system (such as a community, a committee, or a joint project) can also be an Enterprise Object.

Figure 2 shows an example of an Enterprise View for Mission A, in which Agency P builds and operates a spacecraft, Agency Q provides tracking support and Science Institute R performs scientific data analysis.

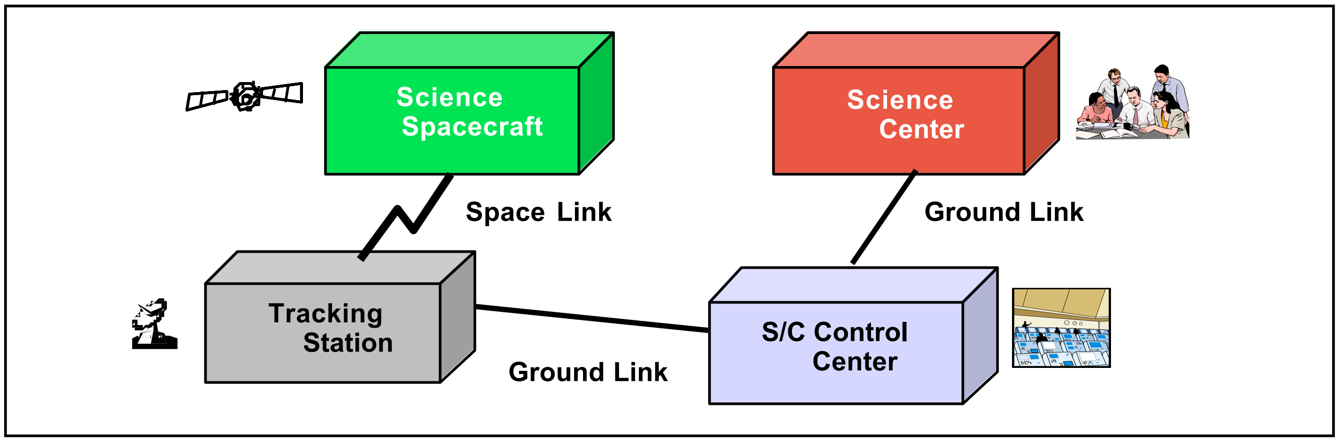

Figure 3. Example of Connectivity View (Mission A)

4. CONNECTIVITY VIEW

The Connectivity View describes the physical elements, how they are connected, and the physical environment of a space data system. The Connectivity View is depicted as a set of Nodes and Links. A Node is an abstract model of a physical entity or component used in a space data system, which is connected to other Nodes by a Link of some sort. A Node represents a system (such as a spacecraft, a tracking system or a control system) or an individual physical element of a system (such as an instrument, a computer, or a piece of equipment). A Node may be composed of other Nodes. A Link is a physical connection between or among Nodes. A Link represents an RF link, a wired link, or a network of some kind (such as the Internet, a LAN, or a bus). Both Nodes and Links have associated behavioral properties, which include performance, location, and possibly motion. The entire set of Nodes and Links is embedded in a physical environment, which has its own properties and behaviors.

The motivation for the Connectivity View is that we have system elements that are in motion through space and consequently connectivity issues associated with pointing, scheduling, long round trip light times, and low signal-to-noise ratios, all of which require special protocols and functionality to deal with. The Connectivity View is used to address all of these physical and performance aspects of space data systems. This is a concrete view of system elements, used in conjunction with more abstract views, such as the Functional View, to show allocation of functions, and with more concrete views, such as the Communications View, to show the protocols that are required to deal with the link and environmental characteristics.

Figure 3 shows Nodes and Links used for Mission A, as shown in Figure 2.

An Enterprise Object owns each Node. Figure 4 shows which Enterprise Object from Figure 2 owns which Node(s) from Figure 3.

5. FUNCTIONAL VIEW

The motivation for the Functional View is to separate functional elements and their logical interactions from the engineering concerns of where functions are housed, how they are connected, which protocols are used, or which language is used to implement them. The Functional View is an abstract view used to address these aspects of space data systems.

The Functional View describes the functional structure of a space data system and how functions interact with each other. The Functional View is depicted as a set of Functional Objects and the logical associations among them. A Functional Object is an abstract model of a functional entity that performs actions and generates or processes data in a space data system. Each Functional Object has a set of associated behaviors and a set of defined interfaces. An Object that only moves data is called a Communications Object and is treated in the Communications View. A Functional Object may be

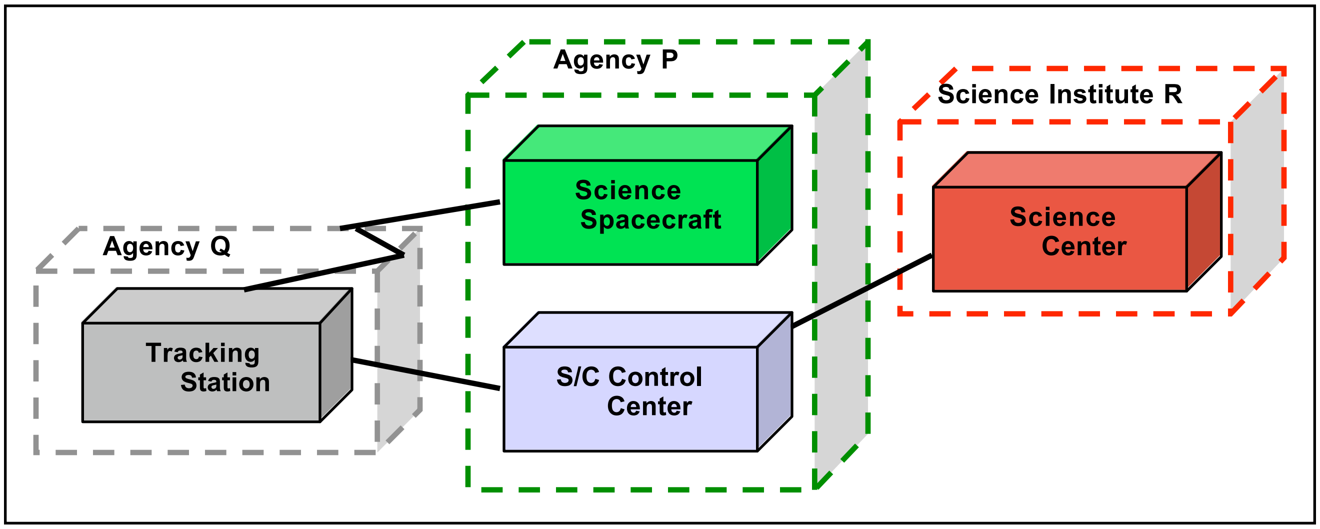

Figure 4. Example of Enterprise and Connectivity Views (Mission A)

realized as either software or hardware. A Functional Object may be composed of other Functional Objects. A Functional Object may use a service provided by other Functional Objects, provide a service to other Functional Objects, or perform actions jointly with other Functional Objects. These kinds of interactions are described in the Functional View.

Figure 5 shows some of the Functional Objects used for Mission A together with the logical associations between them (shown with dotted lines).

Functional Objects actually reside in physical entities (i.e., Nodes) of the system. Overlaying the Functional View on the Connectivity View of the same system will show the distribution of Functional Objects among Nodes. Such an example is shown in Figure 6, in which the Functional Objects from Figure 5 are overlaid on the Connectivity View from Figure 3. The allocation of Functional Objects to Nodes is a part of the system design trade space.

6. INFORMATION VIEW

The motivation for the Information View is to clarify relationships among data objects that are passed among the functional elements, and to define their structures, relationships, and policies. Data objects are managed (that is, stored, located, accessed, and distributed) by information infrastructure elements. The Information View is used to address these aspects of space data systems.

The Information View describes the space data systems from the perspective of the Information Objects that are exchanged among the Functional Objects. It includes descriptions of Information Objects (their structure and syntax), information about the meaning and use of these Objects (contents and semantics), the relationships among Objects, rules for their use and transformation, and policies on access. It also provides descriptions of the Distributed Information Infrastructure (DII) that supports the location, access, delivery, and management of these Information Objects and descriptions of the Information Management Functional Objects that support the operations of DII. Finally, this View shows the relationship between the Information Objects and the Functional Objects that manipulate and exchange them.

Figure 7 shows the relationship between some typical Functional Objects and the Information Objects that they exchange. This example shows a mission planning flow

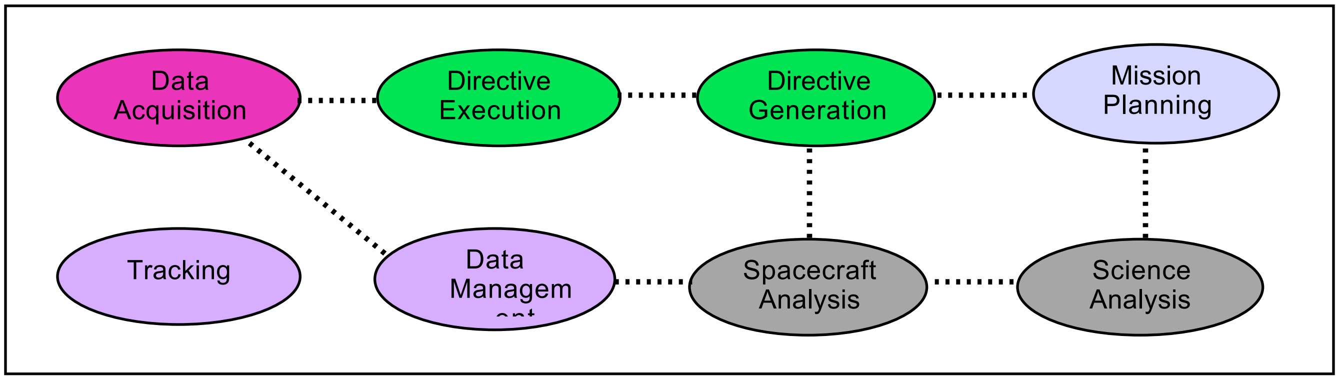

Figure 5. Example of Functional View (Mission A)

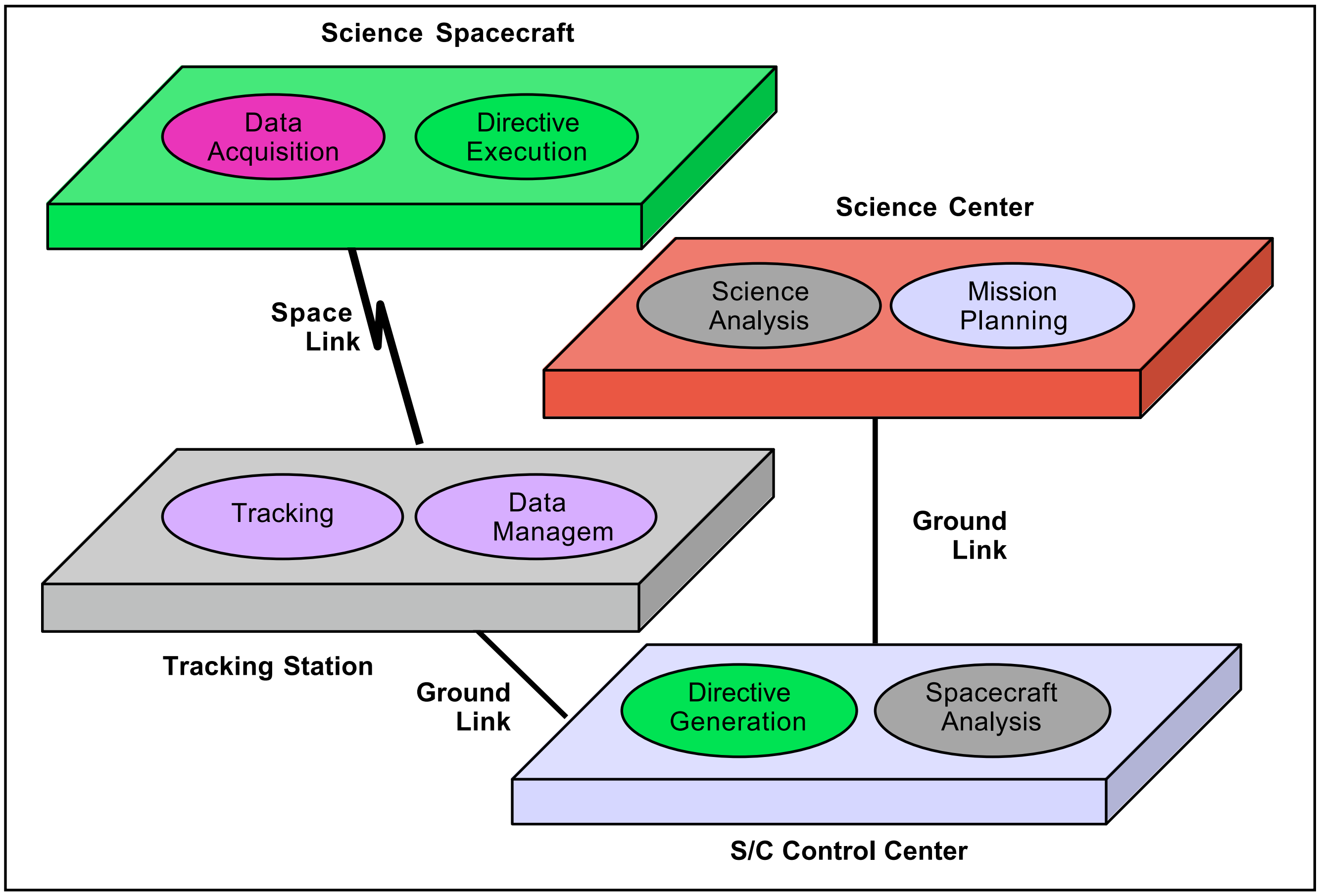

Figure 6. Example of Functional and Connectivity Views (Mission A)

for Mission A, where the green objects are Functional Objects and the blue objects are Information Objects.

7. COMMUNICATIONS VIEW

The motivation for the Communications View is to define the layered sets of communications protocols that support communications among the functional elements. These protocols, and the Communications Objects that implement them, are needed to meet the requirements imposed by the connectivity and operational challenges. The Communications View describes the engineering solutions to these space data systems challenges and is a key area of technical focus within CCSDS. The Connectivity View describes the operating environment and the physical connections among Nodes and Links.

The Communications View describes the mechanisms for information transfer among physical entities (i.e., Nodes) in a space data system. The Communications View is depicted as a set of Communications Objects and interactions among them. A Communications Object is an abstract model of a communications protocol that may be realized as either software or hardware. Communications Objects support information transfer between or among Functional Objects over Links (i.e., physical connections between or among Nodes). A stack of Communications Objects is usually used to support information transfer from a Functional Object to another Functional Object for

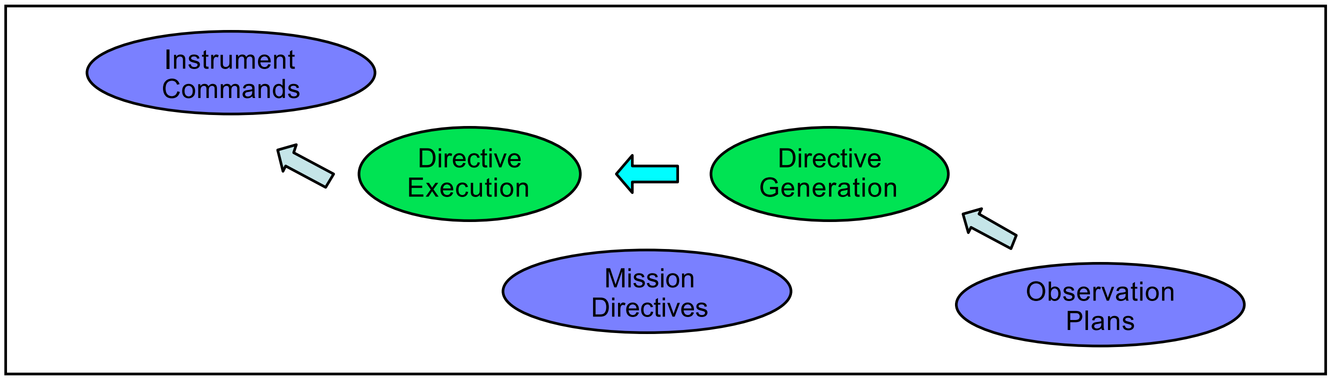

Figure 7. Example of Information and Functional Views (Mission A)

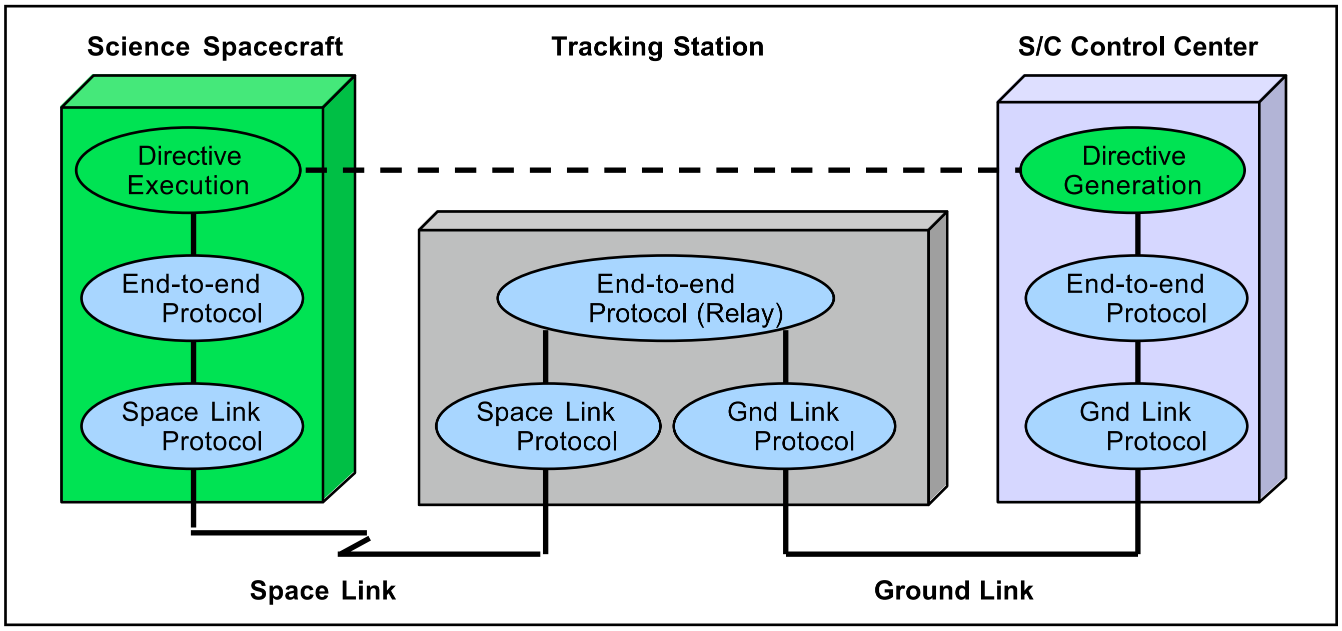

Figure 8. Example of Communication, Functional and Connectivity Views

a sequence of functional interactions. In the communications stack, the topmost Communications Object directly supports the Functional Object, and the lowest Communications Object handles the Link.

The selection of Communications Objects to support information transfer between Functional Objects over a Link heavily depends on the characteristics of the Functional Objects, the Nodes, the physical Link and the space environment. Therefore, it is useful to show the Functional Objects, the Nodes and the Link together with the Communications Objects in the Communications View.

Such an example is shown in Figure 8, in which the Communications View (Communications Objects) are overlaid with a simplified Functional View (Functional Objects) and the Connectivity View (Nodes and Links).

8. APPLYING THE REFERENCE ARCHITECTURE

The RASDS can be used for comparing and analyzing different systems in a systematic way. Each space data system has a set of functions, but there are several design choices on how to implement these functions in the system. For example, some functions can be implemented on the spacecraft or on the ground, and if they are implemented on the spacecraft, they can be centrally located at the central data handling sub-system or distributed among several sub-systems.

A simple example of such a distribution of functions to physical elements is shown in Figure 6 as a distribution of Functional Objects to Nodes. In this example, mission planning is performed at the science center and observation plans are sent from the science center to the spacecraft control center, where observation plans are converted to mission directives by the directive generation Functional Object (see also Figure 7 that shows information transferred between Functional Objects).
Depending on the constraints imposed by system or operational requirements, observation plans may be transferred directly to the spacecraft, where mission directives are generated and executed, instead of generating the low level directives on the ground and shipping these to the spacecraft. In such a case, the Communications Objects (i.e., protocols) to support transfer of information between Functional Objects may need to be re-selected to match the physical and operational environments of information transfer.

In a similar fashion, some highly autonomous missions, or several missions that are collaborating to serve some end science goals, may require distributed information management and access functionality. The interfaces can be conveniently analyzed using the Functional View and the implications of distributing this functionality can be analyzed using the Connectivity View, possibly in conjunction with the Communications View. An example of such a Connectivity View is shown in Figure 9.

The RASDS can be used to present these different designs in a unified way so that engineering issues associated with each of the possible designs can be analyzed systematically. Given a sufficiently complete set of attributes for physical components and links, and an adequate model of the connectivity and protocol performance, it will be possible to model the end to end performance and science return of different mission

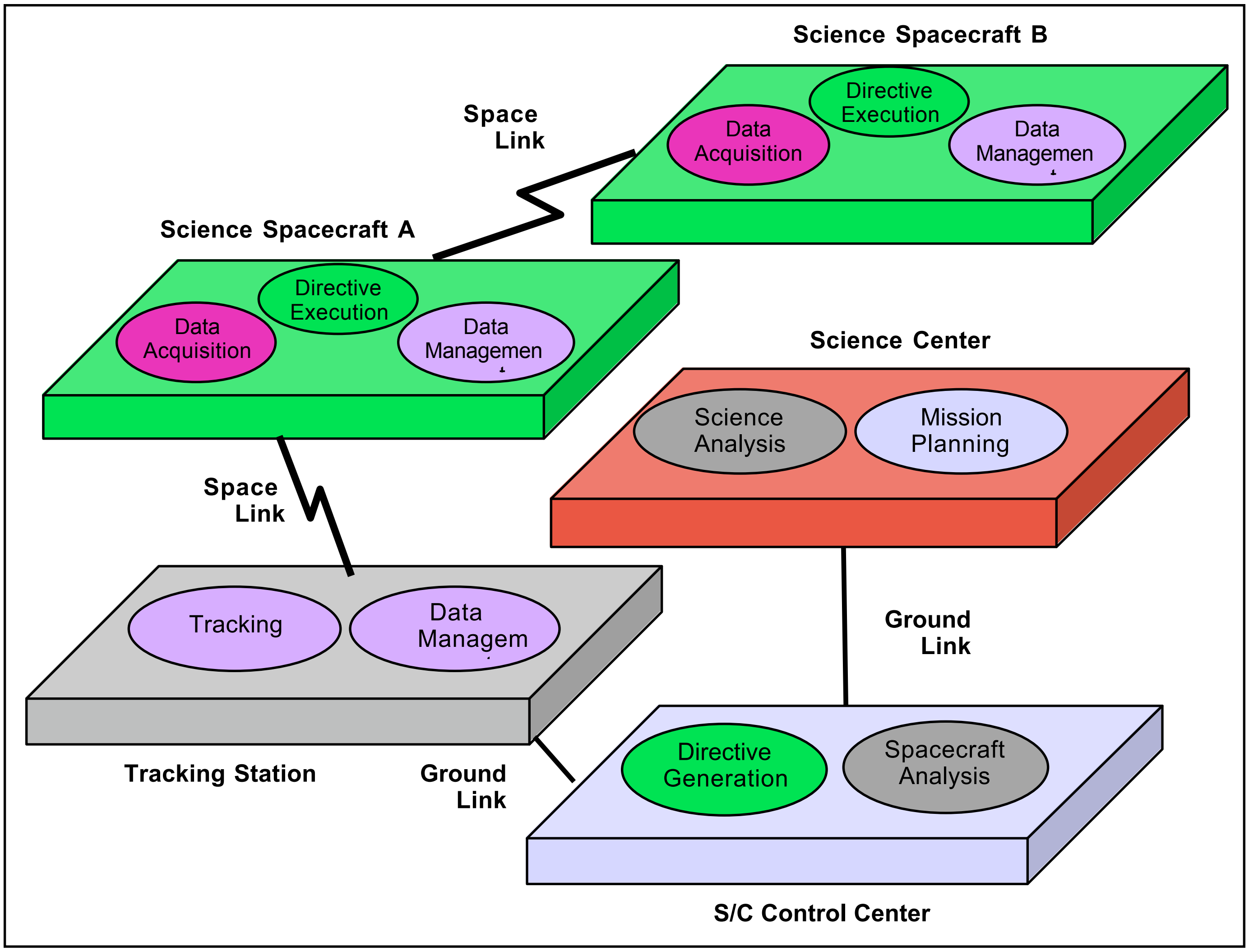

Figure 9. Example of Multi-Mission Functional and Connectivity Views

architectures.

9. COST REDUCTION WITH RASDS

This paper has presented the conceptual framework of RASDS. Since there has been no common architectural framework for space data systems, RASDS will be used as a standard framework by system architects and system developers. This will reduce the cost of system development by eliminating the need of developing individual frameworks.

Our next step is to develop formal methods for describing these architectures (for example, UML profiles and/or XML schemas). With these methods, each View of a space data system will formally be described with the Objects contained in the Views and the interactions among the Objects. The characteristics of Objects, their behaviors, and their interactions will also be formally described. These formal methods will enable sharing and exchange of information on architectures and systems among different organizations or teams, and eliminate the need of re-generating the same information for different purposes, which happens quite frequently in actual system development.

Together with the formal methods for describing architectures, we plan to develop software tools, based on existing commercial or academic tools, for generation and manipulation of architectures. These tools will facilitate generation and manipulation of architectures and information on the architectures. By using the formal methods and tools, information on an architecture or a system will be electronically generated by the architect or developer and then delivered to the engineering teams who use the information for building, testing and using the system. For example, architectural information on a system generated by the architect can be directly fed to a generic simulator, which simulates the behavior of the system using the received architectural information. The same information can also be fed to software tools used for detailed design and documentation of the system in which the processes for design and documentation will be initiated using the received information.

Therefore, the approach described above will greatly facilitate automating the system design processes and this will greatly reduce the cost of system development.

To summarize, we anticipate cost savings from using a common, highly capable approach to describing these complex systems. These saving will come from:

- Using a common, highly capable approach to describing these complex systems
- Using common tools
- Using model based engineering to design these systems
- Reuse of models and architectures
- Sharing architectures and engineering models with partners
- Development of standards to implement cross support
- Reuse of complaint components
- Cross support among agencies & missions.

10. CONCLUSION

This paper has briefly presented the Reference Architecture for Space Data Systems (RASDS) that is being developed by the CCSDS Systems Architecture Working Group (SAWG). The SAWG generated some sample architectures (spacecraft onboard architectures, space link architectures, cross-support architectures) using this RASDS approach, and RASDS was proven to be a powerful tool for describing and relating different space data system architectures.

Some simple examples were provided to show how to apply the RASDS approach to the analysis of mission design trades. The ability to separate the different views and therefore simplify the analysis of different elements in the trade space should prove to be beneficial during mission design. The European Space Agency (ESA) in a European technology harmonization of Ground Software System is now applying the RASDS approach. RASDS will provide high level views and XASTRO, which uses xADL [3] and UML [4] to model the systems, will be used as the method to describe the ground segment reference architecture.

Many aspects of space data systems that are considered in the RASDS have not been addressed in this brief paper, but are covered in the full report of the SAWG. These include security, system management, engineering details, lifecycle issues, IV&V, and other aspects of designing and building real systems. This Reference Architecture offers a consistent way of dealing with a variety of critical system viewpoints, starting with high low level abstractions and work toward more concrete realizations and implementations.

As missions become more complex, and more interdependencies are required between projects and among agencies, having clear architectural models will be essential. Significant costs savings are possible both in the architecting process itself and in the development of standards and systems components that are compliant with these architectures.

11. ACKNOWLEDGEMENTS

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the Institute of Space and Astronautical Science (ISAS) and the National Aeronautics and Space Administration. The authors wish to thank the members of the CCSDS System Architecture Working Group (Eduardo Bergamini, Fred Brosi, Adrian Hooke, Lou Reich and Don Sawyer) for their valuable contribution to the development of the RASDS presented in this paper.

12. REFERENCES

[1] http://www.ccsds.org

[2] *Information Technology - Open Distributed Processing - Reference model: Overview*, International Standard, ISO/IEC 10746-1, December 1998.

[3] *xADL,: Enabling Architecture Centric Tool Integration with XML*, Khare, et al, UCI

[4] *Unified Modeling Language, (UML)*, http://www.omg.org/uml/