Toward a Framework for Modeling Space Systems Architectures

Peter Shames[1] and Joseph Skipper, PhD[2]
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA

This paper describes an approach for developing a framework to describing the kinds of complex system architectures that are involved in designing space systems. This framework leverages earlier work that has been done in CCSDS to develop a Reference Architecture for Space Data Systems (RASDS), but extends it to define the additional views required by the physical aspects of operating systems in space. These modeling concepts were applied in a Model Based Engineering and Design (MBED) project at JPL which demonstrated that a common information model could be used for different purposes and at different levels of abstraction during a design process. The focus in this paper is on the concepts in this modeling framework and how they relate to other current architectural modeling methods.

1. Overview

Since the early 1990's there have been a number of efforts to define powerful and extensible approaches for describing a general class of software intensive system architectures. The DoD has funded many of these, but some have sprung from international or national efforts in ISO or the IEEE. These approaches are typically focused upon architectures of terrestrial systems and they range from broadly applicable, moderately formalized, approaches like the Unified Modeling Language (UML) [1], to more focused approaches such as the Systems Engineering Modeling Language (SysML) [2], and RM-ODP (Reference Model of Open Distributed Processing) [3]. There is a recent UML for RM-ODP (UML4ODP) effort that is providing a UML formalism for ODP [4]. In alignment with the recommendations made in the ANSI/IEEE 1471-2000 Recommended Practice for Architectural Description of Software-Intensive Systems [5], all of these approaches support an appropriate set of viewpoints for developing system architectural descriptions.

All of these standard architectural approaches are intended to describe large-scale terrestrial data systems that are inherently complex, but are typically fixed in one place and often designed and built by a single organization. In the world of space systems there is an even higher level of complexity in that these are most often multi-organizational developments that are best characterized as systems of systems. In further contrast to terrestrial systems the most challenging elements of these systems, the spacecraft, are not fixed in place, but are flying through space at high velocity, must use specialized ground and space communications assets and protocols, are often at great distance from the Earth and are frequently out of contact with their control centers.

These attributes of space communications systems drive architectural complexity and require consideration of issues that are not typical in terrestrial systems, and the set of viewpoints that the existing standard approaches define are not completely adequate to the task of describing space systems. Work has been done during the last couple of years to model space data systems using a methodology called the Reference Architecture for Space Data Systems (RASDS) [6] that is derived

[1] Manager, JPL Data System Standards Program, 4800 Oak Grove Drive, MS 301-265, Pasadena, CA, USA
[2] Senior Member of the Technical Staff, 4800 Oak Grove Drive, MS 301-180, Pasadena, CA, USA

from RM-ODP. This modeling approach augments the RM-ODP viewpoints on a system by adding ones that deal directly with space communications and protocols, physical element connectivity, and their interactions with the environment. The efficacy of RASDS has been demonstrated by its use in several NASA projects to describe the end-to-end architectures of their spacecraft data systems.

Recent work has been performed at JPL in an internal research and development project called Model Based Engineering and Design (MBED) [7] to extend this RASDS approach to capture all of the other physical aspects of space systems in an extended model. Modeling the design of space systems necessitates inclusion of one or more viewpoints that deal with the rest of the physical attributes of these systems and their interaction with the environment. These other attributes include mass, power, propulsion, thermal, structure and dynamic control, in addition to the component (node), connector (link), gravitational and environmental aspects already in the RASDS connectivity viewpoint.

This conceptual approach is intended to be general enough to permit description of civilian, military, and commercial space systems, the spacecraft physical and logical design, ground systems, processing and communications resources, and organizational arrangements. The method focuses upon how to represent the technical end to end system architectural elements and their logical and physical interfaces and interactions. In this paper we will describe this extended RASDS methodology, the set of viewpoints that we have derived, its use in the MBED task, and describe their relationship to RM-ODP and other related methods.

2. Introduction to Architectural Modeling

The IEEE Recommended Practice for Architectural Description of Software-Intensive Systems, IEEE 1471-2000, provides some very useful definitions and guidelines for what system architecture is and for the use of viewpoint specifications to address the identified set of stakeholder concerns.

The scope of this recommended practice encompasses those products of system development that capture architectural information. This includes architectural descriptions that are used for the following:

a) Expression of the system and its evolution

b) Communication among the system stakeholders

c) Evaluation and comparison of architectures in a consistent manner

d) Planning, managing, and executing the activities of system development

e) Expression of the persistent characteristics and supporting principles of a system to guide acceptable change

f) Verification of a system implementation's compliance with an architectural description

g) Recording contributions to the body of knowledge of software-intensive systems architecture

The IEEE 1471-2000 specification describes the process for developing architecture descriptions under a number of scenarios, including precedented and unprecedented design, evolutionary design, and capture of design of existing systems. In all of these scenarios the overall process is the same: identify stakeholders, elicit concerns, identify a set of viewpoints to be used, and then apply these viewpoint specifications to develop a set of relevant views of the system.

IEEE 1471-2000 indicates the utility of defining system, functional, and technical views (among others), however, in order to maintain generality it does not go so far as to define any specific set of views nor what these might be used for. While all of the definitions in IEEE-1471 provide useful guidance as to process and terminology, they provide little in the way of practical direction for

actually defining an architecture methodology for space systems, nor do they offer pragmatic guidelines for describing system architectures, particularly space system architectures. Maier, et al, in ANSI/IEEE 1471 and Systems Engineering [8] address the strengths of IEEE-1471 and how it relates to other more prescriptive methods like RM-ODP, the DoD Architecture Framework (DoDAF) [9], and others. Other methods, such as the Rational Unified Process for System Engineering (RUP-SE) [10] have also adopted the principles in IEEE-1471 and extended them for use within their own domain.

2.1 Reference Model of Open Distributed Processing

In order to provide relevant domain specific guidance on how to select useful viewpoints we need to look to other, more domain specific, approaches that adhere to these general principles. The ISO Reference Model of Open Distributed Processing (RM-ODP) was published in 1996 to provide a useful framework for describing the architecture and design of large scale distributed systems. The RM-ODP, also known as ISO/IEC 10746, provides domain specific guidance that aligns with the principles defined in IEEE 1471-2000. The RM-ODP was developed to provide a useful framework for describing the architecture and design of large scale distributed systems. Among the contributions that RM-ODP provides are the following:

- RM-ODP offers a conceptual framework and an architecture that integrates aspects related to the distribution, interoperation and portability of software systems, in such way that hardware heterogeneity, operating systems, networks, programming languages, databases and management systems are transparent to the user. In this sense, RM-ODP manages complexity through a "separation of concerns", addressing specific problems from different points of view.

- RM-ODP offers a coordinating framework for the standardization of ODP, able to integrate current and future standards, and maintain consistency among them.

- RM-ODP provides a short, clear and explicit specification of concepts and constructs that define semantics, independent of the representation, methodologies, tools and processes used for the development of open distributed applications. RM-ODP offers a vocabulary and a common semantic framework to all the applications' participants (from managers to users, from designers to developers), and encourages the use of formal notations for the definition of those concepts and the specification of the architecture.

A good architectural framework should allow different parts of the design to be worked upon separately if they are independent, but should clearly identify those places where different aspects of the design constrain one another.

RM-ODP has been used in the design of major terrestrial telecom systems (TINA) and other large, multi-user, distributed systems. In the telecommunications industry, TINA (Telecommunications Information Networking Architecture), defined by TINA-C (TINA Consortium) [11], describes an architecture for the development of telecommunication applications based on the concepts defined by RM-ODP. Currently TINA provides the most widespread and accepted architecture in this field. With the support of those technologies, building systems using the RM-ODP concepts is no longer a visionary undertaking. RM-ODP is real.

2.2 Reference Architecture for Space Data Systems

While the terrestrial distributed systems that RM-ODP was designed to describe may be very large and complex space systems are even more so and they don't stand still. RM-ODP provides an excellent framework with which to tackle terrestrial systems, but additional views and viewpoints are essential for describing space systems, largely because there is an entirely new set of concerns. The

Consultative Committee on Space Data Systems (CCSDS) [12] has been working on a space domain adaptation of RM-ODP for the last few years. This is the Reference Architecture for Space Data Systems (RASDS). The RASDS, CCSDS 311x0-R-1, is designed to address these added complexities of space data systems and to focus upon the specification of their architecture.

Flight elements in space systems are always in motion, whether they are in transit to their destination, in orbit around it, landed upon some remote part of the Solar System, roving some distant terrain, or on their way out of it to even more distant domains. There is no avoiding things like (1) solar and lunar eclipses, and (2) occultations that occur when a moving object, such as a planet or the moon, blocks the light coming from a more distant object, such as a star. The physical environment plays a large role because physics acts upon these systems in a way that must be modeled in our design processes and during control system design, planning, commanding and operations. Even such subtle energies as solar radiation pressure, outgassing, and gravity must be analyzed during design and countered during operations.

Because of the long round trip light times (RTLT) command, control and monitoring paradigms must be reconsidered, autonomous systems rise in importance, and any notions of distribution transparency must be completely rethought. Assumptions about immediate, continuous, and interactive communications break down when ground and space communications assets must be scheduled months in advance and command and response round trip times rise from tens of milliseconds to tens of minutes or even hours. Communication protocols designed to work with normal terrestrial communication delays break down as interaction times exceed a few seconds.

The one new viewpoint, which RASDS introduces for space systems, is the Connectivity Viewpoint that deals with system components (Nodes), connectors (Links), the environment within which these systems operate, and the physical interactions among the system elements with the environment. This is a partial sub-set of the RM-ODP Engineering Viewpoint, but it explicitly includes the physical aspects of space data system architectures. RASDS also distinguishes a Communications Viewpoint that is used to address the complexities of communications protocols and end-to-end information system (EEIS) design in space data systems. The other viewpoints in RM-ODP only require minor changes in order to be used for the purpose of describing space systems architectures.

The complexities of operating long lived system in space also requires planned and predictable capabilities "gated" into the system along two axes. First, each element, such as a Crewed Exploration Vehicle, communications network, and Operations Control Center must have a clear evolutionary path that leads to future capabilities. Second, at the starting point, and at some specific point(s) in time in the future, all of the capabilities must work together to produce the desired results.

RASDS provides adaptation to RM-ODP that allow it to describe space data systems, and to deal with issues in the design of these more demanding command, control, and data transfer issues. RASDS is also specifically focused upon a desire to describe architectures, and it intentionally leaves out much of the engineering and technical viewpoint elements in RM-ODP, on the assumption that these additional RM-ODP engineering concepts could be used directly within the RASDS framework where they are required.

3. Adapting RM-ODP and RASDS to Describe Space Systems

The RASDS extensions to RM-ODP provide an excellent framework with which to tackle space data systems, but architecting complete space systems, in general, requires accommodation of other physical attributes and interactions, such as power, propulsion, thermal, and structural. In our effort to extend the RASDS space data system model to encompass other attributes of space systems the one new viewpoint that we introduced is the Physical Viewpoint, which subsumes the RASDS Connectivity Viewpoint. We have also associated the RASDS Connectivity Viewpoint with the

Engineering Viewpoint. Other viewpoints in RASDS and RM-ODP only require minor changes in order to be used for our purposes. In the rest of this paper we will refer to this extended model as the Reference Architecture for Space Data Systems – Extended (RASDS-E).

The concepts in RASDS-E were developed in a Model Based Engineering and Design (MBED) internal research and development task at JPL that was intended to produce a model driven design and engineering process for space systems. Central to this concept is the development of an information model which is rich enough to capture all of the critical elements of space mission design, including requirements, mission goals, observational objectives, activity sequences, technical spacecraft design, development, and operations, space to space and space to ground interactions and communications, and science planning, operations, and processing. RASDS-E is initially focused on the early design phases, but is intended to be of use throughout the mission lifecycle.

3.1 Fundamental Concepts - RASDS-E

As with RM-ODP, a good framework for space system design should allow different parts of the design to be worked on separately if they are independent, but should clearly identify those places where different aspects of the design constrain one another. In order to achieve this, RASDS-E uses several structuring approaches:

-The specification of a complete system in terms of ***viewpoints***.

-The use of a ***common object model*** for the specification of the system from every viewpoint.

-The use of ***views*** to tailor user or domain specific analyses of the system.

-The definition of a ***modeling infrastructure*** that provides support services for system applications, hiding the complexity and problems of defining mission specific models.

-The definition of a set of ***common functions*** that provide general services needed during the design and development of space systems.

-A ***framework*** for the evaluation of conformance of models and designs based on conformance points.

3.2 Viewpoints - RASDS-E

Most space system specifications are so complex and extensive that no single individual can fully comprehend all aspects of the specifications. Furthermore, different stakeholders in the system design have different needs for a given system and different reasons for examining the system's specifications. A mission planner will ask different questions of a system make-up than would a system implementer. The concept of the RASDS-E viewpoints framework is to provide separate viewpoints into the specification of a given space system. These viewpoints each satisfy an audience with interest in a particular set of aspects of the system. Associated with each viewpoint is a viewpoint language that optimizes the vocabulary and presentation for the audience of that viewpoint.

> "A *viewpoint* establishes the conventions by which a view is created, depicted and analyzed. In this way, a view conforms to a viewpoint. The viewpoint determines the languages (including notations, model, or product types) to be used to describe the view, and any associated modeling methods or analysis techniques to be applied to these representations of the view. These languages and techniques are used to yield results relevant to the concerns addressed by the viewpoint. An architectural description (AD) selects one or more viewpoints for use. The selection of viewpoints typically will be based on consideration of the stakeholders to whom the AD is addressed and their concerns." – IEEE 1471-2000

A viewpoint defines a selected set of architectural concepts and structuring rules, in order to focus on particular concerns within a space data system. A viewpoint establishes the purpose and audience for a view and the techniques or methods employed in constructing a view.

The RASDS-E framework extends the RM-ODP and RASDS frameworks to provide six generic and complementary viewpoints on the system and its environment:

- The **enterprise** viewpoint, which focuses on the purpose, scope and policies for the system. It describes the organizational entities, requirements, goals, objectives, scenarios, constraints, and how to meet them.
- The **information** viewpoint, which focuses on the semantics of the information and the information processing performed. It describes the information managed by the space system and the structure, content, semantics, type, and relationships among the data used within the system.
- The **functional** viewpoint, which defines the abstract functional decomposition of the space system into objects, which interact at interfaces. It describes the functionality provided by the space system, the behavior of the functional elements and their functional decomposition.
- The **physical** viewpoint, which defines the physical decomposition of the space system into components, which interact across connectors. It describes the physical aspects of the space system and the external environment within which it operates, the physical behavior (and motion) of the components and their physical decomposition. The connectors may be manifestly physical (nuts and bolts, struts, network or power cables), or they may be more ethereal (RF & optical signals, thermal radiation, gravitational force).
- The **engineering** viewpoint, which focuses on the mechanisms and functions required to engineer and implement the space system and on the allocation of implemented functionality to engineered components of the system, including implementation choices. It describes the distribution of processing performed by the space system to manage the information and provide the functionality.
- The **technology** viewpoint, which focuses on the choice of technology and standards to develop the space system. It describes the standards and technologies chosen to provide the communications, processing, functionality and presentation of information in the space system. It also describes any technology risks that must be assessed during design and development.

A viewpoint is a subdivision of the specification of a complete system, established to bring together those particular pieces of information relevant to some particular area of concern during the design of the system. Although separately specified, the viewpoints are not completely independent; key items in each are identified as related to items in the other viewpoints. However, the viewpoints are sufficiently independent to simplify reasoning about the complete specification.

The mutual consistency among the viewpoints is ensured by the viewpoints and relationships defined by RASDS-E, and the use of a common object model provides the glue that binds them all together. The RASDS-E architecture is based upon a common semantic object model such that each object class and set of relationships is represented once and only once in the model. As a note to the implementer, the intent is that any change to the object model from one view will result in a change in the underlying semantic model, therefore automatically changing all related views. The issue to be addressed is that searching for similar models or elements that may be affected by any change in these complex models can no longer be tolerated. The modeling tools must manage this for us. Figure 1 shows the relationships among the top-level objects in the RASDS-E model.

3.3 Common Object Model - RASDS-E

The RASDS-E viewpoint specifications are expressed in terms of objects. An object is an abstract representation of an entity in the real world. It contains information and offers services. A system is composed of interacting objects. Each viewpoint defines its own objects and their relationships and interactions. In the enterprise viewpoint the objects are organizations and the interactions involve requirements, contracts, and policies. In the functional viewpoint the objects are abstract functions and they interface via abstract interfaces. In the physical viewpoint the objects are components with mass and structural properties that are related to one another by some sort of physical connector.

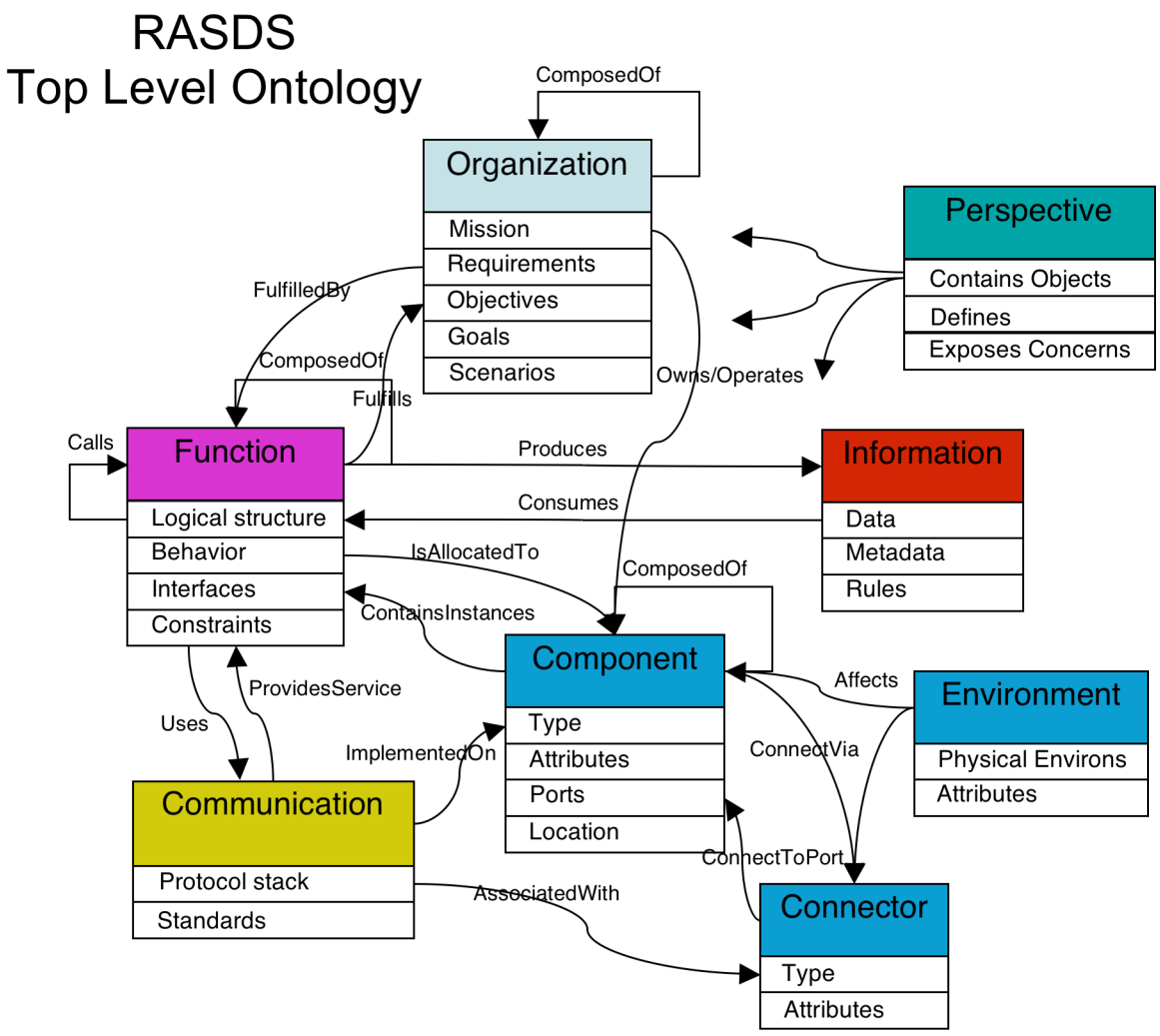

Figure 1 – RASDS Top level Objects

The use of the object paradigm provides abstraction and encapsulation, two important properties for the specification and design of complex systems. Abstraction allows highlighting those aspects of the system relevant from a given perspective, while hiding those of no relevance at that moment. Encapsulation is the property by which the internal implementation details or information contained in an object is accessible only through interactions at the interfaces supported by the object. Because objects are encapsulated, there are no hidden side effects of interactions. It also implies that the internal details of an object are hidden from other objects, which is crucial for dealing with heterogeneity, multiple implementations, interoperability and portability.

3.4 Views - RASDS-E

Viewpoints provide the conventions, rules, and languages for constructing views. A view is a representation of a whole system from the perspective of a related set of concerns. Views are themselves modular and well formed, and each view is intended to correspond to exactly one viewpoint. In some cases objects defined in one viewpoint will have a correspondence with related objects defined in another viewpoint. The user may also define a new view based on the basic concepts defined by RASDS-E if it is impossible to capture all the important aspects of the system with the six viewpoints defined here. Some aspects of a system design may benefit from being examined from two or more views simultaneously.

> "A view may consist of one or more architectural models. Each such architectural model is developed using the methods established by its associated architectural viewpoint. An architectural model may participate in more than one view. NOTE—In a complex system, Architecture Descriptions (AD) may be developed for components of the system, as well as for the system as a whole. In this case, it may be that one AD will have a view corresponding to a particular viewpoint and another AD will have a view corresponding to the same viewpoint. Although the system being described by these two views has the whole-part relationship, this is not an instance of multiple views corresponding to one viewpoint. The ADs are considered separate even though they are related by the systems they describe." IEEE-1471-2000.

What follows is a set of viewpoints that may be used to describe space systems, along with a nominal and incomplete set of views associated with each viewpoint. Not all of these views may be useful for any specific project and other views may be defined as necessary. For each viewpoint the primary stakeholders, concerns, modeling language, and consistency methods are identified. Note that for some analyses elements from multiple viewpoints may be combined into a new view, possibly using a layered representation.

As part of the modeling language each viewpoint specification identifies a set of objects and the relationships that may be expressed among them. And as noted earlier, objects defined in one viewpoint will often have a correspondence to related objects in another viewpoint specification. In fact, tracing these correspondences and relationships is one of the key means by which design consistency and integrity is established. Requirements and scenarios trace to functions, functions trace to the engineering objects that implement them (hardware or software), the physical attributes of hardware engineering objects are considered as part of the structural and thermal analysis of the designed system.

Enterprise viewpoint

Stakeholders: funding source, acquirers, users, and developers
Concerns: what the system is to do and how we organize to design, develop and operate it
Modeling Language: enterprise objects and relationships, roles, policies, constraints, scenarios, requirements
Consistency & Completeness Methods: documented set of completeness rules

> **Organization** view – Includes organizational elements and their roles, structures and relationships. May include agreements, contracts, policies and organizational interactions.
> **Requirements** view – Describes the requirements, goals, and objectives that drive the system. Says what the system must be able to do.
> **Scenario** view – Describes how the system is intended to be used. Includes user views and descriptions of how the system is expected to behave.

Information viewpoint

Stakeholders: users, developers, maintainers, and data system operators
Concerns: structure, semantics, rules, and policies on data
Modeling Language: information objects and relationships, constraints, rules on access and retention

Consistency & Completeness Methods: documented set of completeness rules, every major information object identified in other views is documented here

Metamodel view – An abstract view that defines information elements and their structures and relationships. Defines the classes of data that are created and managed by the system and the data architecture.
Information view – Describes the actual data and information as it is realized and manipulated within the system. Data elements are defined by the metamodel view and functional objects in other views refer to them. Includes policies on access and retention of data where appropriate.

Functional viewpoint

Stakeholders: system engineers, acquirers, developers, users, and maintainers
Concerns: the functions that are required for the system to meet its requirements and execute its scenarios
Modeling Language: functional objects and relationships, interfaces, behaviors, constraints
Consistency & Completeness Methods: every requirement maps to at least one function, no requirement is not mapped to a function, no function is not mapped to a requirement, and there is structural data and control flow consistency

Functional Dataflow view – An abstract view that describes the functional elements in the system, their interactions, behavior, provided services, constraints and data flows among them. Defines which functions the system is capable of performing, regardless of how these functions are actually implemented.
Functional Control view – Describes the control flows and interactions among functional elements within the system. Includes overall system control interactions, interactions between control elements and sensor / effector elements and management interactions.

Engineering viewpoint

Stakeholders: system engineers, sub-system engineers, developers, operators, users, maintainers, and acquirers
Concerns: how the functions that the system must possess are to be engineered, construction and assembly approaches, performance envelopes, suitability, implementability, testability, riskiness, operability
Modeling Language: engineering objects (hardware and software) their connections and relationships, constraints
Consistency & Completeness Methods: every functional element maps to at least one engineering element, no functional element is not mapped, no engineering element is not mapped to a function, system performance is estimated and verified against requirements and scenarios, the assembled system is validated

Allocation view – Describes the allocation of functional objects to engineered physical and computational components within the system, permits analysis of performance and used to verify satisfaction of requirements
Software view - Describes the software engineering aspects of the system, software design and implementation of functionality within software components, select languages and libraries to be used, define APIs, do the engineering of abstract functional objects into tangible software elements. Some functional elements, described using a software language, may actually be implemented as hardware (FPGA, ASIC)
Hardware views – Describes the hardware engineering aspects of the system, hardware design, selection and implementation of all of the physical components to be assembled into the system. There may be many of these views, each specific to a different engineering discipline.
Communications Protocol view – Describes the end-to-end design of the communications protocols and related data transport and data management services, shows the protocol stacks as they are implemented on each of the physical components of the system.
Risk view – Describes the risks associated with the system design, processes, and technologies, assigns additional risk assessment attributes to other elements described in the architecture
Control Engineering view - Analyzes system from the perspective of its controllability, allocation of elements into system under control and control system
Integration and Test view – Looks at the system from the perspective of what must be done to assemble, integrate and test system and sub-systems, and assemblies. Includes verification of proper functionality, driven by scenarios, in satisfaction of requirements.
IV&V view – independent validation and verification of functionality and proper operation of the system in satisfaction of requirements. Does system as designed and developed meet goals and objectives.

Physical viewpoint

Stakeholders: system engineers, sub-system engineers, acquirers, developers, operators, users, and maintainers

Concerns: the physical structures of the system, their connections, and how they interact with the environment
Modeling Language: physical objects (components) and their connections, physical behavior and interactions, the environment, constraints
Consistency & Completeness Methods: every functional element maps to at least one physical element, no functional element is not mapped, no physical element is not mapped to a function, and there is structural integrity and consistency

Data System view – Describes instruments, computers, and data storage components, their data system attributes and the communications connectors (busses, networks, point to point links) that are used in the system.
Telecomm view – Describes the telecomm components (antenna, transceiver), their attributes and their connectors (RF or optical links).
Navigation view – Describes the motion of the major elements of the system (trajectory, path, orbit), including their interaction with external elements and forces that are outside of the control of the system, but that must be modeled with it to understand system behavior (planets, asteroids, solar pressure, gravity)
Structural view – Describes the structural components in the system (s/c bus, struts, panels, articulation), their physical attributes and connectors, along with the relevant structural aspects of other components (mass, stiffness, attachment)
Thermal view – Describes the active and passive thermal components in the system (radiators, coolers, vents) and their connectors (physical and free space radiation) and attributes, along with the thermal properties of other components (i.e. instruments as thermal sources (or sinks), antennas or solar panels as sun shade)
Power view – Describes the active and passive power components in the system (solar panels, batteries, RTGs) within the system and their connectors, along with the power properties of other components (data system and propulsion elements as power sinks and structural panels as grounding plane)
Propulsion view – Describes the active and passive propulsion components in the system (thrusters, gyros, motors, wheels) within the system and their connectors, along with the propulsive properties of other components

Technology viewpoint

Stakeholders: system engineers, sub-system engineers, developers, acquirers, and maintainers
Concerns: the technologies chosen to implement the system and their suitability for the intended purpose, their level of development and risk
Modeling Language: tables of technology items, maturity levels, risk assessments, trade assessments
Consistency & Completeness Methods: documented set of completeness and consistency rules

Standards view – Defines the standards to be adopted during design of the system (e.g. communication protocols, radiation tolerance, soldering). These are essentially constraints on the design and implementation processes.
Infrastructure view – Defines the infrastructure elements that are to support the engineering, design, and fabrication process. May include data system elements (design repositories, frameworks, tools, networks) and hardware elements (chip fabrication, thermal vacuum facility, machine shop, RF testing lab)
Technology Development & Assessment view – Includes description of technology development programs designed to produce algorithms or components that may be included in a system development project. Includes evaluation of properties of selected hardware and software components to determine if they are at a sufficient state of maturity to be adopted for the mission being designed.

One distinction in RASDS, as compared to RM-ODP, is that objects are considered to have a "primary" or "home" viewpoint where they are fully specified and their fundamental characteristics are defined. Representations of these objects may appear in other views. As an example, the full definition of information objects will appear in an information view, but representations may appear in enterprise, functional, or other views.

4.0 MBED Mission Modeling Experiment

This conceptual approach to space system modeling has been partially evaluated in our FY05 MBED IR&D task. This task was intended to demonstrate:

(1) Concurrent engineering of the spacecraft and the science instruments, and
(2) Concurrent engineering of the spacecraft and science instruments at one level of abstraction, and the Subsystem performance models at a lower level of abstraction.

In this task we used an existing system engineering modeling tool, CORE [13], and two different sets of existing performance simulation models. CORE provides partial support for the full set of required viewpoints, but does not directly define any of the viewpoints or specifications that we have discussed. However, it does provide support for system engineering modeling and some amount of behavioral analysis, and has direct support for a set of views, available via frameworks in their modeling environment, that map to several of the key viewpoints that we have identified:

- Element, Relationship, Attribute View (ERA)
  - Information Viewpoint
  - Object and relationship definitions for other viewpoints
- Hierarchy View
  - Enterprise Viewpoint, requirements and organizational
  - Functional and Engineering Viewpoint, element hierarchy
- Functional Flow Block Diagram (FFBD) & Enhanced FFBD (EFFBD)
  - Enterprise Viewpoint, scenarios
  - Functional Viewpoint, data and control flows
- IDEF0 Diagram
  - Functional Viewpoint, functional interactions
  - Enterprise Viewpoint, organizational interactions
- Functional Interface (N2) Diagram
  - Functional Viewpoint, object interactions
- Physical Block Diagram
  - Physical Viewpoint, high level component and link interconnections
- Text View
  - No mapping to RASDS-E

This is not the complete set of viewpoints and views that we wish to have available, nor are the available views exactly what is required in the long run, but the tool provided a very useful and adaptable platform for our experiment. This tool allowed us to capture example spacecraft requirements, the physical and functional architecture, and the high level behavior of the described elements in a machinable way. This has allowed us to simulate at least the coarse grained overall behavior of this system based upon its description, to provide means to assess some elements of end to end performance, and to explore design trades.

While the underlying CORE information model is proprietary, it can be exported in an XML schema [14]. This XML model has a regular and easily understood structure, which makes it amenable to analysis, dissection, and transformation. A part of our MBED task was to use this system requirements and architecture model to drive other existing analysis tools that performed science feasibility analysis and telecom link and power performance analyses. These existing tools were modified to accept newly defined XML schema that incorporated the structuring concepts in our RASDS-E model, and their inputs were derived directly from the exported CORE XML model. The flow of information among these models and tools is shown in Figure 2.

This experiment was useful in that it allowed us to integrate several existing modeling and simulation packages at the data level, thus providing an improved flow of information and assisting in the system engineering evaluations of a modeled spacecraft. However, it also exposed some of the limitations in existing tools and their inability to support extensions to include the required conceptual viewpoints. The approach that was chosen to do these model transformations was

somewhat fragile because it was tied to the specific structures identified in the various XML schema and required a lot of analysis to construct semantically meaningful transformations. Future work will explore how far we can go to improve capture transformation, and analysis of these formalized design descriptions and evaluation of completeness and correctness based upon these system models.

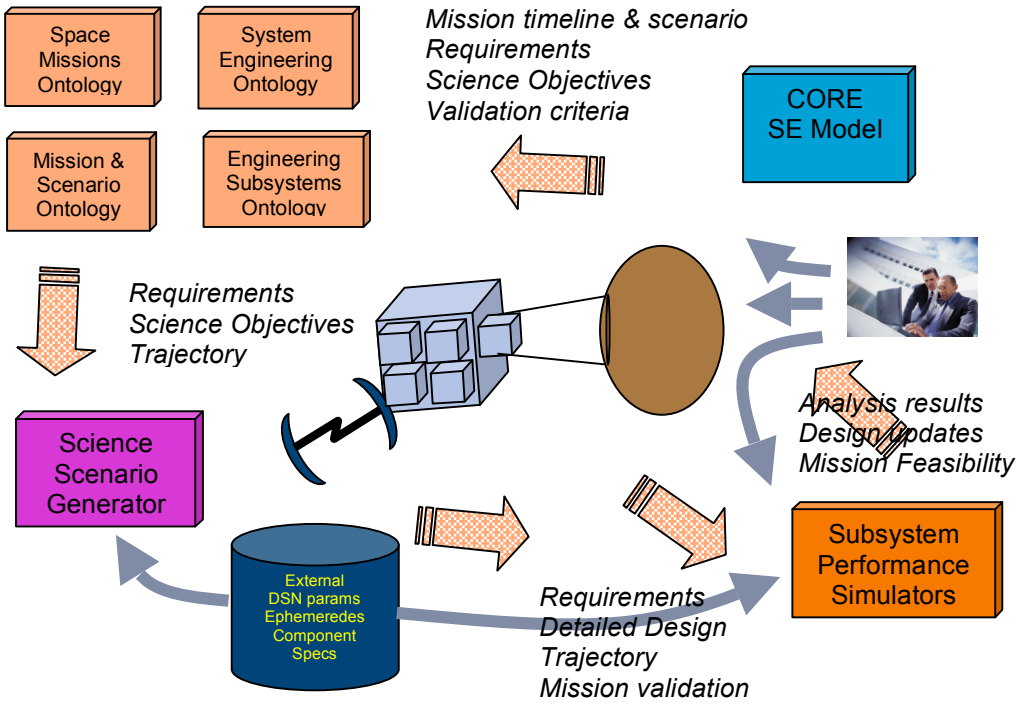

Figure 2 MBED Model and Information Flows

5. Discussion of Methodology and Tool Limitations

What has complicated these system modeling efforts are the limitations in existing methodologies and tools in terms of their suitability for description of the architecture and design of space missions. Of the existing methods the one that appeared most suitable for the initial space data system architecture descriptions was RM-ODP. It's adaptation into the RASDS reference architecture has proven very useful and it is being used for some of the newest CCSDS documents. Even though RASDS is still a draft specification it has been used successfully for several NASA mission system architecture studies. In one case the mission design team had intended to use DoDAF, but found that it had limitations when attempting to describe space mission technical architecture. The RASDS Connectivity and Communications views were used to great effect to augment the DoDAF OV and SV views.

As is typical, most of these mission design efforts have been document driven approaches, using common word processing and presentation software packages. While these are typical of the practice, they have significant limitations in that any models or views that are depicted are essentially just drawings with implied, but not explicit semantic content. The practice, at least for DoDAF based architectural designs and for UML based software architecture efforts, has been changing and a number of tools are available that support UML and DoDAF diagrams. In fact, several UML tool vendors now offer extensions to their development environments that support DoDAF views within the UML environment, leveraging UML methods where they are applicable.

These design environments are capable of capturing and preserving a lot of semantic meaning and of exporting these models via an XML export using the XMI schema [15]. However, full tool interoperability remains elusive, as there is as yet no effective means of exchanging model syntax, semantics, and drawings among these tools.

5.1 SysML, UML, and Related Approaches

The SysML methodology extends UML 2.0 by adding requirements, verification, and parametrics to the UML suite of diagram types. The SysML Specification also supports modeling semantics for continuous behavior as well as discrete. This provides good support in a general way for many front-end system engineering and architecting processes. SysML also has incorporated support for the sorts of viewpoint and view constructs that are required to support the kind of domain specific reference models that we propose in this paper. These definitions are, by design, fully compliant with IEEE 1471-2000. The authors worked with the SysML Submission Team to ensure that these concepts were adequately supported.

The SysML 1.0 specification is still being finalized between the OMG and INCOSE, and we can expect to shortly have commercial tools that support it (already demonstrated by 4 vendors for the 0.9 version of the specification). However, what is needed next is a space system, domain specific, profile that will extend any tools supporting SysML to include explicit support for the sorts of viewpoint specifications and views that have been identified here. As it stands, most of the diagram types that have been defined in SysML are useful, but they are far too general, allow too many degrees of freedom, and provide too little guidance.

We believe that what is needed is a SE and architecture design environment that presents a suitable set of conceptual viewpoints, frameworks, and templates to guide the practitioners. In a sense that is what CORE does and we believe that it one of the aspects of this tool that users have found most valuable. But CORE lacks many of the viewpoints and view languages that are required to model space systems, and a more general approach, leveraging industry standards like SysML, seems like the most fruitful path to explore.

5.2 Relationship to Other Domain Methods

The approach that we have described defines a set of viewpoints, objects, and relationships that will be familiar to most practitioners in the space systems system engineering and architecture community. It is a very rich set of concepts that attempts to support description of all of the core objects, linkages, and relationships that are needed during the early design and development process. At the same time, it is clear that for some of these viewpoints there is already a diverse set of languages and modeling approaches in use. This is particularly the case for the engineering and physical viewpoints.

For example, in the software architecture domain a frequently used analysis approach is the Krutchen 4+1 view model [16]. These views are logical, process, physical, development and use case.

- The logical view shows how the system is decomposed into a set of behavioral abstractions and it may use class, collaboration, or sequence diagrams. This is essentially identical with what we have called the functional viewpoint.
- The process view lets you describe the systems processes, as implemented, and how they communicate. Activity diagrams are often used in this view. This view aligns with part of the engineering viewpoint, in the software view.
- The development view describes the modules in the system and how that are organized and associated into packages and classes. This is also a part of the software view in the engineering viewpoint.

- The physical view describes how the implemented software application is installed and operates in one or more computers. It may use a UML deployment diagram showing nodes that may contain one or more components. This is directly analogous to the allocation view in the engineering viewpoint.
- The use case view includes scenarios and it is used to describe the required functionality of the system. It may employ use cases, activity diagrams, or descriptions of required actions of the system. This is aligned with the enterprise viewpoint scenario view.

While our method provides a quite clean mapping for the basic concepts described in the Krutchen 4+1 view approach it does not provide much guidance for the software development process itself. The field of software development is a rich one and it brings a wealth of new methods such as agile programming, design patterns, and other methods for risk reduction. What we have provided here is a framework for describing the overall architecture of these systems that accommodates these other design processes and artifacts and relates them to the other elements in the system in a much more complete way than these other, more domains focused methods, are able to.

The Rational Unified Process (RUP-SE) and the related Model Driven System Design (MDSD) [17] process also uses the Krutchen 4+1 model, but adds two additional viewpoints, one for workers and operational interactions, and one for geometric assemblages. To place this into a lifecycle context RUP-SE also introduces the concepts of model levels, which they describe as context, analysis, design, and implementation. These "levels" largely relate to lifecycle phases. Note that RUP-SE has also adopted the IEEE-1471 language for describing viewpoints and acknowledges leveraging RM-ODP. The RASDS-E does not explicitly model the operational interactions captured in RUP-SE nor many of the related Operational Viewpoint views modeled in the DoDAF. Where these are required the available UML mappings can be used in a way that would integrate with the rest of the future modeling framework.

5.3 Physical Viewpoint Modeling

In the physical viewpoint it could easily be argued that there need to be separate viewpoints for each of structural, thermal, propulsion, power, telecomm because each of these items belong to different subsystems, have different properties, and are frequently analyzed by different means. All of this is true, but it is also true that a spacecraft is a set of physical components that are assembled using various kinds of connectors and that these components have a variety of different properties or attributes. Some components are purely structural, some are computational, some produce data, some are thermal, some propulsive, and many consume or provide power. But the critical point is that almost every one of these components has a variety of properties and appears in more than one view.

For example, a panel that is part of the spacecraft bus and appears in a structural view may also be a part of the ground plane from the power view and will also appear in the thermal view because it connects heat producing components that are inside the spacecraft and shields them from solar radiation. Similarly all of the instruments and computational elements have physical mass, but they also draw power, produce heat, and create, manage, or transport data. While it is really useful for be able to look at the system from just one viewpoint for analytical purposes, these viewpoints are not truly independent since it may be one element with multiple attributes and changes of state in the power domain will directly affect what happens in the thermal domain. The approach that we have taken in modeling these systems is to explicitly acknowledge these relationships and to attempt to capture them in the core information model.

Recent developments in physical system modeling appear to bear out the value of this approach. Moore et al, in their paper on multi-disciplinary computer aided analysis of thermal, structural and optical performance [18] have identified a need for an integrated approach to modeling physical systems that need very high accuracy results. This is driven by the strongly coupled nature of these

classes of problems combined with unprecedented levels of required optical precision for certain advanced space missions like SIM. Their development efforts have defined a new finite element-based analytical capability, which utilizes NASTRAN syntax to describe common-model multidisciplinary analysis tasks. Capabilities currently under development will capture behavioral aspects of coupled nonlinear radiative heat transfer, structures, and optics problems to a level of accuracy and performance not yet achieved elsewhere. Discussions are underway to understand how their detailed models of structural attributes of system elements can be related to the components, connectors, and behavioral specifications associated with appropriate views in the physical viewpoint.

6. Summary

We have described a conceptual framework for modeling space system architectures that is suitable for capturing all of the elements of a spacecraft, both hardware and software, and of relating them to each other, to the driving requirements and scenarios, and to the environment within which they are to operate. This adaptation of RM-ODP and RASDS viewpoints and concepts to describe space systems appears to offer us some significant advantages:

-First, RASDS-E may help us in thinking about these systems from different perspectives (or viewpoints), greatly improving the requirement collection and analysis phases of the development of applications, and providing a set of well proven concepts for analyzing space systems design and behavior.

-Second, RASDS-E offers a conceptual infrastructure and a common reference model within which different views, expressed in separate languages (those from the viewpoints), can be consistently integrated.

-Third, RASDS-E provides a set of already established reasoning patterns to help specify and design space systems. Those patterns assist us to identify the fundamental entities of the system and the relationships among them. In this sense, RASDS-E encourages us to ask the right questions of the right people, and with the appropriate degrees of abstraction and precision for building useful system specifications.

-Finally RASDS-E provides space system architects, designers and developers with a set of mechanisms and common services to facilitate their jobs, and permits models of these complex systems to be developed.

The intent of this architecture work is not to define one all encompassing model that can describe the Universe, although it may appear that way. Rather the intent is to provide a conceptual framework within which the basic architectural building blocks and elements that must be considered in building space systems can be discussed. This core model is intended to provide the links and relationships among these other views and to help manage the process of information exchange. The proposed modeling framework, based upon an extended SysML / UML model, has the native capability to describe much of the system and software architecture, but it does not have the language for describing physical structures and many of the other physical attributes of the system components that must be analyzed.

However, as was indicated in the discussion on the physical viewpoint, integrated languages for describing physical structures, their thermal properties and optical behavior are emerging now. What is intended in this work is to provide a common architectural framework that can capture these relationships and manage the linkages to these other views with their specialized languages. Given this approach we expect that the system engineers, during various trade study phases, will have the degrees of freedom to explore alternative configurations and assemblages of components. But when

various types of performance analysis must be done the underlying model views, which carry their own domain specific semantics, will enable rapid assessment of structural or thermal, performance.

While this methodology may be directly used in a variety of document driven ways to describe space system architecture, the real power of it will come when there are tools available that will support full description of system architectures that can be captured electronically in a way that permits their analysis, verification, and transformation. The best hope for this appears to be to develop a suitable profile for SysML or UML tools that will stand as a domain specific meta-model. This approach, coupled with extensible libraries of components, has promise of providing a useful framework within which system designs practitioners can operate conveniently.

Acknowledgements

The research described in this paper was carried out at Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute nor imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

The RM-ODP introductory section of this paper is an adaptation (with permission) of an introductory paper on RM-ODP written by Antonio Vallecillo of the Universidad de Málaga av@lcc.uma.es [19]. The terminology used in this paper is drawn from the ISO/IEC 10746 documents defining RM-ODP and also from IEEE-1471-2000, Recommended Practice for Architectural Description of Software-Intensive Systems. Most of the concepts for extending RM-ODP to describe space data systems were developed within a System Architecture Working Group in CCSDS, the Consultative Committee on Space Data Systems, chaired by Dr Takahiro Yamada, during the creation of the Reference Architecture for Space Data Systems (RASDS), CCSDS 311X0-R-1. The work to extend these concepts in order to describe the full architectural framework for space systems was developed in a JPL funded internal Research and Technology project called Model Based Engineering and Design (MBED), whose PI was Steven Wall.

Biography

Peter Shames has been engaged in the process of turning computers into useful tools for scientists for the bulk of his professional career. He manages JPL's Data Systems Standards Program in the Interplanetary Network Directorate (IND). He is the Director of the Consultative Committee for Space Data System's System Engineering Area and he is working within CCSDS to define RASDS, an end-to-end reference architecture and formal methodology for describing space data systems. He has developed architectures for a variety of NASA programs, including JPL's mission operations system, the Hubble Space Telescope science processing and archiving systems, and real time data acquisition. He has served on working groups in the National Academy of Sciences and the Internet Activities Board. Once upon a time he used to know how to program.

Dr. Joseph Skipper is engaged in two roles at JPL. First, he is a practicing systems architecture / C3I (Command, Control, Communications and Information) on the Constellation program and producing program deliverables such as the Systems Engineering Management Plan. Second, he is engaged in the MBED initiative actively constructing various research models and using them to integrate information between science instruments and spacecraft design, and between spacecraft design and subsystem models at lower levels of abstraction. He has a background in object technologies, dynamic simulation, and systems engineering tools. He has been participating in the Systems Modeling Language (SysML) standards initiative.

References

[1] *Unified Modeling Language (UML) V2.0 Superstructure*, Final Adopted Specification, OMG formal/05-07-04, http://www.omg.org

[2] *Systems Modeling Language (SysML) Specification (draft) v 0.9*, SysML Partners SysML-v0.9-PDF-050103.pdf, http://www.omg.org.

[3] *Reference Model of Open Distributed Processing (RM-ODP)*, ISO/IEC 10746-1 to 10746-4, ITU-T Specifications X.901 to x.904, 1998, http://www.rm-odp.net

[4] *International Standards Organization, Information Technology - Open Distributed Processing – Use of UML for ODP System Specification*, Committee Draft, ISO/IEC 19793 – 2005-01-07, ISO/IEC JTC1/SC7/WG19, http://www.rm-odp.net

[5] *Recommended Practice for Architectural Description of Software Intensive Systems*, ANSI/IEEE P1471-2000.

[6] *Reference Architecture for Space Data Systems*, CCSDS 311.0x-R-1, WG Draft, Jan 2006

[7] S. D. Wall, *Model-Based Engineering Design for Space Missions*. 2004 IEEE Aerospace Conference, Big Sky, WY, March, 2004.

[8] M. Maier, D. Emery, R. Hillard, *ANSI/IEEE 1471 and Systems Engineering*, Wiley, Systems Engineering, Vol. 7, No. 3, 2004

[9] *DoD Architecture Framework (DoDAF), Version 1.0,* Deskbook, Vol I, Vol II, US Department of Defense, http://www.aitcnet.org/dodfw/

[10] *Rational Unified Process for System Engineering, RUP-SE 1.1*, Rational, TP164A, May 2002

[11] *Overall Concepts and Principles of TINA*, Telecommunications Information Networking Consortium (TINA-C) 1994

[12] Consultative Committee on Space Data Systems (CCSDS), http://www.ccsds.org

[13] *CORE Architecture Definition Guide, Rel 5.1*, Vitech Corp, http://www.vitechcorp.com, April 2005

[14] *XML Schema, parts 0, 1, 2*, W3C Recommendation, Oct 2004, http://www.w3.org/TR/xmlschema-0/

[15] XML Metadata Interchange (XMI), v 2.1, OMG, Sept 2005, http://www.omg.org/cgi-bin/doc?formal/2005-09-01

[16] Philippe B. Kruchten, *The 4+1 View Model of Architecture,* IEEE Software, vol. 12, no. 6, November 1995, pp. 42-50.

[17] M. Cantor, G. Roose, *Hardware/software Codevelopment Using a Model-Driven Systems Development (MDSD) Approach*, IBM, Dec 2005, http:/www.ibm.com

[18] G. Moore, M. Chainyk, J. Schiermeier, *Multidisciplinary Analysis for Large-scale Optical Design,* Proceedings of SPIE Proceedings of SPIE -- Volume 5528, September 2004, pp. 108-117

[19] A. Vallecillo, *RM-ODP: The ISO Reference Model for Open Distributed Processing,* 2000, www.lcc.uma.es/~av/Publicaciones/00/odpeng.pdf