Architecture of a Distributed Multimission Operations System

Takahiro Yamada

The Institute of Space and Astronautical Science (ISAS)
3-1-1 Yoshinodai
Sagamihara 229, Japan

ABSTRACT

This paper presents an architecture to develop a multimission operations system, which we call **DIOSA**. In this architecture, a component used as a building block is called a functional block. Each functional block has a standard structure, and the interface between functional blocks are defined with a set of standard protocols. This paper shows the structure of the database used by functional blocks, the structure of interfaces between functional blocks, and the structure of system management. Finally, examples of typical functional bloks and an example of a system constructed with this architecture is shown.

Key Words: System architecture, Mission operations, spacecraft control

1. INTRODUCTION

In order to reduce the cost of developing an operations system for a spacecraft, an approch of developing a system by integrating reusable components has been proposed (Holder et al., 1992; Mandl et al., 1992). In order for such an approach to be successful, the function of the components must be defined in a structured model of the entire system, and the interfaces between components must be standardized.

This paper presents an architecture to develop a multilission operations system, which we call **DIOSA** (DIstributed Operations System Architecture). In this architecture, a component used as a building block is called a functional block. Each functional block has a standard structure, and the interfaces between functional blocks are defined with a set of standard protocols.

If the functions provided by a functional block can be customized by only changing parameters, the functional block can be utilized by many missions. The key to do this is the standardization of database. This paper shows an example of a structure of standard spacecraft database. To make a distributed system reliable, the interfaces between components must be simple and understandable. This paper presents a simple interface structure with which functional blocks can communicate with each other easily. Automating management activities is the key to reduce operational labor (Newsome et al., 1992). This paper proposes a scheme for managing a distributed operations system.

Finally, examples of typical functional blocks and an example of a system constructed with this architecture is shown.

2. SYSTEM ARCHITECTURE

2.1 Overall Architecture

To enable the definition of system components and interfaces of a distributed system, the architecture of the entire system needs to be defined. In this subsection, the definition of the concepts of system, complex, domain and functional block is given.

A spacecraft operations system consists of some complexes (Fig.1). A complex is an aggregated set of operations facilities located at one location. Typical examples of a complex are (1) a ground station, (2) a

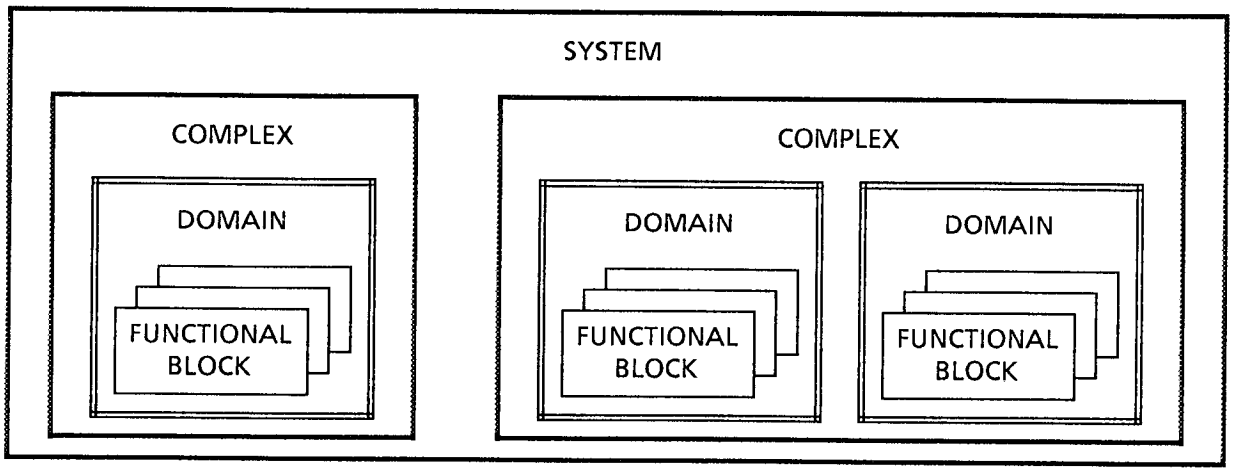

Fig. 1 DIOSA Overall Architecture

spacecraft operations center, and (3) a science analysis center.

A complex is divided into domains. A domain is a set of operations facilities managed as a single system. The system configuration of a domain is controlled independently from that of other domains. In other words, a configuration change of a domain does not affect the configuration of other domains. Typical examples of a domain are (1) a set of equipments for one antenna at a ground station, (2) a spacecraft control system for one spacecraft at a spacecraft operations center, and (3) a payload operations system for one payload at a science analysis center.

The distinction between complex and domain may not be important in some cases. For example, if an entire complex is managed as a single system, the notion of domain is useless. In what follows, however, we assume that the domain is the unit for management.

A domain consists of functional blocks. Functional blocks are basic building blocks of DIOSA. Each functional block performs a set of functions to operate a spacecraft, and has a standard structure which will be defined in the next subsection.

2.2 Structure of a Functional Block

Each functional block has (1) data ports, (2) a management port, and (3) a local SIB (Fig. 2).

The data ports are used for receiving data to be processed by the functional block and for sending data which has been processed by the functional block. The communications protocols to be used for the data ports are discussed in Section 4. The other end of a data port is another functional block.

The management port is used for receiving configuration control information and for sending status information (Newsome et al., 1992). The protocols to be used for the management port are discussed in Section 5. The other end of the management port is usually the Domain Management Functional Block.

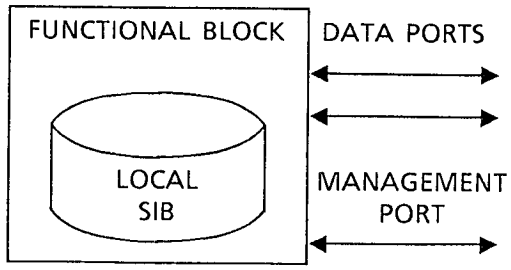

Fig. 2 Structure of Functional Block

The local SIB is a local copy of a subset of the Spacecraft Information Base (SIB) which will be discussed in Section 3. Each functional block retains its own copy of SIB, which is downloaded from the Master SIB of a spacecraft. Most of the basic functional blocks can be used for several spacecraft by changing the local SIB and (maybe) replacing some of the software modules.

3. SPACECRAFT INFORMATION BASE

3.1 Structure of SIB

The Spacecraft Information Base (SIB) is a database which stores all the information needed to operate a spacecraft. SIB consists of three parts, namely Data Definition Part, Behavior Definition Part, and Procedure Definition Part (Fig. 3). The Data Definition Part is equivalent to a traditional command and telemetry database found in most spacecraft operations systems. The Behavior Definition Part and Procedure Definition Part is an online version of the flight operations manual (Cipollone et al., 1992). In the future, higher-level knowledge on spacecraft should be further combined with SIB using a technique proposed by Kaufeler et al. (1992a).

Each part of SIB is generated from the spacecraft specifications and the flight operations manual. It is important that SIB should be used from the spacecraft integration and test phases by the spacecraft integration team and then handed to the flight operations team in an electronic manner.

In the operations system, SIB is maintained in the Master SIB Functional Block, and its appropriate portions are occasionally distributed to other functional blocks (Fig. 4).

3.2 Data Definition Part (SIB Part 1)

The Data Definition Part of SIB defines the structure of binary data contained in telemetry and command packets (or frames). For each command item, any information needed to generate a command packet from its mnemonic expression is stored (Fig. 5). And for each telemetry item, any information needed to extract its value or status from a telemetry packet is stored (Fig. 5). The parameters of the RF links of the spacecraft are also stored in this part.

A standard of packet formats like ESA's Packet Utilization Standard (Kaufeler et al., 1992b) greatly facilitates standardizing this part of SIB, thus increasing portability of SIB from mission to mission.

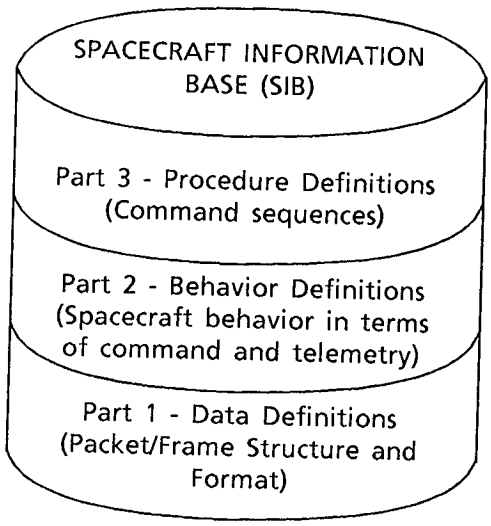

Fig. 3 Spacecraft Information Base (SIB)

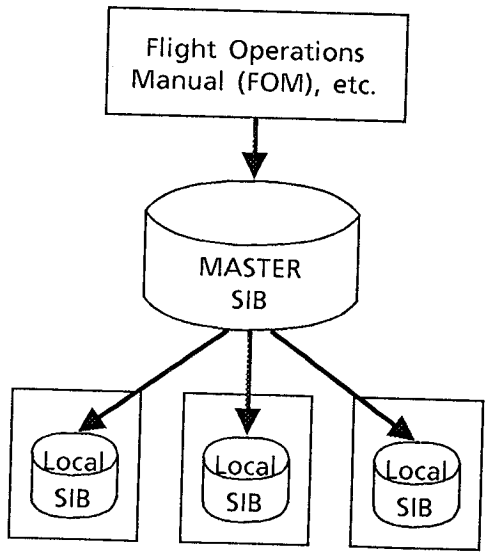

Fig. 4 Distribution of SIB

3.3 Behavior Definition Part (SIB Part 2)

This part defines the behavior of a spacecraft in terms of command and telemetry. This data is used to decide whether or not the spacecraft is acting normally, and to give the operator a message on what actions to be taken in case of an anomaly (Fig. 5).

Examples of information stored in this part are: (1) How the spacecraft reacts to each command in terms of telemetry, (2) Actions to be taken (or commands to be sent) when the spacecraft does not reacts to the transmitted command properly, (3) Telemetry limit values, (4) Actions to be taken (or commands to be sent) if a telemetry limit value is exceeded.

3.4 Procedure Definition Part (SIB Part 3)

This part stores command procedures. A command procedure is a sequence of commands to accomplish an objective. Other command sequences can be called as subprocedures in a command sequence. Information on resource requirements and operational constraints should be stored with each command sequence.

4. INTERFACE STRUCTURE

Standard interfaces between functional blocks are obtained by standardizing data formats and communications protocols. In this section, standard formats and protocols to be used for the data ports of functional blocks are presented.

4.1 Data Types

Data formats to be used for the data ports of functional blocks are standardized in two categories, namely Raw Data Type and Text Data Type (Fig. 6).

Examples of data of the Raw Data Type are command and telemetry packets (or frames) and radiometric data. To increase the portability of some software, any raw data used for spacecraft operations should be formatted in a data unit whose structure resembles that of CCSDS packets. In this way, for example, navigation data generated by a spacecraft and doppler data obtained at a ground station can be displayed on the same screen easily.

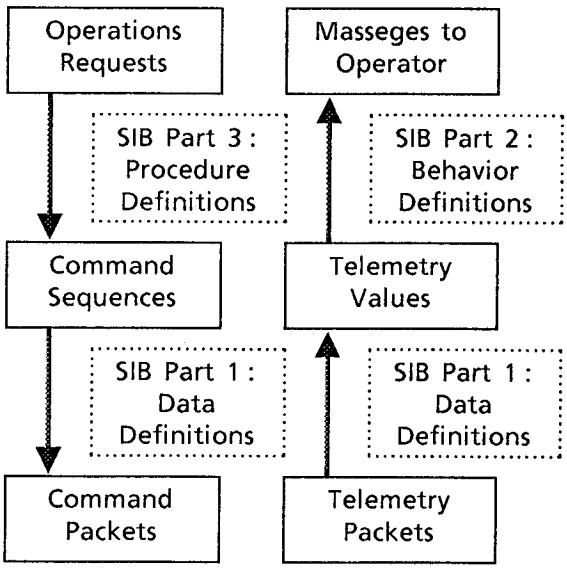

Fig. 5 Usage of SIB

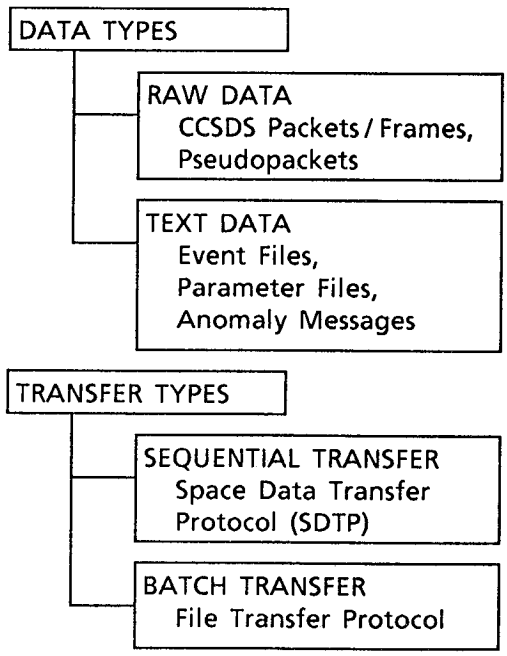

Fig. 6 Data Types and Transfer Types

To process raw packets, some ancillary information needs to be attached to each packets. The type of ancillary information depends on the application data. For telemetry packets, for example, reception time at the ground station and data quality information should be attached to each packet as ancillary information. In **DIOSA**, this information is called Application Specific Ancillary Information (ASAI).

Examples of data of the Text Data Type are event files (sequence of events and commands), parameter files (listing of parameters), anomaly messages (indicating the ocurrence of an anomaly), and state vectors. These kinds of data are transferred as text files with a standard format so that they can be processed with standard UNIX tools like AWK.

4.2 Transfer Types

Communications protocols to be used for the data ports of functional blocks are standardized in two categories, namely Sequential Transfer Type and Batch Transfer Type (Fig. 6).

The protocols of the Sequential Transfer Type are used for transferring data which needs to be transferred while it is being generated. These protocols can be used for delayed transfer as well (e.g. receiving telemetry data stored at a ground station) if the user wants to use the same software for both realtime and delayed data.

A data delivery protocol, which is called the Space Data Transfer Protocol (SDTP), is the standard application layer protocol of **DIOSA** for sequential transfer. This protocol is used together with a standard transport protocol such as TCP/IP or X.25 (Fig. 7). Details of SDTP are described in the next subsection.

The protocols of the Batch Transfer Type are used for transferring files. A standard file transfer protocol such as FTP or FTAM can be used for batch transfer (Fig. 7).

Most raw data will be transferred with the sequential transfer protocols. However, the

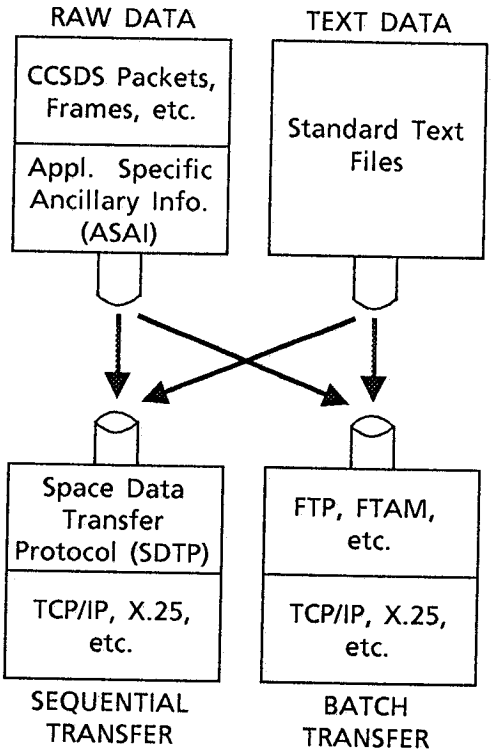

Fig. 7 Protocol Stack Used for Data Ports

batch transfer protocols can be used for transferring raw data which is already stored in a file. Most text data will be transferred with the batch transfer protocols. However, the sequential transfer protocols can be used for transferring text data if it has to be transferred immediately after its generation.

4.3 Space Data Transfer Protocol (SDTP)

The Space Data Transfer Protocol (SDTP) is a connection oriented protocol used for delivering sequential space data (e.g. sequence of CCSDS packets) from a functional block to another functional block. SDTP has a capability of (1) requesting data transfer, (2) specifying Spacecraft ID (SCID), Application Process ID (APID) and other attributes of data, and (3) notifying of any events related to data delivery (e.g. loss of RF signal). The formats of the Protocol Data Units (PDUs) of SDTP is shown in Fig. 8.

SDTP is used as follows. When Functional Block *Tom* wants to receive a sequential data

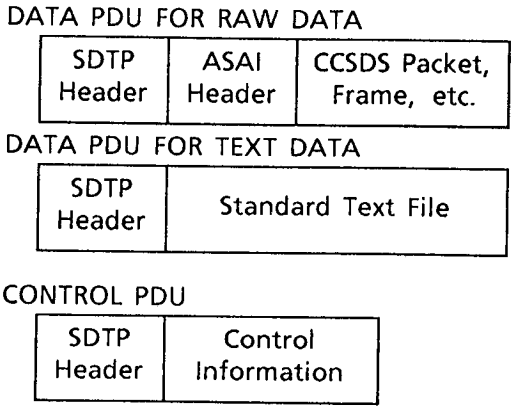

Fig. 8 Format of SDTP_PDU

from Functional Block *George*, *Tom* opens a connection of SDTP with *George* saying "Hi, George. I want to receive packets with APID=5 and SCID=32 in realtime" (Connect Request). Then *George* answers "OK, Tom" (Connect Response) and starts data delivery. When the data delevery is (or must be) terminated, either end of the connection can disconnect the connection.

SDTP can be used together with a data distribution service as explained below as well as in a bilateral mode.

In a distributed space operations system, a data stream often needs to be delivered to several destinations simultaneously (multicasting). For example, some telemetry data may be monitored by several workstations simultaneously. In **DIOSA**, data distribution and multicasting are performed by the Data Distribution Functional Block (DDFB). A DDFB is placed in every domain where it is needed. The DDFB of a domain acts as the data server of the domain. The DDFB receives sequential data from the DDFB of another domain or from another functional block in its own domain, and distributes the received data to functional blocks in its domain (Fig. 9).

In such a situation, when a functional block wants to receive a sequential data, it opens a SDTP connection with the DDFB of that domain requesting transfer of that data. Then the DDFB checks whether or not it is

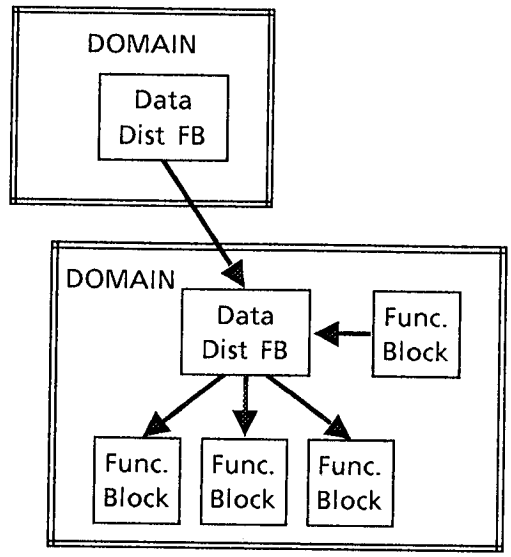

Fig. 9 Data Distribution With Data Distribution Functional Block (DDFB)

receiving that data, and if it is not, it sends a request for that data to another DDFB. Therefore, the requesting functional block does not have to know where the data originally comes from. It always sends requests to the DDFB of its domain.

5. MANAGEMANT STRUCTURE

Each domain has a functional block, called the Domain Managemet Functional Block (DMFB), which manages the functional blocks of the domain. Management within a domain

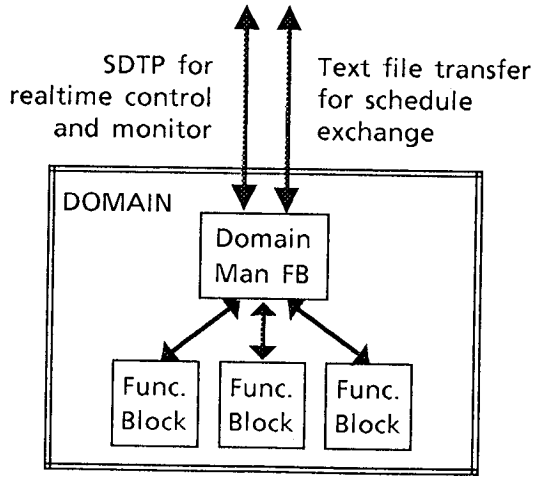

Fig. 10 Intterdomain Management With Protocols of Fig. 7

Table 1 Some Examples of Functional Blocks

Functional Block Name	Typical Location	Function	Data Input	Data Output
Domain Management	Every Domain	Manage functional blocks of the domain	Event file (Text Data, Batch Transf.), Config. change req. (Text Data, Seq. Tsf.)	Status information (Text Data, Seq. Transf.)
Data Distribution	Every Domain	Distribute data	Various data (Raw Data, Seq. Transf.)	Various data (Raw Data, Seq. Transf.)
Command Transmission	Ground Station	Transmit commands	Command data (Raw Data, Seq. Transf.)	Command signal (Analog)
Telemetry Reception	Ground Station	Receive and decode telemetry	Telemetry signal (Analog)	Telemetry data (Raw Data, Seq. Transf.)
Radiometric Data Collection	Ground Station	Collect range and doppler data	Radiometric signal (Analog)	Range/doppler (Raw Data, Seq. Transf.)
Spacecraft Control	Ops Center	Generate commands and verify results	Event file (Text Data, Batch Transf.), Telemetry data (Raw Data, Seq. Transf.)	Command data (Raw Data, Seq. Transf.), Anomaly msg (Text Data, Seq. Transf.)
Timeline Generation	Ops Center	Generate timelines	Ops request (Text Data, Batch Transf.)	Event file (Text Data, Batch Transf.)
Orbit Determination	Ops Center	Determine orbit	Range/doppler (Raw Data, Seq. Transf.)	State vector (Text Data, Batch Transf.)
Data Archive	Science Center	Archive data	Telemetry data (Raw Data, Seq. Transf.)	Telemetry data (Raw Data, Batch Transf.)
Data Analysis	Science Center	Analyze data	Telemetry data (Raw Data, Batch Transf.)	Papers to be published in journals

is performed with the management port of functional blocks and a standard network management protocol like SNMP.

Managinment between domains can be performed either with a network management protocol or with the protocol suit described in Section 4. In the latter case, schedule information is exchanged in Standard Text Files with the File Transfer Protocol, while configuration change messages and status information messages are exchanged with SDTP in realtime (Fig. 10).

6. EXAMPLES OF FUNCTIONAL BLOCKS

In Table 1, some examples of functional blocks are given. Please note that this is not a comprehensive list of functional blocks. An example of an operations system constructed with **DIOSA** is shown in Fig. 11.

7. CONCLUSION

This paper presented the concept of a distributed multimission operations system.

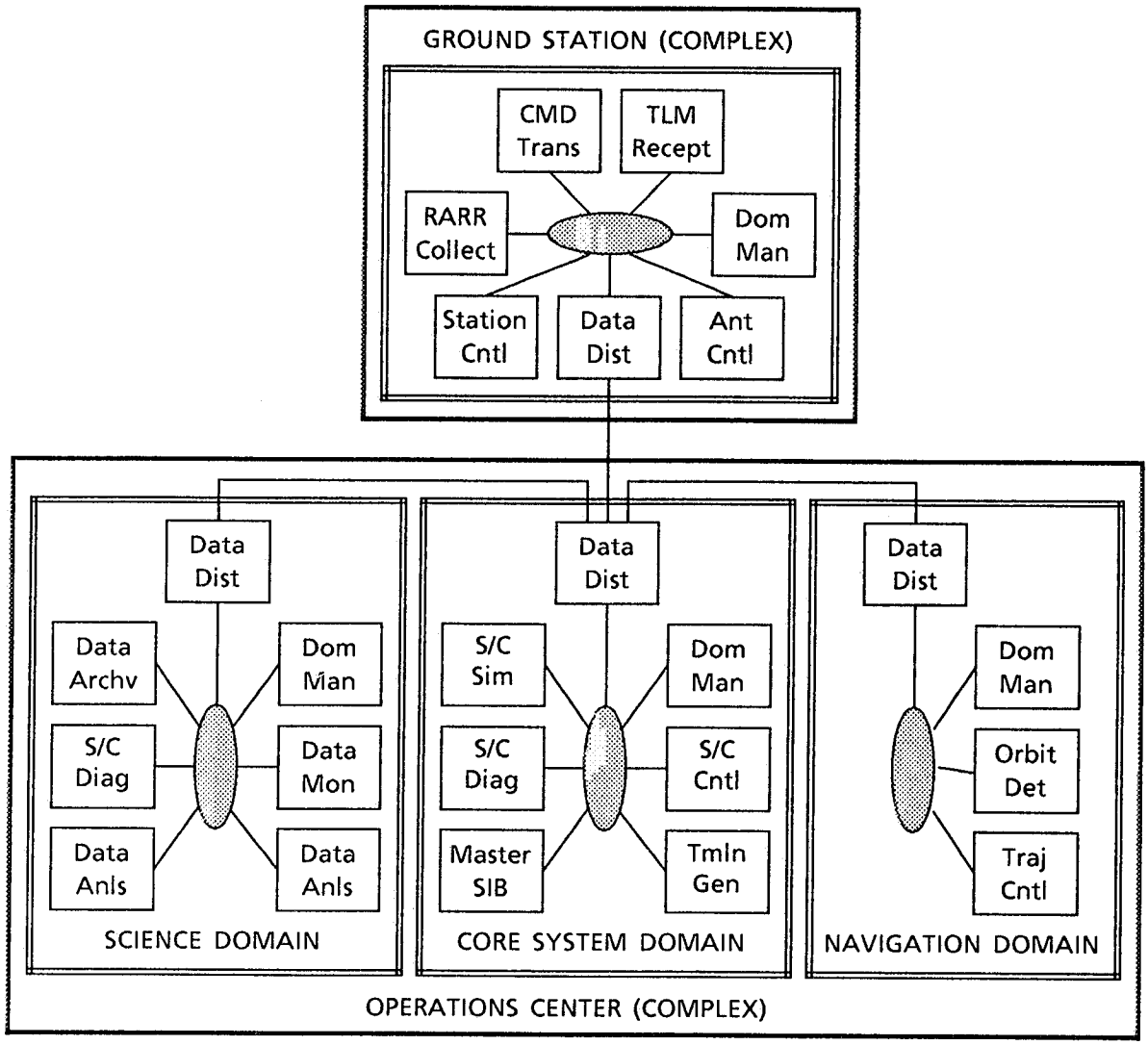

Fig. 11 Example of Spacecraft Operations System

We plan to develop a prototype in a few years to verify the validity of this concept.

8. REFERENCES

Cipollone, G., & McKay, M. (1992). An engineering database management system for spacecraft operations. *Proc. of the Second Int. Symp. on Ground Data Systems for Space Mission Operations*, 787-790.

Holder, B., & Levesque, M. (1992). Making adaptable systems work for mission operations: A case study. *ibid*, 327-331.

Kaufeler, J.-F. et al. (1992a). The advanced technology operations system ATOS. *ibid*, 425-434.

Kaufeler, J.-F. et al. (1992b). The European space agency standard for space packet utilization. *ibid*, 813-818

Mandl, D. et al. (1992). SAMPEX payload operation control center implementation. *ibid*, 63-68.

Newsome, P., & Otranto, J. (1992). The advanced orbiting systems testbed program: results to date. *ibid*, 501-506.

Systems Engineering

* *Presented in Poster Session*