Organizing Space Shuttle Parametric Data for Maintainability

R.C. Angier*
IBM Corporation, Houston, Texas

A model of organization and management of Space Shuttle data is proposed. Shuttle avionics software is parametrically altered by a reconfiguration process for each flight. As the flight rate approaches an operational level, current methods of data management would become increasingly complex. An alternative method is introduced, using modularized standard data, and its implications for data collection, integration, validation, and reconfiguration processes are explored. Information modules are cataloged for later use, and may be combined in several levels for maintenance. For each flight, information modules can then be selected from the catalog at a high level. These concepts take advantage of the reusability of Space Shuttle information to reduce the cost of reconfiguration as flight experience increases.

Background

NOW that Space Shuttle has proved itself in flight testing, a new challenge is beginning: the task of bringing the Shuttle's unique capabilities to full use in a viable Space Transportation System. The success of this project is determined by cargo capacity, performance bounds, and Orbiter turnaround time. It is constrained by per-flight cost objectives, aimed at putting Shuttle services on an economic basis. And it is driven by a steady increase in flight frequency toward an operational level. The ultimate goal is a "space truck" fleet, routinely transporting payloads to and from orbit.

Bridging the gap from flight test to full operations will require many changes. Projected increases in flight rate will strain the capacities of all areas involved in preparation of Shuttle flights. New ways of doing business must be found which are both cost effective and reliable. Project organization, procedures, and capabilities will all be affected by an evolution from the current emphasis on research and development to an emphasis on production.

This paper addresses one aspect of the evolution to an operational era: management of the computer data necessary to prepare for and fly Space Shuttle missions. The need for accurate information is pervasive, affecting computer systems in simulators, control centers, test facilities, and onboard each Orbiter. Integration and flow of data products recur with each flight, and account for a significant portion of flight preparation resources. A new data management model is presented which facilitates the collection, integration, and use of information in such a way that data integrity is enhanced, while costs of data use are reduced.

Shuttle Data Management

Flight Data Preparation

Flight Preparation Activities

The process of preparing flight data for a Shuttle mission involves several tasks, which divide into two phases.

1) Planning and design phase:

a) Mission planning starts with mission objectives and ends with assignment of a specific Orbiter and trajectory to a specific payload set.

b) Flight design follows with the detailed mission planning.

c) Payload and communications data are defined and integrated to meet individual mission requirements.

2) Flight production phase:

a) Onboard flight software, together with its related simulator test environment, is revised to match the Orbiter, payloads, and flight characteristics.

b) The Shuttle Mission Simulator (astronaut training facility) and Mission Control Center (in-flight control) software are modified by specific mission data required for flight-related training of the crew and ground controllers.

c) The Launch Processing System (preflight and launch control) is modified with mission hardware and software characteristics, and preflight servicing of the vehicle and its payloads is performed.

Each of these activities has specialized data needs, involving many thousands of parameters. Many different types of data are required, ranging in concreteness from a generic definition of Orbiter characteristics, used in planning, to the specific description of a vehicle as it is configured on the launch pad. In addition, application data needs often overlap in content, while varying in schedule. The flow of data products among these activities must be streamlined.

Overlapping Mission Flows

As the number of flights per year increases, more and more missions will be in preparation at one time. The degree to which they will overlap is different for each flight preparation activity, depending upon the process time it requires. Resource contention among overlapping missions will also present a complex management problem. Unless the degree of overlap among preparation flows is managed, the multiplexing capacity of each activity may limit the flight rate achievable by the Shuttle program.

Mission Variability

Each Shuttle flight is defined by a carefully balanced set of mission objectives, in a unique combination. These result in designation of the Orbiter, payloads, flight path, and other major variables that define the flight. Yet even under controlled conditions, no two flights will be identical. Orbiter capabilities will change by Orbiter vehicle and with time. Trajectories will vary with payload mass properties, solid rocket booster thrust profiles, and seasonal atmospheric characteristics. Mission activities can be altered by the phase of the moon or by crew preference. The data description of each mission must respond to this variability.

Presented as Paper 81-2101 at the AIAA Computers in Aerospace III Conference, San Diego, Calif., Oct. 26-28, 1981; submitted Nov. 12, 1981; revision received Feb. 8, 1983. Copyright © American Institute of Aeronautics and Astronautics, Inc., 1981. All rights reserved.

*Advisory Systems Analyst, Federal Systems Division.

Late Availability of Information

The reusability of the Shuttle separates it from other space vehicles; it also affects preparation flow. Overall flight preparation time currently exceeds one year, while Orbiter turnaround time is less than four months. Thus several flights of an Orbiter will occur during the preparation cycle of a flight that plans to use it. In order to be reused, an Orbiter must be assessed and refurbished from one flight to the next. Malfunctions during the previous mission can cause capabilities to be downgraded or lost. Component aging effects and failures may require replacement by spares, or revision of operating characteristics. Orbiter monitoring requirements may be revised in response to problems encountered in flight. None of this is known until after the prior mission has flown. Hardware related information used in early flight preparation must therefore be predictive; actual data will only become available in the last few weeks prior to launch.

Need for Flexibility

Due to the unknown condition of the Orbiter following its flight, detailed long-term planning is impractical. For example, a failure could cause an Orbiter to be withdrawn from operation while repairs are being made, forcing reassignment of payloads or Orbiters. To allow adaptation to late changes, to reduce overlap, and to minimize the amount of rework required, a short flight preparation flow must evolve. Its elapsed time should be measured in weeks, compared to the months and sometimes years of Apollo and Skylab missions.

Cost of Data Errors

Underlying all flight preparation activities is the recognition that incorrect data can cause a failure to meet mission objectives, or in the worst case, loss of the vehicle and its crew. Accuracy of reconfiguration data is therefore of high importance. Strategies used to reduce the risk of data errors include tight configuration control, auditing, and extensive testing.

Current Data Management

Evolution

Current methods of managing Shuttle data evolved in response to the needs of STS-1, the first Space Shuttle mission. A multiyear effort was required to specify and develop software for the first flight. In the process, successive iterations refined the mission data. Software functional requirements also changed, at times affecting data needs. Logic and data came under control of the same change management boards, helping to keep them consistent. The result was a process aimed at stabilizing the contents of a software system for a mission.

Configuration Management

Both data and logic are managed by controlled changes to a mission baseline. To incorporate a data change, an existing complete baseline for a flight is assessed, and the affected data are identified along with their new values. These changes are submitted to a control board for review and approval, after which they become part of the next baseline for that flight.

Multiple missions complicate this process. Each new mission is initially built from an earlier mission baseline, to which changes are made to account for the differences. Subsequent changes to a mission may also affect any new missions built from it, making assessment of the change necessary in all related mission baselines. For example, if both STS-11 and STS-12 data baselines are derived from STS-10, then a change in STS-10 flight data must be evaluated for applicability in the other two flights. In addition, it is possible for a change submitted against STS-11 to affect both the earlier and later missions. This type of interaction increases much faster than the number of active mission baselines.

Information Flow

For each flight, a network of activities is started through which a variety of data products flow. Each product consists of a mission-unique collection of data required by a user, which must be rebuilt for each flight, based on manual assessment of change. A simplified view of current mission data flow is shown in Fig. 1. Handcrafting of unique data products for each flight causes the per-mission preparation cost to remain almost static.

An additional cost factor is the consistency among related data products. Given a long lead time for flight preparation, during which vehicle hardware or software logic can change, products generated at different times or from different baselines may be inconsistent. This causes additional manual analysis in tasks that integrate related data products. Late data changes aggravate this problem; all affected data products must be identified and correctly updated, often involving "patches" that bypass the normal preparation flow.

Reconfiguration

Like the Orbiter, the software systems used to prepare and fly the Shuttle are reusable. In its basic design, each system contains sufficient functional capabilities to support several flights. Software can be specialized to a mission by combining mission-unique data with a fixed logic base, in a process termed "reconfiguration." Due to the wide range of its functions, Orbiter Avionics Software exemplifies several aspects of the reconfiguration process.

Reconfigurable Data Requirements

The Shuttle is the most computer-dependent manned space vehicle ever flown. The onboard computer system is deeply embedded in Orbiter hardware design, and involved in most major activities. Among the functions performed by onboard software are flight control, sequencing, guidance, navigation, system monitoring, payload management, and display processing. All of these functions require flight-specific data in order to be adapted to a mission. Reconfiguration data are used to describe the external environment to the computer system.

Data Installation

Reconfigurable data are packaged according to how they are used. Separate flows are currently maintained by application, using a spectrum of methods for the installation of mission data. Flight-critical applications are the most rigidly defined; the identity and derivation of all needed reconfiguration data are specified beforehand, as an integral part of software implementation. Values are directly installed in computer memory data locations. Telemetry data requirements are slightly less rigid, resulting in generated logic sequences which are then compiled and installed on the system. Systems management and payload applications are

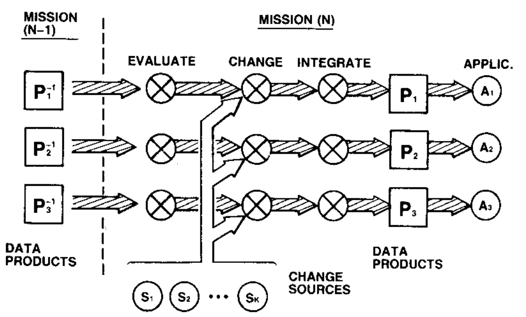

Fig. 1 Current flow of mission data.

more flexible, resulting in variable-sized tables which are used by real-time interpreters.

Data Validation

The critical nature of many flight software functions makes necessary a careful validation of its reconfiguration data. Reviews are conducted to assure that all values have been accurately installed. In addition, the correctness of the data is proven through use of the updated flight software system. Extensive simulation testing is required to achieve this goal, consuming about half of the total resources needed in flight software preparation. Further testing is carried out as part of crew and ground flight controller training simulations.

Once validated, the reconfigured flight software system is installed on the Orbiter. Its operational use begins with preflight checkout, followed by performance of primary Orbiter control functions in the active flight phases from ascent to landing. Ground support is provided during these phases by the Launch Processing System and the Mission Control Center.

Data Management Concepts

General

The previous sections describe management of Space Shuttle data as it is now; the remainder of this paper explores how it can be organized to avoid limitations that now exist. New data management concepts were developed as part of the Software Production Facility,[1] and share its goals of 1) reduced cost and time of reconfiguration, 2) flexibility to handle late changes, and 3) high reliability.

Two strategies were adopted to meet these goals. The first is standardization of capabilities to reduce the amount of work required. The second is automation, to reduce the cost of work still remaining, and to preserve its accuracy.

Standardization

In all areas of the Shuttle project, significant effort can be made to limit the uniqueness of a mission. Instead of defining a separate orbital path for every flight, a few standard orbits can be created, each of which replaces an envelope of ideal orbits. Propellant tank loading can be constrained to increments, rather than be infinitely variable. Crew procedures for deployment of a payload and carrier can be made standard. The effect of these and other efforts can substantially reduce the number of variables in the Shuttle project, and their interactions. By imposing constraints on the use of capabilities, we give up the ability to fine-tune the last bit of performance out of each flight, but gain substantial benefits in efficiency throughout the project.

Organization by Cause

Data changes do not occur randomly; they are driven by factors in the world that the data are modeling. Information can therefore be partitioned into natural sets, each related to a source of change. For example, all seasonally varying data can be grouped together, resulting in a set of information which is both complete and homogeneous. The contents of a set can later be divided among the applications needing its data.

Organizing data around the causes of data change has a number of advantages over current organization by data use. Since related information is grouped together, its joint correctness can be more readily verified. When a change occurs in a driving factor, its related data group can be replaced. This helps assure that all affected data elements are updated and are consistent across applications. Also, since the effects of change are concentrated into packages, considerable flexibility is permitted for late changes to a mission.

Figure 2 shows the effects of reorganization by source of change on the flow of mission data. In contrast to the current mission flow in Fig. 1, change sources are placed in line to the

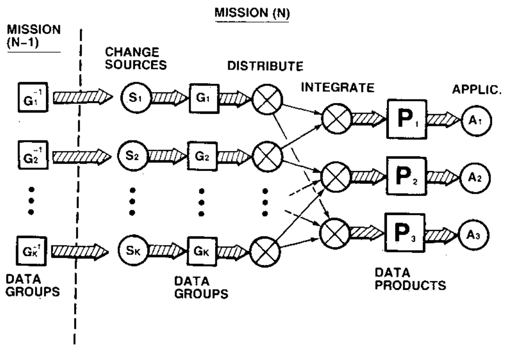

Fig. 2 Flow of mission data organized by source of change.

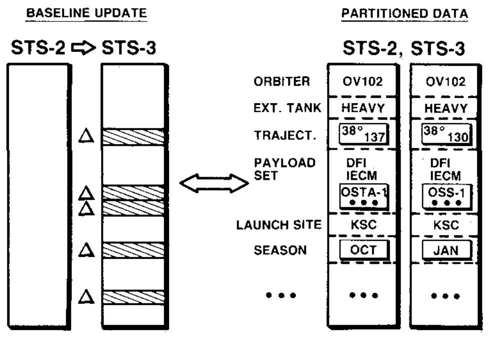

Fig. 3 Equivalent views of mission data baselines.

information flow. They are also independent, so that only the areas which actually change between missions need to provide new data.

Data Cataloging for Reuse

In the current mode of data management, STS-3 is defined by STS-2 flight data, plus a set of changes. An alternative way of looking at this relationship results when the reconfiguration data are partitioned according to sources of change. Figure 3 illustrates the similarity of these views. When the information in each data group is compared between the two missions, it can be seen that both flights describe the same Orbiter, launch site, etc. Some variations appear in selected areas, such as payloads, ascent and orbital trajectories, and seasonal atmosphere. If the data groupings are created and set aside beforehand, each mission can be formed by selection of the appropriate sets of data.

Cataloging

Data cataloging refers to setting aside groups of related data for later use by one or more missions. Each data grouping constitutes a component baseline of information, which is independently configuration controlled. It can be validated and approved by itself, without reference to other groups or specific missions. Alternative occurrences of the same group of data can coexist in the catalog; these make up the variations which are to be selected for a flight. To clearly distinguish between data function and data identity, each collection of data should carry two labels: a group name and an occurrence name. For example, (launch site, Kennedy) and (launch site, Vandenberg) are both occurrences of the data group "launch site."

Data Flow

The impact of a catalog on mission data flow is illustrated by Fig. 4. The basic flow has changed from serial evolution of successive missions to parallel development. With a catalog, new data group occurrences can be supplied one time, independently of mission use. Definition of a new payload or

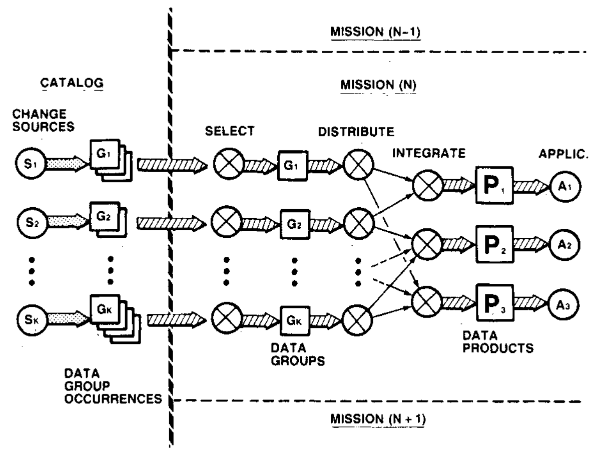

Fig. 4 Data flow using a mission-independent catalog.

relocation of a navigation ground station can be specified in response to external stimuli. These accumulate to form a set of alternative occurrences in each data group.

Each mission is defined independently, using a common catalog. An appropriate set of data group occurrences is selected out of those in the catalog, resulting in a mission-unique data product. The set of data group names selected for a mission forms a mission baseline.

Data Collection

With a data catalog, the collection of new sets of data can occur independently of individual missions. This allows separation of the factors driving data use (selection) from those driving its supply (collection). For example, selection of an Orbiter to fly a mission can be kept separate from the supply of data defining a new Orbiter. Thus the frequency with which data are selected for mission use no longer affects the frequency with which new data must be provided.

A wide range of supply rates exists. Some information, such as seasonal atmosphere, must be selected for every flight, based on launch data, but are rarely, if ever, altered. Others, like aerodynamic performance data, are supplied infrequently (with each new Orbiter). Still others define transient characteristics that must be resupplied regularly, such as engine burn characteristics and inertial measurement unit calibrations. Mission independence of data supply has the potential for freeing many people from the necessity of providing new data for every mission, with increasing frequency, on into the future. The cost of flight preparation can be reduced accordingly.

Data Selection

For each Shuttle mission, a set of reconfiguration data can be constructed out of selected components from the catalog. An appropriate set of occurrences are specified by name. Selection is guided by a predefined list of required data groups, for which occurrences must be specified. Additional selection from remaining data groups is optional, and depends upon specific mission requirements. For example, data occurrences related to the Remote Manipulator System in the payload bay would be selected only by missions requiring it.

Data Reuse

Reuse both depends upon and fuels the success of standardization. To the degree that a few standard occurrences of a data group suffice for varying missions, those occurrences will be reused. Conversely, the existence of a limited set of occurrences to select from, coupled with recognition of the cost of supplying and validating a new set of data, will tend to enforce standardization. Reuse is supported by the desire to accumulate "shelf life" on a proven set of data; increased confidence comes from having successfully flown it before. Also, much of the cost of data development is incurred in its validation prior to use. As later flights take advantage of previously used information, the proportion of new, untried data used in each mission will decline to an asymptotic limit.

Learning Characteristics

Properly controlled, the information system can behave as a learning machine. The catalog of data groups constitutes a memory within the data management system. Its contents are supplied based on long-range planning and the anticipation of data needs. With each flight, the "goodness of fit" of its selected data occurrences can be assessed, and exceptions and corrections fed back to the catalog. Changes in data needs can also be identified, leading to the collection or retirement of information. This is the master control loop of the data management system, with which the catalog can grow in knowledge and refinement as flight experience increases.

Data Hierarchies

Grouping of information through the use of subsets and supersets can simplify data management. Related data groups can be associated together in a higher-level group through a list of occurrence names. This can serve as a tool either to divide a large data group into more manageable parts for maintenance, or to gather related groups into one unit to simplify selection.

For example, groups defining the characteristics of various sensors aboard an Orbiter can be specified by a "sensor" data group, consisting of a list of devices and the low-level data group name associated with each one. Such a group could be maintained by the vehicle subsystem manager who keeps track of sensor assignments; a separate occurrence would be needed for the assignments on each Orbiter. Note that the subsystem manager would not be involved when a specific device, such as rate gyro No. 0073, changed some of its characteristics; that would be the job of a calibration specialist. But he would be affected if it were replaced by a spare rate gyro, an event which may not be significant at the other level. In this way, areas of responsibility can be isolated from each other.

This technique permits higher-level data, in the form of names, to be collected and stored in the catalog. A hierarchical group can also include elemental data that are logically associated with it. When a higher-level data group is selected for a flight, all of its subordinate data are included. Thus the benefits of preassembled parts can be applied to the Shuttle software environment. Most notably, higher-leveled groups can be integrated and verified in advance, independently from the missions that use them.

The use of supersets can be easily extended to still higher levels. Continuing the previous example, a "sensor" data group could be associated with other Orbiter subsystems such as hydraulics, consumables, and electrical power through an "Orbiter" data group. It could, in turn, be part of a "vehicle configuration" data group, which also identifies the external tank and solid rocket boosters making up the Space Shuttle launch vehicle. Finally, the "vehicle configuration" data group could be specified, along with trajectory and cargo manifest, in a "mission" data group. The result of this construction of data is a multilevel hierarchy of data groups, shown in Fig. 5. Independence of functional areas of responsibility is maintained throughout.

The construction of data hierarchies resembles parts assembly in a bill of materials process. Data "parts" are combined into subassemblies, which are then used to make assemblies, and so forth. However, some differences exist with this analogy:

1) In contrast to a simple parts assembly process, the relationships within a data hierarchy can be logical as well as physical. It is just as reasonable to combine geographic data and atmospheric characteristics with guidance parameters (all

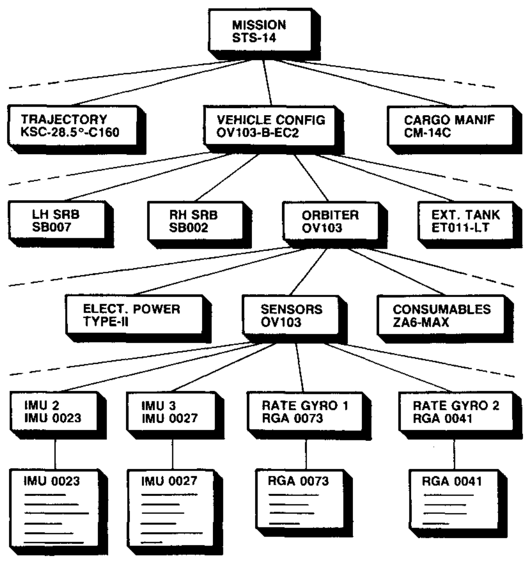

Fig. 5 Multilevel data hierarchy.

are related to approach and landing) as it is to combine Orbiter subsystems.

2) The engineering drawing of a physical assembly describes the relationship of its parts, and directions for their construction. With logical relationships in the data world, a drawing will not suffice. Its equivalents are data interface specifications and data integration algorithms, tailored to a specific assembly process.

3) Unlike a physical assembly process, construction of a data hierarchy can proceed from either the highest or lowest level, or from any point in between. When a data hierarchy is specified in a top-down fashion, each of its components can have a wide range of variability. This is somewhat like allowing the subparts of a part in a manufacturing process to vary in their color, shape, and dimensions. With each downward step through a data hierarchy, components are specified in greater detail. At the "mission" level, for example, the component "cargo set" is general. It will later be replaced by a list of the specific payloads, pallets, structures, and kits manifested to a flight. Each of these are further defined in terms of signal flow, mass properties, etc. This process can continue until the data hierarchy is fully specified.

Data Constraints

Constraints are a natural part of any system; all of its elements must conform to physical laws, logical boundaries, and procedural limits. In recognition of this fact, constraints are an integral component of the proposed data management model. To this end, every data structure consists of a data definition and an associated logic module. A principal function of the logic module is application of constraints to its related data. Limits are applied when information is collected, combined in a data hierarchy, or selected for use in a mission. These will be more fully described in the next section.

Data Management Processes

Configuration Management

With data cataloging, the concept of a mission baseline still exists, in a modifed form. The data for a mission are constructed out of selected data groups, combined to form a whole set. Each selection is accomplished by specifying the name of a data occurrence. Thus the mission baseline is abstracted into a list of names, which can be managed separately from component baselines in the catalog and from other missions. Different skills can be applied to the tasks of data management and mission definition. However, the baselines are not totally independent; they are connected in two ways:

1) Mission to catalog—Specification of a new data occurrence for a mission establishes the need for its collection. This can be done during the planning phase, so that the appropriate data can be "ordered" from its supplier. Lack of selection can also affect a data occurrence; after a period of disuse, it may be retired from the catalog.

2) Catalog to missions—Replacement or repair of a data occurrence in the catalog impacts missions which have selected it. Since the affected flights may be in any stage of the assembly line from planning to planetary orbit, application of the data change must be under separate control for each mission. Both old and new versions of a changed data occurrence should remain available, with a notification sent to all missions that use it. Note that connections among mission baselines are maintained only through the catalog.

Data Integration

Collection Constraints

Included in the definition of a data group are the limits which it must satisfy in order to be valid. A newly supplied data occurrence is testable at two levels. Each datum can be compared to predefined limits which establish its reasonableness. At a higher level, interrelationships among the elements of a group are tested. Functions of reconfiguration data in a group are evaluated against related bounds, which may also be functions. Typical examples are limits on the behavior of a polynomial whose coefficients are reconfigurable, and a test that an array has the characteristics of a rotation matrix. Constraint functions and limit values are contained in the logic module defined for each data group. Both levels of data testing must be satisfied; a data occurrence should not be made available for use until the correctness of its contents can be asserted.

Integration Constraints

Integration of data can be done in stages, whenever information is combined. In this way, the process is manageable, can be observed, and can be corrected. As integration proceeds, it validates successively larger collections of data, each of which can be cataloged for later use. Reliability is built into the data at each stage.

The first step of integration takes place when a data occurrence is collected; interrelationships among its elements are tested for consistency. Subsequent integration steps occur as a hierarchy of data groups is formed by the collection of higher-level data group occurrences. As each higher-level data group is completed, constraints among its components are applied. As in data collection, assertions are tested by a logic module associated with the data group.

Mission-Independent Integration

A significant portion of the integration process can be performed without reference to a specific flight. Data group occurrences entered into the catalog can be combined into data hierarchies, or subassemblies of standard information. Integration constraints are applied to the combined group before it can be used in subsequent integration steps. This process proceeds from the bottom upward, until mission-specific characteristics must be considered. The results of each integration step can be preserved in the data catalog for later use. Examples of the level of integration achievable without specific flight reference are 1) definition of an entry trajectory, 2) combination of a communications satellite and its upper stage booster, and 3) definition of the signal paths through an Orbiter.

Mission-Unique Integration

In contrast to the stability of the data catalog, definition of a flight is influenced by a number of less predictable forces, such as payload availability and changes in mission objectives. However, the proposed organization of data makes this task considerably easier. Three tools are available to assist in mission data management.

1) Mission-unique integration of data assemblies can continue the bottom-up integration methods used in the catalog. This process is particularly useful in constructing the contents of the payload bay, which are unlikely to be reused by another mission in their integrated form.

2) Top-down specification of a flight is a complementary process to bottom-up integration. The primary components of a mission form a high-level data group, which is specified in much the same way as a flight assignment manifest; a specific Orbiter, trajectory, major payloads, and launch date are tied together to define a Shuttle mission. Each of these components can then be refined through further specification at the next lower level. This process continues until cataloged data assemblies are reached. These two methods combine to fill in any remaining portions of a mission data hierarchy.

3) Late changes can be managed by replacement of a data group at any level of the hierarchy; the modularity of data matches the modularity of the hardware system that it models.

The flow of mission data taking into account all aspects of the data model is shown in Fig. 6. In contrast to simple data cataloging (Fig. 4), the task of integration has been moved to the earliest possible point in the flow. As much of this work as possible is done mission-independently to avoid redundant effort. Early integration avoids discovery of incompatibilities late in the flow, when little time is available to deal with them.

Application Sensitivities

As data describing a mission are combined, they are integrated independently of the applications which will eventually use the information. Constraints applied to this point relate to data content, rather than to how the data are used. Additional constraints based on application unique sensitivities can be applied after mission selection.

Reduced Testing

Substantial reductions can be made in the task of verifying the reconfigured software through its execution. Much of the testing has already been done statically during integration of the data, before it is ever applied to software. Data integration cannot completely replace dynamic testing; a higher level of confidence is gained by demonstrating that a mission system performs as predicted. However, each successful dynamic test demonstrates the validity of static tests preceding it, tending to lessen the need for additional testing.

Data Structures

Functional Decomposition

Partitioning

The data management model described above relies on the ability to divide reconfiguration data according to the sources of change. It is recognized that this can be a difficult task. Existing information is oriented to usage, and is often heavily processed before it becomes visible. Significant analysis may be required in determining the true sources of data before its drivers can be understood.[2]

Decomposition

The objective of data partitioning is functional independence of the resultant parts. "The strategy of separating the variables in the environment and providing different mechanisms for dealing with each is almost universal among systems that are successful."[3] Functional independence permits both the scope and time of data change to be managed effectively. Interactions among data are minimized.

When information has been preprocessed, simple partitioning may not be able to effectively separate variables. In such cases, existing reconfiguration data can be derived from its independent sources. A classic example of the need for data derivation is the inertial measurement unit (IMU) on the Orbiter. Data defining each of the three IMUs and their orientations to the onboard software system are dependent upon 1) the unit's error characteristics, 2) its location on the Orbiter, 3) the location of the Orbiter on the Earth, and 4) the Orbiter attitude (vertical or horizontal).

Structural Decomposition

Data groups were defined earlier to be stable, homogeneous, standardized collections of data. These characteristics can be further enhanced by structural decomposition of a data group into specialized components, as shown in Fig. 7. In this process, information can be represented in a very stable form, with transformations to bring it to and from more volatile representations.

Source Isolation

The stored form of a data group can be separated from the way in which it is collected. This allows a data group occurrence to be defined in terms familiar to the data supplier. It can be tested in its initial form, then transformed into a standard representation. For example, the coordinates of a landing strip can be defined by geodetic survey parameters, then transformed to an Earth-centered vector.

Source isolation has several advantages. Dynamic variables can be removed by the transformation, resulting in information which has archival properties. Independent concurrent data sources can also be supported. Finally, evolution of data supply can take place without significant effect on the stored form of the data; replacement of an input transformation will allow collection of data in a new form.

Data Expansion

Once data have been collected into a standard form and validated, additional information can be derived from it. An

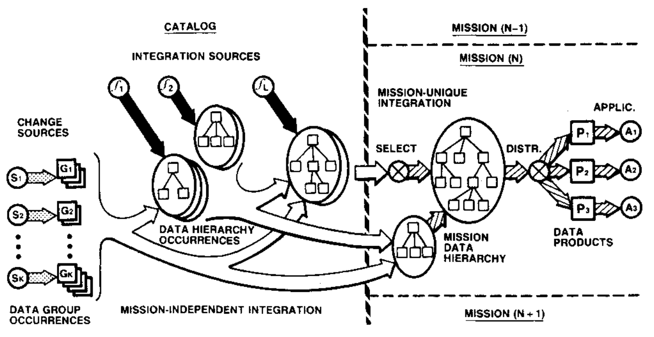

Fig. 6 Data flow using full data management model.

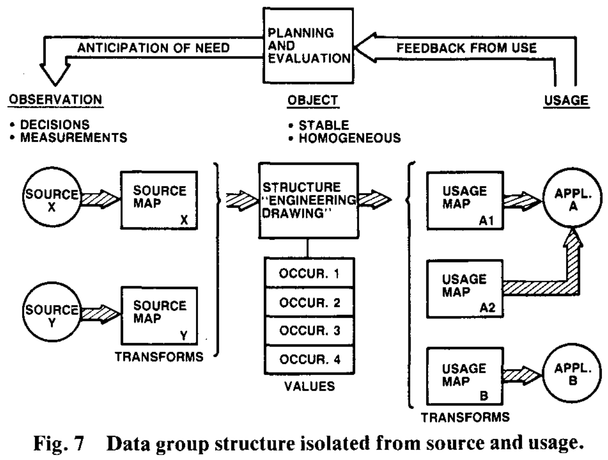

Fig. 7 Data group structure isolated from source and usage.

expansion transformation can simplify the data collection process while providing a reliable means of repeatably generating data.

Data expansion can be used during integration to derive the common properties of a higher-level data group. For example, the joint mass properties of a payload and its carrier can be derived from their individual mass properties (in a standard form) and a description of their connection geometry (data which is supplied directly to the higher level data group). Integration of other properties, such as communications signal flow, thermal and acoustic characteristics, can be managed similarly.

Common Structure

The occurrences of a data group can be tied to a common specification of their contents. Only the varible portion, typically data values, need exist in an occurrence, along with a link to its associated data group definition. In this way, the commonality of contents among occurrences can be maintained. A common structure allows data occurrences to be managed as interchangeable parts.

Usage Isolation

Data need frequently overlap across applications; each has a specific set of information that it requires from a data group. In addition, a single application may have different uses for the same type of object. A payload experiment, for example, can have a considerably different representation to a crew display than to its electronic control sequencer. Both describe the same experiment but in different terms, using overlapping parts of its full description.

Usage isolation separates the natural form of information from its varied applications. A separate transformation can be defined for each application using a data group, and within an application, for each role in which the information is used. This permits separation of the need for information about an object from the object itself. It reduces the impact to the data system from changes in application data requirements.

The usage transformation module is a logical place to define application sensitivities. These are based on limits to how the information is used, rather than its fundamental constraints. Separate transformation and constraints maintain the integrity of each data application.

Adaptability

The specialized structures identified above form a transformation sequence which has some useful properties. Both data suppliers and data users can view information on their own terms and apply their own constraints. Adaptation to changes in either's representation is possible by localized replacement of component structures. Between supplier and user, information can be stored and maintained in a stable form.

Conclusions

A model of data organization based upon the principles of standardization, modularity, and data cataloging provides a reasonable structure for systematic management of Space Shuttle reconfiguration data. Preparation costs can be substantially reduced by conversion of many activities from mission-specific to mission-independent roles. Increased reliability can be achieved by built-in data validation and integration processes. The data organization model also provides the flexibility necessary for responding to change inherent in the Shuttle program.

Acknowledgments

This work was sponsored by the Spacecraft Software Division, Johnson Space Center, under NASA Contract NAS 9-14444. Significant technical contributions were made by John R. Garman of the Spacecraft Software Division, NASA, Houston, Texas, and by Thomas P. O'Brien of IBM Federal Systems Division, Houston, Texas.

References

[1] "Software Production Facility Operations Planning Document," NASA JSC-16763, Vol. I, Book 1, Dec. 1981.

[2] Parnas, D.L., "On the Criteria to be Used in Decomposing Systems into Modules," *Communications of the ACM,* Vol. 15, No. 12, 1972, pp. 1053-1058.

[3] Weinberg, G.M. and Weinberg, D., *On the Design of Stable Systems,* John Wiley & Sons, New York, 1979, p. 169.