INFORMATION-DRIVEN CONTROL OF PRODUCT GENERATION AND RECONCILIATION: Generating Cassini ISS and VIMS Data Products on a Budget

A. Culver, R. Patel, A. Stanboli, H. Lee, S. Noland, J. Diehl, C. Avis, J. Henricks
Caltech/Jet Propulsion Laboratory

Introduction

The lengthy 12-year Cassini mission duration brings severe pressure to reduce operations costs. However, mission costs are pushed upward by complex science desires and vast spacecraft and instrument capabilities which must be used to their fullest. Regardless of the complexities, in the end, science data product generation essentially consists of telemetry processing, data assessment and file delivery. Fundamentally these processes can be information-driven, and as such, they are ideally suited to being automated. Therefore, the key to cost savings is to automate by developing software that uses information for control of product generation rather than to rely solely upon more costly human resources to perform the same tasks. This paper discusses the design and implementation of the automated Cassini control function, a concept which is extensible to any subsystem required to perform systematic, well-defined, data-driven or time-driven processing.

Driving Requirements - Why Automate?

Given the comparatively long duration of the mission (7+ years for cruise and 4 years for the planned tour), it is vital to constrain the operations cost for product generation and delivery. Cassini is unlike shorter duration missions for which it might be cost effective to hire a large workforce for a brief amount of time to manually generate data products. Instead, Cassini enjoys a lengthy cruise phase allowing for relatively long-term development to support operations. The decision to automate was initially considered for three reasons: 1) the ground system is required to limit its workforce to prime shift operations, 2) some products must be generated and delivered during non-prime shift times (as described below), and 3) funding for operations workforce is minimal.

The ground system must generate and deliver Imaging Science Subsystem (ISS) and Visible and Infrared Mapping Spectrometer (VIMS) data products in a timely manner. These data products are delivered to two separate types of recipients: the Navigation team and the Science teams.

The Navigation team uses a subset of the estimated 750,000+ ISS images to perform precise orbit determination, from which critical trajectory/orbital correction maneuvers are planned. These critical Optical Navigation images must be delivered to the Navigation team within 20 minutes of the time the data reaches Earth. For the Science teams, initial product delivery is required within 2 days, and final product delivery is required within 12 days of the time the data reaches Earth. Product accounting (identification and explanation of data gaps) and assessment reports must also be delivered to the Science teams within that same 12-day period. Finally, life-of-mission (LOM) storage is required for all data products generated.

The fundamental requirements to 1) constrain the operations cost and 2) deliver products in a timely manner strongly suggest that an automated ground data system is necessary to support both the ISS and the VIMS instruments for the Cassini Tour phase. Such a system has to be self-monitoring, near "lights out," and capable of being operated during prime shift by a small workforce while, at the same time, meeting the timely product delivery requirements demanded by the Navigation and Science teams. In the cost-capped Cassini budgetary environment, money spent generating data products reduces the resources available for science analysis (the reason for the mission.) Resources spent during cruise to develop automation become investments in the ultimate mission science return.

Development cost constraints dictate that resources not be spent duplicating existing functions. Therefore, in order to cost effectively design and implement an automated system, the ground system is integrated with existing multimission capabilities, and it inherits capabilities from ground software developed before the launch. Using existing multimission capabilities reduces the development cost to acquire telemetry data from the Deep Space Network, but it limits flexibility and introduces interface issues. These interface issues primarily concern operations, but they also impact the ability to reliably automate data production. Specifically, the concerns include the ability to establish and maintain connectivity across flight operations firewalls, the dependency on remote servers maintained by other organizations, and the management of configuration changes. Pre-launch ground software is a good starting point for data product generation, but it requires a transition from a manually executed program for generating a small set of products to an automated system capable of high volume production, data assessment and product reconciliation.

The Processes

The processes implied by the Cassini product delivery requirements include activities such as: scheduling telemetry processing; accessing and deciphering telemetry; product formatting, delivery and quality assessment; reconciling expected data products with products that are received; and accounting for missing data. These processes and their interactions are illustrated in Figure 1. Many of these processes were accomplished manually during the Cassini cruise phase including the Jupiter flyby. Accordingly, human resources were necessary to launch programs and compile information from a variety of sources to complete the tasks. However, the processes within the Cassini mission lend themselves well to the automation of information-driven events. Although these processes are Cassini-specific, this type of understanding of the system is essential before automation can proceed. Once the processes, their interactions, and their dependence on information are defined, the processing can be considered for automation.

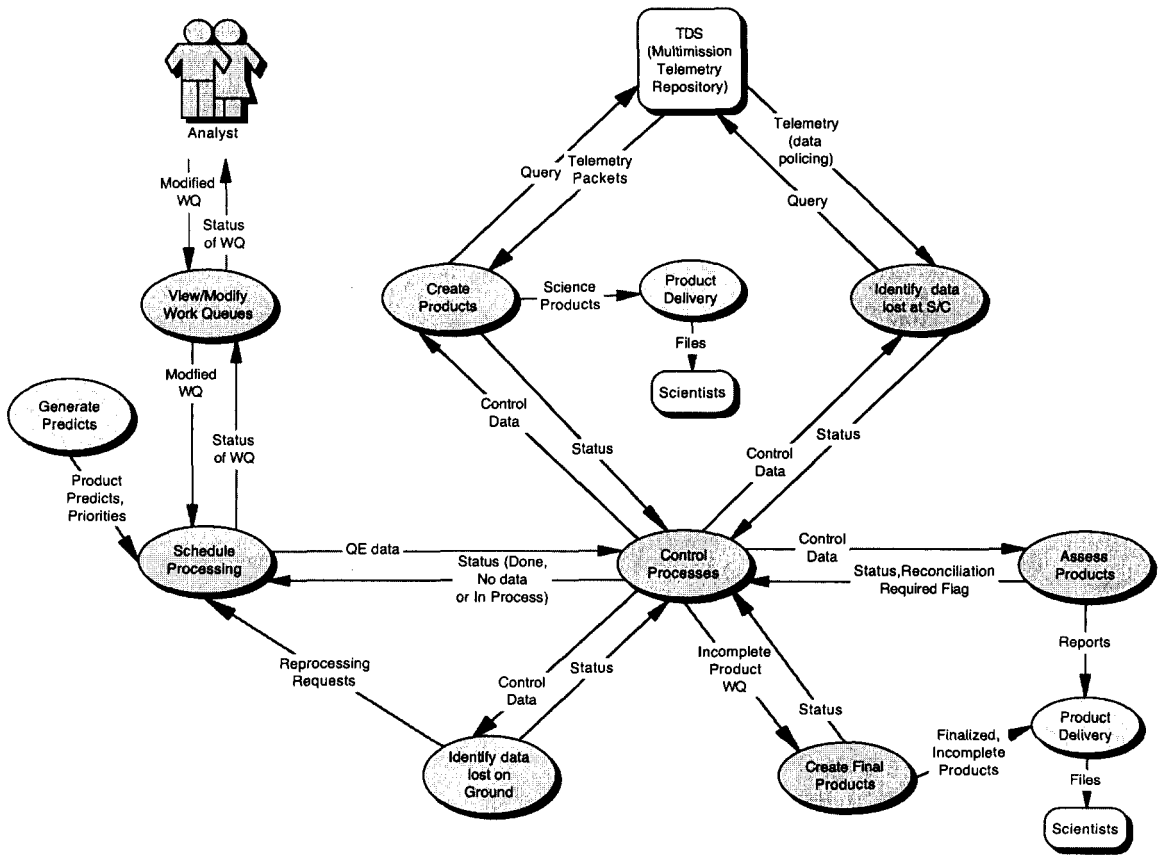

Figure 1. Product generation process flow diagram

The Design Challenge

The Cassini challenge was to create a control system for data production that would be driven by information rather than being driven by human resource considerations. The control system must access and utilize information on predictable events, and act upon information derived from the data itself. More precisely, it must become an automated, self-monitoring, fail-safe system that can accomplish production while still allowing for human intervention when necessary.

Information Needed for Automation

Both static and dynamic types of information are needed to transition formerly manual processes into software components.

Static information needed to drive the automation consists of several categories: processing rules and parameters, job and task sequencing, location of input/output data, system configuration, and Project/Instrument-specific data.

Processing rules and parameters include information such as how soon after a downlink to begin product generation (or reconciliation). This category also includes information regarding the data source, query values and a multitude of parameters required for telemetry processing jobs.

Job and task sequencing includes a sequential list of the tasks (single programs) to be performed to accomplish a job (i.e., a prescribed work product that includes one or more tasks). The relative timing of jobs is essential as well. For example, the system should be capable of waiting until all telemetry processing within the upstream ground data system is complete before attempting reconciliation.

The *locations of input and output data* are also needed for each task within the automated system in order to locate and perform operations on data generated or maintained by other tasks. The tasks also need to distribute output data to external teams while maintaining life-of-mission storage.

System configuration provides a mechanism for tracking software server locations and assignments. For example, when servers are distributed on a multitude of hosts on a shared network, system configuration allows for identification of servers throughout the network. It also includes server assignments, such as servers dedicated to critical optical navigation image processing.

Project data is used as a basis for product label meta-data and includes mission phases, sequence boundaries and product delivery routing information.

Finally, *instrument-specific data* is necessary due to deferred flight software development. Over the span of cruise and tour, different algorithms have been implemented to decipher telemetry from the instruments, and the distinction between algorithm versions is time-based. The automated system must know which algorithm to apply to data generated at any given time.

Dynamic information is also needed to drive the automation and includes the following categories: predicted events from sequence generation, generated product information, system resource availability and task status.

The *predicted events* allow visibility into expected products (including what type, how many and when they will be generated) and visibility into downlink windows which are essential for knowing when data is expected on the ground.

Generated product information includes product metadata and reconciliation assessment results. This information is key to identifying data based on a variety of parameters and provides the mechanism to account for lost data or schedule re-attempts to locate missing data.

Knowledge of *system resources* is necessary for the automated system to a) know which servers are available to handle work, and b) distribute work to idle servers. Similarly, the automated system must have access to task status information in order to a) monitor processing to determine whether a failure has occurred, b) determine subsequent processing, and c) allow for the possibility of interjected directives from humans.

The Design

The key to this design's capability is the use of a database to drive the engine of the product generation and reconciliation processes. This solution uses the database as a persistent, ongoing repository of both static and dynamic information. The database is organized in related tables used by the controlling processes to prepare, schedule, initiate and monitor the data processing tasks, as well as to pass data between tasks and maintain their status and history. A set of software servers distributed across multiple platforms initiates and executes processing based upon the information residing within the database. This design enables the servers to startup, shutdown and restart safely, and to determine uncompleted work based on the information in the database tables, thereby achieving stateless servers. This design minimizes

corruption of the system in the event of a failure. Because the servers primarily communicate through the database, throughput is optimized as processing is distributed across multiple machines. Lastly, and perhaps most importantly, the static information in the database can be re-configured so that new tasks or processing can be added, or processing parameters can be modified without disrupting or restarting the system.

Server Functionality

Five types of software servers work together as the engine to control, monitor and maintain performance for the product generation and reconciliation processes: the Request Analyzer, Scheduler, Control, SubControl and Post Request Processor. These servers are described below, and are also illustrated with their database interactions in Figure 2.

The Request Analyzer ingests predicted instrument and downlink pass event information into the database. This server is responsible for defining all processing jobs, and splits the jobs along project and instrument boundaries as necessary. Recall that a "job" is a prescribed work product that includes one or more processing tasks.

The responsibilities of the Scheduler server are simple: periodically query the database for jobs eligible for processing, and release those eligible jobs to the Control server.

The Control server is the heart of the automated system. It accepts jobs to initiate directly from the Scheduler server. Next, the Control queries the database for available SubControl servers waiting to process jobs, and passes jobs to the appropriate SubControl servers. The Control is responsible for updating the database with processing assigned to each SubControl, and then the Control monitors the system to ensure that each SubControl server is executing jobs as expected.

The SubControl servers are responsible for executing the tasks needed to accomplish product generation and reconciliation. There may be multiple SubControl servers configured for the system, in order to distribute CPU-intensive task processing across multiple platforms. Upon startup, each SubControl waits to accept jobs ready for processing from the Control server. Next, the SubControl queries the database to identify the tasks and task execution sequence needed to complete the jobs. In addition, the task processing parameters are obtained from the database. Then the SubControl constructs any ancillary parameter files required by the task software, and initiates sequential task processing necessary to complete the job. The SubControl is designed to handle and report error conditions. Furthermore, to protect the system in the event of a

Control failure, the SubControl regularly queries the database and accepts unfinished processing jobs.

The Post Request Processor server performs a capability derived from the design. In order to maintain system performance, the database tables containing dynamic information must remain fairly small. When job processing is complete, the Post Request Processor archives the processing history in a static database table, then removes relevant entries for the job from the dynamic database tables. Also, the Post Request Processor archives data products, report files, log files, processing metadata and parameter files once the job is complete, thus releasing needed disk space for future jobs.

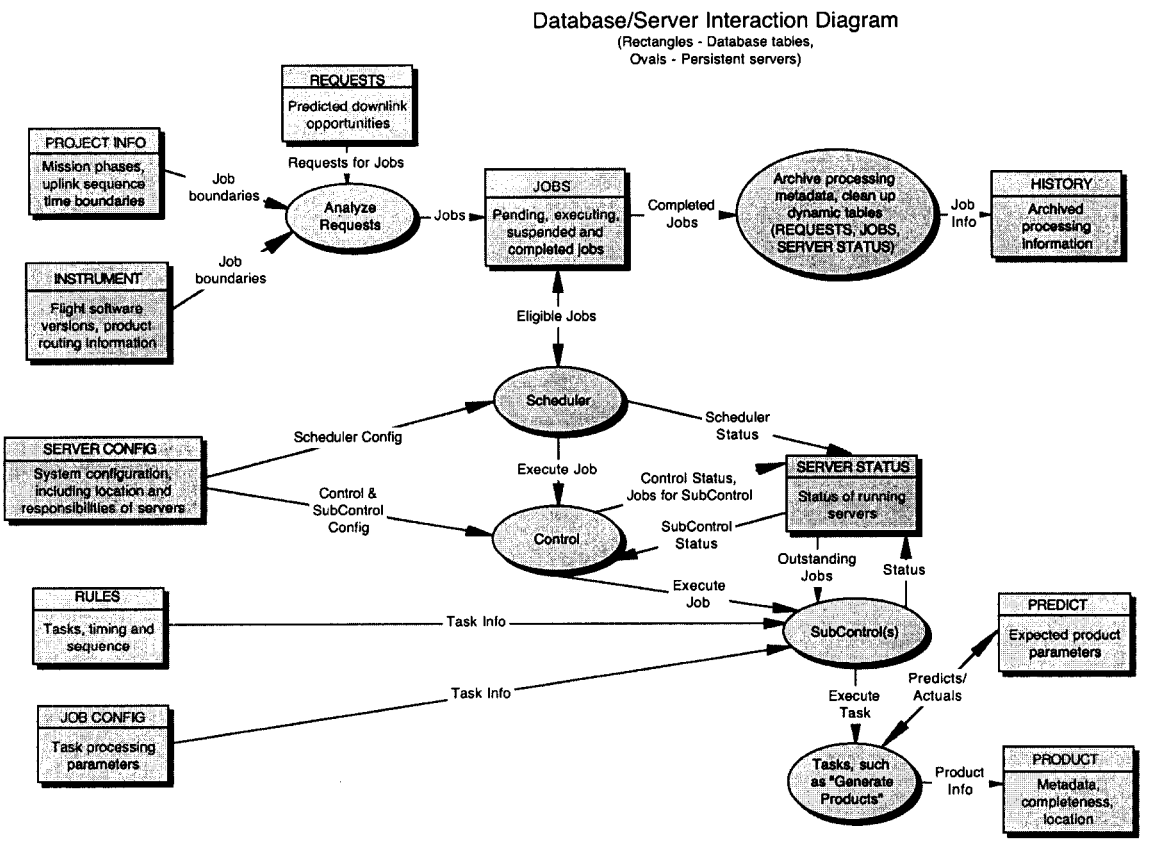

Figure 2. Database/Server interaction diagram

Adaptability

The control system concept is extensible to other missions and other types of data processing once the requirements and processes for that work are defined. By using the database as the

repository for *processing rules/parameters* and *job/task sequencing*, the system is dynamically configurable and adaptable to a variety of other types of ground data processing. Whether processing is sequential and continuously flowing (data-driven) or requires selecting in advance the time when data will be processed (time-driven), once processing rules are defined, the control system is capable of automating production, thus allowing for valuable cost-savings where human resources are concerned.

Conclusion

The control system to accomplish data processing for Cassini ISS and VIMS data products has been implemented and was put into operations in August 2003. It has already proven to be an effective way to minimize operations cost by reducing the amount of workforce needed to schedule, generate and reconcile data products. As the Cassini spacecraft passes through its Approach Science phase and into orbital operations, the automated, self-monitoring system is prepared to deal with an increasingly higher volume of data without requiring an increase in workforce.

When Cassini flew by Jupiter in December 2000, there was an opportunity to gain insight about the level of effort required to support ISS and VIMS data processing manually. Compared to the statistics collected at Jupiter, the automated system is easily capable of handling the expected quadrupling of the data volume experienced at Jupiter, sustaining production indefinitely, and accomplishing that work with 1.5 times the workforce. The reconciliation and assessment reports are more detailed and robust than the manual process at Jupiter. In addition, the data products and reports are delivered in adherence to the timely performance requirements - even the critical optical navigation images, unlike the Jupiter experience when sometimes reconciliation processing lagged months behind the actual data acquisition. Some comparisons from Jupiter include:

JUPITER	SATURN
Manual production	Lights dim production
5,000 data products/month	19,000 data products/month
Prime shift, 2-3 times/week	Round-the-clock production
Limited data accounting	Full data accounting
30+ days to complete reconciliation	12 days to complete reconciliation
1.7 employees needed	2.5 (or fewer) employees needed