

Built But Not Used, Needed But Not Built: Ground System Guidance Based On Cassini-Huygens Experience

Barbara S. Larsen *

Jet Propulsion Laboratory, Pasadena, CA, 91320 U.S.A.

These reflections share insight gleaned from Cassini-Huygens experience in supporting uplink operations tasks with software. Of particular interest are developed applications that were not widely adopted and tasks for which the appropriate application was not planned. After several years of operations, tasks are better understood providing a clearer picture of the mapping of requirements to applications. The impact on system design of the changing user profile due to distributed operations and greater participation of scientists in operations is also explored. Suggestions are made for improving the architecture, requirements, and design of future systems for uplink operations.

I. Introduction

It is common wisdom in the development of complex systems, of which the Cassini-Huygens uplink ground system is certainly an example, that the requirements are fully understood about the time the system is retired. While we anticipate several more years of successful operation in our tour of Saturn, we now have enough experience to reflect on how the design and development would have been done differently if we had known a decade ago what we know now. The focus is not primarily on missteps in software development, since the relevance of many of those would be diminished by changes in technology. It is rather on ways in which people and processes have changed, particularly because of the shift in paradigm due to distributed operations, so that tasks needing software support are different than anticipated. It also looks at the need for software support of small (typically unglamorous) tasks done repetitively by dozens of people. By analyzing needs that were inadequately met, we aim to enlighten the definition efforts of future projects.

Planning (i.e. constructing a conflict-free timeline of activities) is one task which has evolved considerably over the life of the mission. In earlier and simpler missions, planning was done largely by specialists at JPL. In contrast the plan for Cassini was largely constructed in meetings, often by teleconference. The participants included not only engineers but also scientists from multiple instrument teams at various institutions around the globe. They needed not a rigorous analysis of the validity of proposed plan (although this is, of course, eventually essential) but a quick, uncomplicated assessment of the viability of an option. Tools that weren't ready when this early planning was done, that required detailed input, or that ran slowly on dedicated hardware at JPL fell by the wayside. Tools built outside the formal ground system development and not anticipated in the original design sprung up to fill the void.

Another aspect of distributed operations (and of instrument processors onboard) was dispersing the responsibility for sequence (commanding) verification and validation. With over 300 flight rules and a like number of constraints and guidelines, it would have been difficult to accomplish this task without dividing the responsibility. Nonetheless, it is essential that a collective picture of the results be assembled from the pieces of validation before uplink, a crucial undertaking that is still inefficiently supported by software.

This paper first addresses delivered applications in the Mission Sequence Subsystem that are used less than anticipated exploring the reasons that have impeded their adoption. The second section considers tasks for which no formal application was planned. Some of these are now supported by software built outside the formal ground development. Other needs are yet unmet with software.

* Cassini Science Planner, m/s 230-205, 4800 Oak Grove Dr.

II. Underused Applications and Causes for Lack of Use

The Cassini Mission Sequence System has been successfully used to produce over a million commands for the spacecraft. A number of workhorse applications are run day in and day out by engineers, scientists, and science planners to design science pointing, to manage data storage and bandwidth, to convert activities to commands and to validate the commanding. However, as shown in Table 1, other applications within the subsystem are not utilized as extensively as envisioned by the ground system developers. The analysis below considers the reasons that these tools have been less successful in supporting uplink operations and makes suggestions for future systems design.

Table 1. UNDERUSED APPLICATIONS

PROGRAM	INTENDED FUNCTION
APGEN—Activity Plan Generator	Produce a conflict-free timeline of activities
SOA—Science Opportunity Analyzer	Find opportunities in the tour for particular observations. Construct and visualize observation designs.
MSS Utilities CFReader OPS_COMPARE t2t & timeconverter Seq_Review SSFTIME	Human intelligible display of initial/final conditions file Compare sequence product files Time conversions Apply user-defined filters to sequence products

A. Unwarranted Fidelity in Modeling and Simulation

Operation of deep space robotic spacecraft would not be possible without high fidelity modeling and simulation both to ensure achievement of unique science observations and to preserve the health and safety of the overall system. Thus both the science community and the engineering community have vested interests in the availability of tools with high fidelity models. Why not extend this fidelity to planning tools? This fidelity does not come without cost: it extends development times, complicates usage, and slows performance. Strategic planning—the division of time among scientific goals and the negotiation of contested resource—does not require fidelity this extreme. For this integration task, the costs of the “last mile” outweigh the advantages of having a usable product with performance that supports repeated iteration early in the ground system development cycle. Also, the set of characteristics modeled needs to correspond to user concerns for the task at hand. For example, APGEN, Activity Plan Generator, copied too much from engineering simulations of the nuance of pointing design but did not adequately address the points of planning contention such as agreements on secondary axis pointing. It rigorously modeled the format of telemetry recording and downlink, but largely ignored the contention for bandwidth and storage.

For maximum system efficiency, a different performance versus fidelity trade-off should be applied to the planning tools than to the engineering validation tools. During the early planning processes evaluating the validity of the design will be a central concern requiring frequent iteration to resolve both resource and flight rule conflicts. For this purpose, a faster tool with close approximations is more useful than one with a level of fidelity that is too slow to run repeatedly. While it is true that compromises in modeling and simulation fidelity will result in some undetected commanding errors, most can be driven out with lower fidelity tools. Fewer iterations through the high fidelity models are then necessary.

As mentioned previously, the Cassini uplink process is built on successive refinement from plan to preliminary implementation to final validation and polishing. Each of these phases employs tools that require as fundamental input the specification of the intended spacecraft activity. It is obviously inconvenient to re-specify the same activity for each phase of the sequence development. Unfortunately, to avoid this re-entry, the sequence system design called for using the output of the planning process as the input to the command generation. So, for example, the activity planner provides as output that which the sequence generation tool expects as input. The Science Opportunity Analyzer provides as output that which the pointing design tool expects as input. The drawback to this approach is that the planning tools then require complete specification of all the parameters necessary for command generation, which significantly impedes ease of use and requires all planners, including scientists, to fully understand the

parameters of detailed design. To the extent that development dollars allowed, this was mitigated in SOA with different modes of operation. Simplification of planning tools would have made them easier and faster to use and thus more popular. This could have been accomplished by hiding parameters that have a nominal value for a large percentage of instances, such as turn rates and acceleration or turn margin, or that represent optimization rather than basic design. Other mechanisms could have been employed to manipulate the product formats at the interface for further refinement in the detail driven tools.

B. Timely Delivery

For essential tools whose use is required by the project, user response to development schedules that slip or lag needed use is limited to grumbling. They will be obliged to endure bugs, to put up with awkward workarounds, to live without desired features. It cannot be overemphasized that this pattern does not apply to tools whose use is not mandated. Largely because of management decisions to delay the start of development, Cassini planning tools such as Science Opportunity Analyzer were not mature when intensive integration of science activities started in 2001. From the users' perspective these tools lacked too many features to adequately support the analysis required or were too buggy to produce reliable results. As described in the second part of this paper, the planners devised other ways to get the tasks done. They did not, however, revert to the formally delivered tools as these were improved and matured. This is a critical consideration for schedule development of auxiliary tools—if the tool is not available and usable early in the period when needed, there is significant risk that even its eventual adoption will be impeded. If the investment necessary to deliver on time is not possible, late push-up may be useless to compensate.

C. Platform Accessibility

For previous flagship missions, most computing was done on project computers in the JPL operations area. No one other than the ground data system engineers had to worry about platform choice. Users did not generally have experience with alternative platforms nor were they likely to field requirements like cost, portability, or available software environments (such as matlab or IDL or Java 3D). In contrast, the scientists planning activities for the Cassini tour had been exposed in other contexts to hardware that was, in most respects, superior to that in the Cassini ground system. Official software was delivered on Sun workstations that were already several years old when intensive science planning kicked in and that had performance inferior to individuals' personal computers. The supported platforms were too expensive to replace or to procure at secondary sites. The software also could not be operated from the meeting venues where the analysis of opportunities and computations in support of resource allocation were needed in real time. That supported platform was indeed a significant factor in tool adoption was borne out by a substantial increase in demand for the Science Opportunity Analyzer when the decision was finally made to deliver a version for Windows and for Linux. The drivers such as cost and institutional support that steer the selection of a platform for project mandated tools are not necessarily key factors in platform choice for optional support.

D. Awareness and software accessibility

The Cassini ground system development budget did not stretch to cover user friendly delivery and usage documentation. User support was also limited. As a result, many of the remote users are unaware of the availability of utility programs. Development of additional utilities was discouraged by the difficulty of making them accessible in a user friendly fashion.

III. Tasks with Late or Insufficient Software Support

While essential software support for the uplink process is provided by the Cassini ground system, some tasks would be easier with support other than that provided in the ground system design. This is particularly true of those tasks that are fundamentally different in either function or process from previous missions either because Cassini itself is different or because of distributed operations. Two categories of such tasks are illustrated in this section. One is needs which still have inadequate software support and thus require additional staffing to accomplish. The other category describes needs that were met with tools developed outside the formal ground system development. The successful impact of these on uplink processes suggests the need for reconsideration of the ground system architecture to allow adoption into the system of tools developed outside it. Some suggestions for future ground system design follow in the next section.

A. The Current Plan, Viewable Anywhere

The initial plan for spacecraft operations at JPL and science operations at team sites with limited interfaces between was supplanted by a far more complicated scheme early in the mission. Fortunately for the success of distributed operations, Cassini's need for expanded avenues of communication was supported by the rapid emergence of email, teleconferencing and the World Wide Web. Even this was not enough given multiple concurrent planning processes on top of multiple sequence development processes whose participants were intersecting groups of scientists and engineers scattered around the globe also tasked with monitoring the executing sequence (not to mention producing science results). The need to coalesce the overlapping threads of planning into a single, up-to-date, accessible, non-redundant view of the detailed plan was met by creating a central repository of planned activity using an Oracle database with a web-based interface. While the timing of ground system development did not support this approach for the initial suite of applications, the degree to which this architecture eliminated concerns over platform choice and differing installation environments at distributed sites strongly recommends it to current distributed architectures.

B. Summary View of Distributed Analysis Results

The complexity of Cassini compounded by the complexity of distributed operations demands that numerous avenues of sequence verification and validation are pursued through several repetitions. Sequence checks are made by members of the science integration teams, science planners serving as virtual team leads, spacecraft team engineers, sequence leads, the instrument teams and the simulator team. They employ a plethora of software and processes, using sequence subsystem tools, spacecraft subsystem tools, mission planning tools, data base internal checks, scripts, team tools, spreadsheets, and hand inspection. Since each sequence passes through at least five ports of integration and validation, these checks are repeated several times. While this process is inherently complicated, the complexity of the system offers few alternatives.

From my personal perspective, the largest unmet need is for software and automation support of the re-integration of the results of the decoupled validation. The same issue, error, or problem is often discovered by multiple people (which is a good thing) via multiple mechanisms (which is safer but not more efficient than a single catch point) at multiple times. Although by design the number of actual problems should be less at each checkpoint, the large majority of checks err on the side of reporting something which may or may not be a problem to reduce the possibility of missing a true problem. Therefore, even a perfect sequence might have many repetitions of multiple reports of the same problem.

Sequence validation would be much improved in both accuracy and efficiency with a software mechanism, perhaps a database, for collecting, correlating, and recording the resolution of detected problems. The analysis for such a tool would have resulted in a systematic enumeration of the errors to be trapped. This comprehensive list would have provided a central place for adding additional checks as new needs were uncovered. The collective ground system designers would have agreed (although perhaps not easily) on an intentional assignment of the checks to appropriate people, their tools or processes, and phase of sequence development. The design alone would have imposed some uniformity on error reporting that would simplify further handling both by software and by human review.

C. Distributed View of Central Analysis from Distributed Inputs

When the conflict-free timelines are integrated into a sequence, refinements are sometimes necessary. The drivers for these revisions are typically changes in the spacecraft state (especially trajectory) or in the ground environment due to changes in the DSN station allocation. For example, relatively late in the sequence development process, when the DSN allocation is relatively stable, it is usually possible to release data volume margin and allow the collection of more data. This requires negotiation among the various instruments for the available data volume, cleverly called sponge bits. Additional bits can be made available as long as there is either bandwidth to downlink them or storage on the SSR. Efficient allocation thus requires knowing fairly precisely how many additional bits each instrument would like over each period. If each team simply requests what they would like without knowing what other instruments are requesting, the demand is always higher than the supply. A mechanism for nearly immediate analysis based on adjusted bit requests from multiple teams would facilitate distribution of all available bits.

The cutting edge of web-based applications is now exploring pages that allow simultaneous, interactive input from multiple users. As this matures, one could envision web-based applications for spacecraft uplink development wherein each of the distributed teams specifies their desired resource allocation, the central processing tool evaluates the net resource consumption based all user requests, and nearly immediately reports in a view visible to all those

supplying input. Immediate iteration would then be possible so with no more work than the current process more optimal resource allocation could be completed.

D. Automation for mechanical, trivial, repetitive, low-level tasks and process flow

Perhaps because automation of spacecraft operations has vigorously pursued accomplishing even cognitively challenging tasks or perhaps because the Cassini software development budget was stressed just by its major applications, many of the menial steps in sequence development remain manual. A personal aggravation is the hand calculation of duration by subtracting two UTC times. As another example, recall that there is no central mechanism for capturing and reviewing sequence critique. To make the various reports available to all sequence contributors, the results from each of the reviewing parties are posted to a web page devoted to that sequence. Since each sequence goes through at least five cycles and is reviewed by fifteen or more analysts, the posting of that material to the web page is an example of a trivial but frequent task that ought to be automated.

E. Distributed Planning Without Access to Ground System Workstations

After the tour design was available, a science plan representing the allocation of time, pointing, power, and data volume (all of which are over-constrained) was needed. In the past this might have been done by a select group of scientists working together at JPL; Cassini chose to use discipline-focused groups of scientists in order to share the responsibility and divide the workload.¹

The interaction of these groups was different from previous experience. Not only were they not co-located, working largely by telecon, but they were also not at project workstations. For this interactive but distributed integration, they needed fast, easy to use, planning tools accessible from meeting rooms to provide the kinds of information cited in the table below. As described in the first section of this paper, the project planning tools did not satisfy these requirements especially early in the process. Various participants in this effort filled this hole with informal software tools aimed at supporting integration decisions.

Table 2. INTEGRATION NEEDS SUPPORTED BY UNPLANNED SOFTWARE

FUNCTION	SOURCE
Turn Time Estimation	CTV—Cassini Turn Visualizer Brad Wallis, Science Planner
Visualization of Geometric Constraints	i. CTV—Cassini Turn Visualizer Brad Wallis, Science Planner ii. Forbidden Zone Plot Frank Crary, CAPS Scientist
2D Visualization Fields and Particles Opportunities	JCSN MIMI Team (access eventually limited to APL)
Trajectory Based Opportunity Data: Range, phase angle, etc.	DIGIT Dave Seal, Mission Planner

These tools made significant contributions to the integration process. During the intense rush of ground software development to support tour operations, independent development conveyed significant advantages over formal ground system development. In particular, it allowed for very rapid redelivery in early phases when new features are added at a rapid pace and when bugs need to be fixed quickly. (Very rapid in this context means multiple times per month in contrast to the formal uplink subsystem, which was redelivered several times per year). As these applications also achieved maturity and the delivery pace slowed down, the risks associated with circumventing configuration management and regression testing were no longer justified by the cost of rigor. However, they have remained separate from the overall ground system design. Future missions should consider a ground system architecture which explicitly allocates requirements that cannot be met, whether for functionality or delivery speed, to a temporary, more flexible subsystem which does not impose the burdens associated with critical path software but affords some of the advantages of uniform delivery.

F. Electronic Signature

Some critical decisions including flight rule waivers and sequence change requests require consent verified with signature of impacted parties. Since the decision maker for the distributed teams is generally not at JPL and since the time-frame for obtaining the signature is relatively brief, the lead responsible relied on fax transmission of the form for signature with return fax of the signed paperwork. In addition to incurring costs, the reliability of the approach is

incongruous with the importance of the communication. Ownership of the paper was an issue. Prototype attempts at electronic signature were overly ambitious for the development effort available and did not realize user adoption. Very recently, Cassini adopted the JPL multi-mission CM system, which provides among its features electronic signature. It has, however, suffered from the curse of late-adopted software: the advantages are significantly degraded by the pain of re-training, glitches at introduction, and mismatch of available features to expectations.

G. Proliferation of Products in Non-Standard Formats

The JPL portion of the Cassini ground system itself consists of multiple systems, and this complexity is amplified by association with the distributed operations systems of the instrument teams. In system design, the analysis of interfaces focused on connectivity. Process definition concentrated on commitments to product delivery with considerably more focus on content than format of the products. In the absence of detailed interface specifications, many products created by multiple teams have as many formats as originators. An example would be the reporting of the team reviews of sequence products for instrument flight rule satisfaction, correct power modes, and other validation checks. No format is mandated and there is therefore no uniformity. Some teams provide their report as an Excel spreadsheet, but since other teams do not use Microsoft Office these must be converted to PDF for review by other teams. With some checklists as spreadsheets, some as email messages, some otherwise captured, doing anything automated with the raw reports would first entail considerable effort in extracting relevant data. These problems suggest the wisdom of rigorously defining data interfaces even when a software interface is not envisioned at design time.

H. File Access and Management for Uplink Tool Use

The distribution of pointing design and sequence generation among the instrument teams meant that far more users were operating sequence system tools than in previous missions. Furthermore they were dispersed over many installation locales some not under control of the project. Also, since the sequence development time is significantly longer than the sequence duration, multiple sequences each of which have multiple ports are in work at the same time. For these reasons, complaints about initializing tools with the correct inputs were frequent. Some of the remote sites had difficulty obtaining and maintaining complete file sets. Some errors in file selection were observed. Also the process represented inefficient distribution—the identical effort to get the input files and construct configuration files was repeated by dozens of users.

This problem is an example of an unmet need recognized late in ground system development that was met with flexible design approaches. Since most pieces of the uplink system were already in use, a glueware approach was used that built around the existing applications rather than significantly altering their code:

Highlights of the restructured system include the following: (a) an electronic version of the master list of correct ancillary files updated at each stage of sequence development, (b) a script validating the master ancillary list construction and verifying that named files exist, (c) a translator to make a web page reference from the master list, (d) a new tool which enables single point construction for all applications of sequence specific configuration files based on the master list of correct files, (e) alterations to the configuration files themselves and to already existing application wrappers so that those sequence specific specifications can be used, (f) a configuration file naming convention that allows easy recognition of the appropriate configuration file for the work at hand, (g) expansion of the project database to include the master ancillary file lists and the configuration files, (h) a logical file structure that provides the application programs a single view of the ancillary files while allowing different implementations in various subsystem architectures, and (i) active maintenance of the ancillary input files within the ground system in a manner expected by the applications.²

IV. A Flexible Ground System Architecture

Increasing the acceptance of the underused applications would largely have required conventional software development improvements, especially these: defining better requirements, starting development earlier, and spending more money. Ongoing efforts at JPL including SQI—Software Quality Improvement—are addressing needed enhancement in software management and development practices.

However, in order to intentionally exploit the success of ad hoc tools, the ground system architecture must become less uniform both in categorization of software and in development process. Although certain popular models for software development suggest that process variation should be stamped out, the suggestion here is to allow for a more tolerant model. The proposal is not to reject the model of improving overall development with mature, controlled processes or to go back to the heroic development model. Rather it is to acknowledge that novel systems as complex as Cassini Uplink which are constrained by both schedule and budget and for which operations processes are evolving in parallel with software support will miss or misdefine essential functional capabilities. It is to acknowledge that the exceptional people involved will provide unanticipated solutions filling these gaps, but that

those solutions may not conform to standard development practices. The ground system would be enriched by integrating these tools and providing system advantages such as configuration management, hosting and delivery, documentation, regression testing, and user support at tool maturity. Anticipation of these inclusions would mean incorporating into the development processes a strategy for risk assessment, management of added interfaces, and provision of the system capabilities just mentioned for the unplanned tools.

V. Conclusion: Ground System Recommendations

Based on the analysis and examples detailed in this paper, we recommend that ground system designers for complex spacecraft with distributed operations consider the following guidelines:

1. Design the ground system to allow incorporation of unanticipated applications.
2. Recognize that a uniform trade-off of rigor versus development speed may not be suitable for all supported tasks.
3. Recognize that distinct user communities weigh application tradeoffs (such as fidelity vs. performance) differently.
4. Realize that integrating results is as essential to distributed operations as dispersing tasks.
5. Exploit technologies for non-homogenous systems such as glueware and adaptable interface technologies.
6. Look for cost savings in automation of mechanical, trivial, repetitive, low-level tasks.
7. Support tasks in the environment in which they will be performed. A uniform hardware architecture may not be feasible if this goal is taken seriously.
8. Apply rigorous system engineering practices such as interface definition to processes as well as software. The data flows will then support later automation with software.

Acknowledgments

This work was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

Dr. Kari Magee of JPL provided editorial support for this paper.

References

1 tbs
2 tbs