Re-Engineering JPL's Mission Planning Ground System Architecture for Cost Efficient Operations in the 21S1 Century

Jess Fordyce

Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, California, USA
91109-8099
FAX: (818) 393-9900
E-mail: jess@rush.jpl.nasa.gov

ABSTRACT

Over the last several decades, engineers at the Jet Propulsion Laboratory have developed a vast array of analytical tools to design missions to Earth orbit, the moon, sun, planets and various other bodies in our solar system, and beyond. Due in part to the unique objectives and requirements of each new mission, many early tools were developed in an ad-hoc environment to support the. immediate needs of specific projects, with little thought given to developing an overall system architecture, maintenance, 01' reuse by subsequent projects. Nonetheless, the tools that emerged began to represent a rich heritage 01 mission design experience and capability.

In rc.cent years, advances in computer hardware, modern programming languages, and the need for faster and more cost efficient operations for small missions have highlighted the need to streamline, consolidate, and generalize JPL's mission planning software. Realizing this, JPL's Multi-mission Ground Systems Office and Project Design Center have jointly undertaken the task of transforming existing "legacy" software into an integrated, general purpose, multi-mission tool Set.

This paper summarizes ongoing efforts at JPL to re-engineer the mission analysis segment of the mission planning ground system architecture. Issues addressed include: developing a partnership between software developers and users, developing a consensus based architecture, evolutionary change versus revolutionary replacement, reusability, and minimizing future maintenance costs. The status and goals of new developments are discussed, and specific examples of cost savings and improved productivity are provided.

The work dc.scribed in this abstract was performed by the Jet Propulsion Laboratory, California Institute of Technology, under contract to the National Aeronautics and Space Administration.

1. HISTORICAL OVERVIEW

In keeping with JPL's charter to explore the solar system, a variety of miss ions have been la unched to various locations including Earth orbit, the moon, [lie planets, heliocentric obit, and to small bodies such as comets and asteroids. Mission objectives have included initial flyby reconnaissance, detailed orbital mapping, satellite tou rs, and surface landings to name a few. Until recently, activities have centered on a relatively small nu mber of unique, first-of-a-kind projects, and most of the mission analysis software was created on an as needed basis by mission desig n engineers to solve. thei immediate problems. Because of the unique nature of each mission, little thought was given to reuse by subsequent projects. After all, how often was NASA likely to send a Viking style. lander to a planet's surface, conduct a "grand tour" of the solar system like Voyagers 1 and 2, oi explore an outer planet gas giant and it's moons like Galileo?

Computer Systems have also changed greatly over the years. Before switching to mostly UNIX based systems, engineers developed and used software on early platforms such as the IBM 7044, 7094 and finally Univac mai nframes. By comparison to modern engineering workstations, these earlier systems were much more di fficult and costly to use. Frequently, the charges for computer time could greatly exceed the engineering labor rates. As a result, the mission analysis software

that emerged favored minimizing computer run time for both development and operations, at the expense of ease of use, reusability, and therefore engineering labor charges.

2. CURRENT ENVIRONMENT

In the early 1990's JPL's Multi-mission Ground Systems Office (MGSO) began the Multi-mission Software Transition Project (MSTP) to port Univac based software to Sun and Hewlett Packard engineering workstations. The intent of the effort was to preserve much of the legacy software developed by prior projects for use by future missions as the laboratory made the transition from mainframes to micro computers. As work progressed in the mission planning area, it became clear that much of the mission analysis software being ported would not be directly usable by other missions. For example, some software developed for use by the Galileo project to Jupiter and its moons presupposed that the mission was in orbit around Jupiter, and that the spacecraft would fly by the four Galilean satellites. It also modeled the orbits of the Galilean moons as perfectly circular, with no inclination with respect to Jupiter's equator. Clearly, this software was developed under the assumption that saving computing cycles was more important than obtaining a high fidelity result, or to be useful to any subsequent project. Realizing these shortcomings, the task objectives expanded to make the software as applicable to other missions as reasonably possible. This included general features such as making the central body for spacecraft orbits a user input, and causing programs to access standard planetary and satellite ephemerides for the bodies of interest instead of using a hard wired approach.

Other forces were at work during the early 1990's as well. NASA's out year budgets began to show increasing reductions, and it became clear that the days of costly "flagship" missions were over. JPL management repeatedly communicated the top level direction from NASA Headquarters to the work force: conduct missions that are faster, better, and cheaper. Therefore, the emphasis switched to performing a larger number of smaller, less expensive missions. This further highlighted the need to create an institutionally supported, multi-mission software base, since it was clear that the many small missions would not have sufficient time or money to develop their own tools. Perhaps more importantly, the tools would need to be easier to use, thereby allowing project analysts to spend their resources designing a mission rather than struggling with difficult, uncoordinated, and error prone software. By addressing the problem in this light, it became evident that improving ease of use satisfied "faster", integrating applications and streamlining information flow to reduce errors meant "better", and the combination of saving time and reducing errors inherently satisfied "cheaper".

3. FORMING PARTNERSHIPS - USERS, DEVELOPERS, MANAGERS, SPONSORS

As the MST effort to port software from the Univac to UNIX systems neared completion, the budgetary and technical considerations outlined in the preceding sections became increasingly clear. Users and developers of mission analysis software had long known that the system needed to be re-engineered, but the MST effort funded by the Multi-mission Ground Systems Office helped to bring the issues to the forefront. As a "grass roots" initiative, a special meeting of the Mission Design section was called in the fall of 1993 to provide users and developers a forum to discuss their views on the current status of mission analysis software, and to suggest possible courses of action.

The special section meeting was well attended, and it soon became clear that there was a ground swell of support to resolve a number of problems. Meeting members identified six main topics to address including: software standards, organizing known software tools, introducing modern computer graphics techniques, improving use and maintenance of modern computers and networks, coordinating related software efforts to maximize shared benefits, and finally, how to obtain institutional support to build a greatly improved multi-mission software system. Leaders were selected to represent each of the six areas, and a series of panel discussion meetings were held over the next several weeks to discuss issues and form recommendations for a plenary session. Line managers made a strategic decision not to take part in the panel group activities, in order to avoid dominating discussion or discouraging a frank exchange of views.

A follow up section meeting was scheduled by the section manager for the panel groups to report their findings, and make recommendations. Ultimately, all recommendations were accepted in principal, including the recommendation to reorganize the section to create a group to

consolidate related software activities. Although the final recommendations were very close to those of the few people who instigated this distributed effort, significant value was added by involving a large number of concerned people. In the end, users, developers, and managers had a shared vision of what should be done, and how to proceed.

After forming the new group, early benefits were realized as members gained greater awareness of related activities, and how tasks could be coordinated to reduce unnecessary duplication. However, there was still no defined architecture for the desired mission analysis system, nor were there dedicated resources to create one. In order to devise a workable plan, developers realized the need to evaluate the section's current software inventory, and to identify the missing components. This information could then be used as a starting point to define the requirements, which would then be modified based on the inputs of experienced mission analysts. The Mission Design section's burden account and JPL's Project Design Center (PDC, funded by the laboratory's internal burden structure) provided a small amount of funds in fiscal 1995 to carry out this survey effort.

Mission design personnel conducted the survey, and the results were discussed in joint steering group meetings involving users, project representatives, developers, and managers. Participants discussed the status of current tools, and how the system could be improved. Eventually, the teams reached consensus on classifying existing applications in one of three categories: "core" capabilities, utilities and mostly redundant programs that could be made into library functions or included as options in a core program, and a list of programs to ignore, and no longer track. Twenty programs qualified in the "core" category. This did not necessarily mean that these were the programs to keep, or that they even existed in the desired state; rather that the programs represented the kind of capabilities desired. Another sixteen programs fell into the utility or mostly redundant category, and the steering group classified roughly another 200 programs to no longer consider, since they were either too mission specific, or obsolete. At the time of the survey, only a handful of the programs in any of the categories were funded for development or maintenance at any level.

During the summer of 1995, MGSO issued it's annual call for continuous improvement proposals. Using the results of the survey as a starting point, the steering group reconvened to develop proposals to begin implementing a new mission analysis architecture, and to satisfy the specific objectives of the proposal guidelines. Three proposals emerged, reflecting the steering group priorities. The proposal objectives were to: develop new software to automate the production of standard trajectory products; create a three dimensional mission plan visualization capability; and to integrate mission analysis software into a synergistic system. Performance objectives were scoped to fit within budgetary guidelines, and specific references were made to coordinating proposal efforts with other JPL sections, divisions, and external organizations such as Purdue University, the University of Minnesota, and Lewis Research Center. Developers, project managers, users and line managers discussed and iterated the proposal objectives, and letters of endorsement from the involved parties were attached to the final proposals.

Ultimately, all three proposals were selected and funded for fiscal year 1996, which was especially remarkable since no mission analysis proposals had ever been selected in prior years. The Project Design Center also agreed to provide considerable funds to help fill in the gaps of the more specific MGSO proposals. Finally, the mission analysis software effort was funded to implement the plans developed in conjunction with their customers. The remainder of this paper describes the architecture that has emerged, reports the status of ongoing efforts, benefits realized, and plans for future efforts.

DEFINING KEY MISSION ANALYSIS FUNCTIONS

The first major step in defining an architecture was to clearly establish the major functions of mainstream mission analysis software, as graphically illustrated in Figure 1. The first column represents trajectory generation functions, where mission designers develop a flight path to satisfy mission objectives. The second column represents analysis to determine the timing of various events such as maneuvers, science sequences, solar occultations, and down link view periods, and the third column is used to perform general geometric calculations.

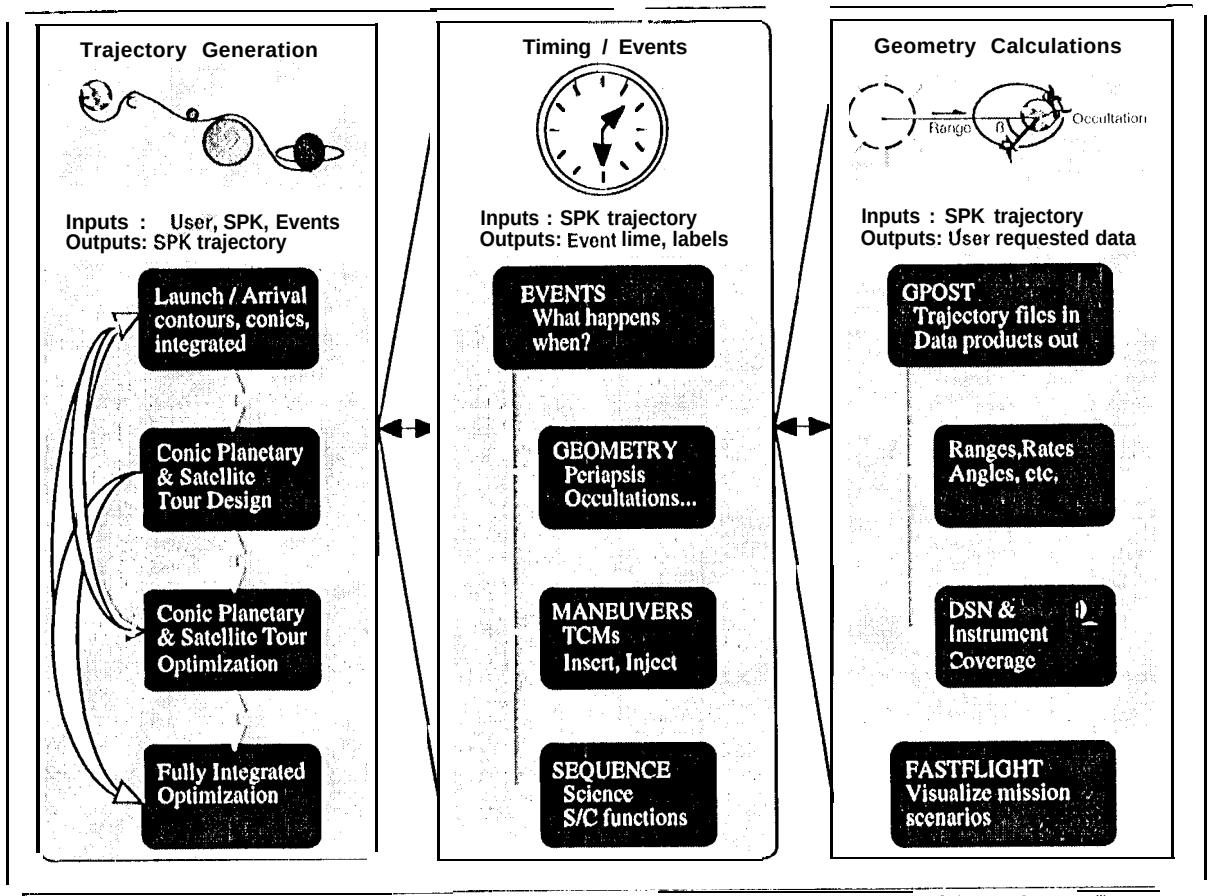


Figure 1: The mission analysis process can be diagrammed as a series of iterative refinements, suggesting a hierarchical, modular structure of trajectory generation, events, and geometric calculation.

The Cassini mission to Saturn exercises much of this capability, so it will be used as an example of how the functions are related. A designer may begin by examining launch / arrival contours commonly referred to as "pork chop" plots to find minimum energy transfers from Earth to Saturn. After determining that a direct trajectory to Saturn requires an unacceptably high launch energy, the analyst may consider using one or more gravity assists to reduce the propulsive requirements. Launch energy contours to other planets or multi-year Earth phasing orbits can be examined to search for alternate opportunities. After determining that a multiple gravity assist trajectory to Saturn might exist, the analyst might use. a conic trajectory optimization program to test the hypothesis. In this Gist., the trajectory involves launch from Earth, two Venus flybys, another Earth flyby, a Jupiter flyby, and then entry into orbit around Saturn. The simple conic program requires only a rudimentary initial estimate of the times of the planetary flybys in order to optimize a trajectory to minimize the total mission Av.

Once a candidate orbit is found, the analyst can further optimize the trajectory subject to a variety of mission constraints such as launch period requirements, minimum flyby altitudes, and many Others. Finally, the trajectory can be refined using more sophisticated, fully integrated acceleration mode.]s (n-body gravitational attraction, higher order gravity harmonics, solar radiation pressure, atmospheric drag models). Ultimately, a viable flight path to Saturn can be found.

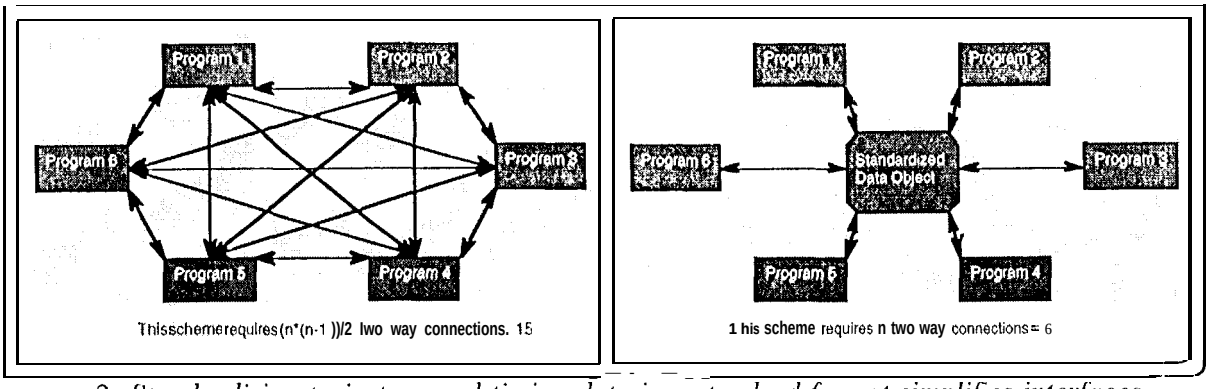
After the spacecraft arrives at Saturn, it enters an orbit that will allow many flybys of Titan and the other Saturnian moons. A conic program is used to search for natural targets of opportunity, and adjust satellite flyby parameters to allow multiple gravity assists to steer the spacecraft towards its next target while. minimizing the total Av necessary to accomplish the mission Many possible satellite (our scenarios exist, and options are chosen to satisfy science goals, as well as spacecraft and mission operations constraints. These constraints are analyzed using software represented by the Timing / Events column, and the Geometry Calculations column After the tour is selected, it, too," can be optimized using the fully integrated trajectory optimizer. At any point in the analysis,

pre-determined constraints or unforeseen circumstances may require the analyst to recycle he entire effort, and iterate the whole process many times until a satisfactory solution is found.

RE-ENGINEERING PHILOSOPHY AND CONSTRAINTS

STANDARDIZING INTERFACES

One of the desired features of the new mission analysis system is seamless exchange of relevant information between related programs. One approach would be to simply write "glue functions", or specific utility programs to take the output of program "A", massage it into the desired format, and then write it out as an input file to be read by program "B". Due to the iterative nature of the mission analysis process, it might also be necessary to create yet another utility to transfer information from B back to A. Given a sufficiently small number of programs, this scenario might be workable. However, as Figure 2 demonstrates, the number of two way interfaces increases rapidly as additional programs are included in the system. In fact, the number of connections scales by (n(n-1)/2) where "n" is the number of programs in the system. Because of this, a standardized data interface was developed based on the JPL's Navigation Ancillary Information Facility "SPICE" kernel capabilities. SPICE is an acronym where each letter represents a significant, unique capability. In the scope of this discussion, the mission analysis system chose to use the "S" kernel (reads and writes SPK files) to represent the trajectories of spacecraft, planets, and satellites, and the "E" kernel to organize relevant events. One of the benefits of the SPK system is that the orbit information can represent a simple conic, or an integrated trajectory with many flybys. Programs that use this capability can then create or read trajectory information, and issues such as changing central bodies for spacecraft orbits are handled automatically. Using this scheme, the number of two way interfaces is simply "n", so it is far easier to maintain an integrated software set.



gure 2: Standardizing trajectory and timing data in a standard format simplifies interfaces.

MODULARIZING MISSION ANALYSIS FUNCTIONS

Examination of the software that remained on the supported list after selection revealed significant overlap of key functions. Figure 3 shows six unique programs that were developed to numerically integrate trajectories, given a specific mission scenario. In some cases, a program has a unique physical model lacking in other programs. For example, CASSAR had a comet outgassing pressure model to use while orbiting a comet, the Planetary Observer Program (POP) set was customized to model terrestrial planets using higher order gravity harmonics, while LUNTRJ and the Goddard program SWINGBY included gravitational attraction of the Earth, moon, and sun. The only program funded for development or maintenance at any level was FAST which has been a core program for nearly 25 years. Between the time the group formed to consolidate efforts and the proposals were funded, the CATO program (Computer Algorithm for Trajectory Optimization) was being developed to optimize fully integrated trajectories involving multiple flybys. CATO was CICERO for Cassini to replace the MOSES multi-conic trajectory optimization program set used by Galileo for the satellite tour at Jupiter.

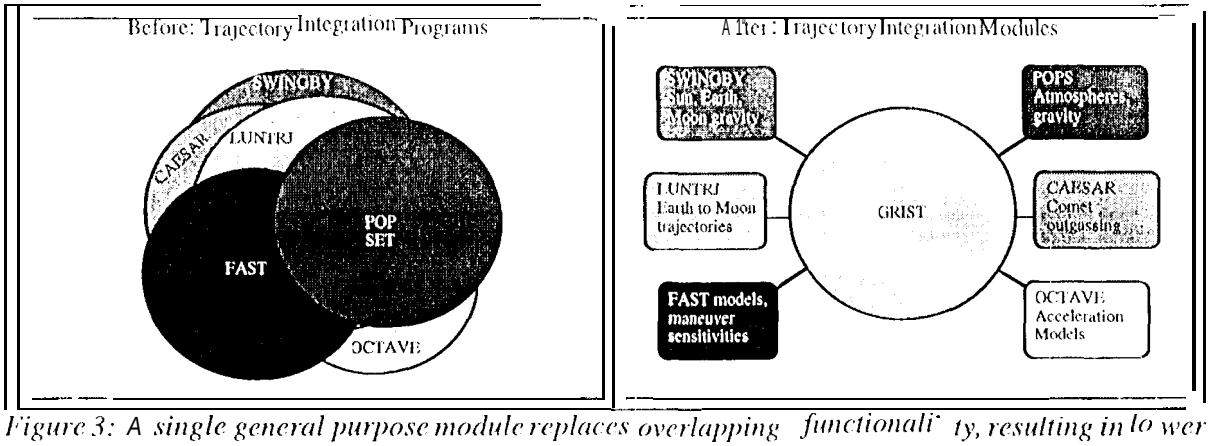


Figure 3: A single general purpose module replaces overlapping functionality, resulting in lower maintenance costs.

Initially, developers tried to adapt FAST to be the trajectory integration module used in CATO, but this plan was abandoned after considerable effort, since the desired features in FAST were distributed all through the complex program. Therefore, a new trajectory integration module called GRIST was written in object oriented FORTRAN 90 to be both a stand alone trajectory integration program, and a module to be called by other programs. Since GRIST was written to be useful to any class of mission, it was designed to integrate trajectories modeling an arbitrary number of gravitating bodies (any combination of sun, Earth, moon, planets and their satellites, comets and asteroids), gravity harmonics, solar radiation pressure, nine different atmospheric models, and other acceleration models as desired. Once GRIST was designed to solve the general problem, it represented a superset of the capabilities of the programs it replaced. If a desired acceleration model is missing in Grist, such as a specific comet outgassing model from CASSAR, it can be included as another term on the right hand side of the differential equation of motion.

The remaining mission analysis programs have other similarities, in addition to trajectory integration. Most programs calculate geometric quantities to help design trajectories, determine instrument coverage or telecommunication links to a ground station, or some other purpose. In existing stand alone programs, a large percentage of the code is typically devoted to keeping track of complicated data, vectors, and coordinate systems to make these calculations. Another function common to many programs is the need to search for and keep track of events that occur, such as propulsive maneuvers, science viewing opportunities, and entry and exit of solar occultation. In some cases, these events are unique to the mission or arc. specified by the user, but many events are calculated based on a geometric quantity (such as occultations and ground station view periods). Realizing this, two more modules were identified as core features, an events finding module (EVENTS), and a geometry calculation module (GPOST - General purpose post processor). In both cases, the modules were designed to be used either as stand alone programs or as part of another program.

Figure 4 shows a top level view of how the pieces fit together. The left-most block represents a generic mission analysis program, which performs a trajectory design function. Using satellite tour design as an example, the application will model a conic obit around a planet, and perform calculations to target multiple flybys of the planet's satellites. As the program is used to build up a tour, events such as maneuvers and satellite encounter times begin to accumulate. Geometric quantities are also calculated, and may be related to viewing geometry of the science instruments, for example. This information helps the mission analyst design the tour to meet the mission objectives. Ultimately, the program will produce some. sort of graphical or tabular output.

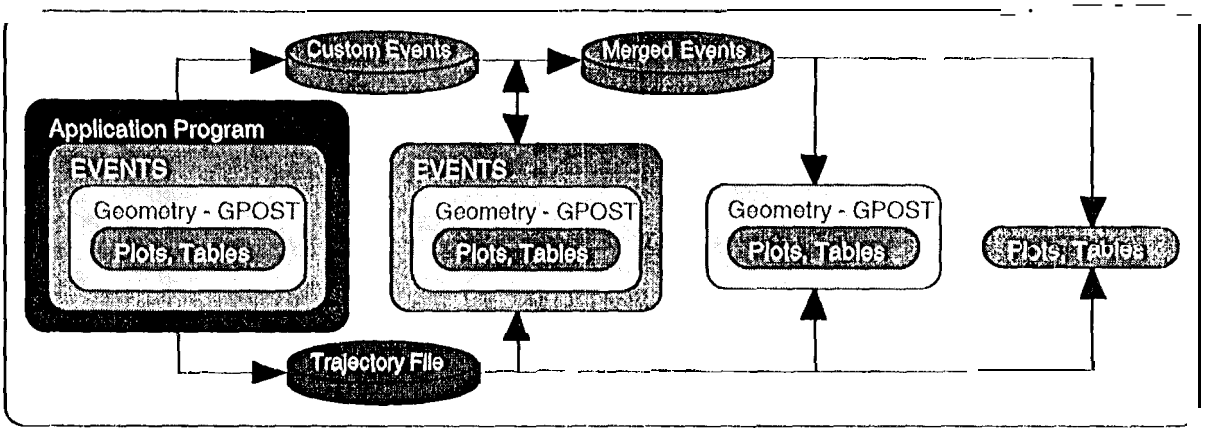


Figure 4: The new mission analysis software architecture shows the inter-relationship and reuse of common modules.

After running the. satellite tour program, the custom events related to the tour are written out in a standardized format to be read by other applications downstream. In the diagram, the custom events and the. standardized trajectory file. are used as inputs to the EVENTS module, which can be used to calculate any number of user specified occurrences such as flux tube crossings. The EVENTS program would then produce a merged events file. containing the information from the tour design program, and the unique occurrences requested. This information could then be passed to GPOST to calculate. standard trajectory products such as ranges and range rates to Earth ground stations for telecommunications analysis, and the range to the. sun for power and thermal considerations. Finally, the information can be plotted or tabulated in a specified format. The main message contained in Figure 4 is that most mission analysis applications have much in common, and that by identifying, the major modules and designing them in a general, multi-mission fashion, they can be readily used to construct new applications while minimizing duplication of effort, and preserving standardized interfaces for a more synergistic system.

COST SAVINGS

The most important consideration in re-engineering JPL's mission analysis software has been to increase the productivity of the mission analyst, thereby saving projects considerable time and money. Equally important, however, is the cost of new developments, and the projected cost of

maintaining the new system. By eliminating unnecessary redundancy and mission specific implementations, there are far fewer programs to maintain, and also fewer programs for the users to learn.

The CATO program is a good example of cost savings realized from the new architecture, originally, CATO was only intended to replace the MOSES software set for optimizing satellite tours around outer planets. The PLATO software set was another large package, but it was intended to optimize trajectories in heliocentric space with planetary flybys. , Conceptually, the two systems were very similar, so a decision was made to make CA'] '() as general in nature as reasonably possible., and thereby replace both sets. in essence, the algorithms were formulated without making any assumptions about what bodies would be involved, whether they were planets, ~.satellites, asteroids, comets, or any combination. Ultimately, the CA'] '() program was delivered to the Cassini project with only an insignificant addition of cost and schedule.

After delivery, several projects realized significant cost savings due to the general purpose nature of CATO. Initially, the Cassini project only intended to use CATO for satellite tour design, but their first use was to optimize. the complex interplanetary Earth-Venus-Venus-Earth-Jupiter-Saturn trajectory and produce the target specification in far less time than planned. Although developed as a Cassini product, the Galileo mission used CATO to perform an end to end optimization of the tour at Jupiter, which was not possible with earlier software. Several small mission studies requiring complex trajectories in the Earth-moon-sun system were able to reduce their propulsive requirements by using CATO to optimize their trajectories. And finally, the New Millennium Deep Space 1 mission using solar electric propulsion (SEP) to math an asteroid successfully used CA"] '() to model the low thrust burns and optimize the trajectory to the target. The New Millennium program will devote. some funds to augment the CATO effort to make it easier to use for SEP analysis, but there are large savings realized by starting with a general purpose, multi-mission tool.

CONCLUSIONS

After decades of relative isolation], JPL flight projects and studies alike are beginning to reap benefits from the new partnership formed between mission analysis software users, developers, managers, and institutional sponsors. By coordinating a large number of tasks, it has become clear that many mission specific applications have much more in common than initially thought. Modern computer platforms, networks and language features have enabled a significant re-engineering of the mission analysis infrastructure, while eliminating unnecessary redundancy and cumbersome interfaces. Benefits include increased productivity, multi-mission applicability, error reduction, and reduced maintenance costs.

More work needs to be done, however, as the new system is integrated both internally, and with the software tools of related disciplines. Although most of the major analytical components are nearing completion, details of the user interface and interprocess communications are still under development. Also, plans are currently being made to integrate the best features of mission analysis software with [hose of the navigation system.