RESOURCE ALLOCATION PLANNING HELPER (RALPH): LESSONS LEARNED

Ralph Durham, Norman B. Reilly, Joe B. Springer
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California
(818) 354-6316
FTS 792-6316
FAX (818) 393-4100

ABSTRACT

The Jet Propulsion Laboratory's (JPL) Resource Allocation Process incorporated the decision making software system RALPH into the planning process four years ago. The current principal task of the Resource Allocation Process includes the planning and apportionment of JPL's Ground Data System composed of the Deep Space Network and Mission Control and Computing Center facilities. The addition of the data-driven, rule-based planning system, RALPH, has expanded the planning horizon from eight weeks to ten years and has resulted in significant labor savings. Use of the system has also resulted in important improvements in science return through enhanced resource utilization. In addition, RALPH has been instrumental in supporting rapid turn around for an increased volume of special "what if" studies.

This paper briefly reviews the status of RALPH and focuses on important lessons learned from the creation of an highly functional design team, through an evolutionary design and implementation period in which we selected, prototyped and ultimately abandoned an 'AI' shell, and through the fundamental changes to the very process that spawned the tool kit. Principal topics include proper integration of software tools within the planning environment, transition from prototype to delivered software, changes in the planning methodology as a result of evolving software capabilities and creation of the ability to develop and process generic requirements to allow planning flexibility.

Also examined are strengthening of resource allocation techniques enabling implementation of effective conflict resolution strategies through an understanding of mission flexibility in the context of resource capacity/availability, characteristics and constraints, and techniques enabling early forecasting of resource loading to permit mission design changes. Finally, we present a discussion of a design which provides the ability to easily alter resource and requirements data tree structures to provide a problem-independent scheduling system applicable to a wide range of scheduling problems.

INTRODUCTION

During the last four years, the Resource Allocation Planning Helper (RALPH) has become an integral part of the Ground Data System Resource Allocation Process (RAP) at the Jet Propulsion Laboratory. (see Figure 1) As a result of the experience of users from the flight projects, the Resource Analysis Team and ground-based radio astronomy, both the process and the RALPH tool kit have begun to change and mature.[1] The experiences gained during this period of transition have provided some interesting

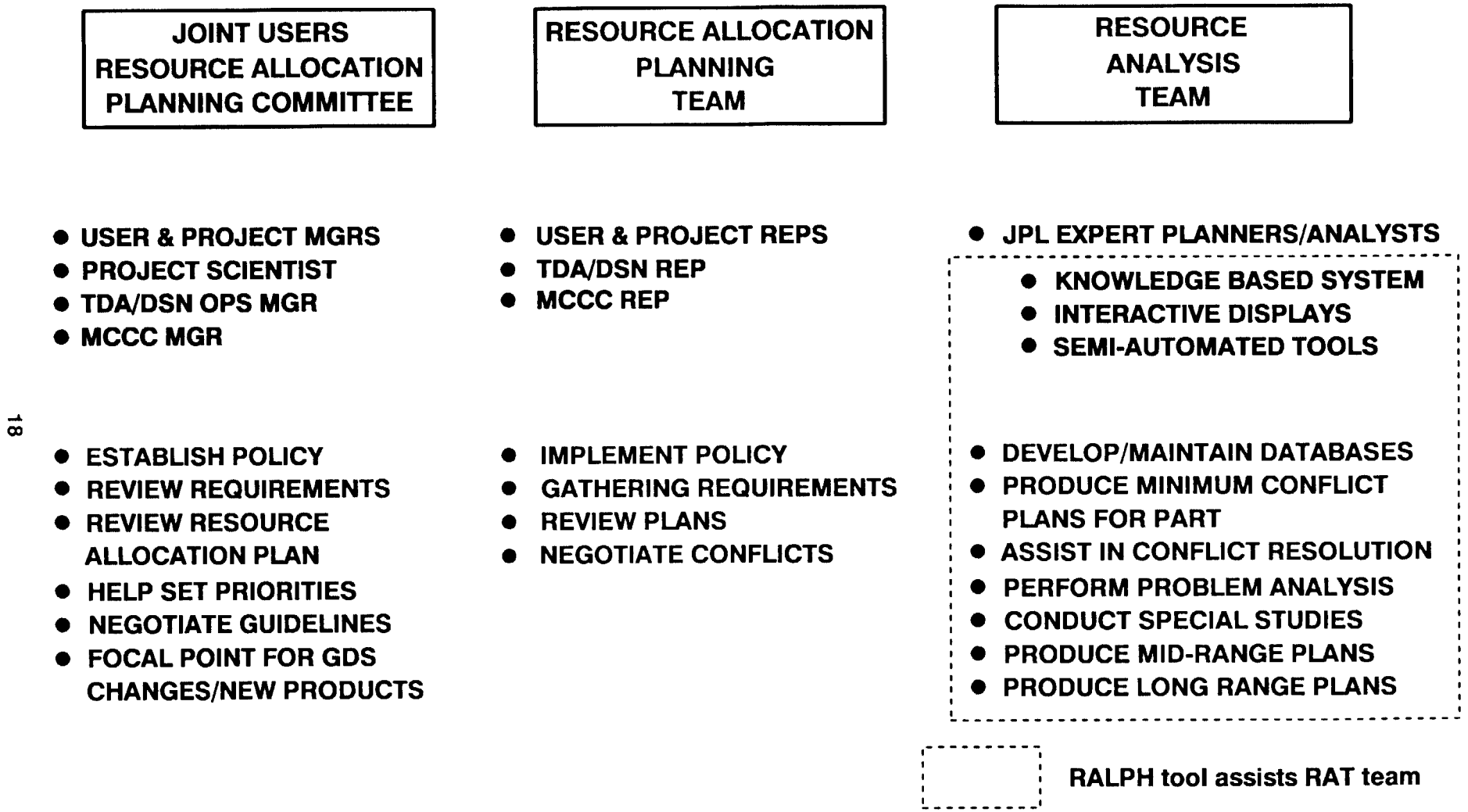

Figure 1 Resource Allocation Planning Process

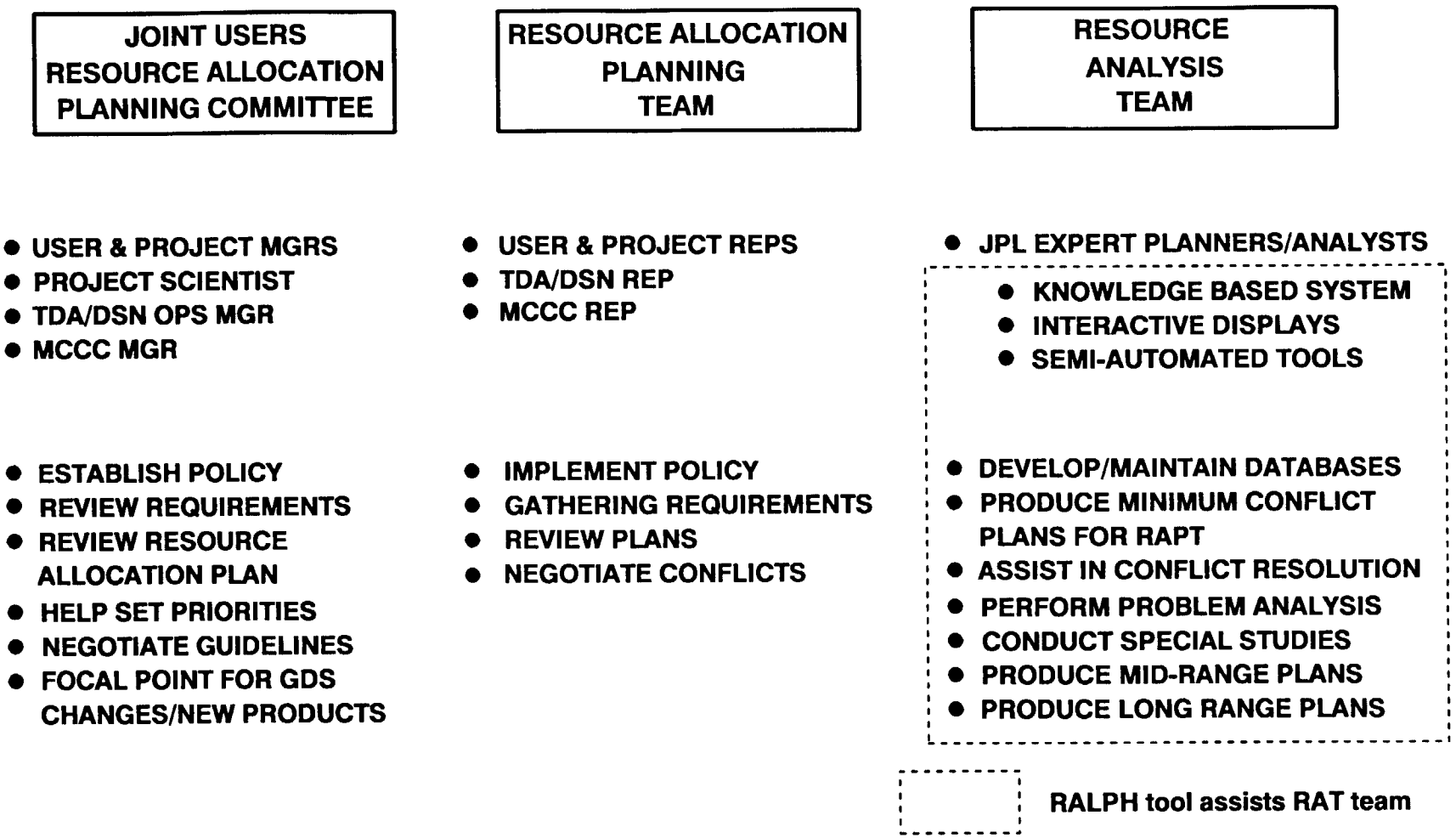

Figure 1 Resource Allocation Planning Process

insights of benefit to those involved in the creation of similar systems.

RALPH has enabled the JPL Resource Analysis Team to extend its planning horizon from the three weeks common in the mid-1980s to ten years. In doing so, it provides a valuable planning tool for ground-based radio astronomy, mission and sequence designers of current unmanned deep space and planetary projects, and planners of future projects. Providing the ability to quickly derive answers to 'what-if' questions posed by JPL and NASA management, RALPH has proven to be a unique and worthwhile resource.

This paper presents a discussion of the transitions that have occurred during an evolutionary design and implementation cycle.

COHESIVE DESIGN TEAM

From the inception of the project in 1985, a relatively small group of developers and users have worked cooperatively toward an illusive goal. Though from different organizations within the JPL matrix, developers and users have built a working relationship based on constant personal communication. To the team, this has meant that formal weekly design meetings are reinforced by daily informal sharing of progress, frustrations, an ever-expanding user wish list, triumphs and failures.

Through the first three years of design and prototyping, purely through an accident of logistics, the teams inhabited the same building, separated only by a flight of stairs. This providential co-location was of tremendous importance to the ultimate success of the project. The development team participated in the 'hands-on' production of planning products during this period, and through these efforts were able to both confirm what they knew and to identify what they did not yet understand.

'LIVE-IN' KNOWLEDGE ENGINEERING

A fortuitous decision in the early stages of RALPH design was that to build a true expert system. As will be explained later, that decision was altered in stages as the design matured, but the effort by development to understand the Planning Methodology was already underway.

Planning is not scheduling; though the end result may well be a schedule, it requires a unique mind-set that does not come easily to some. A plan may be differentiated from a schedule by the process of its creation. Typically, a planner has much more to consider than simply how to fit some irregular pieces together to force them into a confined space. Rather, he must juggle the complexities of intertwined impacts that his decisions may have on the entities being scheduled. A prime example is that Resource Allocation Team plans (schedules) have as their principal aims: (1) the maximizing of science return from each of the spacecraft being tracked, (2) optimization of resource use, and (3) spacecraft health/survival.

The need to understand the planning mind-set became apparent during the earliest stages of the Design Team's work, and, as a result, the developers virtually 'moved in' with the users. Due to the close proximity of our offices, members of the development team had begun to spend virtually all of their time working with the planners. Though much of their activity was typical of the information gathering stages of Knowledge Engineering, their involvement with the

Resource Allocation Team was total. At least two developers have acquired the skills of apprentice planners.

The result of this extended task has been a design that is intuitive, generic and flexible. RALPH is optimized to attack some very specific problems revolving around identification and management of resource conflicts at Deep Space Network (DSN) stations, but has already shown its ability to solve problems not specifically foreseen by the Design Team. As they acquired the planning mind-set, the development team began to realize that the solution to the primary problem, if properly implemented, could be applied readily to other resource scheduling situations.

One of the most commonly applied results is the capability to do 'what if' special studies for a variety of users. To date, those activities have included an analysis of the potential impact of NASA access to a proposed Centre National d'Etudes Spatiales (CNES) 34 meter tracking station on Tahiti, an impact study on proposed DSN support for Phobos and a DSN study to examine the cost-effectiveness of acquiring some additional hardware to minimize station downtime during equipment upgrades. The Phobos study was of some interest because, as a result of its findings, Phobos Project management altered the landing date of Lander #2 and adopted a co-location scenario for the landers. The addition of RALPH support has enabled the team to produce one or two special studies each month as a complement to the regular work load.

Each of these, and other studies, have been possible because the design offers nearly unlimited flexibility that allows the planner to describe virtually any set of resource capabilities and user requirements.

A LITTLE LANGUAGE FOR DATA TREE MANIPULATION

Prototyping the necessary Requirements and Resources data management techniques had proven the value of specialized tree structures to the development team. It became apparent, though, that without resorting to the use of proprietary software with some unacceptable limits, manipulation of data objects required a great deal of coding. The decision to create a little language to manage operations within the RALPH database has proven to have been the correct path.

Implemented in C, the resulting Tree Manipulation Base Routines (TMBR) has provided sufficient power and flexibility that approximately half of the RALPH executables are written in TMBR. The remaining code is C.

The tree structures are central to the final RALPH design. Schedules are appended to the lowest levels of the Resource Capabilities trees as they are created. The text and graphics editors, printer and plotter routines and display drivers all access schedules and user requirements via TMBR commands. Schedule changes following negotiating sessions alter the data structures via TMBR.

CART BEFORE HORSE

When the Resource Allocation Team first identified the need for software support, it seemed apparent that the conditions necessitated a schedule optimizer. The planning staff had spent years putting together schedules manually, but the capability to create a final product that provided the maximum possible support for all users while minimizing negative impacts was very time-consuming.

The inability to identify and evaluate all alternatives had precluded true planning beyond three to eight weeks. Even within this time frame it was difficult, if not impossible, to react in real time to changes in user requirements (science opportunities such as unexpected solar activity or the recent super nova) and facility capabilities (last winter's inopportune loss of the 70m station at Madrid during a period of already heavy contention).

As design work was begun, it became apparent to the team that concentrating on the optimizer would be a tactical error. The resource allocation process would be better served by a tool that could build the schedule from scratch. Optimization, though guided by the planners, had always been a people process, decision making by consensus of representatives of all involved projects. Totally removing the users from the loop would be politically inadvisable.

With the realization that the planner logically had to be done first, the team once more began looking at required functionality. Focus was initially on implementing what we had learned about the planning process, but once again, it was realized that we had not reached the illusive 'square one'. Our planners work was being driven by written requirements levied by users. The planning software would require some sort of interface through which requirements could be input.

The requirements translator, which interprets and reformats user inputs, revealed itself to be a task of a complexity equal to that of the planner. The translator must accept widely varied input in the form of user requirements and synthesize a uniform list of times, durations, antenna designations and split coverage with tracks to support uplink and downlink for distant spacecraft that the planner can overlay on a timeline. Parsing the input file and creating a common format from requirements which may be totally generic (14 hours of tracking on 34m stations during the next seven days), science priority specific, spacecraft or mission event-driven, or innumerable combinations is more time and resource consuming than creating a plan. (see Figure 2)

Another pivotal innovation that has made the system such a valuable tool was the concept of Generic Requirements.

Building a schedule is a series of controlled, rule-based reactions to requirements imposed by the participants. Each scheduling exercise calls for thousands of individual decisions and has thousands of potential solutions. As in a game, the complexity increases geometrically with the number of rules. If projects could be convinced that there were advantages to loosening or reformatting their tracking requirements, or reducing the number of rules in the game, the planner, whether man or machine, would have many more options and could ultimately create a better plan.

That campaign has been won. RALPH was implemented with the ability to interpret inputs and build plans whether driven by specific or generic requirements, but the user projects, realizing the positive impact of generic requirements use them for all tracking except that supporting project-critical events such as encounters and maneuvers.

The goal, then, was to create a requirements translator/planner/optimizer to provide the conflict resolution process a solid starting point. This is not to minimize the quality of RALPH plans for, starting with the highest priority requirements, the planner derives from

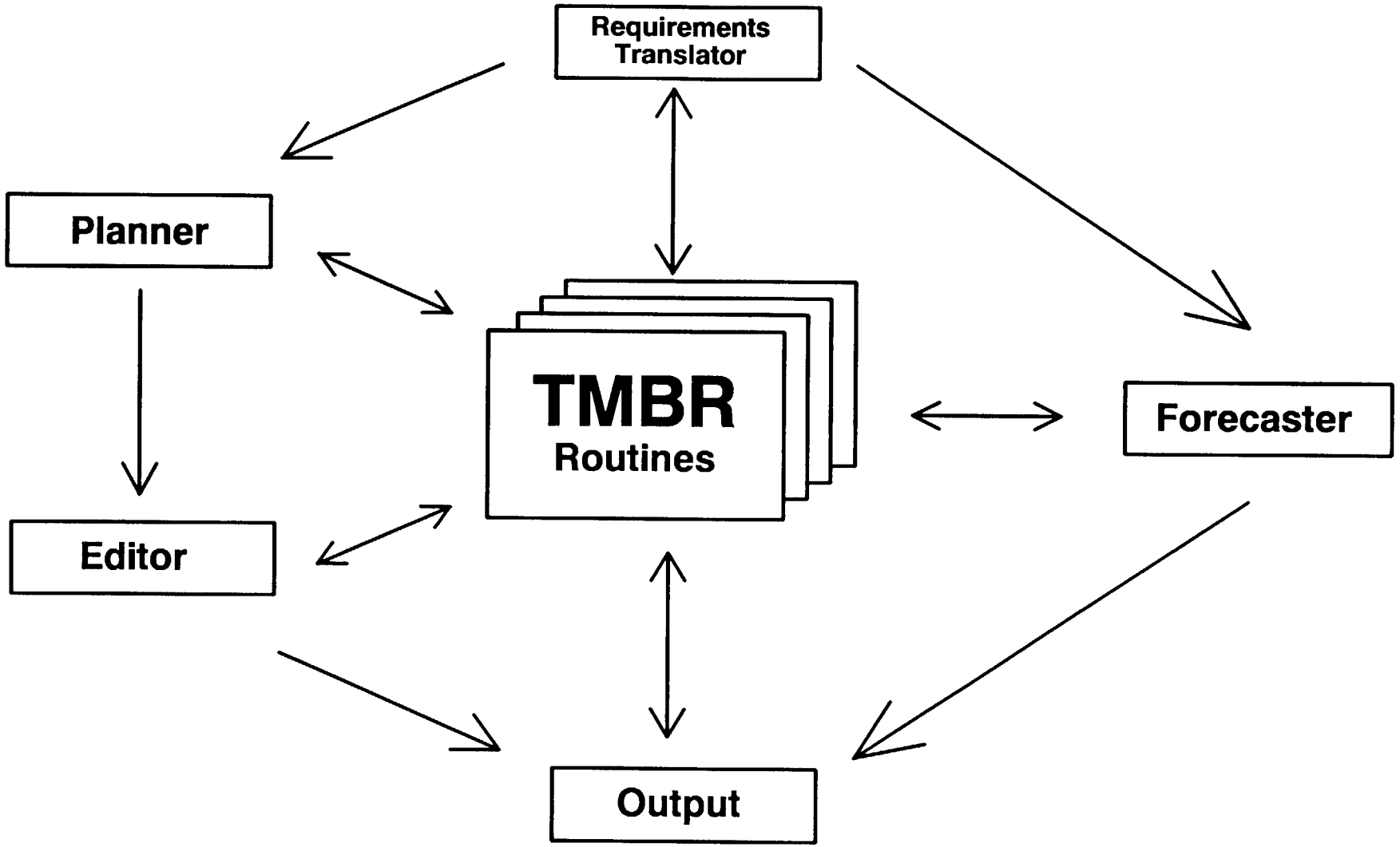

Figure 2 RALPH Architecture

the multitude of possibilities the 'best' option. And, though the planner can be forced to produce a conflict-free product, as we have previously discussed, we believe that the committee of project representatives should resolve the most difficult conflicts though consensus decisions.

HOW FAR SHOULD AUTOMATION GO?

When the RALPH tool kit was originally conceived, one of the ultimate goals was to determine the appropriate mix of decision making by software and by humans. However, resource allocation is a very complex function the result of which ultimately determines or, at least strongly influences, the volume and mix of science and engineering data to be returned from each spacecraft. The science investigation teams for each project could make a valid case for increasing the amount of coverage granted their spacecraft or ground-based activity. For these reasons, and for those we have discussed elsewhere, the final steps in making tracking decisions is, and will continue to be, made by a team of project representatives working with Resource Analysis Team planners.

Human nature, then, is an undeniable obstacle standing in the way of a totally automated planner. With the adoption of generic requirements and with RALPH's event priority logic delivered, there is no technical reason that a set of rules could not be assembled that would allow the software to create a conflict-free schedule.

If total automation were a goal, a second impediment, one that would have to be overcome by careful design, would be the need for a comprehensive rule set defining the relationship and priorities between every combination of supported projects and a second set defining every potential contingency that might alter the requirements of each project. In planning, decision making must take into account the present situation, but must often also consider inter-project trade-offs used recently in granting (or withholding) support.

A planner may, for example, deny Voyager 1 several hours of tracking time during a specific week in favor of ICE, but often does so with at least the informal understanding that that time will be 'repaid' when the total tracking situation allows it. Any such trade-off, including the intention to do a payback, is always driven by the ultimate goal of maximizing science return for all supported projects. Any fully automated system would have to be capable of similar decision making to be acceptable.

START-UP LAG

Valid statistics demonstrating the cost-effectiveness of the RALPH development effort have been compiled and advertised. At the same time, we have recently been overwhelmed by more than normal negotiating time, one of the specific problems that RALPH was built to minimize. We have been able to demonstrate that both the process and the tool are valid and are functioning as designed. We are, however, victims of uncontrollable past circumstances.

Had the capability to do long-range planning existed in the early 1980s, JPL and NASA would have had a tool to permit optimized re-scheduling of payloads when shuttle flights resumed following the post-Challenger hiatus. Long-term planning completed before the Challenger accident had scheduled most upcoming launches, encounters and other

tracking-intensive events to minimize serious conflicts.

With the resumption of STS-based planetary launches, however, the schedule for the late 1980s and early 1990s has become anything but optimal. Magellan (MGN) and Galileo (GLL) were launched six months apart with mission designs vastly different than those originally planned. As a result, both are headed for Venus and periodically have very similar view periods (that is, occupy the same part of the sky from an Earth perspective). Sharing significant parts of the same sector of the sky are two of the Pioneer spacecraft. As GLL approaches Venus for a fly-by and concurrent trajectory correction, planners are faced with providing Galileo nearly continual coverage while providing at least survival coverage for MGN and the Pioneers. This is just the situation planning is designed to avoid.

Software projects implemented to correct or improve an on-going situation should provide some strategy to go back and correct the short-comings of the past. The danger is that sponsor confidence can be badly damaged unless the development organization is able to foresee start-up deficiencies and make them known to management.

The RALPH Design Team had anticipated this 'lag' to some extent and, consequently had taken steps to prepare management. Two additional problems prevented a totally adequate reaction to the coming situation. First, when launches resumed, there was not sufficient lead time to allow flight projects and mission planners to react properly. In addition, these events occurred at a time when Resource Allocation Plans were new to project management and the plan's credibility had to be established before any reaction could be mounted.

When plans with enormous levels of antenna contention began to appear, the negotiating process reacted by slowing down from the excessive work load. It appeared at first that neither the software nor the resource allocation process were working when, in reality, the quality of both was absolutely valid under the circumstances. What was required was a doubling and re-doubling of negotiating time until the most difficult periods had been freed of conflicts.

COST EFFECTIVENESS

While Resource Analysis Team benefits of the RALPH tool kit and planning products produced through its use are easily measured, a significant portion of its positive impact per dollar invested cannot be estimated with any degree of accuracy. The latter is largely because RALPH provides long-range planning capabilities not available in the past from any source. The semi-annual mid and long-range plans as well as special studies, done largely by special request and with quick turn-around, have become working tools for near term decision making and long-range planning by JPL management and NASA Headquarters.

A significant side benefit of RALPH is its support of RAT team activities including data entry, plan generation, and conflict resolution meeting support. The 1988 estimated savings was well over 7000 person-hours a year. A significant part of this is the fact that preparation of weekly plans historically required 25 hours of an experienced planner's time. Now, a RALPH planner with far less experience can produce a plan in five hours, most of which is consumed by data entry.

The editors and plotting routines make possible the production of diagnostic tools never before available. The majority of the 7000 hours, though, is reflected in the fact that the mature state of RALPH allocation plans has reduced the number of negotiating sessions to only one weekly from the former three or four. This reduction is worth well over 6000 hours yearly. The RALPH four year development costs have been about $3 million.

HYBRID ARCHITECTURE

As a design incorporating heavy reliance on data tree structures emerged, the designers realized that any attempt to continue to rely wholly on the original expert system concept was invalid. Rather, conventional algorithmic structures could be used for much of the planning as well the supporting editors, input/output and interpretive modules.

The RALPH design continues to rely on rule-based decision making for such tasks as creating best fit schedules and, at a more detailed level, making support decisions based on sets of contention variables. Each of the TMBR modules, then, has been designed either as a rule-based routine or as a traditional algorithm.

PAINFUL TRANSITIONS

To have the opportunity to even begin an innovative task requires equal doses of optimism and masochism, innovation and conservatism. In a large organization, it also requires management with the foresight to charge an enemy hidden in the mists of uncertainty. Innovation can make you seem a genius or a fool. Innovation is an arena for those who understand that not every attempt is a win, but who believe that the goal is worthy and the aspirants are equal to the test.

RALPH development has been treated somewhat differently than many software prototypes. Whereas the prototyping environment is most often laboratory-like, isolated from the atmosphere of real world production, RALPH has been an 'on-line prototype' from its first delivery in 1986. We have found this to be an optimal state offering the users the latest available technology while at the same time allowing quick turn-around when delivering new features or bug fixes.

Daily use of advanced prototypes also offers the developer a realistic perspective of the true usefulness of his creation. Isolated prototyping, if not carefully designed, can mask the behavior of subject software under the hands of perhaps less sophisticated users who can usually be counted on to do the unexpected in the course of meeting daily production goals.

Throughout the prototyping period, the status of the software, and of its relationship to the process it supported, was under constant peer review. That feedback continues today through both formal and informal feedback from the user community and the RALPH review board.

During four years of on-line prototyping, current configurations on development and production machines had been managed by knowledgeable members of the design team. The sponsoring organization, JPL's Flight Project Support Office (FPSO), determined that RALPH had achieved a state of maturity that demanded the end of informal deliveries and consequent

introduction of formal configuration management (CM).

Though neither users nor developers foresaw this as a problem, we learned over the next two or three months that two distinct philosophies existed. The design team assumed an indeterminate transition period controlled largely by the relative ease of the version 4.0 delivery. This delivery followed a major rewrite during the transition from RALPH's early format to the previously described TMBR/C version, a non-trivial metamorphosis. Configuration Management had assumed a transition day to occur before delivery to test. The acceptance test period was extensive and required several re-deliveries. Ultimately, it was evident to all that a transitional period had occurred naturally and that formal, third party CM was applicable only after true stability of the final prototype version had been achieved.

WHERE WILL IT END?

References have been made to the mid-course corrections that have been necessary to refine the design team's targeting. The result is that, to date, a tool set with enormous positive impact has been created for the users. Through use of the growing set of tools, both the Resources Analysis Team and the projects they support have conceived an extensive list of additional capabilities that would expand the team's abilities even further.

Not the least of these is the suggestion that, in the coming era of growing international cooperation and sharing of resources,[2] the JPL resource allocation process and the RALPH tool kit would be an ideal means to provide optimized planning and scheduling for the world space science community. The Phobos and CNES studies mentioned earlier, as well as support for such multinational efforts as Ulysses, Giotto and Galileo, have provided valuable experience in international cooperation.

CONCLUSION

The RALPH experience has emphasized a number of lessons which, while not unique to this effort, will have been of benefit as we embark on future projects. Tantamount to a secure development environment is a sponsor who understands the risks and potential benefits and is willing to support development through both good and bad times. During development, success of a unique system is highly dependent on a close and lasting relationship between developers and users. Such a circumstance is a virtual guarantee that delivery will be free of misgivings and disappointment.

An expert system is traditionally design to *replace* an expert. The goal of RALPH implementation has been to relieve the 'front end' burden of tedious scheduling from the expert planner and to move him to the 'back end' of the system where his expertise can be concentrated on analysis and on providing recommendations prior to negotiations.

In the creation of a precedent-setting system, it is vital for the developers to take unusual strides to be assured that they understand the full implications of the user's requirements. The system initially conceived and requested by users may not be the solution to his problems, for, as we have experienced, the methodology may change in response to the power of the tool. Proceed with caution, be flexible and schedule deliveries that assume change. Regardless of the care given to the original design decisions, significant changes are likely

when the methods chosen break new ground. And, finally, plan a lengthy transition from prototyping to a more formal, controlled environment.

REFERENCES

1. Berner, C. A., R. Durham and N. B. Reilly, "Ground Data System Resource Allocation Process", 1989 Goddard Conference on Space Applications of Artificial Intelligence, May, 1989

2. Dumas, L. N., G. A. Briggs, M. S. Reid, and J. G. Smith, "Opportunities for International Collaboration in Deep Space Communications and Tracking", Proceedings of the Second Annual AIAA/JPL International Conference on Solar System Exploration, Pasadena, August 1989