A Reusable State-Based Guidance, Navigation and Control Architecture for Planetary Missions[1][2]

Sanford M. Krasner
Jet Propulsion Laboratory
4800 Oak Grove Dr.
Pasadena, CA 91109
818-354-6612
skrasner@jpl.nasa.gov

Abstract—NASA's Jet Propulsion Laboratory has embarked on the Mission Data System (MDS) project to produce a reusable, integrated flight and ground software architecture. This architecture will then be adapted by future JPL planetary projects to form the basis of their flight and ground software. The architecture is based on identifying the states of the system under consideration. States include aspects of the system that must be controlled to accomplish mission objectives, as well as aspects that are uncontrollable but must be known. The architecture identifies methods to measure, estimate, model, and control some of these states. Some states are controlled by goals, and the natural hierarchy of the system is employed by recursively elaborating goals until primitive control actions are reached.

Fault tolerance emerges naturally from this architecture. Failures are detected as discrepancies between state estimates and model-based predictions of state. Fault responses are handled either by re-elaboration of goals, or by failures of goals invoking re-elaboration at higher levels. Failure modes are modelled as possible behaviors of the system, with corresponding state estimation processes.

Architectural patterns are defined for concepts such as states, goals, and measurements. Aspects of state are captured in a state-analysis database. Unified Modelling Language (UML) is used to capture mission requirements as Use Cases and Scenarios. Application of the state-based concepts to specific states is also captured in UML, achieving architectural consistency by adapting base classes for all architectural patterns.

Within the Guidance, Navigation and Control domain of MDS, work has focussed in three areas:

(1.) Re-engineering and re-implementation of legacy Navigation systems within an object-oriented structure that is reusable from mission to mission and common between flight and ground systems;

(2.) Identification of states and mission activities which are common across multiple missions;

(3.) Exploitation of commonality between Attitude Control and Navigation functions, which have historically been separated in previous JPL missions.

These areas will be demonstrated on a simulated reference spacecraft and mission and then adapted by customer missions. Early deliveries will have levels of autonomy similar to existing JPL spacecraft, in order to demonstrate the applicability of the state-based concepts. This architecture should greatly simplify the implementation of existing levels of autonomy and should support significantly increased autonomy in future deliveries. Studies are ongoing to determine the mission requirements for highly integrated attitude and trajectory control functions, leading eventually to "6-degrees-of-freedom" control. These functions will eventually be implemented within the MDS architecture.

The first customer mission is Europa Orbiter, to be followed by Pluto/Kuiper and Solar Probe. Discussions are also ongoing to adapt MDS for the Space Interferometry Mission and for control of the Deep Space Network.

First prototype demonstration of a simple detumble capability was demonstrated in November 1999. As of December, 1999, work was being done on the next increment, controlling attitude to bring the Sun into the Sun-sensor field-of-view. Subsequent deliveries will include

---

[1] 0-7803-5846-5/00/$10.00 © 2000 IEEE

[2] The work described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

increasing capability leading to a delivery to the Europa Orbiter project in Nov. 2001.

TABLE OF CONTENTS

1. INTRODUCTION

In order to reduce the cost of developing and operating planetary missions, NASA is attempting to build Faster, Better, Cheaper projects. As part of this initiative, the Jet Propulsion Laboratory has begun the Mission Data System (MDS) project. [1] MDS attempts to reduce the cost of mission software development and operation by taking advantage of two forms of software reuse: reuse of code and reuse of concepts or frameworks. Code is reused from mission to mission and common code is used in both ground and flight applications. Common frameworks are reused in various domains throughout the flight and ground software. In most cases, MDS is not inventing new ways of performing mission functions. For instance, MDS will not pioneer new forms of attitude control or navigation. Instead, MDS will focus on systematic and reusable ways of describing and implementing capabilities found on previous missions.

The MDS software architecture is "state-based". It is organized around the identification, estimation and control of the states of the project system. This system includes the ground and flight segments. In general, these states have been used implicitly in the designs of previous missions; MDS will focus on making these states explicit.

In addition to the state-based architecture frameworks, MDS is divided into a number of application domains, including the Guidance, Navigation and Control domain. In past missions, navigation (orbit determination and control) has been entirely a ground-based function, based on radiometric data and optical navigation images captured by the spacecraft; attitude control and maneuver execution has been entirely a spacecraft-based function. On the New Millennium Deep Space 1 mission (DS1), optical navigation image processing, orbit determination and some level of orbit control planning was done onboard the spacecraft. Future missions will require more integration of navigation and attitude control functions, ultimately leading to coupled orbit and attitude control for rendezvous and landing missions.

2. STATE-BASED DESIGN

For MDS, *state* is defined as "a representation of the momentary condition of an evolving system". States may range from the simple on-off state of a power switch to the trajectory of a spacecraft or a planet. It may even be possible to manage mission science data acquisition in terms of the state of the mission data repository, e.g. whether a particular observation has been acquired.

An iterative process of state discovery will drive the design of the MDS frameworks, and will drive the adaptation for a particular mission. Once a state has been identified, further analysis of the state will identify other states that must be defined and analyzed. States will be implemented in software as specific instances of the general state pattern. In object-oriented terminology, the architectural pattern for a state is defined by a state class definition; each type of state is a subclass of the State class, and will be instantiated in state objects.

For each state, the following characteristics must be identified and, in some cases, implemented in software:

State Representation and Value Domain,
Goals, Goal Elaborations and State Control Actions
State Estimation, Measurement and Uncertainty representation,
State Dynamics and Measurement Models.

State Representation and Value Domain

The representation of each state must be defined as well as the range of the values the state can assume. (For instance, an enumerated ON/OFF for a power switch, a continuous value for a temperature or pressure, a quaternion for a spacecraft attitude, etc.) This range is known as the value domain of the state, and will be the basis for goal constraints defined below.

Goals

A state may be controllable or uncontrollable *within the context of the system being described.* If a state is controllable, actions can be taken in mission operations to change its value. Controllable states may include power switches (which can be controlled by hardware commands), spacecraft attitude (which may be controlled by thruster firings or reaction wheel activation) or spacecraft trajectory (which may be controlled by delta-velocity maneuvers). Uncontrollable states are externally imposed – we can observe them but cannot change their values. Uncontrollable states are included in the mission system definition because their values are needed to estimate or control the controllable states. It is possible to control the relation of a controllable state (such as spacecraft trajectory) to an uncontrollable state (such as the trajectory of Jupiter). Uncontrollable states may include equipment alignments

onboard the spacecraft, or the trajectories of the planets and asteroids.

MDS missions will be controlled by a network of goals on controllable states. A goal is defined as "a constraint on the value of a state" and represents an assertion about a state over a time range. This assertion is generally enforced by a software control loop. A goal for a power switch state may be that "the switch is closed from time T1 to T2"; a goal on a spacecraft attitude may be that a "spacecraft body vector is pointed at Jupiter from T1 to T2". The beginning and ending times for the time range are identified as "timepoints". Timepoints may be specified as absolute times, or as relative time relations (e.g. a specified maximum or minimum time between timepoints).

Note that identifying a celestial body (e.g. Jupiter) as a target implies a set of uncontrollable states whose values must be known to perform that control. In this case, the states may be the spacecraft trajectory relative to the Sun and Jupiter's trajectory relative to the Sun.

Since a goal represents an assertion that is enforced over a time-span, other activities can assume that the condition holds true. This leads to an elaboration process, in which execution of a particular goal (e.g. pointing at Jupiter) requires that other goals be enforced (e.g. that gyro and star tracker power switches be maintained on).

Goal Elaborations and State Control Actions

Each controllable state must define a controller, which will execute any control law required to enforce a goal constraint. The state must also specify any conditions on other states that must be enforced in order to control this state; these are normally expressed as "subgoals" – goals delegated to other states. As cited above, a goal to maintain attitude may result in subgoals to maintain power to sensors and actuators. The recursive process of elaborating goals into nets of subgoals eventually bottoms out in "primitive" or "executable" goals. These goals can be directly enforced by a controller and require no further elaboration.

Primitive goals correspond to the action commands typically used in sequences on previous missions; these are executed by standard control algorithms, which are capable of issuing power switch commands, slewing to commanded attitudes, etc. In general, these primitive goals are expected to follow patterns of "transitioning to desired state" (e.g. "turning switch on", "slewing to point at Jupiter"), followed by "maintaining desired state" (e.g. "periodically checking that switch is still on", "holding attitude pointing at Jupiter"). Since the desired state is specified explicitly, failure to maintain a goal constraint can be used as the basis for fault detection and correction.

When primitive goals are stated as transitioning to or maintaining a particular state, they make it explicit that state estimation and control processes are required; i.e. that a

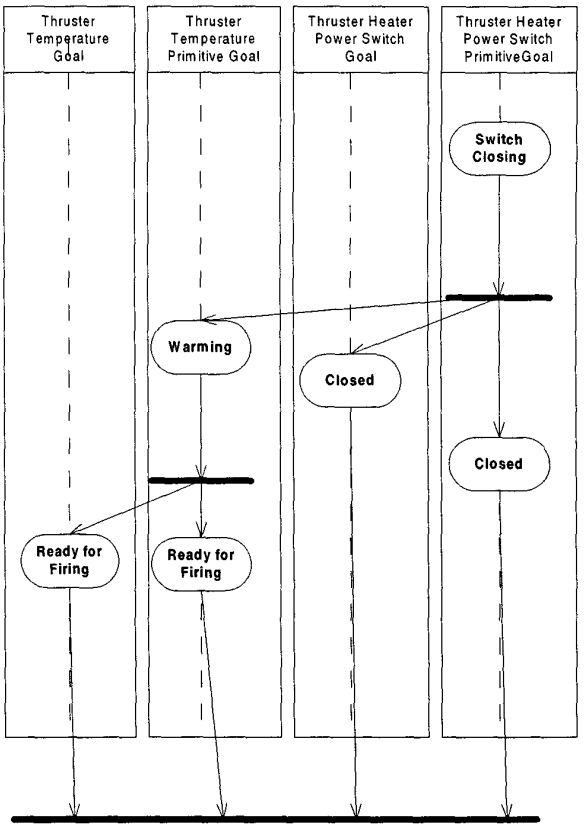


Figure 1 - Thruster Temperature Goal Net

control loop is required. This is a common notion in the attitude control domain but is not as common in other domains. A sequence command may throw a power switch. The corresponding primitive goal is to maintain a switch state; one or more switch commands may be invoked by the control loop for that switch state.

Experience with the Detumble prototype shows that primitive goals correspond directly to the states of the underlying controller. Firing of the timepoint to initiate a goal translates into an event that triggers a state transition. This has allowed the controller transition logic to be auto-generated from a UML statechart. Primitive goals are the states of the state chart; timepoint firings are the transition events.

The set of goals active on the spacecraft at a particular time is represented in a temporal constraint network [2]. Figure 1 illustrates a simple goal network, represented as a UML activity diagram. In this diagram, the goals are represented as rounded squares or "activities. Timepoints are represented as black bars – "forks" or "joins" in activity diagram notation.

In the temporal constraint network, each goal is represented as an activity that occurs between two "timepoints"; during this time interval, the state constraint stated by the goal is enforced. A timepoint "fires", initiating an activity, when all of its outgoing goals are being satisfied. Timepoints may also be constrained by earliest and latest times, as well as smallest and largest delays between timepoints. This allows the network to implement absolute-time, relative-time, or event-driven control.

In this simple network, two states are defined – thruster heater power switch and thruster temperature. Thruster temperature is defined to be "ready for firing" when the power switch has been closed for a specified period of time. The goal on the thruster heater power switch is that it be closed. Elaboration of this goal has invoked primitive goals that the switch is closing (i.e. being commanded closed, followed by measurements to determine if it is closed); followed by a primitive goal that the switch remain closed. The closing primitive goal is executed by commanding the switch closed, then monitoring status measurements to determine if the desired state ("closed") has been reached. The "maintaining closed" goal is executed by continuing to monitor switch state, and re-commanding if necessary.

Reaching the "closed" state triggers a primitive goal on the thruster temperature to be in a "warming" state. Once sufficient time has elapsed, the temperature is declared "ready for firing" and a primitive goal is executed to keep it ready.

State Estimation, Measurement and Uncertainty Representation

Each state object must provide state estimation methods. These methods allow other objects to query the value of a state at a particular time. States may be estimated from measurements that are available to the software, or may be supplied by external sources (such as ground calibrations of device characteristics, astronomer's observations of planetary ephemerides, etc.)

The MDS architecture makes a strong distinction between measurements and state estimates. A measurement is a sample taken at a particular time. It may be a function of several states and does not directly say anything about the value of those states immediately before or after the sample. A state has a value at all times. The value at any time can be estimated, based on measurements and models of the dynamics of the state. (Of course, some states may have "degenerate" dynamics models – they are assumed constant. The dynamics for a calibration state may be simply that the value is constant between calibrations.)

In addition to state estimates, each state is also responsible for representing the uncertainty of the estimate. Since the representations of uncertainty are highly state-specific, the MDS architecture requires that state estimates be capable of responding to stereotyped queries about uncertainty, such as the probability that a state is within a particular value domain (range). In addition, each state must have a way of representing that the state estimate is unknown or highly uncertain.

Among other applications, state estimate uncertainty will be used to control the configuration and initialization of the spacecraft. Controllers will be able to wait until needed estimates become available, and then initiate control. Goal elaborations will be able to set goals on estimate knowledge or uncertainty and control actions will be triggered when necessary estimates are available. Sensor and estimators will be configured to provide estimates and enable control. These should simplify the process of initializing the spacecraft, at launch or after a fault. The order of events will flow naturally out of the chains of dependency on estimates.

Estimates of state performed onboard will be telemetered to the ground as "state models", which summarize the behavior of the state over contiguous time periods. A state model may represent a curve-fit or other approximation to the state evolution; this makes it adaptable to successive compression methods to reduce storage and downlink volumes.

Some states will be estimated on the ground, (for instance, spacecraft trajectories via ground-based navigation.) MDS plans to use the same framework for ground and flight estimation. MDS also plans to use the same frameworks for transporting state estimates between ground and flight systems, making them consistent between both systems.

State Dynamics & Measurement Models

In order to support state estimation, each state will include dynamics models of how the state evolves over time. This model will include the effect of control actions as well as all environmental factors influencing the state value. This allows an estimator to propagate the effect of control actions (e.g. thruster firings) and to use filters to combine state propagations with measurements to provide state estimates.

3. Integrated Guidance, Navigation and Control

Why Integrated Guidance, Navigation and Control?

On previous planetary missions, attitude control (determination and control of rotational orientation) and navigation (trajectory determination and control) have been treated as separate disciplines. This has been done largely because of the need to close loops at significantly different time scales. Attitude control requires loop times of fractions of a second; navigation loops (from orbit determination to maneuver execution) typically take hours to days. In addition, navigation has required data types only available on ground (radiometric observables) or data types requiring large amounts of processing and human intervention (such

as optical navigation images). For ballistic (chemical-propulsion) missions, trajectory corrections are done hours to months apart, so it is feasible to close loops through the ground.

Inevitably, there has been some coupling between the attitude control and navigation disciplines. For instance, attitude control thruster firings perturb the spacecraft trajectory and must be incorporated into the trajectory model. Navigation teams have routinely included both completed and planned attitude control activities (such as spacecraft turns) into the trajectory estimate and propagation models. Navigation software has incorporated both models and measurements of propulsion and attitude control behavior. This includes predictions and actual telemetry on items such as duration of thruster firings, temperature, pressure and fuel flow rates. On Cassini, attitude control thrusting has been modelled redundantly by the Attitude Control and Navigation teams. MDS seeks to eliminate this redundancy, and possible miscommunication, by coordinating and reusing common states and models for propulsion, attitude, and trajectory behavior.

With increasingly complicated missions, and increasingly more capable onboard computers, navigation functions have begun to migrate onto the spacecraft. Ten years ago, Galileo carried a simplified planetary ephemeris computation to allow pointing at the Earth or Jupiter; this required a laborious process to translate navigation ephemerides to a reduced-order representation. Onboard navigation was greatly expanded on New Millennium Deep Space 1 (DS1). DS1 performs onboard processing of optical navigation images and incorporates these, along with thrusting perturbations, into orbit estimates. These orbit estimates are used autonomously to compute and execute trajectory control maneuvers, using ion propulsion and chemical propulsion systems. Since planetary and spacecraft ephemerides are available onboard to support navigation, these ephemerides are used directly to provide pointing. Separate reduced-order representations are unnecessary.

Future missions will require even more integration of attitude control and navigation functions. Ion propulsion missions will perform trajectory control continuously over months, rather than at discrete times. Rendezvous and landing operations on asteroids will be extremely sensitive to trajectory perturbations due to thruster firings, and will require onboard processing of navigation observables (optical navigation images, radar distance measurements, and even onboard radiometric measurements). Aerobraking and landing missions show strong relations between spacecraft orientation and orbital changes, and require trajectory correction maneuvers with turnarounds of hours or less.

These considerations led the MDS project to identify an integrated Guidance, Navigation & Control (GNC) domain. This domain applies the MDS state-based concepts to attitude control and navigation functions.

Next Generation Navigation

Prior to initiation of the MDS project, JPL's Telecommunications and Mission Operations Directorate had begun a project to build the Multi-Mission Operational Navigation Tool Environment (MONTE) [4]. MONTE will replace the legacy Navigation code, some of it 30 years old, with a multi-mission object-oriented design. MONTE is being integrated into MDS as part of the GNC domain. It will also be used by non-MDS missions.

Initial GN&C Capabilities

Initial GNC activities are focusing on implementing previous levels of mission capability within the MDS architecture. For the next 2 years, the focus will be on a reference mission to fly a ballistic (chemical-propulsion) interplanetary mission, with ground-based navigation; this capability will support the interplanetary cruise phase of the Europa orbiter mission.

The first GNC capabilities to be implemented will provide the ability to initialize the spacecraft attitude control capabilities. This includes activities typically done following launch vehicle separation:

> Detumbling the spacecraft (controlling rate using gyros and thrusters),
>
> Using a Sun sensor to establish Sun direction (and bringing the Sun into the sensor field-of-view),
>
> Using a star sensor to establish inertial pointing reference,
>
> Pointing a communication antenna to Earth.

Subsequent capabilities will include performance of trajectory correction maneuvers, and addition of fault detection and redundancy management.

MDS goal elaboration capabilities are being used to achieve the actuator and sensor configurations necessary to support these mission activities and to orchestrate the necessary state transitions. The use of event-driven goal nets should simplify the task of building spacecraft sequences and state transition software.

The integration of attitude control and navigation is beginning with these initial steps. Since both Navigation and Attitude Control deal with rotations of coordinate frames relative to each other, common software will be used for coordinate manipulation in both areas.

Navigation ephemerides will be used to represent spacecraft trajectories and the trajectories of celestial bodies of interest: Earth and other planets, asteroids, moons, etc. The

navigation system includes representations of targets on rotating bodies. A camera onboard the spacecraft can track a target on Jupiter's moon Europa while the spacecraft orbits Jupiter.

These capabilities will be integrated into the MDS architecture by representing each of these potential targets as a state. The state will allow users to query information about the target (e.g. direction, distance, angular rate, velocity) and uncertainty of this information at any time.

Trajectory targeting is traditionally done by specifying a desired condition on spacecraft position and/or velocity relative to a celestial body. One of the long-range objectives for MDS GN&C is to formalize these trajectory targets as goals on one or more states. Casting the problem in this framework should provide smoother coordination between a ground navigation system and on-board functions and facilitate the migration of some or all of the trajectory and maneuver design functions from ground to the on-board flight software.

A somewhat simpler objective will be to include "goals" on the geometric relation between a spacecraft and a target body. This would allow the navigation software to signal events when particular geometric relations apply. This event-detection logic is already in the MONTE software and is commonly used by scientists designing observations. For instance, it would be possible to trigger imaging at particular distances from a target or at particular phase angles. (Phase angle is the Sun-target-spacecraft angle, and determines lighting conditions on the target.) This application of goals is somewhat counter-intuitive, since it does not invoke any direct control actions. However, it is consistent with the MDS notion of monitoring a condition on a state. As with current mission design, analysis and design would be required to predict if and when the desired geometry would occur.

A particularly interesting challenge will be to determine how to represent ephemeris uncertainties in forms that are usable by non-navigation experts. These uncertainties may be used, for instance, in determining how many images are required in a mosaic to cover the error ellipse for a target, or in determining what deadband to use to maintain pointing at a target. Ephemeris uncertainty has not been widely used in previous non-navigation software applications, so there is still some question as to how it will be used.

Onboard Time Usage

With the introduction of onboard Navigation, it is important to be more rigorous about onboard definition and usage of time. In previous missions, the correlation of onboard clock to other time frames (e.g. International Atomic Time or TAI) was measured by the ground and used to adjust ground-produced products. Spacecraft telemetry time-tags were adjusted to allow interpretation in TAI or Universal Time Coordinated (UTC). Command sequences to the spacecraft were adjusted to translate desired UTC times into spacecraft clock values.

Onboard ephemerides require an onboard time source that can provide time in the navigation time frames. This also makes it possible to command directly in the time frame of interest, rather than going through the ground-based TAI-spacecraft clock translation. On DS1, this onboard time system made it possible to incorporate late onboard navigation updates into the command sequence, as was done with the Braille asteroid flyby.

On DS1, onboard time maintenance was done by providing offset-and-frequency corrections to the onboard clock, and by incorporating occasional corrections from the ground-based clock correlation process. Since MDS is designed to run on a distributed onboard hardware set with a number of processors and local clocks, the clock-correlation process will be generalized to allow correlation of multiple clocks. All correlations will be ultimately traceable to a TAI reference. This reflects the MDS philosophy of finding general patterns and reusing them extensively.

Future Capabilities in GNC

Introduction of onboard orbit determination and control enables new capabilities in the trajectory control loop. Precedents for these capabilities have already been demonstrated on DS1. Rather than relying solely on a priori analytical models of non-gravitational accelerations induced by thruster firings, solar pressure and atmospheric friction (during atmospheric braking or aerobraking phases), these perturbations can be estimated from in situ measurements and incorporated in real-time. More timely information can be introduced into the orbit determination solution, and these perturbations can be compensated in the trajectory correction maneuvers.

MDS GNC will be able to compute non-gravitational accelerations due to thruster firings, based on propellant pressure and temperatures, and will be able to keep track of propellant mass depletion. Chemical trajectory control maneuvers may be terminated based on actual accumulated changes to trajectory, rather than solely on ground-predicted times or accelerometer data. This may simplify the process of ground modelling of maneuvers, and should improve accuracy of trajectory control.

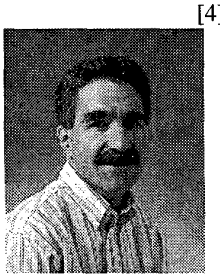
4. Failure Detection and Correction

JPL planetary missions are characterized by long communications delays, due to the long distances involved (light-hours in some cases) and to the fact that communication with the spacecraft cannot be maintained continuously. For this reason, JPL has put a high priority on onboard fault detection, identification and reconfiguration, commonly referred to as "fault protection". Fault protection will be incorporated within the general MDS architecture.

Since the architecture envisions the use of behavior models, fault detection will generally work by determining if a device or process is adhering to its models – if its goals are being achieved. An inability to maintain a goal may result in finding alternative or degraded modes of operations, such as using redundant equipment or functional redundancy for estimation or control.

If a feasible configuration (e.g. redundant equipment or alternate modes) cannot be established to maintain a goal, the goal can be declared as "failed". This will result in notification to each of the goals that are dependent on the failed subgoal. The usual response to this failure will be re-elaboration of the parent goal, with the new information that certain subgoals are not achievable. Again, an alternate path may be found, or the goal may be failed. Thus, faults will propagate within the state/goal architecture, and correction will be achieved as low in the hierarchy as possible.

5. Current Status and Future Plans



As of November 1999, MDS has completed the first prototype of an integrated application of the state-based architecture. This "Detumble" prototype implements a primitive goal network to switch on a simulated gyro package and propulsion system, wait appropriate warm-up times, and activate controllers to null spacecraft rates. No goal elaborations are implemented, and there is no redundancy or fault protection. This prototype demonstrates the lowest layer of the state/goal structure, and its interaction with traditional control laws. The Detumble prototype has also demonstrated the application of the MDS development process. This includes identification and analysis of states; use of Unified Modelling Language to represent operational scenarios, collaborating classes and statechart behavior; and use of code auto-generated from the UML specification.

The next development step calls for adding a Sun search capability, followed by goal elaboration for the detumble and Sun search activities. Following this, the GNC domain will incrementally add capabilities. By November 2000, GNC plans to provide attitude control and navigation capabilities to point at and track celestial bodies, and perform chemical trajectory correction maneuvers.

The MONTE development and MDS integration will continue in parallel with the onboard MDS software development. This will support use of MONTE by non-MDS missions, use of the navigation functions in the MDS ground system, and eventual onboard incorporation of some navigation functions (such as optical navigation and maneuver planning and execution).

By November 2001, the flight software capabilities will be extended to include detection and correction of faults and use of redundant components. Ground navigation components will be available for trajectory estimation and control during interplanetary cruise. This software will be delivered to the Europa Orbiter project as the basis for their adaptation.

References

[1] "Software Architecture Themes In JPL's Mission Data System", D. Dvorak, R. Rasmussen, G. Reeves, A. Sacks, *Proceedings of the IEEE Aerospace Conference*, Big Sky Montana, 2000

[2] Temporal Constraint Network: A mechanism for specifying ordering and timing of activities", E. Gamble, E. Gat, S. Chien, D. Dvorak, R. Rasmussen, JPL Internal Document

[3] R. Dechter, I. Meiri, and J. Pearl, 1991. "Temporal Constraint Networks", *Artificial Intelligence*, 49, 61–95

[4] "Next Generation Navigation Software", R. Vaughan, JPL Internal Document

Sanford Krasner specializes in system design and integration of spacecraft flight software. He has worked on Galileo, Cassini, Mars Observer and Deep Space 1 missions. He is currently the Guidance, Navigation & Control lead for the Mission Data System Project.