Application of State Analysis and Goal-Based Operations to a MER Mission Scenario

J. Richard Morris*
Michel D. Ingham†, Andrew H. Mishkin‡, Robert D. Rasmussen§, and Thomas W. Starbird**
Jet Propulsion Laboratory, NASA, Pasadena, CA, 91109, USA

State Analysis is a model-based systems engineering methodology employing a rigorous discovery process which articulates operations concepts and operability needs as an integrated part of system design. The process produces requirements on system and software design in the form of explicit models which describe the behavior of states and the relationships among them. By applying State Analysis to an actual MER flight mission scenario, this study addresses the specific real world challenges of complex space operations and explores technologies that can be brought to bear on future missions. The paper describes the tools currently used on a daily basis for MER operations planning and provides an in-depth description of the planning process, in the context of a Martian day's worth of rover engineering activities, resource modeling, flight rules, science observations, and more. It then describes how State Analysis allows for the specification of a corresponding goal-based sequence that accomplishes the same objectives, with several important additional benefits.

Nomenclature

APXS	=	Alpha Particle X-Ray Spectrometer
ECam	=	Engineering Camera
Hazcam	=	Hazard Avoidance Camera
IDD	=	Instrument Deployment Device
MAPGEN	=	MER Activity Plan Generator
MB	=	Moessbauer Spectrometer
MDS	=	Mission Data System
MER	=	Mars Exploration Rover(s)
MI	=	Microscopic Imager
Mini-TES	=	Miniature Thermal Emission Spectrometer
Navcam	=	Navigation Camera
ODY	=	Mars Odyssey Orbiter
Pancam	=	Panoramic Camera
PMA	=	Pancam Mast Assembly
RAT	=	Rock Abrasion Tool
RML	=	Rover Mark-up Language
RoSE	=	Rover Sequence Editor
RSVP	=	Rover Sequence Visualization Program
Seqgen	=	Sequence Generator
sol	=	Martian Solar Day

* Software Systems Engineer, Planning and Sequencing Systems Group, M/S 301-250D.
† Senior Software Engineer, Flight Software Development and Technology Group, M/S 301-225, AIAA Member.
‡ Principle Engineer, Planning and Execution Systems Section, M/S 301-250D.
§ Principal Engineer, Information Technologies & Software Systems Division, M/S 301-225.
** Principle Engineer, Planning and Execution Systems Section, M/S 301-250D.

I. Introduction

As the complexity of space missions grows, it is becoming ever more apparent that the associated risks do not evaporate upon a successful launch or even landing. One of the most complex and ambitious space missions to date, the Mars Exploration Rovers (MER), two robotic explorers operating daily on the surface of Mars, has taken the operations challenge to a new level. As the effort to realize NASA's vision to return humans to the moon and on to Mars begins in earnest, it is clear that an operations challenge beyond that of even MER is upon us.

This newest challenge incorporates the scalability of current systems with respect to flight rules and operations personnel, difficulty in observing system behavior as a result of the increased automation required for scalability, the simultaneous operability of multiple assets, and the distributed operations required by concurrent missions and geographically dispersed expertise. A solution that addresses all of these challenges would incorporate a common architecture for distributed operations, enable autonomy for scalability, and model system state explicitly for operability. This paper presents State Analysis as such a solution.

The driving questions motivating this study include: How would a goal-based approach to operations benefit the MER uplink process? How would a Goal-based operations process work? Can State Analysis products (e.g., State Effects Diagrams and Models) provide additional benefit to the mission operations team?

The objective of this study is to initiate the incremental process by which we eventually answer those questions in concrete terms. Here we attempt to map a MER command sequence into a goal-based sequence (Goal Network), provide a comparison of the two operations approaches, and further refine our understanding of the scope, process, and tools associated with the Goal-based adaptation of a real system.

II. State Analysis

A. State-based Control Architecture

State Analysis defines a uniform control architecture whose core principle is the explicit modeling of system state. State Analysis goes on to provide a rigorous methodology by which the architectural elements are populated. The resulting product set uniformly represents the systems engineers' understanding of the control system in terms

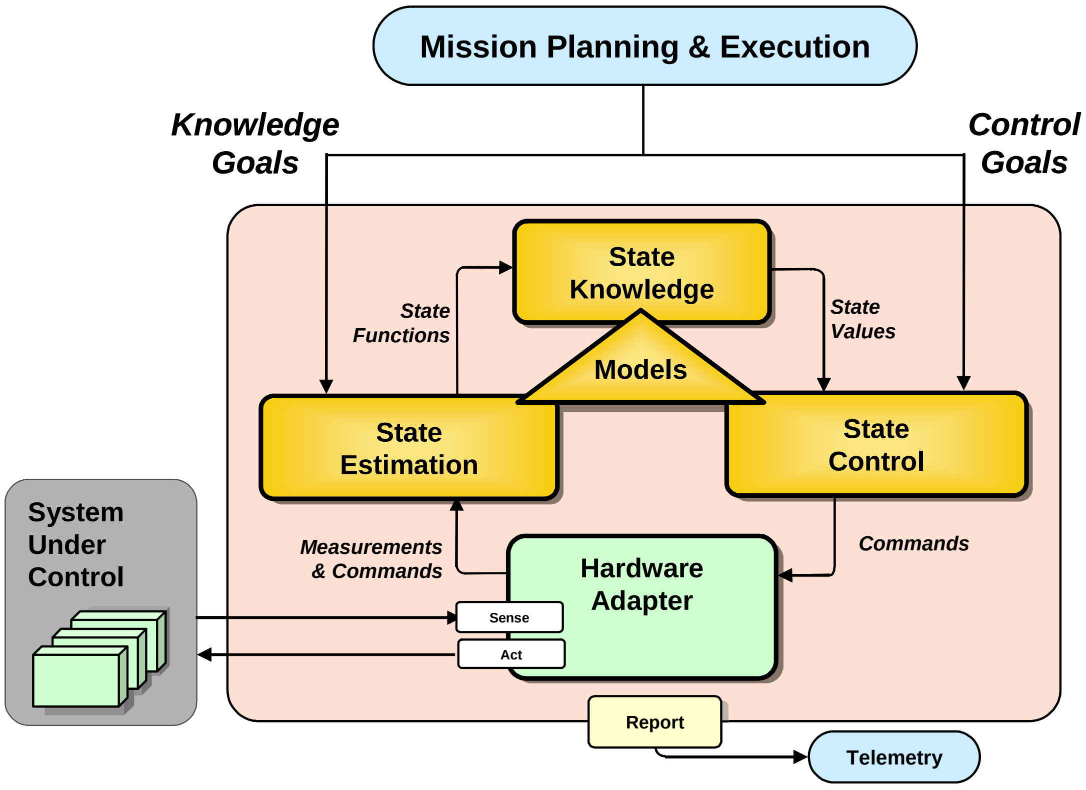


Figure 1. The state-based control architecture.

of the system under control.

Several key features of the control architecture can be seen in Fig. 1. *The representation of State knowledge is explicit*, in the form of continuous-time State Functions, such that the system's true state is defined at any point in time. *State estimation is separate from state control*, which promotes objective assessment of system state and simplifies design through modularity. *Hardware adapters provide the sole interface between the system under control and the control system*, forming an explicit boundary of the control architecture. *Models are ubiquitous throughout the architecture*. Whether overt and explicit, or hidden quietly in the minds of the engineers, models have always existed. Modeling is thinking. *Control is performed through Goal-directed operation*, enabling the system as well as operators themselves to directly reason about control objectives. *The architecture provides for a straightforward mapping of the architectural elements into components in a modular software architecture*, such as the Mission Data System (MDS). Although a detailed description of MDS is beyond the scope of this paper, we should note that the application of State Analysis in conjunction with MDS serves to bridge the fundamental gap between the requirements on software specified by systems engineers and the implementation of these requirements by software engineers.

B. State Discovery

State Analysis begins with the modeling of states of the system under control through an iterative process which incrementally constructs the model. The states include any physical value which we may want to control, which affect control, and may need to be estimated. The steps in this process are as follows:

1) Identify needs – define the high-level objectives for controlling the system.
2) Identify state variables that capture what needs to be controlled to meet the objectives, and define their representation.
3) Define state models for the identified state variables – these may uncover additional state variables that affect the identified state variables.
4) Identify measurements needed to estimate the state variables, and define their representation.
5) Define measurement models for the identified measurements – these may uncover additional state variables.
6) Identify commands needed to control the state variables, and define their representation.
7) Define command models for the identified commands – these may uncover additional state variables.
8) Repeat steps 2-7 on all newly discovered state variables, until every state variable and effect we care about is accounted for.
9) Return to step 1, this time to identify supporting objectives suggested by affecting states (a process called 'goal elaboration,' described later), and proceed with additional iterations of the process until the scope of the mission has been covered.

In State Effects Diagrams, state variables are depicted as ovals, measurements as triangles, commands as inverted triangles, and effects as arrows.

C. Goal-based Operations

A Goal is an expression of operator intent. It is a constraint on a State Variable's value history over a time interval. A Goal expresses *what* to do, not *how* to do it. The expression of a Goal is more compact and inspectable than a traditional command sequence, making it easier to see interactions and conflicts between activities. This benefits operators and software alike, both of which are required to reason about directives and resolve conflicts between their probable and desired result. Because goals specify intent in terms of state, they are inherently closed-loop, again giving both the operator and the software the ability to directly monitor the system for faults. Goals give the real-time control layer flexibility in achieving the goal.

The time interval associated with a goal is defined by a start and end *time point*, a potentially variable moment in time. Operators construct plans by instantiating goal types and interconnecting goal instances by either sharing time points or relating time points with temporal constraints, thus forming a *goal network*.

Because of the cause and effect relationship between states as modeled during the state discovery process, goals levied on one state variable may require supporting goals on other state variables which affect it. The process of expansion of a goal into supporting goals is directly informed by the state effects model and is called *elaboration*.

Goal elaborations are defined based on engineering judgment, our model of the system under control, and the following five rules:

1) A goal on a state variable may elaborate into concurrent control goals on directly affecting state variables (concurrent goals share the same start and end time points).

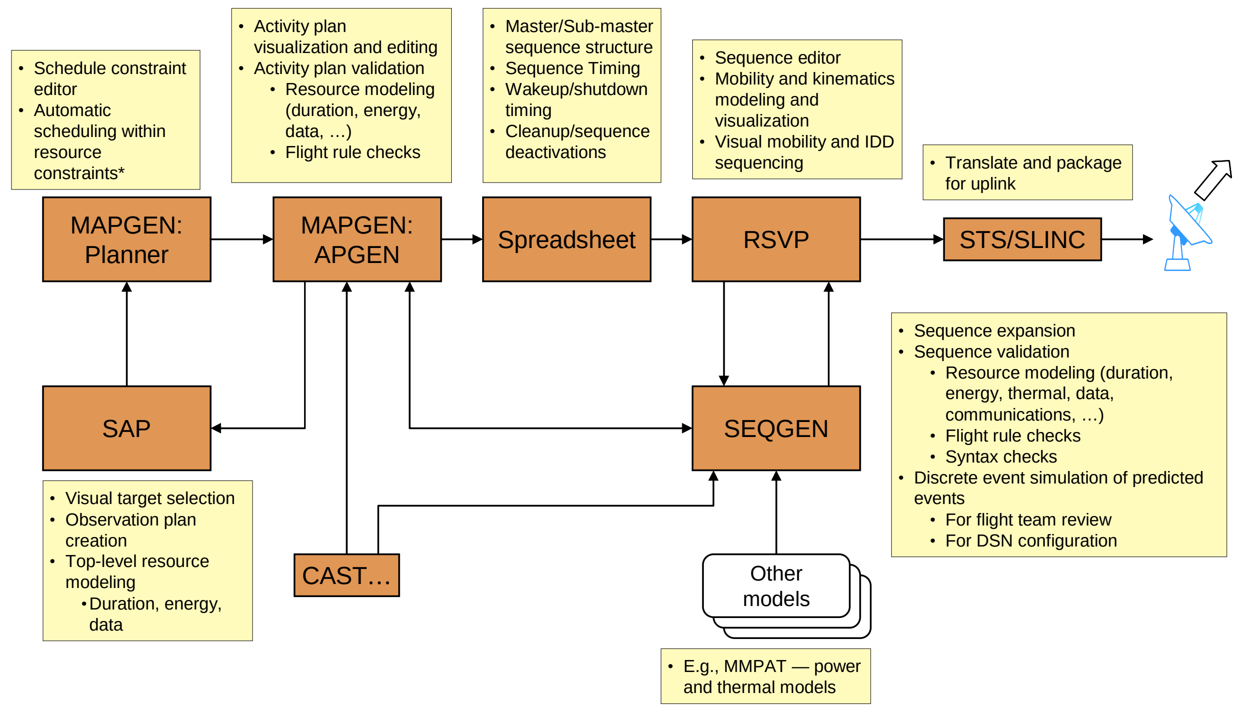
2) A control goal on a state variable elaborates to a concurrent knowledge goal on the same state variable (or they may be specified jointly in a single control and knowledge goal).
3) A knowledge goal on a state variable may elaborate to concurrent knowledge goals on its affecting and affected state variables.
4) Any goal may elaborate into preceding goals (typically on the same state variable). For example, a "maintenance"-type goal on a state variable may elaborate to a "transition"-type goal on the same state variable, which has an ending time point coincident with the starting time point of the "maintenance"-type goal.
5) A goal's elaboration can include uncertain temporal constraints to reserve time in the schedule for actions required by the goal.

Flight rules and constraints, resource management, fault monitoring, and control are all accounted for within goal elaborations. This consistent representation allows for direct reasoning and detection of conflicts between plans, and their likely result. Furthermore, given the model, state effects can be propagated not only across states variables, but also forward in time. This gives the planning system the opportunity to discover conflicts before they occur. Given a particular goal, and the control model, a controller can predict its likely performance over time. This process is called state *projection*. Since activities and resources are both expressed in terms of goals, projection becomes the process by which we model resource usage. By applying our elaboration rules, the effects of activities become goals on resources. This process can be automated and used onboard.

III. MER Operations Process

A. Overview

The MER uplink process is a sequential flow of products from tool to tool, usually with a perl script in between each, which translate products from one tool's format to the next. The process is executed on a daily basis and begins with the downlink team having already processed the latest data and determined the current state of the spacecraft. Over the course of two years on Mars, the process has evolved significantly. Described here is the process as it exists today.



• Schedule constraint editor
• Automatic scheduling within resource constraints*
• Activity plan visualization and editing
• Activity plan validation
• Resource modeling (duration, energy, data, …)
• Flight rule checks
• Master/Sub-master sequence structure
• Sequence Timing
• Wakeup/shutdown timing
• Cleanup/sequence deactivations
• Sequence editor
• Mobility and kinematics modeling and visualization
• Visual mobility and IDD sequencing
• Translate and package for uplink
MAPGEN: Planner
MAPGEN: APGEN
Spreadsheet
RSVP
STS/SLINC
SAP
SEQGEN
• Sequence expansion
• Sequence validation
• Resource modeling (duration, energy, thermal, data, communications, …)
• Flight rule checks
• Syntax checks
• Discrete event simulation of predicted events
• For flight team review
• For DSN configuration
• Visual target selection
• Observation plan creation
• Top-level resource modeling
• Duration, energy, data
CAST…
Other models
• E.g., MMPAT — power and thermal models

Figure 2. The MER Uplink Process.

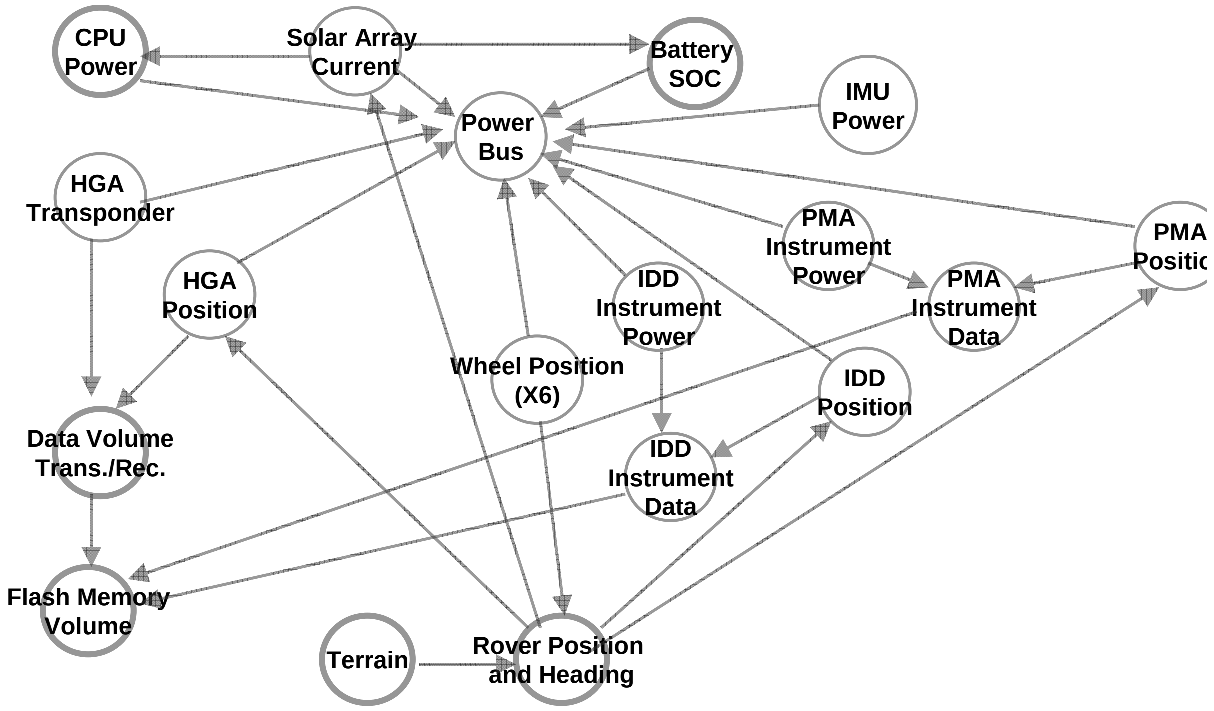


Figure 3. State Effects Diagram for Some Important MER State Variables.

B. Science Planning

The first order of business is science planning and the specification of science activities in the Science Activity Planner (SAP). Most significantly, SAP models the durations of activities and the volume of data each will produce. Within SAP, related activities are grouped together in Observations.

C. Activity Planning

Activity planning begins with a perl script which translates form the SAP format to a format used by the MAPGEN activity planning and scheduling tool. The relationship between activities and observations as specified in SAP carries over into MAPGEN. In addition to modeling activity durations and data volumes as in SAP, MAPGEN models a number of power related resources such as battery state of charge, solar array current, and battery minimum usable amp-hours. Also modeled in MAPGEN are conflicts between activities which claim the same discrete resources. For example, Mini-TES and pancam activities both require the use of the PMA.

D. Sequencing

Sequencing begins with annotation of the sequence summary. The MAPGEN tool creates a time ordered list of activities, which a perl script then translates into a file of tab-separated values called the sequence summary. The script assumes a one-to-one mapping of activities to sequences. The sequence ID is a parameter found within each activity, which can be edited in either SAP or MAPGEN. Annotation of the sequence summary adds directives to the spreadsheet which inform the subsequent translation from TSV to Rover Mark-Up Language (RML). RML is the XML file format used to specify sequences and commands. The translation from the sequence summary to RML is done by another perl script.

At this point the RML is ready to be refined using the Rover Sequence Editor (RoSE). RoSE allows users to instantiate commands from the command dictionary, piece them together in an ordered list, and place time constraints between adjacent commands.

Once complete, sequences in RML are translated, by a Perl script, into the actual uplink products radiated to the spacecraft.

IV. State Analysis Applied to MER

A. Opportunity Sol 63

In order to provide the richest set of MER activities from which to choose examples, Opportunity sol 63 was chosen as a representative sol due to the wealth of diverse activities executed in that particular plan. The plan included mobility, use of the IDD, a UHF communications relay with the Mars Odyssey Orbiter (ODY), pancam imaging, use of the Moessbauer Spectrometer (MB), remote sensing with the Mini-TES, CPU shutdown and wake-up, and direct-from-Earth communications using the high-gain antennae (HGA).

First we note some of the more visible state variables associated with a rover mission (Fig. 3), by starting with a position and heading state variable and applying the state discovery process described above. We then show other SED's concerning some important subsystems, such as power (Fig. 4), mobility (Fig. 5), and data management (Fig. 6). Note the mixture of resource state variables such as power with activity state variables such as position and heading.

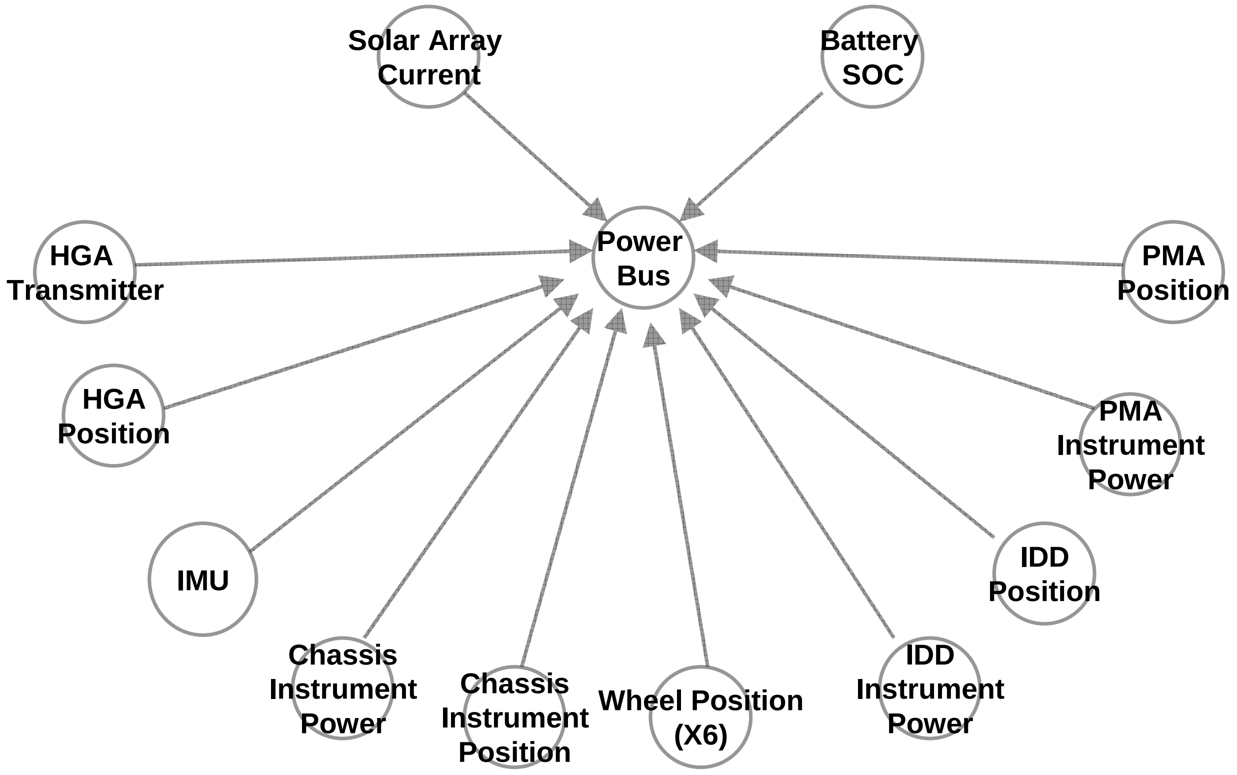


Figure 4. State Effects Diagram for Power-Related State Variables.

We now focus our exploration on an image mosaic using the microscopic imager (MI). The MI is particularly interesting because it involves several diverse activities, some of which are quite analogous to other rover activities. The MI is a camera much like the pancam, navcams, and hazcams, which perform mosaics in much the same way. Furthermore, it requires the IDD to place the MI on the appropriate target.

We begin by applying the elaboration rules mentioned earlier to the SED for science instruments which applies to pancam and MI alike. The result is individual goals for MI stacks interleaved with goals on IDD position as shown in Fig. 10. Note the introduction of the concept of a macro-goal. A macro-goal is a goal used to group related goals, but does not apply to a specific state variable.

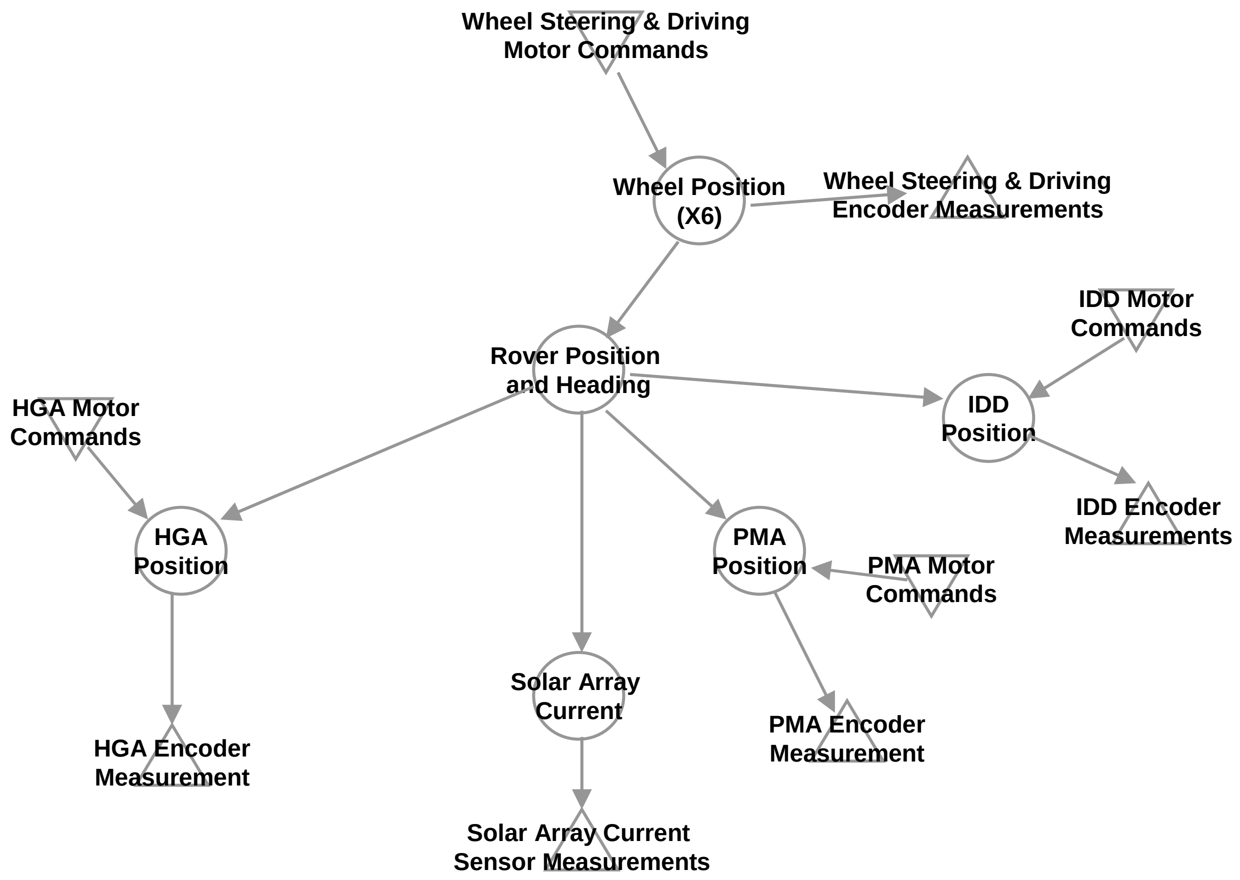


Figure 5. State Effects Diagram for Mobility-Related State Variables.

We then apply more iterations of our elaboration rules to the MI stack sub-goals, obtaining the elaboration diagrams shown in Fig. 11, Fig. 12, and Fig. 13.

B. State Effects Diagrams

State effects diagrams provide several important advantages. One model means consistency across project phases, tools, and roles. A member of the downlink team may be interested in assessing a trade between data volumes and data rates for communications, whereas a member of the uplink team may be more interested in trades between battery energy available and science observations. An engineer creating detailed elaboration and goal specifications during the formulation phase of a mission may be interested in state effects diagrams at a different level of detail than a member of the tactical operations team assembling plans on a daily basis.

State effects diagrams elevate design specification to a level that uplink personnel can understand in a timely manner. When implementing an activity a member of the uplink team will ask “what do I need to get this done”. This question is typically answered not by a tool, but by another operator. In effect, the team is going

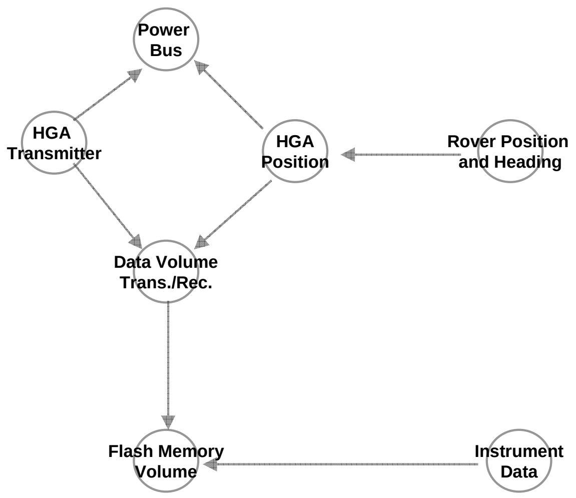


Figure 6. State Effects Diagram for Data-Related State Variables.

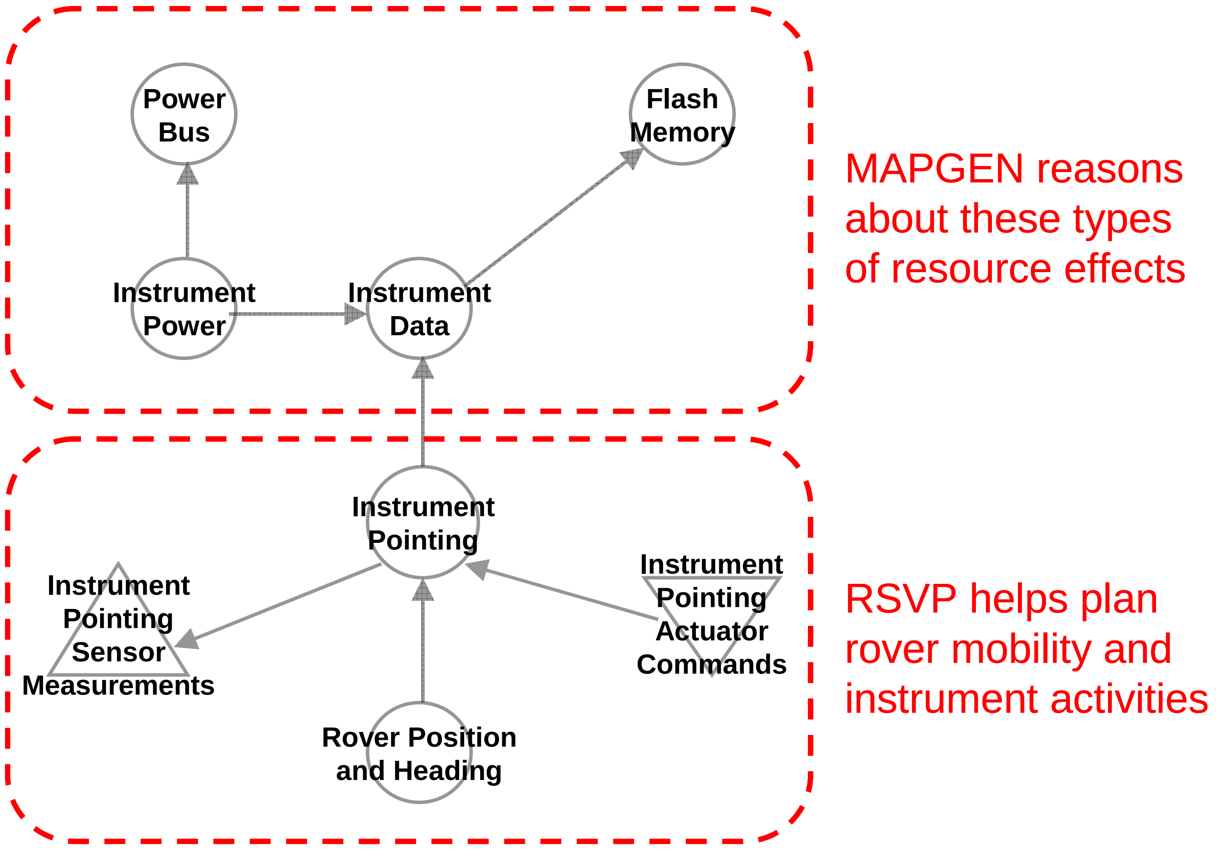

Power Bus
Flash Memory
MAPGEN reasons about these types of resource effects
Instrument Power
Instrument Data
Instrument Pointing
Instrument Pointing Sensor Measurements
Instrument Pointing Actuator Commands
RSVP helps plan rover mobility and instrument activities
Rover Position and Heading

Figure 7. Typical State Effects Diagram for State Variables related to a Science Instrument.

through the state discovery process every shift. This ad-hoc state discovery is error-prone and tedious. Why not capture the discovery process and shorten the panning process?

C. Goal-Based Operations

Since goals represent both activities and resources, actions and effects can be reasoned about side-by-side. Goals can overlap, because they eventually get merged by the planning system. This allows for optimization by consolidating redundant activities. State effects and elaborations allow the operators to trace resource violations back to the source. The mapping from activities to sequences is unclear. Sequence design patterns are not enforced much less standardized. This means that the state effects documented with state analysis are not enforced. Elaboration relationships allow for reusable patterns and enforce state effects, and macro-goals capture hierarchy design. Furthermore, the plan can be checked against an actual design document thus creating a direct verifiable link between design and operations.

Goal-based operations means consistency across tools, eliminating the need for conversion scripts. In the current product-flow paradigm, development progresses form one tool to the next through the exchange of data files. Reversing the flow through

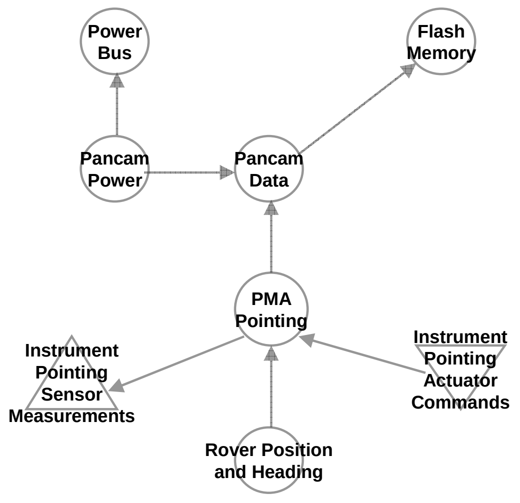

Power Bus
Flash Memory
Pancam Power
Pancam Data
PMA Pointing
Instrument Pointing Sensor Measurements
Instrument Pointing Actuator Commands
Rover Position and Heading

Figure 8. State Effects Diagram for Pancam-related State Variables.

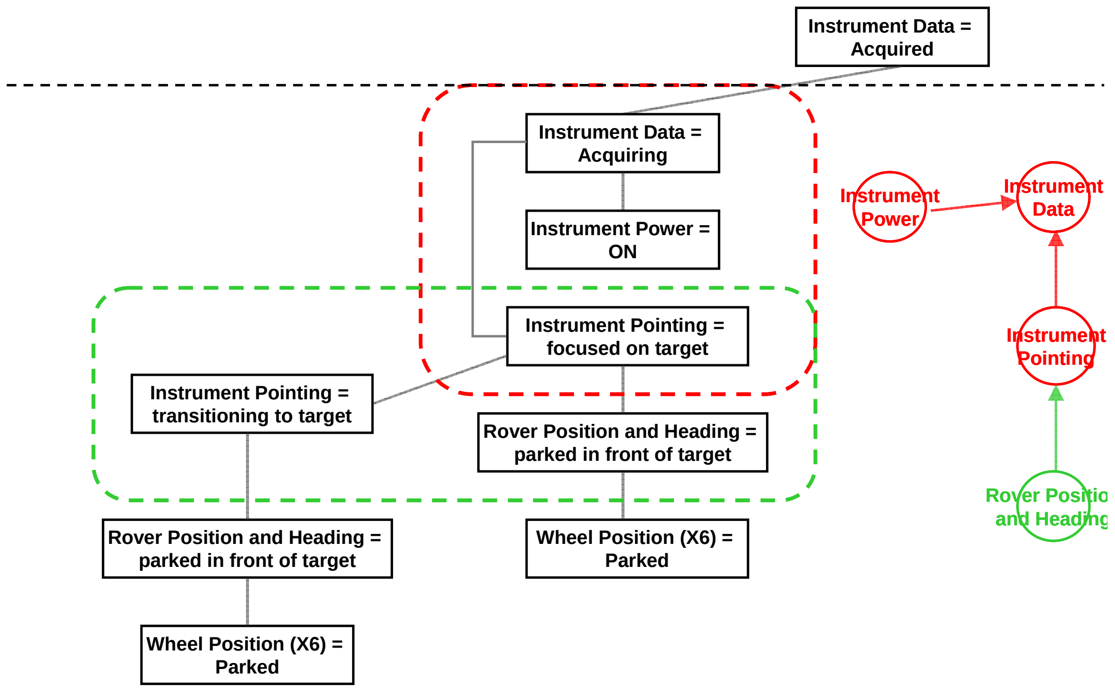

Instrument Data = Acquired
Instrument Data = Acquiring
Instrument Power = ON
Instrument Pointing = focused on target
Instrument Pointing = transitioning to target
Rover Position and Heading = parked in front of target
Rover Position and Heading = parked in front of target
Wheel Position (X6) = Parked
Wheel Position (X6) = Parked
Instrument Power
Instrument Data
Instrument Pointing
Rover Position and Heading

Figure 9. Elaboration Diagram for an Science Data Goal.

this process is awkward at best, and is usually avoided. Fixes are often made in place without the benefit of the functionality of preceding tools. Goals facilitate a shift to a work-flow paradigm, where one product set managed by a common tool undergoes successive stages of refinement. Reversing to address problems is straightforward. Suppose we need to re-plan after sequences have been delivered, under the current approach used by MER, we would have to add an activity in SAP, plan and re-model resources in MAPGEN, perform the human translation from activities to sequences, insert an instantiation of the sequence into the sequence hierarchy, and create and edit the sequence in RoSE. In a goal-based work-flow approach, we would simply add a goal, and let the tool elaborate the goal and model resources.

V. Extreme MER

Some potential benefits of state analysis and goal-based operations go beyond the ambitions of the MER process. Theoretically, we can cut the MER uplink process in half. With goal-based operations, modeling and control are expressed in the same language (goals), eliminating the need for the translation from activities to sequences. These two pieces typically take up equal amounts of time within the process.

If a state-based software framework like MDS is used onboard the spacecraft, both the flight software (state variables) and the plan (goals) are built directly from the same model (state effects diagrams), enabling operators to more easily understand the flight software and what it will do with the commands they send, mitigating operator need for resident flight software experts, "hand-me-down" understanding, and/or guess work. Resource modeling (projection) can be done onboard, even during plan execution, so problems can be forestalled, as opposed to "I hope nothing happens on Mars that I didn't model in MAPGEN". This can prevent the loss of planning cycles used to recover from faults.

Can we use Projection for long-term strategic planning? Suppose we want to be at a particular outcrop in time for the weekend, such that we can build a plan which will cover several days, thus allowing the engineers to take time off. Suppose we also want to have enough onboard memory available for this lengthy science campaign. Or even suppose we want to have a particular image mosaic on the ground in time for a big press conference. Why don't we use MAPGEN to plan such things a week at a time? We don't typically model these things so far in advance because the fidelity of our models, such as power, break down over time. But with state projection operating onboard the spacecraft, battery energy and solar array input measurements can inform planning onboard.

VI. Conclusion

References

Proceedings

[1]Ingham, M., Rasmussen, R., Bennett, M., and Moncada, A., “Engineering Complex Embedded Systems with State Analysis and the Mission Data System”, AIAA Journal of Aerospace Computing, Information and Communication, Vol. 2, No. 12, Dec. 2005, pp. 507-536.

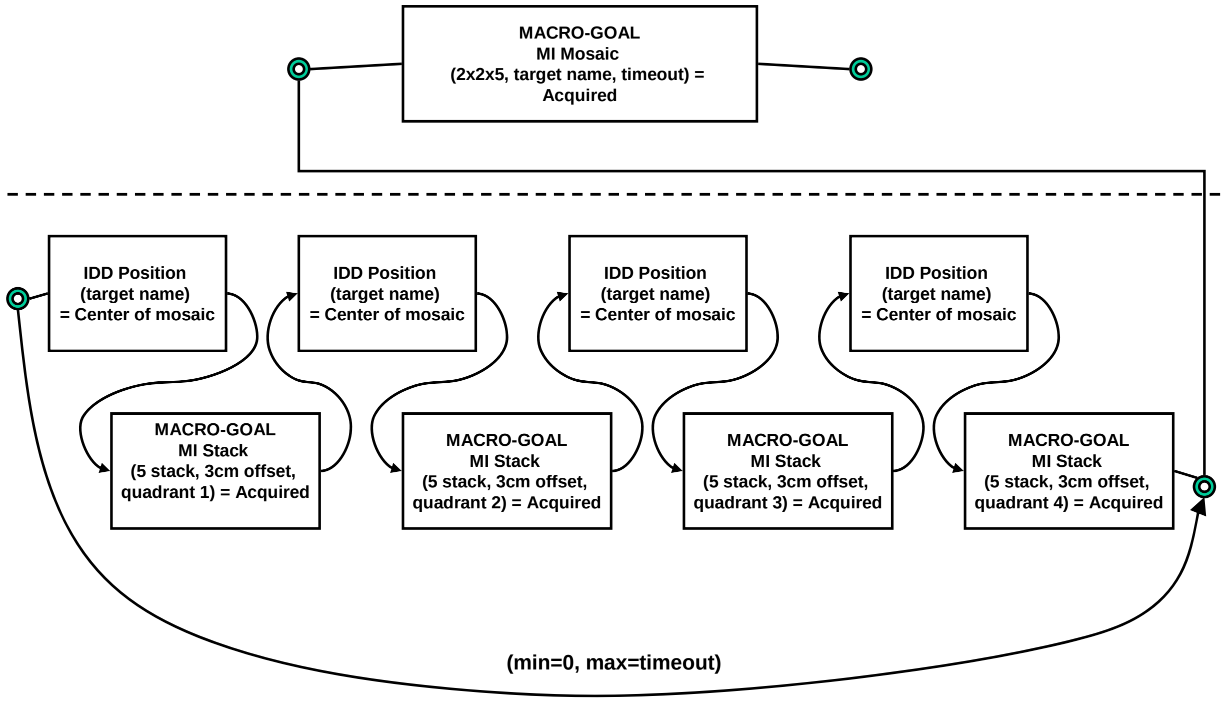

MACRO-GOAL
MI Mosaic
(2x2x5, target name, timeout) = Acquired
IDD Position (target name) = Center of mosaic
IDD Position (target name) = Center of mosaic
IDD Position (target name) = Center of mosaic
IDD Position (target name) = Center of mosaic
MACRO-GOAL MI Stack (5 stack, 3cm offset, quadrant 1) = Acquired
MACRO-GOAL MI Stack (5 stack, 3cm offset, quadrant 2) = Acquired
MACRO-GOAL MI Stack (5 stack, 3cm offset, quadrant 3) = Acquired
MACRO-GOAL MI Stack (5 stack, 3cm offset, quadrant 4) = Acquired
(min=0, max=timeout)

Figure 10. Elaboration Diagram for an MI Mosaic Goal.

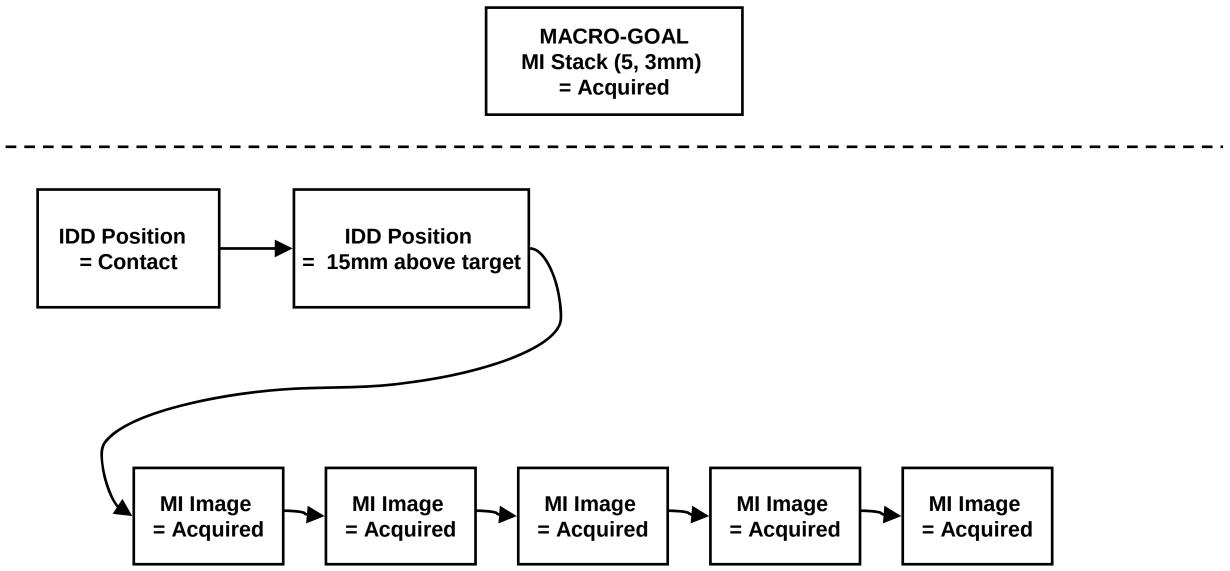


Figure 11. Elaboration Diagram for an MI Mosaic Goal.

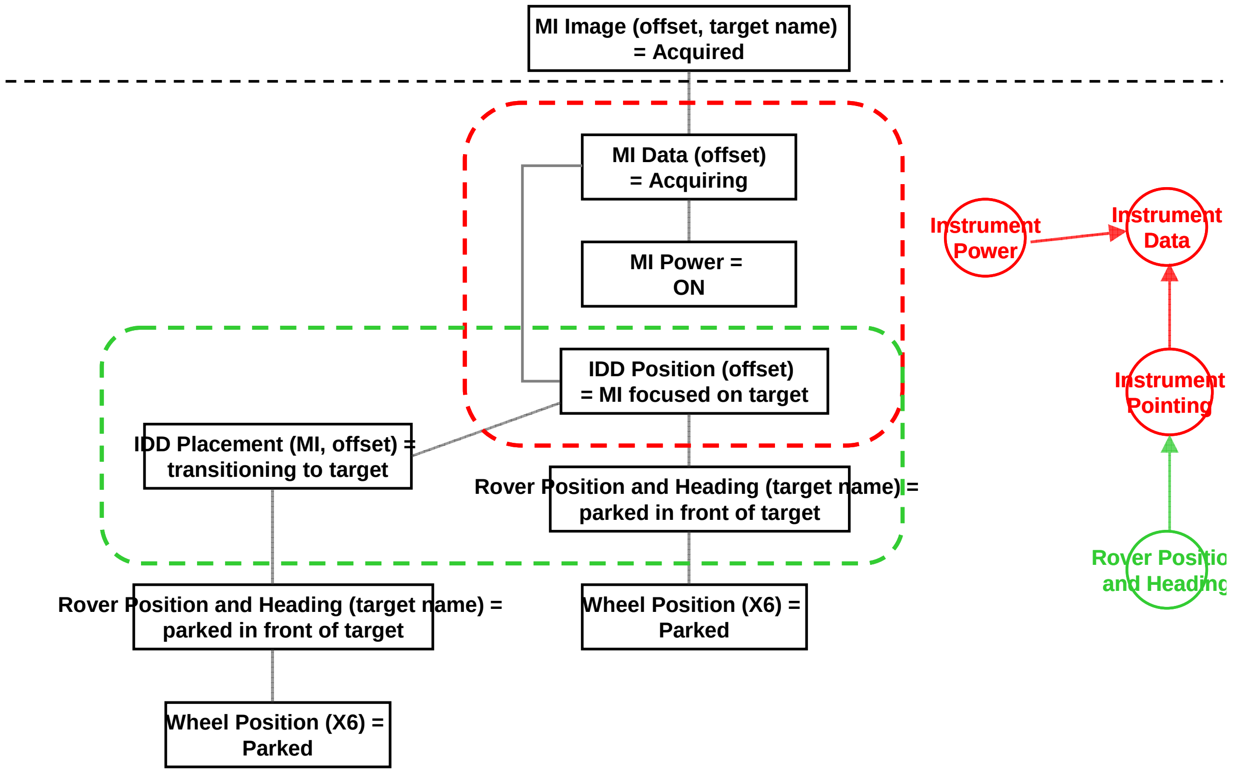
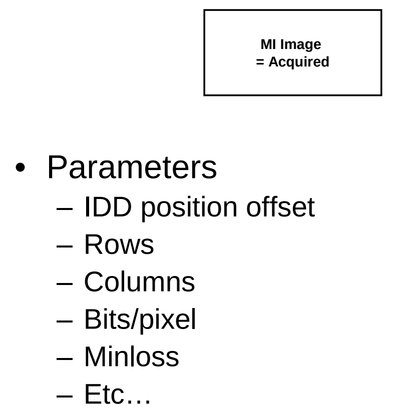


Figure 12. Elaboration Diagram for an MI Image Goal.



MI Image
= Acquired

- Parameters
  - IDD position offset
  - Rows
  - Columns
  - Bits/pixel
  - Minloss
  - Etc…

Figure 13. An MI Image Goal.

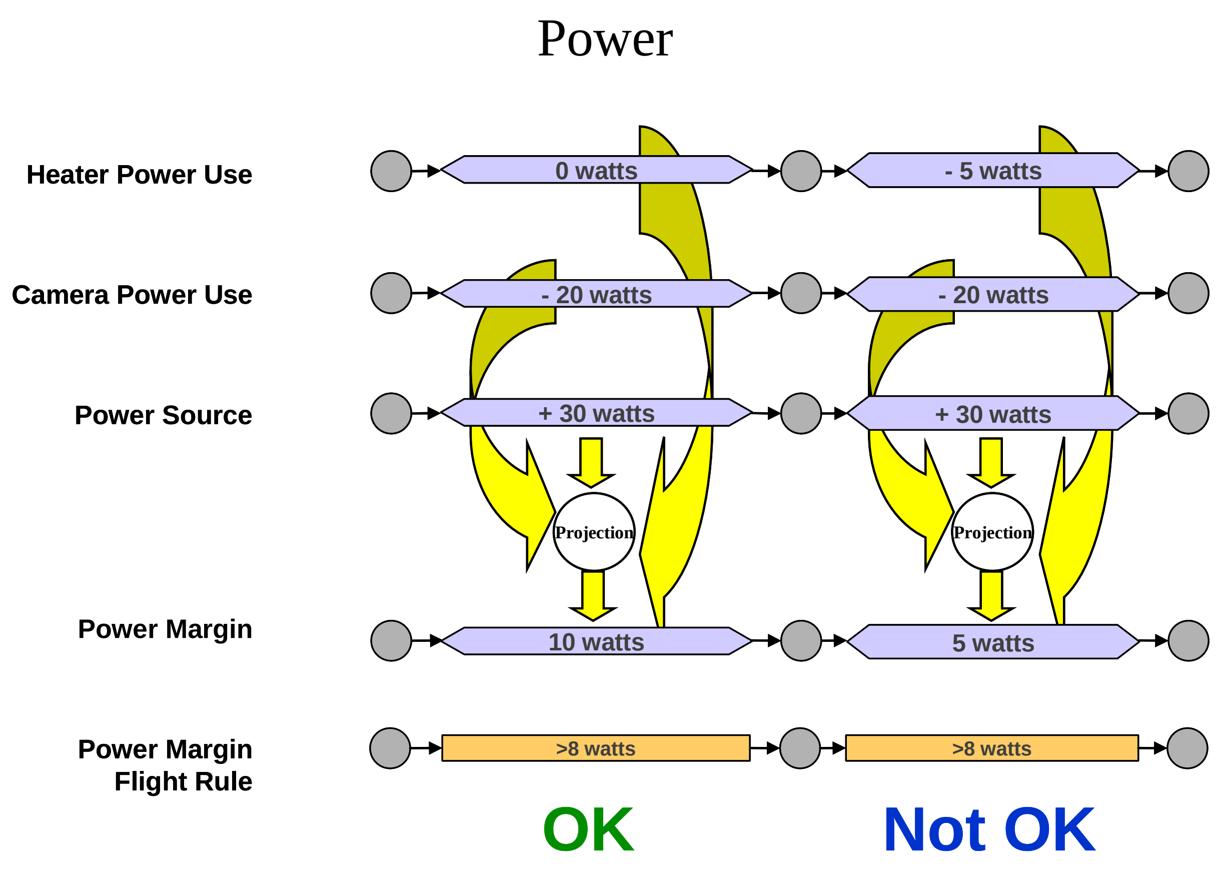


Figure 14. Projection for Power Goals.

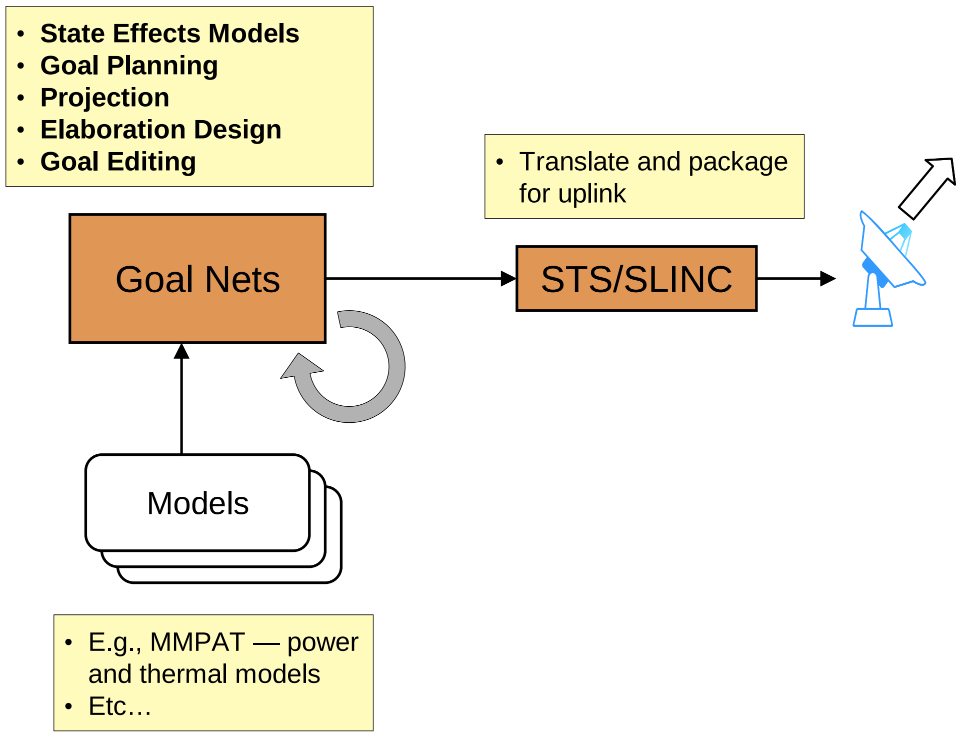


Figure 15. A Goal-Based Uplink Process.