SEQ_GEN: A Comprehensive Multimission Sequencing System

Jose Salcedo, Thomas Starbird

Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California

Abstract

SEQ_GEN is a user-interactive computer program used to plan and generate a sequence of commands for the spacecraft. Desired activities are specified by the user of SEQ_GEN; SEQ_GEN in turn expands these activities, deriving the spacecraft commands necessary to accomplish the desired activities. SEQ_GEN models the effects on the spacecraft of the commands, predicting the state as a function of time, flagging any conflicts and rule violations. These states, conflicts, and violations are viewable both graphically and textually at the user's request. SEQ_GEN also displays the entire sequence graphically, showing each requested activity as a bar on its graphical timeline. SEQ_GEN includes a full-screen editor, allowing the user to make changes to the requested activities. After a change has been made to the sequence, SEQ_GEN immediately revalidates the sequence, updating its models and calculations along with its displays based on these changes. Because SEQ_GEN is user-interactive and because it has the ability to recalculate spacecraft states immediately, the user is able to perform "what-if" sessions easily.

SEQ_GEN, a multimission tool, is adaptable to any flight project. A flight project writes its adaptation files containing project unique information including in its simplest form, only spacecraft commands. For more involved projects the adaptation files may also contain flight and mission rules, description of the spacecraft and ground models, and the definition of activities. SEQ_GEN operates at whatever level of detail the adaptation files imply. Simple adaptations are straight forward to do. There is, however, no limit to the complexity of activity definitions or of spacecraft models; both may involve unlimited logical decision points. Commands and activities may involve any number of parameters of a wide variety of data types, including integer, float, time, boolean, and character strings.

SEQ_GEN will be used by the Mars Pathfinder, Cassini, and VIM (Voyager Interstellar Mission) projects in an effort to speed up adaptation time and to keep sequence generation costs down.

SEQ_GEN is hosted on UNIX workstations. It uses MOTIF and X for windowing, and was designed and coded in an object-oriented style in the language C++.

Introduction

SEQ_GEN is a flexible software tool that can be used in several roles in the uplink planning and sequence generation process. In this paper, we address various tasks that are done during uplink planning and sequence generation, and show how SEQ_GEN supports each of them. We begin with comments that apply to all uses of SEQ_GEN.

Typically, SEQ_GEN is used interactively. The user sees results of SEQ_GEN computations on a graphical timeline (see Figure 1). If there are conflicts or rule violations, the user changes the sequence by using SEQ_GEN's editor to alter, add, or delete requests. SEQ_GEN then recomputes the state of state of the spacecraft, and reevaluates the rules. This process is repeated until the user is satisfied with the sequence, at which

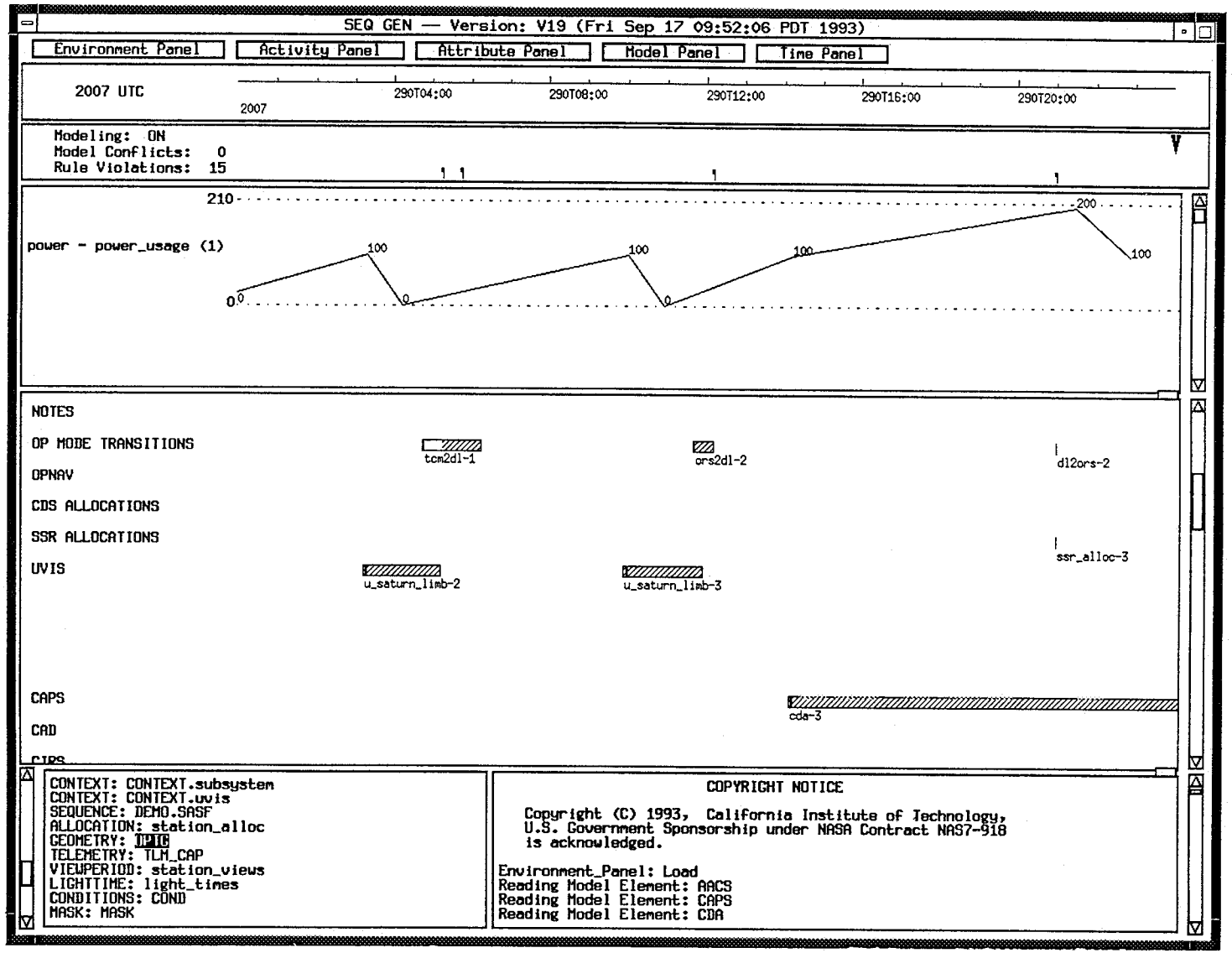

Figure 1. SEQ_GEN TIMELINE

time SEQ_GEN writes results into computer files.

A key feature of SEQ_GEN is that it is a multimission program. Adaptation of SEQ_GEN for use by a specific flight project in a specific role is done by supplying SEQ_GEN with files of data about the project. Only the information pertinent to the intended use of SEQ_GEN is required. In this paper we use the term "adapter" to denote the person or persons that supply the information about the flight project. The term "user" denotes a person who is using an adapted SEQ_GEN.

Now we discuss how SEQ_GEN supports various uplink planning and sequence generation tasks, showing SEQ_GEN's flexibility.

Generating Command-Level Sequences

Command-Level Editing

One simple use of SEQ_GEN is as an editor, enabling a person to write a sequence of spacecraft commands. The adapter provides the list of all spacecraft commands for the flight project. If the commands have parameters, those are named by the adapter. The adapter can specify what the allowable values are for each parameter of each command, and what type of value is appropriate (decimal integer, hexadecimal, octal, binary , floating point, duration, time, character string, boolean, or a one-dimensional array of any of the previous types). The adapter can also specify the default value of each parameter. In this way, SEQ_GEN "knows" a project's spacecraft commands.

When the user wants to add a command to the sequence, SEQ_GEN lists all the commands, letting the user choose one. SEQ_GEN displays the name and description of each parameter (as supplied by the adapter), to guide the user in specifying the requested command. SEQ_GEN will use whatever information the adapter has supplied concerning the parameters of the command. For example, if the adapter has supplied the allowable range, SEQ_GEN will warn the user when a value given by the user is not in range. SEQ_GEN's editor enables a person to form a request, consisting of one or more commands (and also "activities"; see below) and to add that request to the sequence.

Command-level Sequence

One output of SEQ_GEN in the simple adaptation described above is a file of all the commands (in mnemonic form), in time order, ready to be translated to bits and sent to the spacecraft.

In addition, SEQ_GEN produces a timeline (see Figure 1), both interactively on the screen, and on paper. The timeline shows visually the position in time of each request in the sequence.

Such an adapted SEQ_GEN is useful for building sequences for use before launch in the testing of the spacecraft. It is also useful for simple projects where command-level sequence planning is adequate, and where any constraints on interactions of commands can be checked by hand.

Spacecraft Clock

If precise timing of commands with respect to the spacecraft's clock is vital, the adapter can define the units of the clock and their nominal durations. The definition is then used in some of SEQ_GEN's calculations. For example, there is an option in SEQ_GEN to align all commands' times to the nearest whole unit in the spacecraft's clock. The relation between Universal Time and the values of the spacecraft's clock is given to SEQ_GEN at run-time, to account for differences in the clock's rate from its nominal rate.

Merging Sequences

Another feature of SEQ_GEN is the ability to merge sequence files. For example, SEQ_GEN could be used individually by different flight team members making their individual request files. Those files can then be merged to produce a single time-ordered file with all of the requests. Each request retains its requestor's name (or other identifying string), so that the individuals can check that their requests were properly handled.

Different requestors could include members of the engineering team (for example, an attitude control analyst requesting a calibration), or of the navigation team (requesting a maneuver), or of science teams (requesting scientific observations).

Predicting Events

It is often useful to predict the effects of the commands in a sequence. The adapter can supply models to SEQ_GEN that enable predictions of the state of the spacecraft based on the commands in the sequence.

Flexibility of Models

A nice feature of SEQ_GEN is the variability possible in the models. One possibility, of course, is to have no models at all. In this case, as discussed above, SEQ_GEN's output is the time-ordered list of commands in the sequence.

Models of varying complexity can be added. For example, if the amount of power being used at any time during the sequence is of interest, a power model could be added. The adapter defines a model element by specifying its attributes (i.e., state variables). An attribute can be of any type (same choices as for parameters of a command; see above), and the adapter can define the allowable range of each (in which case SEQ_GEN will give a warning to the user if the attribute's value ever becomes out of range). For each spacecraft command that affects an attribute, the adapter describes the effect, using a simple language provided by SEQ_GEN. The language includes the basic programming language constructs, such as IF statements and loops. In addition, the language C can be used by the adapter to specify calculations. No compiling or linking of SEQ_GEN is needed to incorporate the adapter's compiled C code; the linking of the adapter's code is dynamic, done at run-time.

The effect of a command can depend on the state of the model before the command. The most common effect of a command is to change the value of an attribute.

Simple models, such as ones that keep track of whether a switch is "on" or "off", are simple for the adapter to specify. Each project can model the details appropriate for its sequencing needs.

The modeling done in SEQ_GEN is a discrete event simulation, where the commands in the sequence are the triggering events. SEQ_GEN processes each command by interpreting the simple language in which the adapter has written the effect of the command, and by calling any C functions the adapter may have used. The adapter can use SEQ_GEN's "stimulus" concept to promote the effect of a command to future time or to several model elements.

SEQ_GEN has built-in the ability to read files of Deep Space Network view periods and allocations, a file that contains predictions of downlink data rate capability, and a file that contains trajectory events, such as occultations. The adapter can write effects of such events in the same way as writing effects of commands. For interplanetary missions, where the light time is non-negligible, SEQ_GEN has the capability of adjusting times between ground time

and spacecraft time using a file giving the light time.

Predicted Events File

SEQ_GEN produces a comprehensive file that contains the results of the modeling (see Figure 2). The file is a time-ordered list that contains an entry whenever an attribute of a model is set to a value. The entry consists of the time, the values of the attributes of the model element, and an indication of the causal command. The file also lists all commands in the sequence. (Activities and rule violations are also in the file; see below.) The file can be used to review a sequence.

Interactive Display of Models

The user of SEQ_GEN can turn the modeling on or off at will. The user can also have SEQ_GEN display a graph of the value of any one or more attributes above the timeline of requests (see Figure 1). The user can change what to display any time during the SEQ_GEN session. When the user changes the sequence, SEQ_GEN models the part of the sequence being viewed and updates all the displays.

Thus the adapter has great flexibility in what models to build and how detailed to make them, and the user has complete flexibility in choosing what model attributes to display on the screen during the session.

Different Users, Different Models

Even on a single flight project, different adaptations of SEQ_GEN could be used. For example, an attitude control expert may include more detailed models of attitude, but omit models of interest only to a scientist, and vice versa.

Checking Rules

The adapter can add "rules", which are stated in terms of the model attributes. SEQ_GEN has eight types of rules. A rule contains a boolean-valued expression of model attributes. During modeling of a sequence, if the expression becomes true (or remains true for too long, or for not long enough, or becomes true too many times, or not enough times, or becomes true before some other state has occurred for long enough), the rule is considered violated. An indication of the violation occurs above the timeline of requests (see Figure 1). The user can click on the indication to get details of the violation. Rule violations are also included in the Predicted Events File.

By defining rules, the adapter enables SEQ_GEN to perform some of the validation of a sequence.

For situations where none of the eight built-in types of rule adequately reflects the constraint desired to be checked, the adapter can use logic in the models themselves to declare a conflict. An indication of conflict appears above the timeline (see Figure 1), and appears in the Predicted Events File.

Thus SEQ_GEN is flexible in the rules it can check. Just as different users could use different models, so they could use rules tailored to their interest.

Making High-Level Requests; Activity Types

SEQ_GEN offers flexibility in the level at which a user requests commands for the sequence. The adapter can define "activity types" (also called "blocks"), which can then be used in users' requests.

A simple activity type is a list of spacecraft commands, with their relative timing specified. The activity type has a name. By requesting an activity of that name, the user is effectively adding all the commands in the activity type's definition to the sequence, timed relative to the time specified for the request.

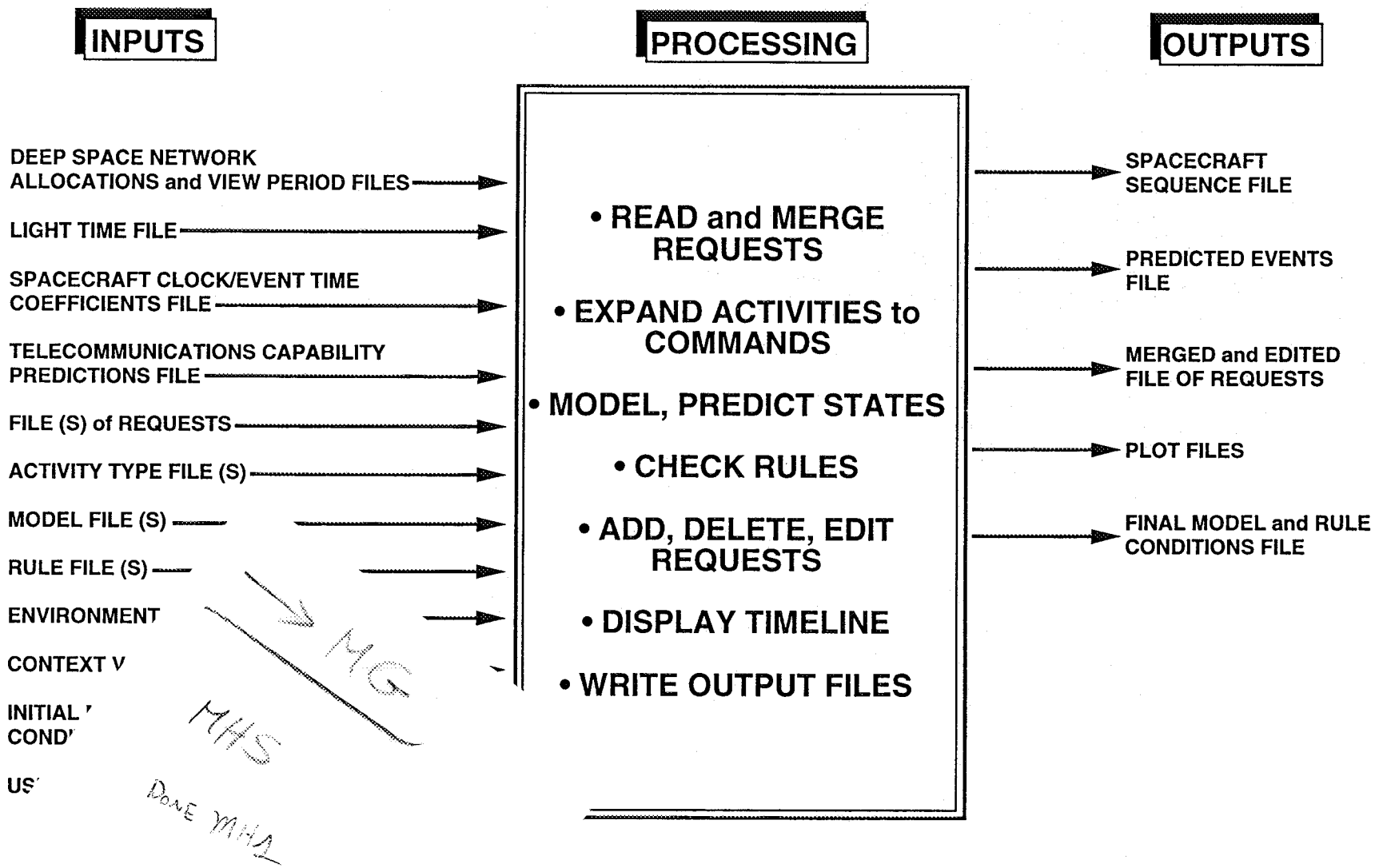

Figure 2. SEQ_GEN FUNCTIONS

SEQ_GEN is flexible in how complicated the definition of an activity type can be. An activity type can have parameters. The user, when requesting an activity of that type, is prompted by SEQ_GEN's editor for values of the parameters. The values can be used for parameters in commands that appear in the definition of the activity type. The values can also be used in logical constructs (such as IF statements) that govern what commands will be used in the activity. For example, in an activity type that represents a maneuver of the spacecraft, a parameter could be an option determining whether or not to turn on the gyroscopes.

The definition of an activity type can refer to other activity types (i.e., activities can be nested).

Activity types are "expanded" by SEQ_GEN to produce commands. The commands are modeled along with any commands requested explicitly by the user.

Using activities allows the user to think at a higher level than individual commands. Also, the definition of an activity type can be written or checked by experts, and tested before use. A person who is not an expert can then safely use the activity.

Some activity types represent on-board programs that can be invoked in a sequence to yield several commands. Such an activity type, called an on-board block, is expanded by SEQ_GEN for modeling, but is not expanded on the Spacecraft Sequence File (see below).

Writing the Spacecraft Sequence File

Another output of SEQ_GEN is a computer file called the Spacecraft Sequence File. This file contains (a mnemonic representation of) the information that must actually go to the spacecraft, i.e., spacecraft commands and calls to on-board blocks. Conversion of this file to binary in a form packaged for transmission to the spacecraft is not a function of SEQ_GEN.

Planning without Commands; d_commands

Activity types can actually be defined even if spacecraft commands have not been defined. SEQ_GEN has the concept of d_commands (dummy commands), which are requested by the user and modeled by SEQ_GEN as commands are, but which are not placed in the Spacecraft Sequence File. In this way, an adaptation of SEQ_GEN can be made wherein activity types are defined in terms of d_commands, which can trigger abstract or approximate models. An example of an abstract model is one telling whether a maneuver is in progress. Such an adaptation is useful for planning sequences early in the planning stage, or early in the life of the project.

Both actual commands and d_commands can be used in the same activity type and in a single sequence. Thus modeling and rule checking involving actual commands can be supplemented by modeling and rule checking of abstractions.

Changing Adaptation

The adaptation information is given in ASCII files (plus optional C code in the model or activity definitions). The adaptation can be changed as the mission progresses. Another program, called SEQ_ADAPT, is being developed to aid the adapter in producing syntactically correct and consistent adaptation files.

History and Use of SEQ_GEN

SEQ_GEN (under different names) has its historical roots in the Mariner Mars 1971 project, a Mars orbiter. Most major later projects at the Jet Propulsion Laboratory, including Voyager and Galileo, wrote new versions specific to the project. In the last few years, the

current version, which is a multimission version, was developed.

Its activity features were used on Mars Observer. It will be used on Mars Pathfinder, VIM (Voyager Interstellar Mission), and Cassini. SEQ_GEN is hosted on Sun SPARC and Hewlitt-Packard workstations.

Development of SEQ_GEN

SEQ_GEN has about 55,000 lines of code, written in C++ in an object-oriented style (Wirfs-Brock et al., 1990). SEQ_GEN is Category A; it was developed with full rigor and testing.

Summary

SEQ_GEN is a comprehensive and flexible tool for use in uplink planning and sequence generation. SEQ_GEN is flexible in that

- it can be adapted for use in any flight project, or for different classes of user in a single project
- it can be adapted in several versions, with or without spacecraft commands, models, rules, and activity types
- models can be simple or detailed
- models can be of actual spacecraft parts and/or of abstract quantities
- models can be triggered by spacecraft commands or by d_commands
- adaptation does not require compiling or linking of SEQ_GEN

Acknowledgement

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

The SEQ_GEN program was developed by Russ Brill, Imin Lin, Win Lombard, Bob Oliphant, John Sisno, Jose Salcedo, and Tom Starbird.

References

McLaughlin, W.I. and Wolff, D.M., "Automating the Uplink Process for Planetary Missions", AIAA 89-0580, AIAA 27th Aerospace Sciences Meeting, Reno, January 9-12, 1989.

Salcedo, J., "Version 19 SEQ_GEN User Guide," D-11261, December 1, 1993 (JPL internal document)

Wirfs-Brock, R., Wilkerson, B., & Wiener, L.(1990). *Designing Object-Oriented Software.* Englewood Cliffs, New Jersey: Prentice Hall.