Sneak Circuit and Software Sneak Analysis

S. G. Godoy* and G. J. Engels†
Boeing Aerospace Co., Houston, Texas

An overview of the development and application of a systems analysis technique will be presented in this paper. The technique is based on an aerospace discovery that topological criteria exist that can be used to recognize unplanned operational modes of a system. Software sneak analysis is based on successful electrical sneak circuit analysis techniques which have been applied to over 70 commercial, military, nuclear, and space systems. The similarities of electrical current and software logic flow implied the possibility of an analogous technique for software analysis. The technique involves encoding software data for computer processing. The computer processing produces simplified, topological network trees which represent the software system. The network trees are analyzed by the application of sneak clues to identify and report all sneak conditions. There have been seven successful applications of software sneak analysis to computer systems involving over twelve different computer languages. The results obtained from a variety of complex electrical and software systems analyses will also be presented as positive collaboration for this analysis technique.

Sneak Circuit Analysis: Background[1-3]

A SNEAK circuit is a latent path or condition in an electrical/electronic system which inhibits a desired condition or initiates an unintended or unwanted action. A sneak circuit is not caused by component failures, but is a condition that has been inadvertently designed into an electrical/electronic system. Sneak circuits often exist because many subsystem designers lack the overall system visibility required to electrically interface all subsystems properly. Some sneak circuits are evidenced as "glitches" or spurious operation modes and can occur in mature, thoroughly tested systems after long use. Sometimes sneaks are the real cause of problems blamed on electromagnetic interference or grounding "bugs."

The unpredictable nature of sneak circuit conditions prompted the National Aeronautics and Space Administration (NASA) to fund an investigation to determine a method of identifying sneak circuit conditions before their occurrence could pose any possible threat to the safety of Apollo, Skylab, or Apollo-Soyuz test project crew members.

In November 1967, the Boeing Aerospace Company began development of a technique of circuitry analysis to detect sneak circuits based on two significant aspects of historical sneak circuits: 1) sneak circuits are universal in complex electrical systems and their analogs and 2) topological criteria exist which enable recognition of all planned or unplanned operational modes within a system. The technique involved: 1) The simplification and reduction of electrical system detail schematics to topological network trees, 2) the recognition of the basic topographical patterns inherent in all circuitry, and 3) the application of the appropriate clues which have been found to characterize sneak circuit conditions.

Sneak Circuit Methodology

Network Tree Production

The first major consideration that must be satisfied to identify sneak circuit conditions is to insure that the data being used for the analysis represent the actual "as-built" circuitry of the system. Functional, integrated, and system level schematics do not always represent the actual constructed hardware. Detail manufacturing and installation schematics must be used, because these drawings specify exactly what was built, contingent on quality control checks, tests, and inspection. However, manufacturing and installation schematics rarely show complete circuits. The schematics are laid out to facilitate hookup by technicians without regard to circuit or segment function. As a result, analysis from detail schematics is extremely difficult. So many details and unapparent continuities exist in these drawings that an analyst becomes entangled and lost in the maze. Yet, these schematics are the data that must be used if analytical results are to be based on true electrical continuity. The first task of the sneak analyst is, therefore, to convert this detailed, accurate information into a form usable for analytical work. The magnitude of data manipulation required for this conversion necessitates the use of computer automation in most cases.

Automation has been used in sneak circuit analysis since 1970 as the basic method of tree production from manufacturing detail data. Computer programs have been developed to allow encoding of simple continuities in discrete "from-to" segments extracted from detail schematics and wire lists. The encoding can be accomplished without knowledge of circuit function. The computer connects associated points into paths and collects the paths into node sets. The node sets represent interconnected nodes that make up each circuit. Plotter output of node sets and other reports are generated by the computer to enable the analyst to easily sketch accurate topological trees. The computer reports also provide complete indexing of every component and data point to its associated tree. This feature is especially useful in cross-indexing functionally related or interdependent trees, in incorporating changes, and in troubleshooting during operational support.

Topological Pattern Identification

Once the network trees have been produced, the next task of the analyst is to idenitfy the basic topological patterns that appear in each tree. Five basic patterns exist: 1) single line (no-node) topograph, 2) ground dome, 3) power dome, 4) combination dome, and 5) "H" pattern. These patterns are illustrated in Fig. 1. One of these patterns or several in combination will characterize the circuitry shown in any given network tree. Alhough, at first glance, a given circuit may

Presented as Paper 77-1386 at the AIAA/NASA/IEEE/ACM Computers in Aerospace Conference, Los Angeles, Calif., Oct. 31-Nov. 2, 1977; submitted Oct. 31, 1977; revision received April 19, 1978. Copyright © American Institute of Aeronautics and Astronautics, Inc., 1977. All rights reserved.

Index categories: Testing, Flight and Ground; Computer Technology; Spacecraft Testing, Flight and Ground.

*Engineer, Space Systems Division. Associate Fellow AIAA.

†Specialist Engineer, Space Systems Division.

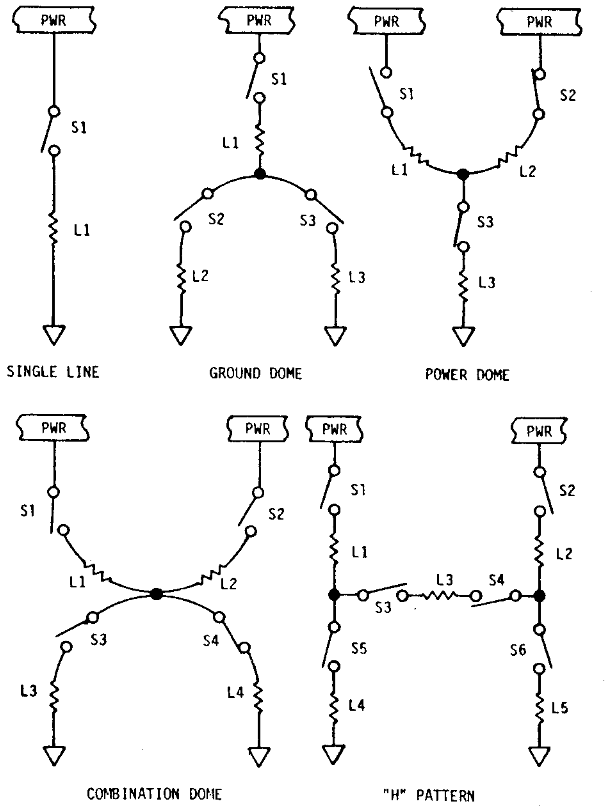


Fig. 1 Basic topographs.

appear more complex than these basic patterns, closer inspection reveals that the circuit is actually composed of these basic patterns in combination. As the sneak circuit analyst examines each node in the network tree, he must identify the topographical pattern or patterns incorporating the node and apply the basic clues that have been found to typify sneak circuits involving that particular pattern.

Clue Application

Associated with each pattern is a list of clues to help the analyst identify sneak conditions. These clues are questions that the analyst must ask about the circuitry in question. As an example, the single line topography (Fig. 1), would have clues such as:

1) Can switch S1 be open when load L1 is desired?
2) Can switch S1 be closed when load L1 is not desired?
3) Does the label of S1 reflect the true function of L1?
4) And so forth.

Obviously, sneak circuits are rarely encountered in this topograph because of its simplicity. Of course, this an elementary example and is given primarily as the default case which covers circuitry not included by the other topographs.

With each successive topograph, the clue list becomes longer and more complicated. The clue list for the "H" pattern includes over 60 clues. This pattern, because of its complexity, is associated with more sneak circuits than any of the previous patterns. Almost half of the cirtical sneak circuits identified to date can be attributed to the "H" patterns. Such a design configuration should be avoided whenever possible. The possibility of current reversal through the "H" crossbar is the most commonly used clue associated with "H" pattern sneak circuits and will be illustrated in the following example. (Fig. 2).

The Reluctant Redstone

The circuitry presented in Fig. 2 represents part of the ignition control circuitry for a Redstone booster used during the Mercury program in the early 1960's.[4] This schematic would not be encoded for sneak circuit analysis because it is not considered to be a "manufacturing level" schematic. It does serve to illustrate, however, the difficulty involved in separating a given circuit from its surrounding circuitry and simplifying it to the extent necessary for an effective analysis. The network tree for the circuitry outlined with dotted lines is presented in Fig. 3. As can be seen in this tree, the computer suppresses all connectors, terminal blocks, unnecessary nodes, splices, and any other extraneous circuit elements that do not affect the actual electrical continuities involved. Only active circuit elements remain on the completed network tree. When the network tree has been obtained, a sneak circuit analyst examines each node in sequence, determining which of the basic topographs the node is a part of and applying the clues that pertain to those topographs. After all such clues have been applied, all sneak circuits present in that network tree will be identified. In the Redstone booster case, the pertinent clue is the "H" pattern clue relating to reverse currents as previously discussed. This clue would be stated as: "Is it possible for current to flow in both directions through the "H" crossbar?"

If the answer to this question is yes, a sneak circuit is not automatically indicated, as this may have been the design intent. The analyst must determine whether or not this a desired condition within all foreseeable circumstances. In the case of the Redstone circuit, the condition definitely was not desired. The design intent was that the Cutoff Command relay contacts or Pad Abort switch should energize the Engine Cutoff coil and the Abort Indicator coil. When the Launch Command contacts in this tree close, they should turn on the Launch indicator lights only. The possible reverse current can only exist when the ground below the indicator lights is lost. This would occur if the Tail Plug umbilical opened before the Control umbilical separated. If this happened, current would flow as indicated by the arrows through the Launch Command relay contacts, the Launch indicator lights, through the suppression diode of the Abort Indicator coil, and finally through the Engine Cutoff coil to the ground. By this means, the Launch Command can activate the Engine Cutoff coil if the Tail Plug umbilical separates before the Control umbilical.

Unlikely, as this event may appear, it did occur on November 21, 1960.[4] After more than 50 sequentially successful Redstone booster launches, a Redstone booster with a Mercury capsule on it was launched. When the ignition command was given, the booster fired. After lifting several inches from the pad, the engine inexplicably cut off. The booster settled back on the pad. The Mercury capsule separated and deployed its parachutes. Damage was slight, but a highly explosive rocket with no means of control sat on the pad for 28 hours. No one dared to approach the Redstone booster until its batteries drained down and the liquid oxygen evaporated. Extensive investigation revealed that the Tail Plug umbilical had separated 29 milliseconds prior to Control umbilical disconnect. This was enough time for the sneak circuit shown in Fig. 3 to occur and abort the mission.

Sneak Circuit Analysis: Results

Since 1967, sneak circuit analyses[5-8] have been performed on more than 70 complex space systems, nuclear power systems, military systems, and commercial systems. Without exception, critical sneak circuit conditions have been identified in every system investigated, even those that have been thoroughly analyzed by other means, comprehensively tested, simulated, breadboarded, and operated for a number of years. Sneak circuit analysis has proved applicable to almost any complex electrical system involving either analog or digital circuitry or both.

The success of electrical sneak circuit analysis encouraged attempts to apply topological pattern recognition techniques to other complex systems. In particular, the similarites of

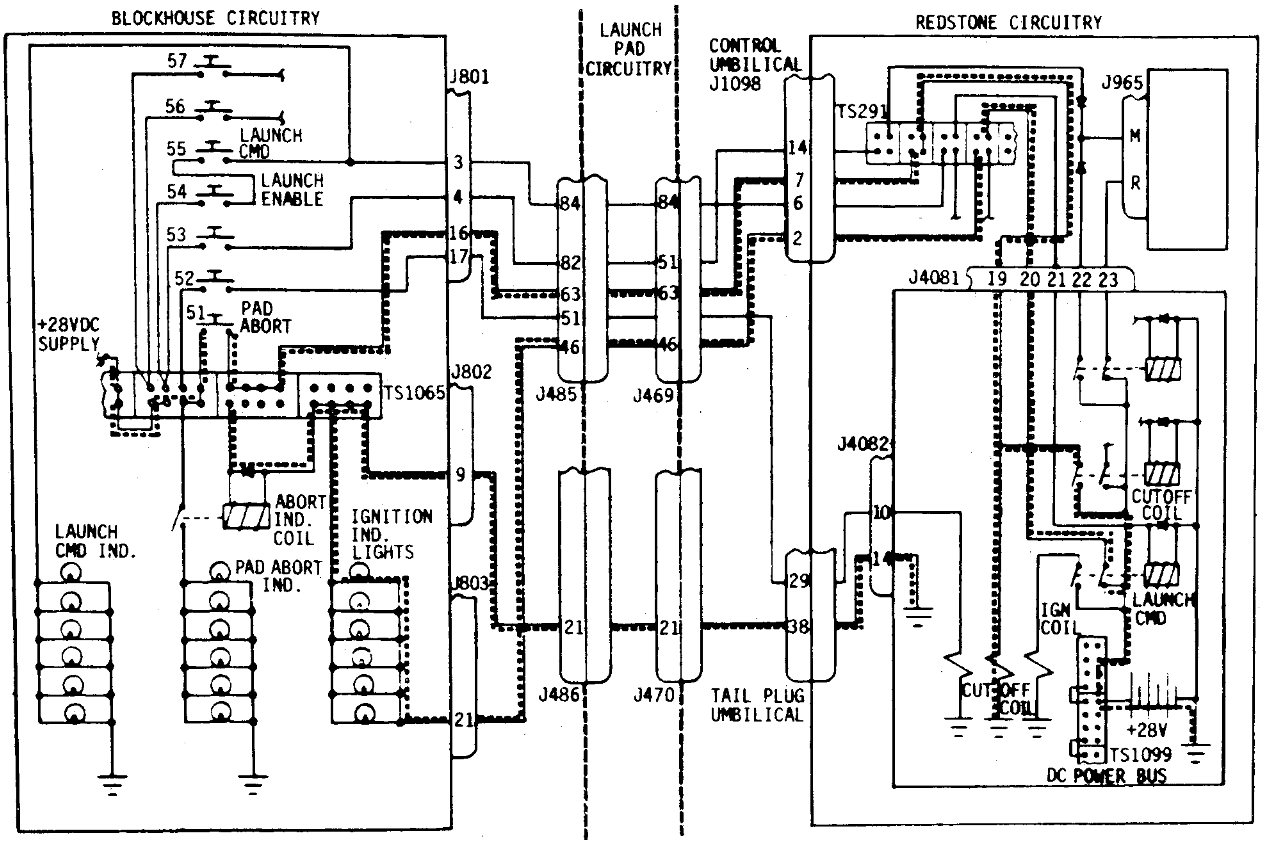


Fig. 2 Reluctant Redstone simplified schematic.

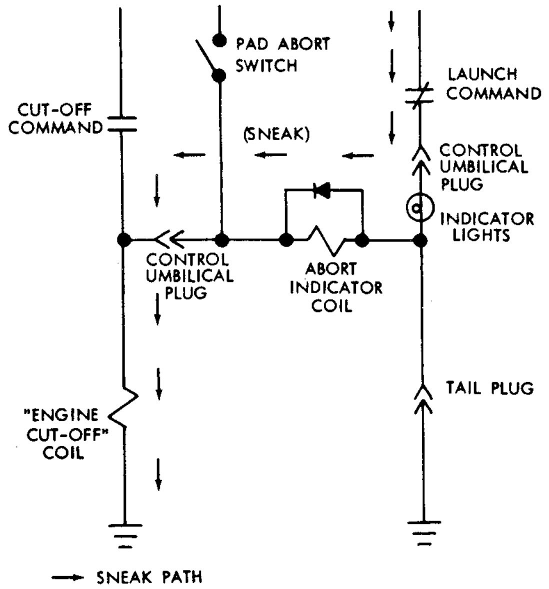


Fig. 3 Redstone network tree.

electrical current and software logical flow implied the possibilty of an analogous technique for software analysis. With the increasingly complex software systems being generated and the frequent interaction of complex software and hardware systems, it became necessary to develop such a technique.

Software Sneak Analysis

In 1970, after three years of successful application of the sneak circuit analysis technique, the Boeing Aerospace Company began seriously studying methods of analyzing software systems with analogous techniques.[9-11] For several years, general approaches to software sneak analysis (SSA) were investigated and case histories of software problems were studied. In 1975, a feasibility study was performed which resulted in the development of a formal technique involving the use of mathematical graph theory, electrical sneak theory, and computerized search algorithms which are applied to a software package to identify software sneaks. A software sneak is defined as a logic control path which causes an unwanted operation to occur or which bypasses desired operation without regard to failures of the hardware system to respond as programmed.

During the feasibility study, the software sneak analysis techniques were applied to several test routines in three different assembler languages. The subroutines contained a number of known problem areas which were successfully identified by the SSA techniques. In addition, 46 software problems, not previously identified by other methods, were uncovered. The feasibility study concluded that SSA: 1) is a viable means of identifying certain classes of software problems 2) works equally well on different software languages, and 3) does not require execution of the software to detect problems. SSA was successfully applied only to individual subroutines during the feasibility study. Since multiple subroutine software systems are likely to involve different categories of problems from a single subroutine, it was felt that SSA should next be applied to a total operating software system. In August of 1976, an analysis of the terminal configured vehicle (TCV) software system was performed for NASA Langley Research Center. During this analysis, the SSA techniques were modified and expanded to include subroutine-to-subroutine interactions to enable an analysis of a total software system package. The technique was modified to provide sufficient cross-referencing to allow tracing of both logic flow and data flow through the software network trees. In this manner, the effect of a given piece of data can be traced through the network trees to evaluate its impact/effect on the total software system.

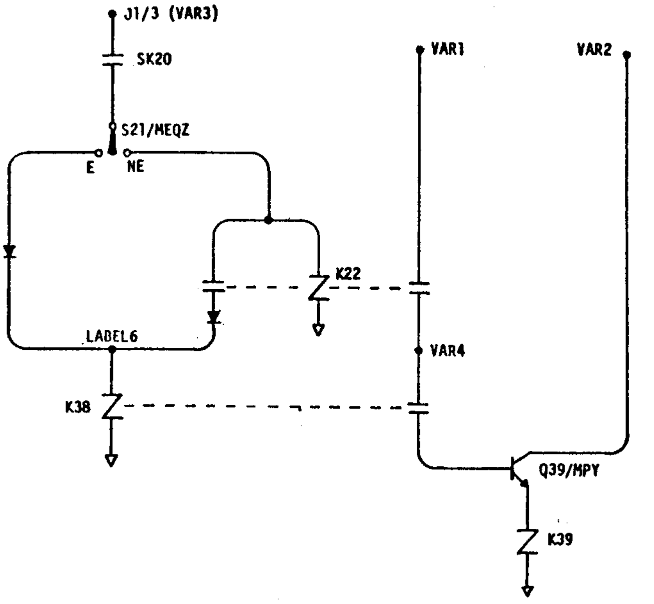


Fig. 4 Example software sneak.

At Boeing, the software sneak analysis technique has evolved along lines very similar to electrical sneak circuit analysis. Topological network trees are used with electrical symbology representing the software commands to allow easy cross-analysis between hardware and software trees and to allow the use of a single standardized analysis procedure.

Software Sneak Methodology

Since topological pattern recognition is the keystone of both sneak circuit analysis and software sneak analysis, the overall methodologies are quite similar. The software package to be analyzed must be: 1) codified and reduced to a standardized topographical format, 2) the basic topological patterns identified, and 3) the appropriate problem clues applied to each pattern.

Network Tree Production

Unlike electrical system elements, software elements occur in a wide variety of formats. High-level languages, such as Fortran, Cobol, and PL1, and the lower level languages, such as the many assembly languages, must all be represented by some common symbology to allow an efficient analysis. Fortunately, regardless of the language used, there are only a limited number of discrete operations that any given command can invoke. Arithmetic operations, branches, conditional branches, data manipulation statements, etc. form the building blocks of all other commands. Since a single symbology was to be chosen more or less arbitrarily to represent these basic elements, electrical circuit symbols were chosen to facilitate total system analysis by having all the system's network trees appearing in a single format. Table 1 illustrates the basic symbols used. Unusual software instructions can have other electrical symbols assigned for a given analysis. Coding techniques, computer processing, and network tree production is performed in much the same manner as in electrical systems using this symbology.

A software network tree appears the same as its electrical counterpart. Figure 4 below illustrates a software sneak which occurred in the operating software of a military aircraft. The network tree of the actual software code makes the sneak readily apparent. This historical problem was uncovered during the software system integrated testing when bad output from a particular subroutine was detected. Obviously, it would be preferable to identify as many potential software problems as possible before the software is run.

Topological Pattern Recognition

In identifying the basic topological patterns in software nodal sets, the inapplicability of the "H" pattern (see Fig. 1) to software becomes apparent. The primary characteristic of the "H" pattern is that current can flow in both directions in the crossbar. Since the node-to-node branches in software always have a directed flow of logic, there is nothing analogous to a reverse current flow. In software, therefore, the "H" pattern breaks up into the other basic topographs. In all other respects, however, topographical pattern recognition is the same as in electrical sneak ciruit analysis.

Table 1 Software-electrical analogies

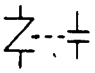
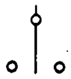
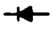
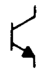
Electrical symbol	Software item
	Data or logic transfer
	Test
	Label or data
	Path
	Program flow
	Arithmetic

Table 2 Typical problem categories

Conventional	Digital logic	Software logic
• Relay race	• Incorrect signal polarities	• Unused paths
• Backfeed	• Load exceed drive capabilities	• Inaccessible paths
• Current reversal	• Wiring interconnections incorrect	• Improper initialization
• Ground switch	• Improper voltage levels	• Lack of data storage usage synchronization
• Bus ties	• Excessive fanout	• Bypass of desired paths
• Power supply crossties	• Potential race conditions	• Improper branch sequencing
• Conflicting commands	• Lack of system synchronization	• Potential undesirable loops
• Ambiguous labels	• Inadequate system reset	• Infinite looping
• False indicators		• Incorrect sequencing of data processing
• Intermittent or random glitches		• Unnecessary (redundant) instructions

Table 3 Sample sneak circuit and software sneak results

	Sneaks	Design concern	Document error
Electrical systems:			
Skylab	259	91	307
N reactor safety subsystems	18	13	22
Bay Area Rapid Transit	25	21	24
F-8 digital-fly-by-wire	19	7	28
E-3 AWACS power system	57	2	194
Software systems:			
f1 Terminal configured vehicle	20	1	3
Incipient failure detector	4	1	1
Sneak circuit subroutines	7	...	...
B1 avionics subroutines	35	...	...

Clue Application

Once the basic topographs have been identified, the clue lists compiled during the historical case studies are applied, and software sneaks are identified. The basic clues are generally the same as in sneak circuit analysis, but the sneaks identified are quite different. Table 2 identifies some of the categories of problems found in software, as compared with analog circuitry. Selected results from several analyses are tabulated in Table. 3.

Integration of Hardware/Software Analysis

After a sneak circuit analysis and a software sneak analysis have been performed on a system, the interactions of the hardware with the system software can readily be determined. The analyst has at his disposal diagramatic representations of these two elements of the system in a single standardized format. The effect of a control operation that is initiated by some hardware element can be traced through the hardware trees until it impacts the system software. The logic flow can then be traced through the software trees to determine its ultimate impact on the system. Similarly, the logic sequence of a software-initiated action can be followed through the software and electrical network trees until its eventual total system impact can be assessed.

The joint analysis of a system's software and electrical circuitry previously described is termed simply Sneak Analysis. This system safety tool helps provide visibility of the interactions of a system's hardware and software and hence will help reduce the difficulties involved in the proper integration of two such diverse, complex systems designs. As hardware and software systems increase in complexity, the use of interface bridging analysis tools, such as Sneak Analysis, becomes imperative to help provide assurance of safety of the total system.

References

[1] Clardy, R. C., "Sneak Circuit Analysis Development and Application," *1976 Region V IEEE Conference Digest,* 1976, pp. 112-116.

[2] Rankin, J. P., "Sneak Circuit Analysis," *Nuclear Safety,* Vol.14, Sept.-Oct. 1973, pp. 461-487.

[3] Hill, E. J. and Bose, L. J., "Sneak Circuit Analysis of Military Systems," *Proceedings of the 2nd International System Safety Conference,* July 1975, pp. 351-372.

[4] "First Mercury-Redstone Flight Test Fails On Pad," *Aviation Week*, Nov. 28, 1960.

[5] "Sneak Circuit Analysis of N Reactor (Prepared for Atomic Energy Commission Richland Operations Office)," Boeing Aerospace Company, Rept. D2-118542-1, July 1974.

[6] "Skylab Saturn Workshop Sneak Circuit Analysis," Boeing Aerospace Company, Rept. D2-118461-1, May 1973.

[7] "Sneak Circuit Analysis of the AWACS Electrical Power System," Boeing Aerospace Company, Doc. No. D2-118547-1, Oct. 1974.

[8] "Sneak Circuit Analysis of Bay Area Rapid Transit Door Control System," Boeing Aerospace Company, Doc. No. D2-118611-1, March 1977.

[9] "Software Sneak Analysis," CDRL No. A005 of Contract F2901-75-C-0069, Oct. 1975.

[10] "Application of Software Sneak Analysis to the Incipient Failure Detection System," Boeing Aerospace Company, Houston, Texas, Sept. 1976.

[11] "Application of the Software Sneak Analysis to the Terminal Configured Vehicle System," Boeing Aerospace Company, Doc. No. D2-118594-1, Aug. 1976.