AN EXPERT SYSTEMS APPROACH TO DECISION SUPPORT
IN A TIME-DEPENDENT, DATA SAMPLING ENVIRONMENT
(A BRIEF DISCUSSION)

Captain Peter G. Raeth
Air Force Systems Command
Armament Division
3246th Test Wing /TZWT
Electronic Warfare Test Division
Eglin AFB, FL 32542-5424

ABSTRACT

What framework can be devised to assist analysts in coping with the growing volume of data needed to support the analysis of today's complex systems? Such a framework should organize and use an analyst's knowledge in an automated environment and go beyond merely plotting raw data or charting statistics derived from raw data.

This paper discusses the author's approach to the use of raw data which greatly reduces the volume of data while increasing its value for analysis and decision-making. Consistent application of an analyst's knowledge in an automated environment readily transforms raw data into information and the information into recommendations. The approach discussed in this paper concentrates first on capturing the analyst's knowledge. The method permits the incremental, and fairly random, capture of knowledge at several levels of detail. The human view of the knowledge base is transformed into the computer view which is then applied to the raw data. Two previous papers based on the partial implementation of these techniques are referenced.

INTRODUCTION

In order to make recommendations about a system or to determine how a system reacted during a given period of time, analysts must process a huge volume of raw data. The sheer volume of data can be staggering, especially in fields such as electronic warfare systems testing. The resulting difficulty in sifting useful information from the data leaves little time for coming to meaningful conclusions. The analyst needs a consistent method of applying knowledge about a system. This method should work well in an automated environment which converts raw data into meaningful information and recommendations. The intent is to quickly determine what a system did during a given period of time and make recommendations accordingly.

The flow of data from its raw form to suggested recommendations is illustrated in Figure 1. As the data move through this scenario, they become lower in volume but higher in value relative to the goal of the analysis.

Figure 1 illustrates three transformation phases: raw data collection, state analysis, and recommendation determination. The computer must pass the raw data through the state analysis and recommendation phases and deliver appropriate reports using the knowledge supplied by the analyst. Time periods which contain no active states and later, no offered recommendations, can be deleted from consideration. Each of the three phases will now be discussed.

RAW DATA COLLECTION

The collection and organization of raw data is a crucial step in any automated analysis. Data from each system in the test must be gathered, recorded, converted, and passed on to the state analysis phase (Figure 2).
### List Of Active System States

| C1 | C2 | ... | Cn |
|----|----|-----|----|
| T1 |
| T2 |
| T3 |
| ... |
| Dij |
| Tj |

### List Of Offered Recommendations

| R1 | R2 | ... | Rz |
|----|----|-----|----|
| T1 |
| T2 |
| T3 |
| ... |
| ... |

### Definitions

- **C**: Channel Number
- **T**: Time Stamp
- **Dij**: Data Value in Cell \(ij\)
- **S**: System State
- **MSij**: State Mode in Cell \(ij\) (Active Or Inactive)
- **R**: Recommendation
- **Mrij**: Recommendation Made in Cell \(ij\) (Offered Or Not)

**NOTE:** Time blocks with no active states or recommendations can be deleted.

---

**Figure 1**

Transforming Raw Data Into Relevant Information
System Under Test (#1) ... System Under Test (#n)
| Data Sampling Process |
|-----------------------|
| \ /                   |
| Analog Recording      | Analog Recording |
| | Analog-to-Digital Conversion |
| \ /                   |
| Digital Recording     | Digital Recording |
| ______________________|
| Software to Unpack Data |
| \ /                   |
| Channel vs Time Data Matrix |
| | Computational Software |
| \ /                   |
| Original Data Merged With Related Data |
| \ /                   |
| To State Analysis Inference Engine |

Figure 2
The Development of Raw Data for the State Analysis Inference Engine
There are two types of raw data, measured and derived. Measured data come directly from system sampling. Derived data are the result of computations performed on measured data. Examples of derived data are Fourier Transforms, averages, slopes, standard deviations, and other results which can be directly computed using measured data. Derived data are simply merged with measured data to create additional data channels. This results in a two-dimensional data matrix where the channels are the columns and the time stamps are the rows. The state analysis phase addresses this matrix, either directly or indirectly, to determine what the system did during any given period of time. The data to be collected during the raw data collection phase are determined by the requirements of the state analysis phase.

**STATE ANALYSIS**

Two things must happen to support the state analysis phase. First, the knowledge of the analyst must be captured. The supporting mechanism must permit incremental, and fairly random, knowledge capture. Second, this knowledge must be used in a consistent manner, along with the raw data, to determine what the system did during any given period of time.

**Capturing the Analyst's Knowledge.** The knowledge of the analyst must be gathered and organized into a detailed description of how system states are recognized in the raw data. Based on extensive interviews with experienced analysts, three basic categories make up this knowledge base:

- **States:** internal or external conditions of the system which remain active for a given period of time. [See MacFarlane for a good discussion.]

- **Events:** correlations of raw data which must be observed in time sequence before a state is determined to be active.

- **Specification Sets:** triplets which identify a data channel, a data value, and a numerical comparison operator (\(<\), \(\rangle\), \(=\), etc.). A list of these sets make up an event. The sets in an event must all be recognized at the same time before the event is determined to have occurred.

Accept-Only-If and Reject-Only-If rules take the form:

\[
\text{If } Q_1 \text{ then If } Q_2 \text{ then } \ldots \text{ If } Q_n \text{ then accept/reject this state, event, or set.}
\]

Where \(Q_k\) can be a state name or a specification set.

An example of a Reject-Only-If rule is:

If BAD-TIME then Reject. Means that the state, event, or set that the rule is tied to will be determined to not exist only if the time stamp is incorrect.

An example of an Accept-Only-If rule is:

If SUB-SYSTEM-B then Accept. Means that Sub-System-B must be active before the indicated state, event, or set can be meaningfully identified.

In addition to the three basic categories mentioned above, there are four categories of related knowledge which must also be collected:

- **Dwell Time:** the minimum length of time an event must last.

- **Transition Time:** the maximum length of time between the start of one event and the beginning of the next event.

- **Accept-Only-If Rules:** rules which must be satisfied before a state, event, or set is determined to exist. These rules are tied to a particular state, event, or set and are invoked only if the given state, event, or set is indicated in the raw data. Acceptance is not accomplished unless all the rules are satisfied.

- **Reject-Only-If Rules:** rules which must be satisfied before a state, event, or set is determined to not exist. These rules are tied to a particular state, event, or set and are invoked only if the given state, event, or set is not indicated in the raw data. Rejection is not accomplished unless all the rules are satisfied.

If no rules are given in the Accept-Only-If or Reject-Only-If categories, then the inference engine can proceed with the analysis without evaluating additional information. Accept-Only-If and Reject-Only-If rules allow the analyst to specify judgment factors which can, in specific instances, override the integrity constraints.

Accept-Only-If and Reject-Only-If rules take the form:

\[
\text{If } Q_1 \text{ then If } Q_2 \text{ then } \ldots \text{ If } Q_n \text{ then accept/reject this state, event, or set.}
\]

Where \(Q_k\) can be a state name or a specification set.

An example of a Reject-Only-If rule is:

If BAD-TIME then Reject. Means that the state, event, or set that the rule is tied to will be determined to not exist only if the time stamp is incorrect.

An example of an Accept-Only-If rule is:

If SUB-SYSTEM-B then Accept. Means that Sub-System-B must be active before the indicated state, event, or set can be meaningfully identified.
It is very important that an analyst-oriented mechanism be devised for the capture of these seven categories of knowledge. The mechanism chosen must use the language of the analyst and must be simple enough for the analyst to incrementally add knowledge in a fairly random fashion. One mechanism for doing this is the expanded spreadsheet shown in Figure 3.

| STATE NAME | EVENT TIMING | SPECIFICATION SETS |
|------------|--------------|---------------------|
|            | DWELL TIME   | CHN VALUE COP       |
|            | TRANSITION TIME | CHN VALUE COP      |

(Spreadsheet-like Input Screen)

Action Applies To Covered:
(Move Cursor To Cover Appropriate State, Event, Or Set)

|   | F1 | F2 |
|---|----|----|
| F3 | F4 |
| F5 | F6 |
| F7 | F8 |
| F9 | F10 |

**REQUIRED OPERATIONS**
odd <Fx> = Edit Accept Rules
even <Fx> = Edit Reject Rules
ddd <ALT><Fx> = Insert Above/Before
even <ALT><Fx> = Delete State, Event, Or Set
ddd <CTRL><Fx> = Edit State, Event, Or Set

<> = Keyboard Key To Press
COP = Comparison Operators ( =, <, >, etc)
CHN = Data Channel

**Figure 3**
Knowledge Entry Mechanism
For The State Analysis Phase
Using the extended spreadsheet shown in Figure 3, the analyst could easily capture state names, their related event, and specification set groupings, as well as their Accept-Only-If and Reject-Only-If rules. Functions which support the entry of this knowledge are: Edit-Accept-Rule, Edit-Reject-Rule, Insert, and Delete. These functions apply to states, events, and sets and can be easily activated via function keys. To indicate what to edit and where to insert or delete, the analyst simply moves the cursor to cover cells of the spreadsheet, then presses the appropriate function key.

The Computer View of the State Analysis Knowledge Base. The spreadsheet shown in Figure 3 is transformed by the computer into a multidimensional network database. An example of this database is shown in Figure 4. The states are listed vertically, the events for each state are listed horizontally, and the specification sets for each event are listed diagonally. The Accept-Only-If and Reject-Only-If rules are connected to the node for their respective state, event, or set. There is no logical limit to the number of states, events, sets, or rules. The only physical limitations are the amount of time and memory available to the analysis.

RECOMMENDATION DETERMINATION

Recommendations are based on the current and prior existence of states and other recommendations (taken either discretely or statistically accumulated). Each recommendation is associated with a list of criteria which must be satisfied before that recommendation can be suggested. Each criteria has an associated 'history' factor. This factor is a period of past time counting from the present time over which the given criteria may have been active. For instance, it may be important to know whether or not a switch was in the

---

Figure 4
The Multi-Dimensional Network Database Representing the State Analysis Knowledge Base
'On' position during the past five seconds. The 'history' factor can be specified as greater than zero if history is important or made equal to zero if only the present condition of the criteria is needed. A mechanism for capturing and organizing this type of knowledge can take the form of a relational database's input screen and the database management software's editing, deleting, and inserting functions. Figure 5 illustrates the input screen.

| RECOMMENDATION1: | recommendation name |
|------------------|---------------------|
| CRITERIA1:       | name of active state/recommendation |
| HISTORY1:        | numeric factor       |
| CRITERIA2:       | name of active state/recommendation |
| HISTORY2:        | numeric factor       |
| etc              |                     |

| RECOMMENDATION2: |                      |
|------------------|----------------------|
| CRITERIA1:       |                      |
| HISTORY1:        |                      |
| CRITERIA2:       |                      |
| HISTORY2:        |                      |
| etc              |                      |

| RECOMMENDATIONx: |      |
|------------------|------|
| CRITERIA1:       |      |
| HISTORY1:        |      |
| CRITERIA2:       |      |
| HISTORY2:        |      |
| etc              |      |

Figure 5
Input Screen for Capturing Recommendations and Their Associated Criteria

The database of recommendations and their associated criteria are transformed by the computer into a binary tree. Each recommendation's criteria are first sorted alphabetically. Then, the recommendations themselves are sorted according to the number of criteria associated with each recommendation (least number on top). The tree is constructed by placing the first recommendation on the tree with its associated criteria. The top criterion is the tree root and the suggested recommendation is the leaf. In a similar fashion, the other recommendations are placed on the tree with their associated criteria, according to how they match recommendations and criteria already placed on the tree. This binary tree does not depend on any given order of recommendations or criteria. Each hangs on the tree regardless of whether it is a suggested recommendation or a criteria. An example recommendation knowledge base (after sorting) and its resulting binary tree is shown in Figure 6.

The tree shown in Figure 6 is traversed in a top-down, depth-first fashion. The left branch of each node is the branch taken if the recommendation or state making up the given criteria is currently active. The right side is taken if the recommendation or state is not currently active. This process continues until a suggested recommendation is achieved. Then, the recommendation report and the active recommendation list are updated. The process continues down the right branch of the node until the leaves of the tree are reached. Each time a recommendation or state is added to or removed from the active list, the binary recommendation tree must be traversed. In a previous paper, Raeth & Hardin discussed their partial implementation and use of this recommendation process.
SUMMARY

This paper has discussed a method of using raw data and an analyst's knowledge in an automated environment to turn data into information and information into suggested recommendations. [For a good tutorial on expert systems in general, see Gevarter.] The contributions made by this paper are:

1) a way of getting meaning out of raw data samples.
2) a way of helping the analyst cope with the growing mounds of raw data while preserving the detail of the data.
3) a means of applying the analyst's knowledge consistently.
4) a means of identifying, structuring, and collecting [Waldron] the knowledge used to perform state analysis and to offer recommendations based on active states.

The current direction of this project is to document the software already produced for state analysis and to move that software into full production.

ACKNOWLEDGEMENTS

The author wishes to thank the following people:
Vicky Deiter, Doug Nation, and Marilyn Raeth for their review of this paper.
Ross Anderson for his encouragement and support.
Patricia Johnston and John Holland for their discussions on Goldenbird analysis.
James Graves (Georgia Tech Research Institute) for his insights into decision criteria.

BIBLIOGRAPHY

Gal, Annie and Jack Minker 'A Natural Language Database Interface that Provides Cooperative Answers' IEEE Conference on Artificial Intelligence Applications, 1985

Gevarter, William E. 'An Overview of Expert Systems' National Technical Information Service; Wash, DC; NTIS/DSC # PB83-217562; May 82

MacFarlane, A.G.J. Engineering Systems Analysis Reading, MA; Addison-Wesley; 1964

Raeth, Peter G. and Patricia A. Johnston 'Using Expert Systems Technology to Develop a Statistical Strategist' 1st Meeting of the Mississippi Academy of Science, 1987

Raeth, Peter G. and James M. Hardin 'Using Expert Systems Technology to Automate Goldenbird Stripchart Analysis' Association of Old Crows Joint Western-Mountain Region Technical Symposium, 1987

Waldron, Vincent R. 'Process Tracing as a Method for Initial Knowledge Acquisition' IEEE Conference on Artificial Intelligence Applications, 1985