TEST REPORT

FOR THE

SCENARIST™
AUTOMATED SCENARIO GENERATION SYSTEM

PROJECT TITLE: RESEARCH IN ARTIFICIAL INTELLIGENCE FOR NON-COMMUNICATIONS ELECTRONIC WARFARE SYSTEMS

Contract No. DAAB07-89-C-P017

April 30, 1991

Prepared for:

US ARMY COMMUNICATIONS-ELECTRONICS COMMAND
Fort Monmouth, New Jersey

Prepared by:

VISTA RESEARCH CORPORATION
5055 E. Broadway Boulevard, Suite D205
Tucson, Arizona 85711
(602)790-0500

Contents

Foreword

This report was prepared by the staff of Vista Research Corporation, under Contract No. DAAB07-89-C-P017 to the US Army Communications-Electronics Command. Project staff included Dr. J. George Caldwell, Principal Investigator, Mr. William N. Goodhue, Ms. Sharon K. Hoting, Dr. William O. Rasmussen, Mr. Eric Weiss, and Mr. Fletcher Aleong. Government monitoring of the project was provided by Dr. Frank Elmer, Head of the Advanced Concepts Division of the Center for Electronic Warfare/Reconnaissance, Surveillance and Target Acquisition (EW/RSTA). The CECOM Project Manager is Dr. Frank Elmer.

Special thanks is extended to Captain John Aloisio of the US Army Intelligence Center and School for providing information about the TRAILBLAZER system. This information was used to define the TRAILBLAZER units and organization for input to the Scenarist, and to define the rules governing the placement of the TRAILBLAZER units.

1.0. Introduction

1.1. Purpose of Document

This document describes the results of a test of the Scenarist automated scenario-generation system. The test was conducted in accordance with the test plan described in the report, Test Plan for the Scenarist Automated Scenario Generation System (Vista Research Corporation, February 28, 1991).

1.2. System Purpose

The Scenarist automated scenario generation system is a workstation-based aid for use by military scientists to develop tactical military scenarios. As used in this report, a "tactical military scenario" is a specification of the initial positions, activities, and planned movements of tactical military units. The specification may be at various levels of detail, ranging from a specification of the positions of high-echelon tactical units (e.g., divisions, brigades, battalions) to the positions and activities of low-level units (e.g., companies) or individual platforms or items of equipment (e.g., vehicles and radars).

The immediate application of the Scenarist is to develop laydowns of noncommunications equipment for use by the US Army Communications-Electronics Command (CECOM) in its analysis of electronic warfare systems and concepts. Later applications may include development of scenarios for use in a wide range of military applications, including force planning; research, development, test and evaluation; operational planning and evaluation; and training and education.

2.0. Applicable Documents

2.1. Government Documents

The following documents of the exact issue shown form a part of this specification to the extent specified herein. In the event of conflict between the documents referenced herein and the contents of this specification, the contents of this specification shall be considered a superseding requirement.

STANDARDS:

DoD-STD-2167A Military Standard: Defense System Software Development

OTHER PUBLICATIONS:

31 August 1989 Statement of Work (SOW) for contract, Research in Artificial Intelligence (AI) for Non-Communications Electronic Warfare Systems, CECOM contract no. DAAB07-89-C-P017

29 March 1990 TRAILBLAZER Operations (FM 34-10-3), Headquarters, Department of the Army, Washington, DC

1 November 1989 TRAILBLAZER User's Guide, Volume I, Operators, Project Manager Signals Warfare, ATTN: SFAE-IEW-SG, Vint Hill Farms Station, Warrenton, VA

1 November 1989 TRAILBLAZER User's Guide, Volume II, Supervisors, Project Manager Signals Warfare, ATTN: SFAE-IEW-SG, Vint Hill Farms Station, Warrenton, VA

1 November 1989 TRAILBLAZER User's Guide, Volume III, Commanders and Staffs, Project Manager Signals Warfare, ATTN: SFAE-IEW-SG, Vint Hill Farms Station, Warrenton, VA

1 November 1989 TRAILBLAZER User's Guide, Volume IV, Interoperability, Project Manager Signals Warfare, ATTN: SFAE-IEW-SG, Vint Hill Farms Station, Warrenton, VA

2.2. Non-Government Documents

The following documents of the exact issue shown form a part of this specification to the extent specified herein. In the event of conflict between the documents referenced herein and the contents of this specification, the contents of this specification shall be considered a superseding requirement.

OTHER PUBLICATIONS:

May 31, 1990 Preliminary System Design (System/ Segment Design Document) for the Scenarist Automated Scenario Generation System, Vista Research Corporation, Tucson, Arizona

February 28, 1990 Test Plan for the Scenarist

Automated Scenario Generation System,
Vista Research Corporation, Tucson,
Arizona

3.0. Proposed Test Procedures

This section identifies the procedures that were proposed in the test plan document to be executed in the test. The paragraphs that follow (under the heading "Test Procedures") were extracted from Section 9.0 of the test plan document (with correction of the TRAILBLAZER echelon terminology, and reference to TRAILBLAZER Master Control Stations (MCSs) as "items").

Test Procedures

1. Input Map Data. Digital map data have already been extracted from the GRASS sample map data files describing the area near Spearfish, SD. These data are stored on floppy diskettes. We will create from these files a set of maps of varying resolution. There are two types of maps used by the Scenarist -- a terrain-type map (that identifies lakes, urban areas, etc.) and an elevation map. We will then create a Scenarist "project file" naming the lowest-resolution maps (i.e., the lowest-resolution terrain-type map and the lowest-resolution elevation map). The Scenarist run will begin using these maps.

2. Define a Division-Level Military Unit and Input It. We propose to define a division, specifying all of the organizational detail necessary to define the location of TRAILBLAZER units. We will first define the division as a "generic" division, and then "copy" it to a specific division. The specific division will be given a location, an objective, and an avenue of approach to the objective. The threat will be characterized simply in terms of a Forward Edge of the Battle Area (FEBA).

The generic division will contain the TRAILBLAZER units and equipment in "canonical" locations, i.e., the locations will be specified without regard to terrain, mission, or threat.

That the Scenarist maintains the organizational structure of the defined unit will be demonstrated by requesting the Scenarist to display a single TRAILBLAZER unit.

3. Specify Placement Rules and Input Them. We obtained five documents from the US Army Intelligence Center and School that describe the tactical doctrine for employing TRAILBLAZER units and equipment. We will analyze these documents and develop a set of rules for placing this system in the division. The rules may or may not require the coding of additional functions to define the factors in terms of which the rules are defined. If factors are needed in addition to those that already exist in the current

Scenarist prototype (terrain type, elevation, distance to FEBA, horizon angle), functions will be coded to compute those factors.

4. Apply Rules to Reposition Items. The Scenarist will be executed to apply the rules to reposition the TRAILBLAZER units and platforms/equipment in a fashion that takes into account terrain features, mission, and threat.

5. Output the Item Locations. The locations and identification of the TRAILBLAZER units and platforms/equipment will be written to a file, printed out in hardcopy, and written to a floppy diskette.

6. Demonstrate User-Positioning of an Item. One of the TRAILBLAZER units will be moved from its current position. The rules will be re-executed to reposition all of the TRAILBLAZER units in the division. It will be noted that the user-specified position will be unchanged.

7. Relocate the Division. The division will be moved to another location on the battlefield. It will be observed that the locations of all subordinate units and equipment are automatically redetermined.

4.0. Test Results

This section presents a detailed description of the procedures performed in the test and the test results. A subsection is presented for each of the test procedures identified in the preceding section.

4.1. Input Map Data

The map data set that was used for the test was the map data provided as a sample data set with the US Army's GRASS geographic information system. The data set includes a variety of attributes for a 14 kilometer by 19 kilometer area near Spearfish, South Dakota. The attributes of interest to this application were cellular data on terrain type, elevation, and roads, and vector data on roads. The resolution of the terrain-type and elevation data was 30 meters (i.e., an attribute value was available for each 30 meter by 30 meter area, or "cell"), and the resolution of the road data was 100 meters.

The Scenarist operates on a sequence of maps of varying resolution, starting from low resolution (e.g., each cell is one or two kilometers wide) to high resolution (i.e., each cell is of the highest resolution available, in this case either 30 meters or 100 meters).

The process of preparing the map data files used by the Scenarist involved three basic steps:

1. Execution of the program WINL7 to extract a "linear" road file from the GRASS data set. This file contains vector representations of the roads. It is not used for analysis purposes, but simply as input to a background map to make it easier for the analyst to orient himself.

2. Execution of the program WIN7 to extract three cellular maps: a terrain-type map, an elevation map, and a road map. These maps are used by the Scenarist in its unit repositioning algorithm.
3. Use of the program s03xcomp to prepare a series of terrain-type, elevation, and road files of lower and lower resolution. Each successive lower-resolution file is obtained by aggregating square blocks of neighboring cells of size two or three cells on a side. This aggregation process is continued until a set of map files is obtained that can cover the entire area of interest with cellular maps having fewer than 32 cells on a side.

In order for the Scenarist to use map data, it must be in a specific format. The exact procedures used to prepare map files for input to the Scenarist will vary from application to application, depending on the source of the mapping data. The programs WINL7 and WIN7 are designed exclusively for producing Scenarist-format map files from GRASS map files. The user would have to develop other programs to produce Scenarist map files from other sources.

For the GRASS sample data set, the process of producing the Scenarist map files was as follows.

1. Extract Binary Map Files from GRASS System. The GRASS system was used to output binary data files containing cellular and vector data. A total of 12 cellular and two vector data files were extracted. The cellular data files were named: road (roads), quad (quadrant), rail (railroads), soil (soils), vege (vegetation), land (landuse), geol (geology), strm (streams), elev (elevation), slop (slope), tern (terrain), and aspc (aspect). The vector data files were named strml (streams) and roadl (roads). The terrain-type, elevation, and road cellular data files were the files named tern, elev, and road, respectively. The road vector data file was the file named roadl.

2. Use the Program WINL7 to Produce a Vector Road File. The program WINL7 was executed to produce a vector road file in Scenarist format from the vector road file in GRASS format. In running the program, the user first specifies the "data topic," in this case, "Road." The program indicates the Universal Transverse Mercator (UTM) coordinate range of the data (east-west and north-south) and requests the location of the top left (upper-left-hand) corner of the map. The extreme northern and western coordinates were selected (590010 and 4928000). The program also requests the width of the block of cells (called a "window") to extract from the GRASS file. This window must be square. A value of 25,000 meters was input. The program requests the name of the output file (geol02.fil). The program then produces a vector road file that is in the Scenarist format, except for the header, which is not used by the Scenarist. The "edlin" editing program was used to delete this header, and the resulting file (still called geol02.fil) was ready for use. (The index "02" is used throughout this test to refer to the fact that this is the second test problem analyzed by the Scenarist.)

3. Use the Program WIN7 to Produce Three High-Resolution Cellular Map Files: a Terrain-Type File, an Elevation File, and a Road File. The program WIN7 was used to extract terrain-type, elevation, and road files from the GRASS data. The GRASS data are stored in rectangular matrices (arrays). The array sizes are 466 rows and 634 columns for terrain type; 466 rows and 633 columns for elevation; and 160 columns and 220 rows for roads. The cell width is 30 meters for terrain type; 30 meters for elevation; and 100 meters for roads. The WIN7 program requests the user to specify the northwest corner and width (in meters) of the "window" (square block) of cells to be selected from the GRASS data set. As above, the extreme northwest point of the data set was selected, and the width of 25,000 meters was specified. The files were named geod0201.fil (terrain-type), geoc0201.fil (elevation) and roadtemp (roads). (Note: Use of the prefix geod for terrain type and geoc for elevation stems from the fact that all discrete-variable (categorical) cellular map data are stored in a discrete-variable matrix (geodisc), and all continuous-variable data are stored in a continuous-variable matrix (geocont). In the early versions of the Scenarist, terrain type and elevation were the only cellular map types. The letters d and c are still used for terrain type and elevation, even though there is now more than one discrete cellular map type (i.e., terrain type (file prefix geod) and roads (file prefix geor).)

Note that the program WIN7 can select only square windows, and that the window must fall wholly within the coordinate limits. Since the GRASS data set was not square, it was not possible (under these conditions) to select a square window that included all of the data. Of the 466 (rows) x 634 (columns) terrain-type matrix, WIN7 selected a square matrix of size 465 x 465; of the 466 x 633 elevation matrix, WIN7 selected a 465 x 465 matrix; of the 160 x 220 road matrix, WIN7 selected a 159 x 159 matrix. In a future application, it is recommended that the WIN7 program be modified to allow the window to be an arbitrary rectangle.

As noted, the three map files were named geod0201.fil (terrain type), geoc0201.fil (elevation), and roadtemp (roads). The terrain-type and elevation files were each of width 465 cells, with a cellwidth of 30 meters. The road file was of width 159 cells, 100 meters wide. The terrain-type and elevation files were satisfactory, but there was a problem with the road file. This data file had evidently been miscoded, so that the value "0," which was supposed to signify "no data," obviously meant "no road." The program s03xcomp was used to recode the data as follows:

1. The codes 1 (primary_route_undivided), 2 (road_or_street_class_3)), 3 (road_or_street_class_4), and 5 (cloverleaf_or_interchange) were recoded as 2 (road present);
2. The code 0 (no data) was recoded as 1 (no_road)
3. The code 0 was reserved for "no data," but there were no such cells.

The input requested by s03xcomp is: (1) compression factor: 1; (2) number of attributes to check for: 4; (3) attribute numbers: 1,2,3,5.

The input data file for s03xcomp was roadtemp. The output data file was named geor0201.fil. The program s03xcomp does not change legends, so the

legend for geor0201.fil was the same as for roadtemp. The program "edlin" was used to change the legend to read:

1: no_data
2: no_road
3: road.

The edlin line editor was also used to change the file name on line 1 of the file (from roadtemp) to geor0201.fil).

4. Use the Program s03xcomp to Produce a Series of Lower-Resolution Cellular Map Files. As noted above, the Scenarist operates on a sequence of map files, from low resolution to high resolution. The next step is to create a sequence of lower resolution files from the three high-resolution files obtained in the previous step. The program s03xcomp was designed for this purpose: it accepts as input a cellular map file in Scenarist format and produces a lower-resolution map file. It does this by aggregating (combining) neighboring blocks of cells. Specifically, the user specifies a "compression factor," which is the width of the block to be aggregated. The program then performs an aggregation in any of three ways:

1. For continuous data, it computes a mean
2. For discrete data it computes either:
    a. the mode; or
    b. an indicator variable having the value 2 if any of a specified set of attribute values is present, and 1 if none of the values is present.

The user may specify whether the value "0" signifies "no data." If 0 specifies "no data," and any of the cells of a block has the value 0, then the mean and mode also have the value 0 (i.e., "no data"). In this case, the indicator value is 0 if none of the specified set of values is present, but it is 2 if any of the specified set of values is present.

The first run of s03xcomp was used to compress the file geod0201.fil by a factor of 3. The input was as follows:

geod0201.fil (name of high-resolution input file
    having 465 columns, 30 m resolution)
0 (to compress discrete data)
3 (compression factor)
0 (signifies "no data")
0 (compute mode)
geod0202.fil (name of lower-resolution output file
    having 155 columns, 90 m resolution)

The additional runs of s03xcomp made to produce even lower-resolution terrain-type files are summarized below:

Input: geod0202.fil,0,2,0,0; output geod0203.fil (77 columns, 180 m resolution)
Input: geod0203.fil,0,2,0,0; output geod0204.fil (38
columns, 360 m resolution)
Input: geod0204.fil,0,2,0,0; output geod0205.fil (19

columns, 720 m resolution)
input: geod0205.fil,0,2,0,0; output geod0206.fil (9
columns, 1440 m resolution)

Input: geoc0201.fil (465 columns, 30 m resolution), 0,3,0,0;
output geoc0202.fil (155 columns, 90 m resolution)
Input: geoc0202.fil,0,2,0,0; output geoc0203.fil (77
columns, 180 m resolution)
Input: geoc0203.fil,0,2,0,0; output geoc0204.fil (38
columns, 360 m resolution)
Input: geoc0204.fil,0,2,0,0; output geoc0205.fil (19
columns, 720 m resolution)
Input: geoc0205.fil,0,2,0,0; output geoc0206.fil (9
columns, 1440 m resolution)

Input: geor0201.fil (159 columns, 100 m resolution), 0,2,0,1,1,2;
output geor0202.fil (79 columns, 200 m
resolution)
Input: geor0202.fil,0,2,0,1,1,2; output geor0203.fil (39
columns, 400 m resolution)
Input: geor0203.fil,0,2,0,1,1,2; output geor0204.fil (19
columns, 800 m resolution);
Input: geor0204.fil,0,2,0,1,1,2; output geor0205.fil (9
columns, 1600 m resolution);

Note that the process of aggregation stops when the entire area of interest is covered by a map of size 32 cells by 32 cells or less. The 9-cell by 9-cell maps are the highest-resolution maps that do this -- the product of 32 cells of width 1440 m is 46,080 m, or 46 km. (The 19-cell by 19-cell maps cover only about half of the area required to place a division -- 32 cells x 720 m = 23 km).

The names of the map cells are entered (using the edlin line editor program) into the "project files." A total of six project files was set up:

proj0201.fil uses geod0201.fil,geoc0201.fil,geor0201.fil
proj0202.fil uses geod0202.fil,geoc0202.fil,geor0201.fil
proj0203.fil uses geod0203.fil,geoc0203.fil,geor0202.fil
proj0204.fil uses geod0204.fil,geoc0204.fil,geor0203.fil
proj0205.fil uses geod0205.fil,geoc0205.fil,geor0204.fil
proj0206.fil uses geod0206.fil,geoc0206.fil,goer0205.fil

Note that a project file uses files of similar resolution, if they are available. Since the highest resolution of the road file is 100 m, the map resolutions in the project file proj0201.fil vary (i.e., the maximum resolution is 30 m for terrain-type and elevation, but 100 m for roads).

In running the Scenarist, the low-resolution files are used first; that is, proj0206.fil will be the first project file accessed, and proj0201.fil will be the last.

4.2. Define a Division-Level Military Unit and Input It

From the US Army Intelligence Center and School, we obtained a number of documents describing the organizational structure and tactical doctrine for placing TRAILBLAZER units on the battlefield (see references under the heading "Applicable Documents," above). Based on a review of these documents, we developed a specification of the organizational structure of a division and its subordinate units in sufficient detail to allow the development of placement rules for TRAILBLAZER units.

In summary, the organizational structure, and the subordinate units explicitly represented, is as follows:

1. Division contains two front-line brigades and a military intelligence (MI) battalion. Represent the division headquarters company explicitly.

2. MI battalion contains an electronic warfare (EW) company. Represent the battalion headquarters company explicitly.

3. EW company contains a SIGINT processing platoon (SPP). Represent the EW headquarters platoon explicitly.

4. SPP contains a TRAILBLAZER (TB) section and a TEAMPACK (TP) section.

5. TB section contains five TRAILBLAZER master control stations (MCSs), represented in the Scenarist echelon structure as squad/teams.

6. TP section contains three TEAMPACK teams, represented in the Scenarist echelon structure as squad/teams.

In the above organizational structure representation, a subordinate unit was included if it or one of its subordinate units contained a TRAILBLAZER unit, or if the TRAILBLAZER positioning rules might refer to the location of it or one of its subordinate units.

During the process of defining the organizational structure, it is necessary to specify "canonical" positions of all units and equipment. A canonical position (or configuration, or laydown) is a placement of the units and equipment in a fashion that is consistent with military tactical doctrine in the absence of any information about terrain, mission, or threat. According to the documents, the five TRAILBLAZER master control stations (MCSs) should be arranged in a "W" formation in the forward area of operations of the division (i.e., in the maneuver brigade areas), with the frontal MCSs about 5-10 km from the forward line of own troops (FLOT) and the rear ones 10-15 km back of the forward ones. Also, a distance of 15 km should be maintained between the MCSs.

The locations of all subordinate units are specified in standardized coordinates, relative to the corners of the parent unit (corners at (0,0), (1,0), (1,1), and (0,1)). In terms of these standardized coordinates, the

TRAILBLAZER MCS locations are (.1,.1), (.5,.1) and (.9,.1) for the frontal MCSs and (.3,.3), (.7,.3) for the rear MCSs.

Apart from the TRAILBLAZER units, other aspects of the canonical laydown are as follows. The division main and MI battalion headquarters companies were placed in the division rear (standardized coordinates (.4,.8)). The EW company headquarters and the SIGINT Processing Platoon headquarters were placed near each other, near the rear of the first brigade (standardized coordinates (.25,.35) and (.2,.3)). The TRAILBLAZER and TEAMPACK section symbol location points were placed at (.2,.3) and (.3,.3). The TEAMPACK teams were placed along the division front, at (.1,.05), (.4,.05), and (.6,.05).

Because the placement rules for TRAILBLAZER are concerned with the MCSs' coverage of the target area, an "objective" was defined for the TRAILBLAZER section (and for the TEAMPACK section, as well). The objective is specified in terms of a "bounding rectangle," in real coordinates. The objective was specified as a division-sized area in front of the BLUE division.

Appendix A contains a complete description of the division organizational structure, in terms of the input data required by the Scenarist program to create a file containing all of the unit organizational data. The specification of the TRAILBLAZER input data starts at section B of the deployment description. (Section A of the deployment description presents the input for a "Beqaa Valley" scenario that was used in the initial testing of the Scenarist. The TRAILBLAZER units were simply added to the Beqaa Valley unit file to avoid the proliferation of unit data files. The Beqaa Valley specific unit file is called spec01.fil -- the "01" signifying problem 01. The Spearfish/TRAILBLAZER specific unit file is called spec02.fil -- the "02" signifying problem 02. File spec02.fil was formed by copying file spec01.fil, changing the locations of the units to place them on the Spearfish map, and adding the TRAILBLAZER units.)

Note that in defining the TRAILBLAZER units, the specification of the "type" of a unit is constrained by whatever types have already been included in the unit file in the Beqaa Valley example. For example, the generic division defined in the Beqaa Valley example was specified to be of type 1. Both the specific BLUE and RED divisions of the Beqaa Valley example were defined to be of that type. That division was defined without any mention of TRAILBLAZER units, and so a new division had to be defined for this TRAILBLAZER test. The type of the new division was specified as "2."

Similarly, it was not desired to use the same organizational structure for the brigades in this test, so the brigades were specified to be of type "2" to distinguish them from the type 1 brigades of the Beqaa Valley example.

The unit type designator determines what symbols and labels are used for a unit. The unit number designator is part of the unit's code, and is used in accessing the unit (by code), during Scenarist processing. No two units may have the same code, or else the Scenarist would only access the first

such one in the file. The unit's number is arbitrary. It does not have to begin with 1, or be in sequence -- all that is necessary is that no two units have the same code. It is recommended for simplicity, however, that the subordinate units of the same type within a unit be numbered serially, beginning with 1. The unit's type must be an integer between 1 and 10 (because the type is used as the index of an array of size 10). The unit's idno (and parent unit's idno) may be any number of the user's choosing (e.g., a familiar numerical designator). It may be used by the user in selecting units for display, but is not used in any way by the Scenarist. In the deployment of Appendix A, the concatenation of side, echelon, and number was used as the idno.

The "type" of a unit is a characteristic that is independent of its parent unit. On the other hand, the "number" of a unit is with reference to its parent unit. That is, if two sections are of the same type, they are, quite literally, the same type of unit, regardless of what parent units they belong to.

Since no MI battalions, EW companies, or SIGINT processing platoons had previously been defined, these units were assigned types "1."

In defining the division and its subordinate units, it is necessary to specify the number of subordinate units of each geographic type (e.g., the number, NALLAREA, of "all-area" subordinate units), and the values of certain positional parameters, which vary by geographic type. These parameter values specify the canonical position of a subordinate unit -- the position suggested by tactical doctrine in the absence of any information about terrain features, mission, or threat.

The geographic types and positions specified in the Appendix A deployment are consistent with the information provided in the TRAILBLAZER documents referenced above, with one major exception. This exception arose from the fact that the area covered by the Spearfish, SD, map (about 14 km square) was quite small relative to the usual size of a division. For a division width of 30-50 km, only one or two MCSs would appear on the map. The problem with this situation is that some of the rules for positioning of TRAILBLAZER units concern interrelationships among the units (e.g., line-of-sight), and we were concerned that the test would not be very meaningful if only two units fell on the map. For this reason, we decided to place the division in a 20 km (front) x 22 km (rear) area in such a way that three MCSs fell on the map area.

A problem that arises with compressing the division size is that one of the guidelines for placing the MCSs is that there should be a separation of about 15 km between the MCSs. We decided that the purpose of the test could still be satisfied if we simply scaled the division down as described below, and dropped the requirement that the MCSs be separated by 15 km. Instead, we replaced this guideline with the requirement that, for the scaled-down example, the MCSs must be separated by only 5 km.

The preceding discussion relates to rules for determining the canonical laydown of the TRAILBLAZER units -- rules that do not account for terrain, mission, or threat. Rules that do account for terrain, mission, and threat are discussed in the following subsection.

At the time when the test plan was being written, it was mistakenly believed that the TRAILBLAZER stations were part of a "TRAILBLAZER platoon," and it was intended to represent the stations as equipment in these platoons. After reviewing the TRAILBLAZER documents, however, it was realized that the stations were included in a TRAILBLAZER section, and that the stations were considered to be squads or teams. That is how they have been defined in Appendix A.

The data presented in Appendix were input into the Scenarist program using the "Define Unit" and "Copy Unit" functions of the "Units" menu. For each type of unit needed, the Define Unit function was used to create a generic unit of that type (and enter it in file genu02.fil). Then, the defined generic unit was copied into the specific unit file (spec02.fil), adding whatever additional specific features were desired (e.g., unit corner locations, objectives). The files genu02.fil and spec02.fil were specified in the project files, proj02xx.fil (where "xx" signifies 01,02,...,06). The project files will be described in detail later, in the section on the test runs.

As part of the test, it was necessary to define labels and symbols for the units used in the TRAILBLAZER problem. The symbols are coded (in C) in the function _symbol. The symbol number (specified as a "case" in _symbol) is entered into the appropriate row (unit echelon) and column (unit type) of the matrix stored in the file symb01.fil.

During the conduct of the TRAILBLAZER test, it was realized that the organization of the "symbol" file (symb01.fil, which contains both unit symbols and labels) was poor. The numbering of the symbols (i.e., the case number in _symbol) bears no relationship to echelon (i.e., the row in symb01.fil). This fact unnecessarily complicates the process of defining symbols and coding them (in the function _symbol). It would be substantially less confusing if the cases in _symbol were in echelon order. Also, it would be very desirable to have "default" symbols for each echelon level. Currently, arbitrary case numbers are entered into rows and columns of symb01.fil for which no symbol has been defined in _symbol. This could be very confusing to a new Scenarist user. Since the test revealed several areas in which the Scenarist design had to be modified (e.g., incorporation of a capability to accept road cell maps; incorporation of a capability to handle global constraints; incorporation of a capability to reposition a map such that its center falls in the center of a unit), no effort was expended on this "clean-up" effort. It is recommended as a future development effort, however, since it would enhance the "user-friendliness" of the system.

4.3. Specify Placement Rules and Input Them

In addition to providing information about the organization and canonical placement of TRAILBLAZER units, the TRAILBLAZER documents were reviewed to determine the ways in which the positions of TRAILBLAZER units would be modified to take into account terrain, mission, and threat. The following rules were extracted from these documents:
1. Accessibility: In order for a TRAILBLAZER MCS to have access to a location, the gradient to that location cannot exceed 30 degrees.

2. Separation: The TRAILBLAZER MCSs should be separated by a distance of at least 5 km. (Note: TRAILBLAZER documents specify 15 km. The 15 km was reduced to 5 km for the reasons discussed above.)
3. Line-of-Sight betweens MCSs: There should be line-of-sight from each MCS to at least two other MCSs.
4. Line-of-Sight to Technical Control Analysis Center (SIGINT Processing Platoon (SPP)): There should be line-of-sight from at least two MCSs to the SPP.
5. Line-of-Sight to Target Area: The MCSs should be positioned so that it is possible to provide good coverage of high-payoff targets.
6. Freedom from Obstacles: The MCSs should have freedom from obstacles that may interfere with coverage (550 m from bodies of water, 1.5 km from cliffs, 10-50 km from mountains).
7. Deployment in Forward Area of Operations: The MCSs should be deployed in the forward area of the division's operations, i.e., within the maneuver brigade areas.
8. Antenna Position: The antenna has to be near the tree line.
9. Distance from FLOT: The distance from an MCS to the FLOT should not be less than 5 km. (Note: this requirement was reduced to 2 km because of the down-scaling of the division.)
10. Distance from Rear Area: The MCSs should be located in the forward area of operations of the division, i.e., in the area occupied by the two maneuver brigades in the division front.

The above rules were derived from the TRAILBLAZER documents, without regard to the structure of the Scenarist. In the Scenarist, not all of the factors (concepts) mentioned above are considered, and some factors are implemented (defined, quantified) in a fashion that differs from the implementation of the documentation. For example, the Scenarist has not been set up to include data on trees (although it could be modified to do so in the future, if this were considered desirable and the data were available).

In order for a factor to be used in a rule, it must be defined in terms of variables present in the Scenarist, and its value computed. The computation of the factor values is done in the Scenarist in C-language functions. For the TRAILBLAZER application, some factors were already available from the earlier development of the Scenarist (terrain type, elevation), but it was necessary to define and implement several new factors. The following additional C functions were defined to compute factors needed for TRAILBLAZER placement rules:

1. _accessibility: This function computes the accessibility of a unit grid cell by a TRAILBLAZER MCS. (The unit is divided into a grid of cells; the

width of the gridcell is the same as the width of the elevation map cell.) A cell is considered to be accessible if:

1. It contains a road; or
2. Its terrain type is "plains" and the slope from the neighboring cell toward the rear of the division is less than 30 degrees and that neighboring cell is accessible.

The function _accessibility is executed once (for each processing of the repositioning rules), to create an accessibility matrix that specifies the accessibility for every cell of the unit. During the processing of the rules, the accessibility of a unit grid cell is determined simply by accessing the appropriate row and column of the accessibility matrix.

Note that in order to compute accessibility, it is necessary to know whether a map cell contains a road. In the version of the Scenarist that existed prior to the TRAILBLAZER test, the only cellular data that were input to the Scenarist were terrain type and elevation. During the course of the TRAILBLAZER test, the Scenarist was modified to accept road cellular data. (Vector (linear) road data were previously input to the Scenarist and used in the creation of a vector map. The Scenarist was not designed, however, to utilize the vector road data. This approach is a standard one in geographic information systems -- use vector data for maps, but cellular data for analysis.)

The values returned by _accessibility are 0 (no data), 1 (not accessible) or 2 (accessible).

2. _LOS: This function determines whether a line-of-sight (LOS) condition exists between two specified points (input parameters).

The values returned by _LOS are 0 (no data), 1 (no LOS), or 2 (LOS exists).

3. _lostotarget: This function determines whether satisfactory line-of-sight conditions exist between a TRAILBLAZER MCS and its target area. The criterion used for "satisfactory line-of-sight conditions between MCS and its target area" is that satisfactory LOS conditions exist if LOS exists from the MCS to at least two of five equally spaced points along the division front.

We considered implementing an alternative criterion for satisfactory LOS conditions, viz., LOS exists between the MCS and at least two of five equally spaced points along the nearest side of the bounding rectangle of the objective defined for the TRAILBLAZER unit. Implementation of this criterion was rejected, however, because of the amount of effort that would have been required to implement it. The problem that exists in implementing it is that the MCS locations are in standardized (unit) coordinates, and the objective bounding rectangle corners are specified in real (map) coordinates. It would hence have been necessary to develop a function to convert real coordinates to standard coordinates (the function _transfstdtoreal performs the inverse transformation). While this could have been done without much trouble, it was considered not worth

the trouble, since the adopted criterion was considered adequate, was easier to implement (since no coordinate transformation from real to standard coordinates was required), and would involve less computation.

This point merits some additional discussion, since it will arise again in future Scenarist applications. The rationale for selecting one method over another is not just to save programming and testing time; program running time must also be considered. In the present case, the rejected alternative may possess an advantage over the selected alternative on theoretical grounds (i.e., its "face validity" is greater, since it relates to a target area rather than the division front), but it has a disadvantage on practical grounds (i.e., longer computer running times). Since either criterion was considered adequate but the _lostotarget function would be called many times, the running-time performance was given heavier weight and the adopted criterion was considered superior, overall, to the rejected alternative.

The function _lostotarget returns three values: 0 (no data), 1 (unsatisfactory LOS from MCS to target area), or 2 (satisfactory LOS from MCS to target area).

4. _lostootherunits: This function determines whether satisfactory LOS conditions exist between a TRAILBLAZER MCS and the other MCSs. The criterion for assessing whether satisfactory LOS conditions exist is the following: satisfactory LOS conditions are considered to exist if a LOS condition exists from the MCS to at least two of the four other MCSs.

This function returns three values: 0 (no data), 1 (unsatisfactory LOS from MCS to other MCSs), or 2 (satisfactory LOS from MCS to other MCSs).

5. _lostoheadquarters: This function determines whether satisfactory LOS conditions exist between the set of five TRAILBLAZER MCSs and the SIGINT Processing Platoon (SPP). "Satisfactory LOS conditions" are considered to exist if a LOS condition exists from at least two MCSs to the symbol location point of the SPP. The symbol location point was used as a reference point for the SPP since no SPP headquarters unit had been defined as part of the SPP unit. That is, the only defined subordinate units defined for the SPP were the TRAILBLAZER section and the TEAMPACK section. It could be considered preferable from a face validity viewpoint to have defined the SPP headquarters as a subordinate unit of the SPP. In this case, the LOS-to-headquarters criterion would be stated in terms of LOS conditions between the MCSs and the location of the SPP headquarters unit instead of the location of the SPP symbol location point. The theoretical preference for the former is that the SPP headquarters unit would be a military object, whereas the symbol location point is merely a Scenarist artifact (viz., a map symbol location, not a unit location). ("Face validity" refers to the degree to which the entities and processes of a model bear a close correspondence to the entities and processes of the real world.) In any event, the symbol location point is an SPP attribute, and the effect of using the symbol location point vs. an SPP headquarters

location is identical, because the SPP headquarters would have been located exactly at the SPP symbol location point.

Some additional consideration should be given to the use of symbol location points as reference points. In the current version of the Scenarist, the user may relocate units without parent units, or subordinate units of geographic type 6 (subunits of absolute radius) or 7 (point objects -- platforms or equipment). The SPP is an "all-area" unit, and so the only way of relocating its symbol location point is to delete and redefine the EW company (where the SPP symbol location point is defined). This is substantially more effort than relocating subunits, because not only the EW company but all of its subordinate units (the SPP, the TRAILBLAZER section, and the TEAMPACK section) would have to be redefined. This was not a problem in the test (since the SPP symbol location point was not relocated), but it could present a problem in a future application. Based on the realization that this could be a problem, however, it is recommended that in future applications all units that will be referred to in the rules should be explicitly defined. In the present test, several of them were (viz., the division headquarters company, the battalion headquarters company, and EW headquarters platoon). It was not realized when the units were defined that we would be making reference to the SPP headquarters, and so it was not explicitly defined. It may be desirable to adopt the policy of defining a headquarters unit for all units, so that this problem cannot arise.

The function _lostoheadquarters returns three values: 0 (no data), 1 (unsatisfactory LOS to headquarters), or 2 (satisfactory LOS to headquarters).

6. _disttootherunits: This function computes the distance from an MCS to the nearest other MCS. The return value is the distance, in meters.

7. _disttofront: This function computes the distance from an MCS to the division front. The return value is the distance, in meters.

8. _inforwardarea: This function determines whether an MCS is located in the front half of the TRAILBLAZER section area. (Note: This criterion differs a little from the one derived from the documents. It is about the same, but much simpler to implement, for the following reasons. The TRAILBLAZER section area coincides with the division area, since the TRAILBLAZER section and its parent units (all the way up to the division) are all "all-area" units. The problem of determining the division front/rear boundary involves some computation. First, the division code must be determined (the division code is the first four digits of the TRAILBLAZER section code). Then, the division must be accessed from the specific unit file. If it exists (i.e., has been defined and is in the specific unit file), then the coordinates of its front/rear boundary line must be accessed, and a computation done to see whether the MCS lies in front of this line. If the division had not been defined, no decision could be made (i.e., "no data"). To avoid all of these computations, it was much simpler, and about as accurate, to simply test whether the MCS was located in the front half of the TRAILBLAZER section area.)

It returns two values: 0 (not in forward area), or 1 (in forward area).

9. _roads: This function determines the availability of roads at a location. This is determined by examining the code of the road map cell containing the location. This function returns three values: 0 (no data), 1 (no road exists in the cell), or 2 (a road is present in the cell).

Prior to the TRAILBLAZER application, a number of other functions had been defined, and two of them were used in computing the preceding functions. Those already-defined functions were:

10. _terrain: This function determines the type of terrain at a location. Its return values are: 0 (no data), 1 (plains), 2 (hills), 3 (woods), 4 (mountains), 5 (urban) and 6 (water).

11. _elevation: This function returns the elevation at a location. Its return values are 0 (no data) or the elevation in meters.

The goal of the Scenarist was to determine rules for placement that took into account terrain, mission, and threat. The factors that relate to specific terrain are 1-5 and 9-11. The factor that relates to specific mission and specific enemy threat is factor 3 (_lostotarget). Factors 6 (_disttootherunits) and 7 (_disttofront) relate to basic tactical doctrine for relative positioning of the TRAILBLAZER system, independent of specific terrain, mission, or enemy threat.

Having determined quantitative definitions for the concepts (factors) that were present in the rules extracted from the TRAILBLAZER documents, we will now specify the rules that were defined in terms of these factors and input to the Scenarist. Before doing so, however, it is helpful to divide these rules into two categories -- "local-constraint" rules, and "global-constraint" rules. Local-constraint rules are rules that involve factors whose values can be determined for a subordinate unit (an MCS, in the TRAILBLAZER application) without any knowledge of the positions of any other subordinate units. The values of these factors will be unaffected by any subsequent repositioning of the other subordinate units (MCSs) in the unit. Examples of such factors are terrain type, LOS to target, or distance to the FEBA.

On the other hand, global-constraint rules are rules that involve factors whose values are dependent on the positions of other subordinate units (in this case, MCSs). The values of such factors may change for a particular MCS if the location of another MCS is changed, <u>even though the location of that particular MCS does not change</u>.

The reason for considering rules in the two categories is that the Scenarist processes the two types of rules differently. It first processes the local-constraint rules, conducting a "spiral search" around the initial location of a subunit (MCS) in the search for a suitable location. This search can be done independently of the locations of the other MCSs (because

of the definition of a local-constraint rule) and the suitability of a found location will not change if some other MCSs are repositioned at a later time.

After the local-constraint rules and the spiral search have been applied for all subunits (MCSs), the Scenarist proceeds to process the global-constraint rules. The "problem" with a global-constraint rule is that, even though an MCS may be in a suitable location at one time, the suitability of that location may change if some other MCS is repositioned (e.g., the MCS may no longer have LOS to at least two other MCSs). In the language of optimization theory, the constraints are not "separable" with respect to the MCSs. This situation dramatically increases the difficulty of finding a suitable configuration for the set of five MCSs, because the optimization cannot be accomplished simply by conducting a local optimization (search) for each MCS independently.

It was never the intent of the Scenarist project to determine a mathematically rigorous solution to a type of optimization problem. Quite otherwise, it was the intent to use an expert (rule-based) system to determine configurations (laydowns) that were consistent with the rules. Nevertheless, the issue of satisfying global constraints cannot be avoided, since they are explicit in the tactical doctrine for placing TRAILBLAZER equipment. The problem was, then, to develop rules and actions that could produce "reasonable" laydowns in the presence of global constraints.

In the Scenarist development that had occurred prior to the TRAILBLAZER test, all of the rules had been local-constraint rules -- all subunit positioning was accomplished simply by conducting a single spiral search for each subunit. The Scenarist design did not call for the handling of global constraints. Because global constraints were an essential feature of TRAILBLAZER placement, however, it was considered necessary to modify the Scenarist design to accommodate them. This modification was undertaken and accomplished in the test.

It was decided to handle global constraints in the following manner. First, the local constraints would be processed for all subunits (MCSs). Then, the status (suitability of the location) of each MCS with respect to all constraints -- both local and global -- would be examined, and an "action" would be taken. The particular action would be a function of the status of the MCS with respect to all constraints. After assessing the suitability and taking a corresponding action for one MCS, the next MCS would be processed in the same way. After processing all five MCSs in this way, a single "global search" iteration was said to have been completed. Then, two more global search iterations were conducted (for a total of three global search iterations).

The "action" to be taken at each processing stage was quite simple. If the MCS location was suitable (i.e., all local constraints and all global constraints were satisfied), then the MCS was left at that location. If the MCS location was not suitable for any reason, then all eight of its neighboring unit gridcells were examined. If any neighboring cells were found having equal or higher elevation, the MCS was moved to the neighboring

cell having highest elevation. Otherwise the MCS was left in the same location. This process terminated either when three complete iterations was accomplished, or no movement occurred for any MCS on a particular iteration.

The preceding action was implemented because it seems quite reasonable -- most of the TRAILBLAZER global constraints are LOS-related, and LOS is likely to improve if the unit moves to higher ground. If all of the MCSs end up on local maxima, then the process stops. If, after three iterations, the locations are not suitable for all MCS, the process stops. In this case, the user would have to examine the configuration, and either accept it or manually reposition the units.

The preceding process is repeated for each of the six map resolutions, starting with the lowest-resolution map set (specified in the project file proj0206.fil) and finishing with the highest-resolution map set (specified in the project file proj0201.fil). For each resolution, the adjustments to the subunit positions are smaller and smaller, since the unit gridsize is the same as the cellwidth of the elevation map (so that the relocation of a unit to a neighboring grid cell involves a move of distance equal to the elevation map cell width (or square root of 2 times this distance, if the move is made to a "corner" neighbor)).

The reason for terminating the global search after three iterations is that, for the low-resolution map set, the moves can distort the configuration of the TRAILBLAZER units substantially from a "W" formation. To avoid massive distortions, either a limit must be placed on the number of iterations, or the action must involve the specification of an entire five-MCS TRAILBLAZER configuration (rather than the movement of a single MCS at a time). While the latter approach is feasible (it could be implemented, for example, by selecting a new system configuration from a prespecified set of alternative configurations), it was considered far too ambitious for the present test.

Note that the number of iterations could be increased for the higher-resolution map sets without appreciably changing the overall system configuration, because the MCS movements become smaller and smaller as the map resolution increases. There is little point in doing so, however, since the LOS to faraway points is not likely to change much from small changes in the MCS location. The main reason for the relocations on the higher-resolution maps is to satisfy local constraints, not global ones.

TRAILBLAZER Rules Input to the Scenarist

The following rules were input to the Scenarist. These rules were implemented both in C-language code and in the CLIPS expert system. The functions defining the factor values and the action were implemented only in C code.

Local Rules

A location is unsuitable if:

1. For any unit:
    a. The terrain type is water, mountain, or urban

2. For TRAILBLAZER squad/teams (i.e., units of side 1 (BLUE), echelon 11 (TRAILBLAZER squad/teams) and type 1 (TBMCS):
    a. The location is not accessible
    b. The location does not have LOS to the objective defined for the parent unit (i.e., the TRAILBLAZER section containing the squads)
    c. The distance to the front is less than 2000 m.
    d. The location is not in the front half of the TRAILBLAZER section area.

Otherwise, the location is suitable.

Global Rules

A location is unsuitable if:

1. For TRAILBLAZER squad/teams:
    a. The squad/team has LOS to fewer than two other TRAILBLAZER squad/teams in the TRAILBLAZER section containing the squad/teams
    b. Fewer than two squad/teams of the TRAILBLAZER section containing this subunit have LOS to the symbol location point of the TRAILBLAZER section containing the squad/teams.
    c. The minimum distance between TRAILBLAZER squad/teams is less than 5000 m.

Otherwise, the location is suitable.

4.4. Apply Rules to Reposition Items

In order to use the Scenarist to apply rules to reposition items (subordinate units or equipment), it is necessary to specify what map files, unit files, symbol files, suitability functions, rule functions, action functions, and CLIPS rule files are to be input at the beginning of the run. The map files can be changed during the course of the run. These files and functions are specified in a "project" file. The six project file used in the test of the Scenarist were as follows.

proj0201.fil (the last file used in the run; it contains the highest-resolution maps):

1. intro.fil (contains the Scenarist introduction)
2. titl01.fil (no longer used)
3. geod0201.fil (terrain-type cell map)
4. geoc0201.fil (elevation cell map)
5. geor0201.fil (road cell map)
6. geoa02.fil (area objects for vector (background) map)
7. geol02.fil (linear objects for vector map)
8. geop02.fil (point objects for vector map)
9. genu01.fil (generic unit file)
10. spec02.fil (specific unit file)
11. scra01.fil (scratch file)

12. symb01.fil (symbol and label file)

13. plat01.fil (platform file)
14. eqpt01.fil (equipment file)
15. feba02.fil (FEBA file)
16. _suitability0201 (suitability function)
17. _preprocessing0201 (preprocessing function)
18. _action0201 (action function)
19. _clipssuitability0201 (CLIPS interface function)
20. clip0201.fil (CLIPS rule file)

Some comments are in order. First, the suffix of the first part of each file name (e.g., 0201) is a two or four-digit code referring to problem number (e.g., 02) and (optionally) to a subproblem number (e.g., 01). The project number 02 was assigned to the test (i.e., the TRAILBLAZER application). The number 01 was used for the Beqaa Valley example used during the Scenarist development. In some instances, files that were used in problem 01 could also be used in project 02, either as-is or with additions. Such files include any files that do not contain map coordinate references, such as the generic unit file (genu01.fil), the scratch file (scra01.fil), the symbol file (symb01.fil), the platform file (plat01.fil), and the equipment file (eqpt01.fil).

Note that, since the file genu01.fil from problem 01 was used in problem 02, it now contains the generic units for both problems. Similarly, the file symb01.fil contains symbols for both problems. At some point, the user may want to start with a new generic unit file (i.e., if the new application has nothing to do with TRAILBLAZER), in order to reduce storage space or unit accessing time.

Some of the file names of files specific to problem 02 have subproject suffixes, and some do not. All of the cellular map file filenames have subproject suffixes: the subproblem suffix specifies the map resolution level. The vector map file filenames have no subproblem suffixes since there is only one set of such files (i.e., there are no resolutions or alternative files).

The platform and equipment files are empty in the current application, because the TRAILBLAZER MCS (the lowest-level item in the deployment) is a squad/team (echelon 11, in the Scenarist echelon structure).

The suitability function, preprocessing function, action function, CLIPS interface function, and CLIPS rule file names have subproblem suffixes (all 01), since it was thought that alternative functions and files would be used during the course of the test. They were not, so, in retrospect, the subproblem suffix for these functions and files could have been omitted.

The other five project files used in the TRAILBLAZER test contain the same file and function names as proj0201.fil, except for the cellular maps:

proj0202.fil contains geod0202.fil, geoc0202.fil, and

geor0201.fil (Note: proj0202.fil contains the same road file as proj0201.fil, because the resolution of the Spearfish road map was 100 m and the resolution of the terrain-type and elevation maps was 30 m. After the first aggregation stage, the terrain-type and elevation maps had resolution 90 m, which was close to the resolution of the road map. Since the resolutions of all maps used in an analysis should be as similar as possible, the highest-resolution road map file, geor0201.fil, was used with geod0202.fil and geoc0202.fil.)

proj0203.fil contains geod0203.fil, geoc0203.fil, and geor0202.fil

proj0204.fil contains geod0204.fil, geoc0204.fil, and geor0203.fil

proj0205.fil contains geod0205.fil, geoc0205.fil, and geor0204.fil

proj0206.fil contains geod0206.fil, geoc0206.fil, and geor0205.fil

The function of the project files is simply to provide the user with a convenient way of starting the Scenarist processing, without having to input all of the map, unit, and rule files and functions every time. The map files can be changed during the course of a run, but if changes are desired in other files or any of the functions, the Scenarist program would have to be exited, appropriate changes made to the project file, and the program re-executed.

In the Scenarist runs to be described below, the run begins with the project file containing the lowest-resolution maps, i.e., with proj0206.fil. Then, during the course of the run, successively higher-resolution map files are called in. Note that when this is done, maps of comparable resolution should be used, and the run should proceed to use maps of the next-higher resolution. For example, the user could start with project file proj0201.fil and, after applying the rules with its map files (geod0201.fil, geoc0201.fil, and geor0201.fil) proceed to call in the next-higher-resolution map files geod0202.fil and geoc0202.fil (keeping the road file geor0201.fil, whose resolution matches that of the terrain-type and elevation map files). Alternatively, the user could exit the program and start a new run with project file proj0202.fil.

When the repositioning rules have been applied at a given stage, the user has the option of either writing the changed locations of the relocated subunits (MCSs) into the file spec02.fil (over the original locations), or not saving the changes. If he is proceeding to relocate the units with

a higher-resolution map, he should save the changes. If the changes are saved, however, the original file will be modified. If it is desired to return to the original specific unit file, a backup version of it should be saved. Such a backup version has in fact been saved, and is called spec02.can (suffix "can" for "canonical"). If it is desired to return to the original specific unit file, the backup file should be copied into spec02.fil (i.e., copy spec02.can to spec02.fil). Similarly, once the process of repositioning is completed, the user will want to save the file spec02.fil so that it is not inadvertently overwritten as just suggested. It could be copied, for example, to a file name spec02.rep (suffix "rep" for "repositioned").

STEP-BY-STEP EXECUTION OF THE SCENARIST TO REPOSITION ITEMS.

With the project files ready, the Scenarist may be executed to reposition the TRAILBLAZER MCSs. The series of steps to reposition the TRAILBLAZER items is as follows.

1. Select project file proj0206.fil. This file contains the lowest-resolution map files, geod0206.fil (terrain type), geoc0206.fil (elevation), and geor0205.fil (roads).

2. Draw one or more maps (terrain type, elevation, roads, or a vector map), according to the user's preference. Note that the vector map (which contains only roads) takes a couple of minutes to draw, because of the very large number of road segments represented in the GRASS data base. Furthermore, it is not helpful to add labels to this map at a low resolution, because the labels overwrite each other and quickly blot out the map. Because of the importance of elevation and roads to the TRAILBLAZER application, the user will probably wish to use an elevation map or road map throughout most of the analysis. Note that individual roads are not apparent on the low-resolution maps (although they are quite apparent on the high-resolution maps).

3. Place one or more of the units on a map. A list of codes for all of the defined units may be obtained by executing the "Display Units" function (for screen output) or the "Output Units" function (printer output). Note that the location point of the map is in the upper-left-hand-corner of the screen, so that part of the units and the FEBA fall off the screen. The map location can be shifted so that the unit falls in the middle of the screen.

Changing the map location may be done in either of two ways, using the function "Change Map Location." First, the location point may be specified as the upper-left hand-corner of the map, and the coordinates (580000,4950000) specified. Or, alternatively, the location point may be specified as the center of the map. To use the latter option, it is helpful to know the locations of various items. The locations of all items in the data base may be obtained by executing the "Output Units" function. Note that the location of the fourth TRAILBLAZER MCS (which does fall on the map) is (595400,4919800). In the analysis that follows, we shall select

maps of higher and higher resolution, and observe the repositioning of this MCS. In each case, we will select the option to use the center of the map as the map location point, and specify the location of MCS no. 4 as the location point. (Note: for the first three map sets, all three MCSs on the map can be seen on the screen without relocating the map. For the higher resolution map sets, however, MCS no. 4 falls off the map in its original location, so that relocation is necessary to see it.)

By selecting either of the preceding map location point options, the user can see a large-scale display of the entire division or any subordinate units on the battlefield. Since the TRAILBLAZER MCSs are contained in the TRAILBLAZER section (code 1 1 1 2 0 0 1 1 1 1 0), it is interesting to display this unit. The user can see all of the MCSs in their canonical locations, the objective (target area) of the section, and the FEBA. Note that two of the MCSs are off the Spearfish map (i.e., they fall in "no data" cells). These MCS are of little interest in the test, since they will always remain in their canonical locations. The interesting cases are the three MCSs in the part of the map having data. In the repositioning that follows, we will, as mentioned above, display the results only for MCS no. 4 (even though the rules are applied to all of them).

4. Redraw a map (to get a clean display). Execute the repositioning rules, by selecting the function, "Reposition Subunits by Rules." Save the new positions (i.e., specify that the units should be saved, when presented with that option).

5. Change to the next-higher-resolution map set, by executing the function "Change Map Files." Select the following map files: geod0205.fil (terrain-type map); geoc0205.fil (elevation map); and geor0204.fil (road map). (Alternatively, exit the program and restart it with project file proj0205.fil, which specifies these maps.) Draw a map. (If desired, the map location may be specified to center the map at (595400,4919800) (the location of MCS no. 4) prior to drawing the map.) Execute the rules. Save the new positions.

6. Change to the next-higher-resolution map set (geod0204.fil, geoc0204.fil, geor0203.fil), by using the function "Change Map Files" or restarting the program and selecting project file proj0204.fil. Draw a map. (If desired, locate the map to center point (595400,4919800) prior to drawing the map.) Execute the rules. Save the new positions. Note: For the higher-resolution map files, the running time can range up to a couple of minutes to extract the map from the data files, and to process the rules.

7. Change to the next-higher-resolution map set (geod0203.fil, geoc0203.fil, geor0202.fil), by using the function "Change Map Files" or restarting the program and selecting project file proj0203.fil. Locate the map to have center point (595400,4919800). Draw a map. Execute the rules. Save the new positions.

8. Change to the next-higher-resolution map set (geod0202.fil, geoc0202.fil, geor0201.fil), by using the function "Change Map Files" or restarting the program and slecting project file proj0202.fil. Locate the map to have center point (595400,4919800). Draw a map. Execute the rules. Save the new positions.

9. Change to the next-higher-resolution map set (geod0201.fil, geoc0201.fil, geor0201.fil), by using the function "Change Map Files" or restarting the program and slecting project file proj0201.fil. Locate the map to have center point (595400,4919800). Draw a map. Execute the rules. Save the new positions.

4.5. Output the Item Locations.

10. Execute the function "Output Units." Specify the name of the output file to be out0201.fil (or any other desired name), or specify that the output is to the printer. All of the units in the file, including the repositioned items(TRAILBLAZER MCSs) will be written to this file (or to the printer, if so specified), in ASCII format.

In an application in which the Scenarist is being used to develop laydowns that will be input to another computer model (e.g., a tactical combat model), the user would wish to format the unit and equipment locations as required by that other model. To do so, the function _outputunits would be replaced by a function that produced a file in the desired format.

Appendix B contains an example of the output produced by the Output Units function. The sample output is the formatted versions of all units defined in Appendix A and stored in the generic and specific unit files.

4.6. Demonstrate User-Positioning of an Item.

11. Exit the program. Copy the file spec02.can to spec02.fil (hence resetting all items to canonical locations). Restart the program using the second-lowest-resolution map set (project file proj0205.fil, which includes maps geod0205.fil, geoc0205.fil, and geor0204.fil). Draw a map. Place the TRAILBLAZER section (code 1 1 1 2 0 1 1 1 1 0) on the map. Execute the function "Reposition Subunits by User," and select the option to reposition subunits of a specific unit.

The current (old) locations (in standard coordinates) of all of the MCSs are printed out, along with their radii and "placement codes":

```
     Subunit 1: .1 .1 25 0
     Subunit 2: .3 .3 25 0
     Subunit 3: .5 .1 25 0
     Subunit 4: .7 .3 25 0
     Subunit 5: .9 .1 25 0
```

Reposition the subunits as follows. The placement code 1 (user-reviewed, unchanged) means that the user has reviewed the subunit position but not moved it; the placement code 2 (user-suggested placement) means that the user is relocating the subunit, but that the rules are allowed to move it

from that position; the placement code 3 (user-mandated placement) means that the user is relocating the subunit, and the rules are not allowed to alter the position.

```
Subunit 1: .1 .1 25 1
Subunit 2: .3 .3 25 1
Subunit 3: .5 .05 25 2
Subunit 4: .65 .25 25 2
Subunit 5: .9 .05 25 3
```

Note that subunits 3 and 5 have been moved to within about half a kilometer of the front, in violation of the rule that the MCSs must be at least 2 km behind the front. As we will see, the rules will relocate subunit 3, but will not alter the position of subunit 5.

Save the reconfigured unit. Output the units to file out0202.fil.

Draw a map. Execute the function, "Reposition Subunits by Rules." Observe that subunit 3 is relocated to a position over 2 km in back of the front, but that the position of subunit 5 is unchanged. Save the reconfigured unit. Output the units to file out0203.fil.

Exit the program. Type the file out0201.fil (MS-DOS command "type out0201.fil |more"), noting the position (in real coordinates) of subunit 5 of unit 11: (591800,4924600). This is the original position of subunit 5. Now, type the file out0202.fil, noting the position to which the subunit 5 had been relocated: (591900,4925700). Finally, type the file out0203.fil, noting that this position is unchanged.

This completes this test procedure. During this test procedure, it was realized that it would be a desirable feature to be able to relocate subunits in real (map) coordinates, not just in standard coordinates. This modification is recommended for a later date.

4.7. Relocate the Division. Exit the program. Copy the file spec02.can to spec02.fil (hence resetting all items to canonical locations). Restart the program using the lowest-resolution map set (project file proj0206.fil). Draw a map. Place the division (code 1 1 1 2 0 0 0 0 0 0 0) on the map. Execute the function "Reposition Unit by User," to shift the division 5000 m to the east, i.e., to add 5000 to each x-coordinate of the unit corners. The program prints out the following coordinates:

```
610000 4925000
590000 4927000
588000 4905000
608000 4903000
```

Input the following coordinates:

```
615000 4925000
595000 4927000
593000 4905000
613000 4903000
```

Save the unit. Output the unit to file out0204.fil. Exit the program. Type the file out0204.fil, and note that, for every subordinate unit in the division, every x-coordinate is 5000 meters greater than the

corresponding coordinate in file out0201.fil (the original canonical unit). If the unit is displayed in a subsequent run, it will be translated 500 m to the east.

5.0. Summary

The test procedures and results described in the preceding section demonstrate the ability of the Scenarist to successfully perform every procedure specified in the test plan. In summary, the test was 100% successful, and demonstrates the ability of the Scenarist to reposition units using high-resolution data on a microcomputer.

The Scenarist program was designed to graphically demonstrate the relocation of a subunit by the rules, and to show which rules were violated if the canonical location placed it in an unsuitable location. These features were nicely demonstrated by the Beqaa Valley sample data set used in the course of developing the Scenarist system, and to a degree in this test, which used the GRASS/Spearfish map data. In any event, the Spearfish example did demonstrate the ability to use high-resolution data (comparable in resolution to Defense Mapping Agency (DMA) data), to have acceptable running times on a microcomputer, and to position and relocate a "realistic" system on digital terrain.

Appendix A

Scenarist Test Deployment

(Data used to define the generic units
and specific units used in the test.)

Appendix A. Scenarist Test Deployment

A. Units Used in Beqaa Valley Problem

```
Generic Unit file:genu01.fil
Specific Unit file: spec01.fil
FEBA File: feba01.fil

1. Define a Generic Division of Type 1

Name: Division1
Code: 1 1 1 1 0 0 0 0 0 0 0
Type: 1
Front/Rear boundary points: .5 .5
NFRONT: 2
NFRONT sets of (echelon,no.,type,idno): 5 1 1 0 5 2 1 0
NFRONT-1 boundary points: .5 .5
NREAR: 0
NALLAREA: 0
NMAJORSUBAREA: 0
NMINORSUBAREAREL: 0
NMINORSUBAREAABS: 0
NPOINT: 0

2. Copy Generic Division of Type 1 to a Specific Division, BLUE

New Unit's Name: Division1
Old Code:    1 1 1 1 0 0 0 0 0 0 0
New Code:    1 1 1 1 0 0 0 0 0 0 0
Parent Code: 1 1 1 0 0 0 0 0 0 0 0
Idno,parentidno: 141 131
4 Corners: 92000 30000 96000 46000 74000 48000 72000 34000
(Note: for Spearfish, SD, map use (in file spec02.fil):
610000 4925000 590000 4927000 588000 4905000 608000 4903000)
Objective: yes (i.e., input a "1")
(Note: for Spearfish, SD, map use 0 (in file spec02.fil)
Mission type: 1
2 Corners: 124000 36000 128000 32000
No. pts on avenue of approach: 1
Coords: 112000 38000

3. Copy Generic Division of Type 1 to a Specific Division, RED

New Unit's Name: RedDivision1
Old Code:    1 1 1 1 0 0 0 0 0 0 0
New Code:    2 1 1 1 0 0 0 0 0 0 0
Parent Code: 2 1 1 0 0 0 0 0 0 0 0
Idno,parentidno: 241 231
4 Corners: 110000 44000 106000 28000 128000 30000 130000 40000

(Note: for Spearfish, SD, map use (in file spec02.fil):
591000 4937000 611000 4935000 613000 4955000 593000 4957000)
Objective: no (i.e., hit "enter")

4. Define a Generic Brigade of Type 1
```

```
Name: MechBde1
Code: 1 1 1 1 1 0 0 0 0 0 0
Type: 1
Front/Rear Boundary Points: 0 0
NFRONT: 0
NREAR: 0
NALLAREA: 2
NALLAREA sets of Echelon,no.,type,idno: 8 1 1 0 8 2 2 0
NALLAREA sets of symbol location pts: .4 .4 .6 .55
(Note: this example is not realistic -- in reality, the
radar units are division assets, not brigade assets.)
NMAJORSUBAREA: 0
NMINORSUBAREAREL: 0
NMINORSUBAREAABS: 0
NPOINT: 0
```

5. Copy Generic Brigade of Type 1 to a Specific Brigade (Right Front of Division), BLUE

```
Name: MechBde1
Old Code:    1 1 1 1 1 0 0 0 0 0 0
New Code:    1 1 1 1 1 0 0 0 0 0 0
Parent Code: 1 1 1 1 0 0 0 0 0 0 0
Idno,parentidno: 151 141
Objective: no
```

6. Copy Generic Brigade of Type 1 to a Specific Brigade (Left Front of Division), BLUE

```
Name: MechBde2
Old Code:    1 1 1 1 1 0 0 0 0 0 0
New Code:    1 1 1 1 2 0 0 0 0 0 0
Parent Code: 1 1 1 1 0 0 0 0 0 0 0
Idno,parentidno: 152 141
Objective: no
```

7. Define a Generic Field Artillery Battery of Type 1

```
Name: FA_Batt1
Code: 1 1 1 1 1 0 0 1 0 0 0
Type: 1
Front/Rear Boundary Points: 0 0
NFRONT: 0
NREAR: 0
NALLAREA: 0

NMAJORSUBAREA: 0
NMINORSUBAREAREL: 0
NMINORSUBAREAABS: 4
NMINORSUBAREAABS sets of (echelon,no.,type,idno):
10 1 1 0 10 2 2 0 10 3 2 0 10 4 3 0
```

```
NMINORSUBAREAABS sets of (location point x,y, radius):
.4 .4 50 .13 .4 50 .87 .4 50 .4 .75 50
NPOINT: 0
```

8. Copy Generic Field Artillery Battery of Type 1 to Specific Field Artillery Battery (in brigade MechBde1, on right front of Division1), BLUE

```
Name: FA_Batt1
Old Code:    1 1 1 1 1 0 0 1 0 0 0
New Code:    1 1 1 1 1 0 0 1 0 0 0
Parent Code: 1 1 1 1 1 0 0 0 0 0 0
Idno,parentidno: 181 151
Objective: no
```

9. Define a Generic Air Defense Artillery Battery of Type 1

```
Name: ADA_Batt1
Code: 1 1 1 1 1 0 0 2 0 0 0
Type: 2
Front/Rear Boundary Points: 0 0
NFRONT: 0
NREAR: 0
NALLAREA: 0
NMAJORSUBAREA: 0
NMINORSUBAREAREL: 0
NMINORSUBAREAABS: 4
NMINORSUBAREAABS sets of (echelon,no.,type,idno):
10 1 4 0 10 2 5 0 10 3 5 0 10 4 5 0
NMINORSUBAREAABS sets of (location point x,y, radius):
.6 .55 50 .13 .2 50 .87 .2 50 .53 .5 50
NPOINT: 0
```

10. Copy Generic Air Defense Artillery Battery of Type 1 to Specific Field Artillery Battery (in brigade MechBde1, on right front of Division1), BLUE

```
Name: ADA_Batt1
Old Code:    1 1 1 1 1 0 0 2 0 0 0
New Code:    1 1 1 1 1 0 0 2 0 0 0
Parent Code: 1 1 1 1 1 0 0 0 0 0 0
Idno,parentidno: 182 151
Objective: no
```

12. FEBA

```
No. of points on FEBA: 3

Points: 99000 28000 101000 37000 104000 46000
(Note: for Spearfish, SD, map use (in file feba02.fil):
585000 4932000 600000 4931000 615000 493000)
```

B. Units Used in TRAILBLAZER Problem

```
Generic Unit file:genu02.fil
Specific Unit file: spec02.fil
FEBA File: feba02.fil

1. Define a Generic Division of Type 2
(Note: this division is assigned type 2 to distinguish it from the division
defined above)

Name: Division2
Code: 1 1 1 2 0 0 0 0 0 0 0
(Note: this division is assigned number 2 to distinguish it from the
division defined above.  In order to access units, the Scenarist requires
that every unit have a unique code.)
Type: 2
Front/Rear boundary points: .4 .4
NFRONT: 2
NFRONT sets of (echelon,no.,type,idno("ENTI"):5 1 2 0 5 2 2 0
NFRONT-1 boundary points: .5 .5
NREAR: 0
NALLAREA: 1
NALLAREA sets of ENTI: 7 1 1 0  (Note: this is the MI Bn)
NALLAREA sets of symbol location points: .4 .8
NMAJORSUBAREA: 0
NMINORSUBAREAREL: 0
NMINORSUBAREAABS: 1  (Note: this is the DivHQCo)
NMINORSUBAREAABS sets of ENTI: 8 1 3 0
NMINORSUBAREAABS sets of loc pt & rad: .5 .9 200
NPOINT: 0

2. Copy Generic Division of Type 2 to a Specific Division, BLUE

New Unit's Name: Division2
Old Code:    1 1 1 2 0 0 0 0 0 0 0
New Code:    1 1 1 2 0 0 0 0 0 0 0
Parent Code: 1 1 1 0 0 0 0 0 0 0 0
Idno,parentidno: 142 131
4 Corners: 610000 4925000 590000 4927000 588000 4905000 608000 4903000
Objective: no

3. Define a Generic Military Intelligence (MI) Battalion of Type 1

Name: MIBn1

Code: 1 1 1 2 0 0 1 0 0 0 0
Type: 1
Front/Rear boundary points: 0 0
NFRONT: 0
NREAR: 0
NALLAREA: 1
NALLAREA sets of ENTI: 8 1 5 0  (Note: this is the EW Co)
NALLAREA sets of symbol location points: .25 .35
NMAJORSUBAREA: 0
```

```
NMINORSUBAREAREL: 0
NMINORSUBAREAABS: 1  (Note: this is the MIBnHQCo)
NMINORSUBAREAABS sets of ENTI: 8 2 4 0
NMINORSUBAREAABS sets of loc pt & rad: .4 .8 100
NPOINT: 0

4. Copy Generic MI Battalion of Type 1 to a Specific MI Battalion, BLUE

Name: MIBn1
Old Code: 1 1 1 2 0 0 1 0 0 0 0
New Code: 1 1 1 2 0 0 1 0 0 0
Parent Code: 1 1 1 2 0 0 0 0 0 0 0
Idno, parentidno: 171, 142
Objective: no

5. Define a Generic Electronic Warfare (EW) Company of Type 1

Name: EWCo1
Code: 1 1 1 2 0 0 1 1 0 0 0
Type: 1
Front/Rear boundary points: 0 0
NFRONT: 0
NREAR: 0
NALLAREA: 1
NALLAREA sets of ENTI: 9 1 2 0  (Note: this is the SIGINT Processing Platoon
(SPP))
NALLAREA sets of symbol location points: .2 .3
NMAJORSUBAREA: 0
NMINORSUBAREAREL: 0
NMINORSUBAREAABS: 1  (Note: this is the EWHQPlt)
NMINORSUBAREAABS sets of ENTI: 9 2 1 0
NMINORSUBAREAABS sets of loc pt & rad: .25 .35 50
NPOINT: 0

6. Copy Generic EW Company of Type 1 to a Specific EW Company, BLUE

Name: EWCo1
Old Code: 1 1 1 2 0 0 1 1 0 0 0
New Code: 1 1 1 2 0 0 1 1 0 0 0
Parent Code: 1 1 1 2 0 0 1 0 0 0 0

Idno, parent idno: 181 171
Objective: no

7. Define a Generic SIGINT Processing Platoon (SPP) of Type 1

Name: SPP1
Code: 1 1 1 2 0 0 1 1 1 0 0
Type: 1
Front/Rear boundary points: 0 0
NFRONT: 0
NREAR: 0
```

NALLAREA: 2
NALLAREA sets of ENTI: 10 1 6 0 10 2 7 0 (Note: these are the TRAILBLAZER and TEAMPACK sections)
NALLAREA sets of symbol location points: .2 .3 .3 .3
NMAJORSUBAREA: 0
NMINORSUBAREAREL: 0
NMINORSUBAREAABS: 0
NPOINT: 0

8. Copy Generic SPP of Type 1 to a Specific SPP, BLUE

Name: SPP1
Old Code: 1 1 1 2 0 0 1 1 1 0 0
New Code: 1 1 1 2 0 0 1 1 1 0 0
Parent Code: 1 1 1 2 0 0 1 1 0 0 0
Idno, parentidno: 191 181
Objective: no

9. Define a Generic TRAILBLAZER Section of Type 1

Name: TBSec1
Code: 1 1 1 2 0 0 1 1 1 1 0
Type: 1
Front/Rear boundary points: 0 0
NFRONT: 0
NREAR: 0
NALLAREA: 0
NMAJORSUBAREA: 0
NMINORSUBAREAREL: 0
NMINORSUBAREAABS: 5 (Note: these are the five TRAILBLAZER Master Control Stations)
NMINORSUBAREAABS sets of ENTI: 11 1 1 0 11 2 1 0 11 3 1 0 11 4 1 0 11 5 1 0
NMINORSUBAREAABS sets of loc coords, radius: .1 .1 25 .3 .3 25 .5 .1 25 .7 .3 25 .9 .1 25 (Note: these coordinates correspond to the "W" configuration of the canonical TRAILBLAZER laydown)
NPOINT: 0

10. Copy Generic TRAILBLAZER Section of Type 1 to a Specific TRAILBLAZER Section, BLUE

Name: TBSec1
Old Code: 1 1 1 2 0 0 1 1 1 1 0
New Code: 1 1 1 2 0 0 1 1 1 1 0
Parent Code: 1 1 1 2 0 0 1 1 1 0 0
Idno, parentidno: 1101 191
Objective: yes
Mission type: 2
Coordinates of bounding rectangle of objective: 591000 4937000 613000 4955000

11. Define a Generic TEAMPACK Section of Type 2 (Note: no rules were developed for the TEAMPACK -- just the following canonical laydown. The section type is 2 to distinguish it from the TRAILBLAZER section, which is a section of type 1.)

Name: TPSec1
Code: 1 1 1 2 0 0 1 1 1 2 0
Type: 2
Front/Rear boundary points: 0 0
NFRONT: 0
NREAR: 0
NALLAREA: 0
NMAJORSUBAREA: 0
NMINORSUBAREAREL: 0
NMINORSUBAREAABS: 3 (Note: these are the three TEAMPACK teams)
NMINORSUBAREAABS sets of ENTI: 11 1 2 0 11 2 2 0 11 3 2 0
NMINORSUBAREAABS sets of loc coords, radius: .1 .05 25 .4 .05 25 .6 .05 25
NPOINT: 0

12. Copy Generic TEAMPACK Section of Type 2 to a Specific TEAMPACK Section, BLUE

Name: TPSec1
Old Code: 1 1 1 2 0 0 1 1 1 2 0
New Code: 1 1 1 2 0 0 1 1 1 2 0
Parent Code: 1 1 1 2 0 0 1 1 1 0 0
Idno, parentidno: 1102 191
Objective: yes
Mission type: 2
Coordinates of bounding rectangle of objective: 591000 4937000 613000 4955000

13. FEBA
No. of points on FEBA: 3
Points: 585000 4932000 600000 4931000 615000 493000)

Code:

1: Side
2: Army
3: Corps
4: Division
5: Brigade
6: Regiment
7: Battalion
8: Company/Battery
9: Platoon
10: Section
11: Squad/Team
(12: Platform)
(13: Equipment)

Appendix B. Sample Output File

(Formatted printout of all units defined in Appendix A, and stored in the generic unit and specific unit files.)