NASTRAN Binary Output Processor

Richard T. Wigginton
Andrew LeBlanc
Vijay Vasani
Electronic Data Systems Corporation

ABSTRACT

The NASTRAN Binary Output Processor (NBOP) was developed to provide an effective, flexible system to expedite postprocessing results from MSC/NASTRAN. The NBOP system consists of ISPF[1] functions, FORTRAN programs and DMAP ALTERS. The current version of NBOP is designed to run in a computing environment consisting of multiple large scale MVS/XA IBM mainframes using an IBM 3270 network for time sharing (TSO) access by engineers. NBOP provides a user a friendly, automatic system to rebuild MSC/NASTRAN binary output datasets generated by OUTPUT2 and OUTPUT4 DMAP statements. The system provides facilities to read an existing binary output dataset and extract specific data blocks. NBOP then creates a new binary output dataset placing the extracted data blocks in the new dataset in any desired order and optionally changing the data block names.

The NBOP system was originally developed at the request of the users of ODYSSEY[2] an optimization system developed by GMR/EDS. ODYSSEY utilizes MSC/NASTRAN through a custom DMAP program that provides multiple load, boundary, and analysis capabilities. This custom DMAP, because it combines the features of statics and normal mode solution sequences, by necessity, uses data block names some consider non-standard. ODYSSEY users wanted to access a variety of programs to postprocess ODYSSEY/NASTRAN results stored in the binary output dataset. These postprocessing programs usually had very specific NASTRAN data block input requirements. NBOP now provides the capability to take the binary output from the custom ODYSSEY DMAP and automatically create new binary output datasets acceptable as input to almost any appropriate MSC/NASTRAN postprocessing program. The NBOP system has proven very useful to the ODYSSEY user community and provides facilities often helpful to the general NASTRAN user. The NBOP design and implementation compromises documented in this paper have significance to anyone integrating MSC/NASTRAN into existing design systems.

[1] Interactive System Productivity Facility (ISPF), a software product of International Business Machines, Inc.

[2] A.k. Gupta, R.T. Wigginton, C.J. Wouden, J.F. Yang, M.E. Botkin and R.V. Lust, "ODYSSEY: A Structural Optimization Program Using MSC/NASTRAN", MSC/NASTRAN World User Conference, Los Angeles, California, 1988.system

NASTRAN Binary Output Processor

Richard T. Wigginton
Andrew LeBlanc
Vijay Vasani
Electronic Data Systems Corporation

MSC/NASTRAN[1,2] manipulates internal data through DMAP[3] statements using named "files" or data blocks. These data blocks are given an arbitrary name for identification. The output information created by DMAP modules is written positionally into data blocks. The first output block in a specific DMAP statement will always contain the same information regardless of what block name is coded. Other than the special data blocks generated by the MSC/NASTRAN prefix, all DMAP data block names are not standard. They are determined by conventional usage.

It is useful to be able to pass information contained in some internal MSC/NASTRAN data block to an external program for postprocessing. OUTPUT2 and OUTPUT4 DMAP statements provide this capability by causing selected data blocks to be specifically written into a dataset. These DMAP output facilities usually write data in binary format, sometimes called unformatted or internal form, because this approach allows a FORTRAN program to read data using unformatted reads. In unformatted reads, the data is moved directly from the external dataset into memory without conversion, a significant processing efficiency. Throughout this paper these OUTPUT2/4 generated datasets are called either binary output datasets or binary datasets.

In binary output datasets MSC/NASTRAN data blocks are always stored by name. Postprocessing programs use those names to identify the data blocks. Generally, MSC/NASTRAN postprocessing programs are designed to read the binary output dataset looking for very specific data block names. Other programs are designed to expect data blocks to be stored in a specific order. A few require both specific names and specific data block order. If the need to use a particular postprocessor is anticipated before the MSC/NASTRAN run, it is a simple matter to obtain and include the appropriate DMAP ALTERs[4]. These ALTERs will write the data blocks required by the postprocessing program to a binary dataset.When the ALTERs appropriate to a particular postprocessor have not been anticipated before the run, a problem occurs. More problems occur when several MSC/NASTRAN postprocessors are required, the MSC/NASTRAN solution uses non-standard names, or a looping solution produces several data blocks with the same name. The point is data block name and order conflicts do occur. When that happens, it can be prohibitively expensive to rerun MSC/NASTRAN with different ALTERS for each postprocessor.

At General Motors this problem was recognized many years ago, and a system of standard ALTERS was created to build special binary output datasets for each standard solution sequence. More than forty postprocessors currently in use at GM use these special GM standard binary output datasets. The standardization

approach has been so effective that, the postprocessing of standard solution sequences, for most users, is automatic. Unfortunately, some vendor-supplied MSC/NASTRAN postprocessing programs do not use the same data blocks or names as GM standards do. Some of the special purpose programs written by GM and EDS do not use GM standards. Some custom DMAP programs in use at GM can not follow all the GM standard naming conventions or data block order requirements.

ODYSSEY[5], the GMR/EDS optimization system, uses a custom DMAP program to provide combined analysis (statics, inertia relief and normal modes) with multiple load and boundary conditions. There were many requests from ODYSSEY users to postprocess ODYSSEY/NASTRAN[6] data using GM standards. There were also requests to provide the capability to postprocess ODYSSEY/NASTRAN data using the programs provided by outside vendors or by using in-house programs that did not follow GM standards. One basic problem with ODYSSEY/NASTRAN binary output is that the combined analysis capability requires the use of different data block names than the standard solution sequences. Another problem with ODYSSEY/NASTRAN output is looping. The data blocks from modal analysis results are duplicated each time a different boundary condition is analyzed.

When ODYSSEY/NASTRAN was first introduced, a serious attempt was made to create the ODYSSEY/NASTRAN binary output compatible with GM standards. Generally, postprocessing with GM standard programs worked as long as the data block names, required by specific postprocessors, were on the binary output dataset. Because the GM standards were devised for standard MSC/NASTRAN solution sequences, there were some obvious block name problems. For example, ODYSSEY/NASTRAN custom DMAP uses a conventional block name for static strain energy, but modal strain energy uses a special block name. GM standards use the same conventional block name for both static strain energy and modal strain energy. Therefore, static strain energy could be processed by GM standard postprocessing programs, but modal strain energy could not be postprocessed. The boundary looping problem is another example. In ODYSSEY/NASTRAN binary output, only the data blocks from the first modal subcase could be processed by GM standard programs. Some nonstandard postprocessing programs could not use ODYSSEY/NASTRAN binary output at all because the block names required by these programs were so different. Often the appropriate information was available but stored under a different data block name.

Several unsuccessful approaches were considered for providing the needed flexibility for ODYSSEY/NASTRAN postprocessing. An obvious solution was to modify all postprocessing programs to accept the binary output from ODYSSEY/NASTRAN as an alternative.While this could have been done, we felt this approach would not be cost effective. Several programs would have required minor modifications. Other programs would have required major changes to handle reading the data blocks from the looping, multiple boundary condition solutions. We could get outside vendors to make the required modifications but not without paying for the development effort involved. It became obvious that some way was needed to restructure the ODYSSEY/NASTRAN output data blocks to a form acceptable to any postprocessing program.

There were four basic approaches initially considered to provide ODYSSEY/NASTRAN with the required output flexibility.

Option 1 - Use a FORTRAN program to read a generalized binary output dataset created by the ODYSSEY/NASTRAN analysis run. This program would create a new customized binary dataset for input into each specific postprocessing program. The FORTRAN program would need to be able to extract, rename and reorder the data blocks. The generalized binary output from the ODYSSEY/NASTRAN analysis run would need to contain every data block that might be used for postprocessing. In essence, the generalized binary output dataset becomes a neutral file for MSC/NASTRAN results.

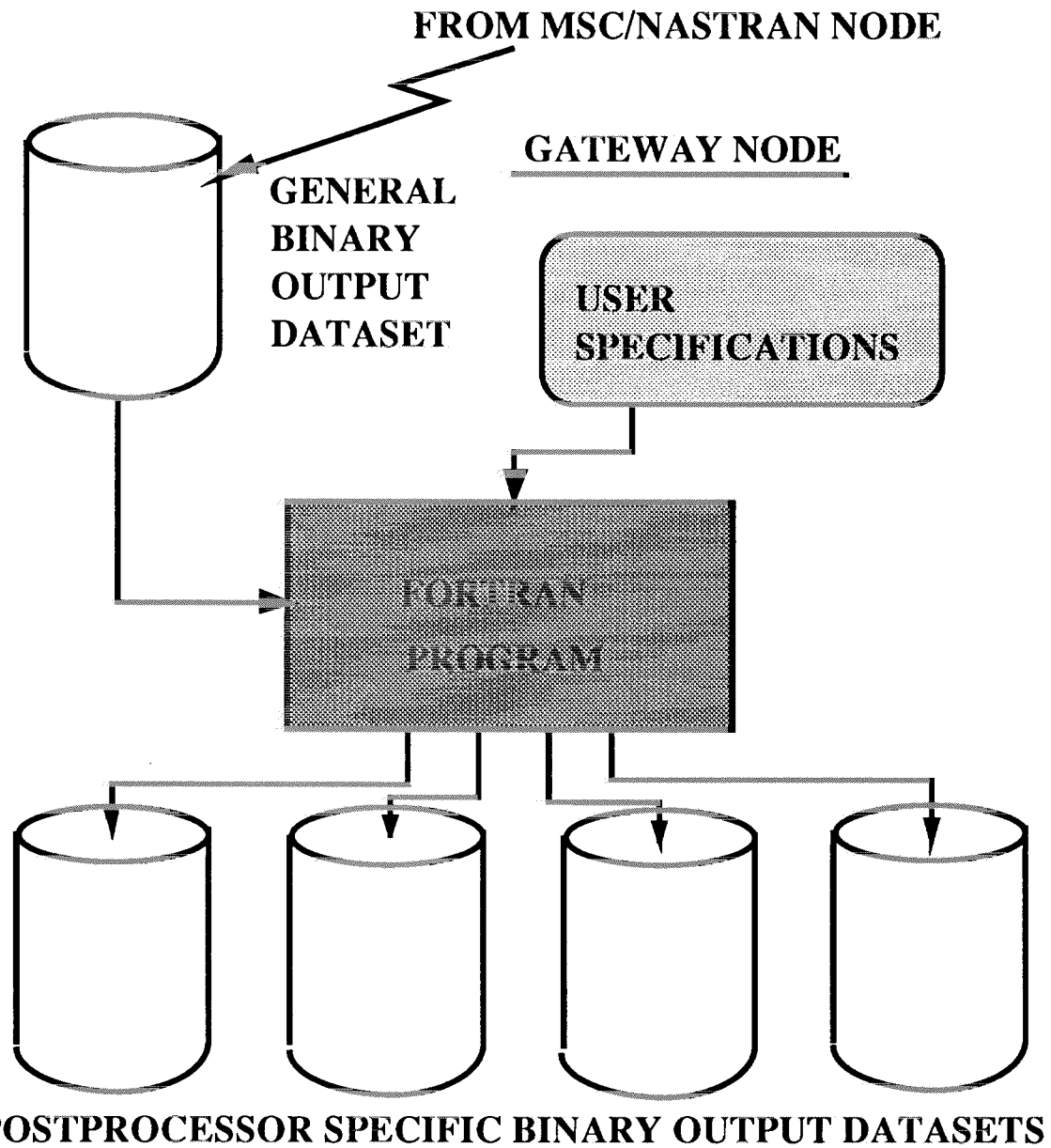

Figure 1: OPTION 1

Option 2 - Use a MSC/NASTRAN DMAP program to read the original binary output, using INPUTT2 and INPUTT4 DMAP statements. This program would write out selected blocks using OUTPUT2 and OUTPUT4 DMAP statements to create a new customized binary dataset for input into each specific postprocessor. The initial binary output could be less general than option 1 because the DMAP could process data blocks into special forms. However, a system would have to be devised to build the required custom DMAP program. A special MSC/NASTRAN run would be required to generate postprocessor input binary datasets, but several datasets could be output in a single MSC/NASTRAN run.

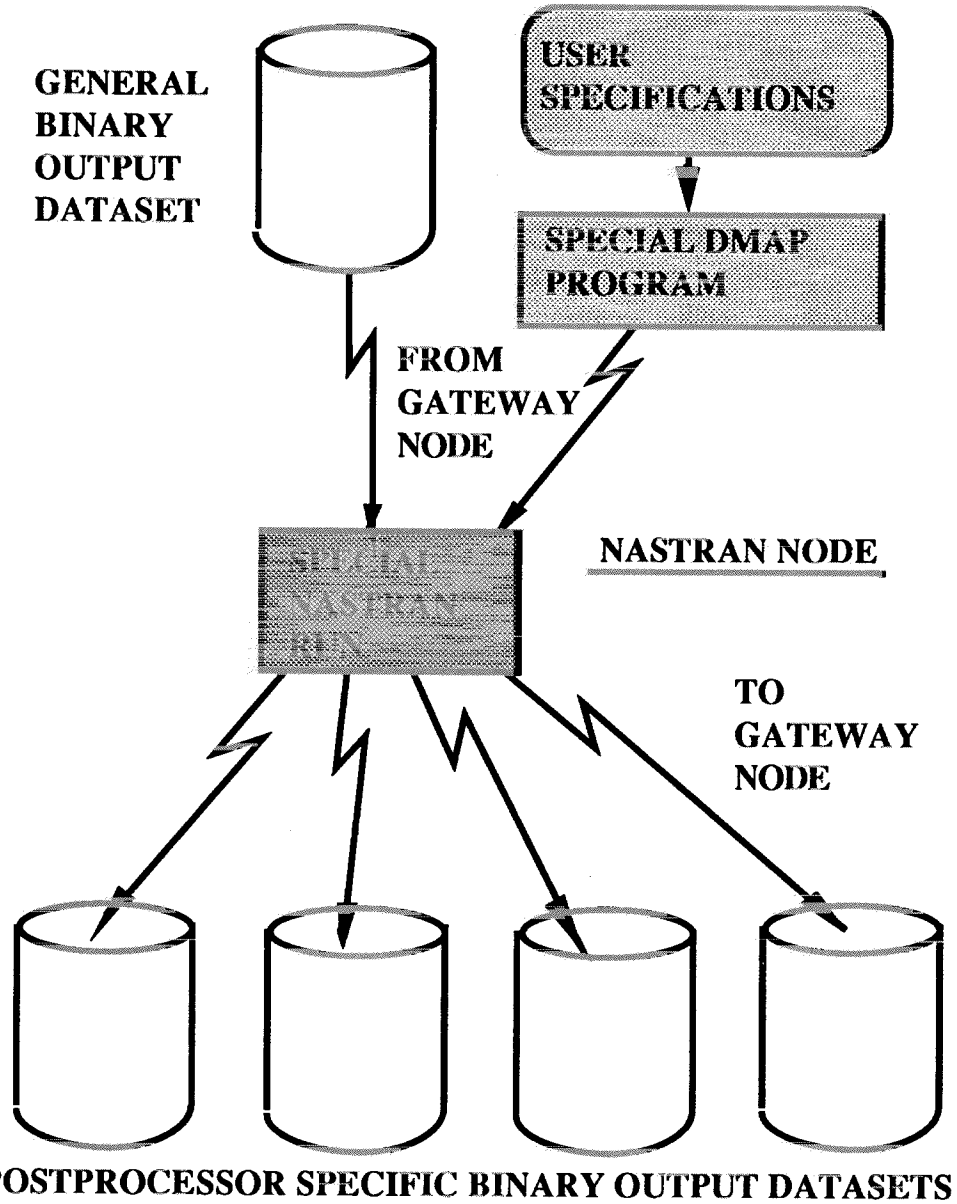

Figure 2: OPTION 2

Option 3 - Use a MSC/NASTRAN restart DMAP program to create customized binary datasets for input into postprocessors. The data blocks would be read from the MSC/NASTRAN database and output as required. The control over what dataset(s) were produced would be through a system of parameters or a system to automatically assemble the appropriate DMAP program as suggested in option 2. A special MSC/NASTRAN run would be required to generate the postprocessor input binary datasets. Then, datasets could be output in a single MSC/NASTRAN run.

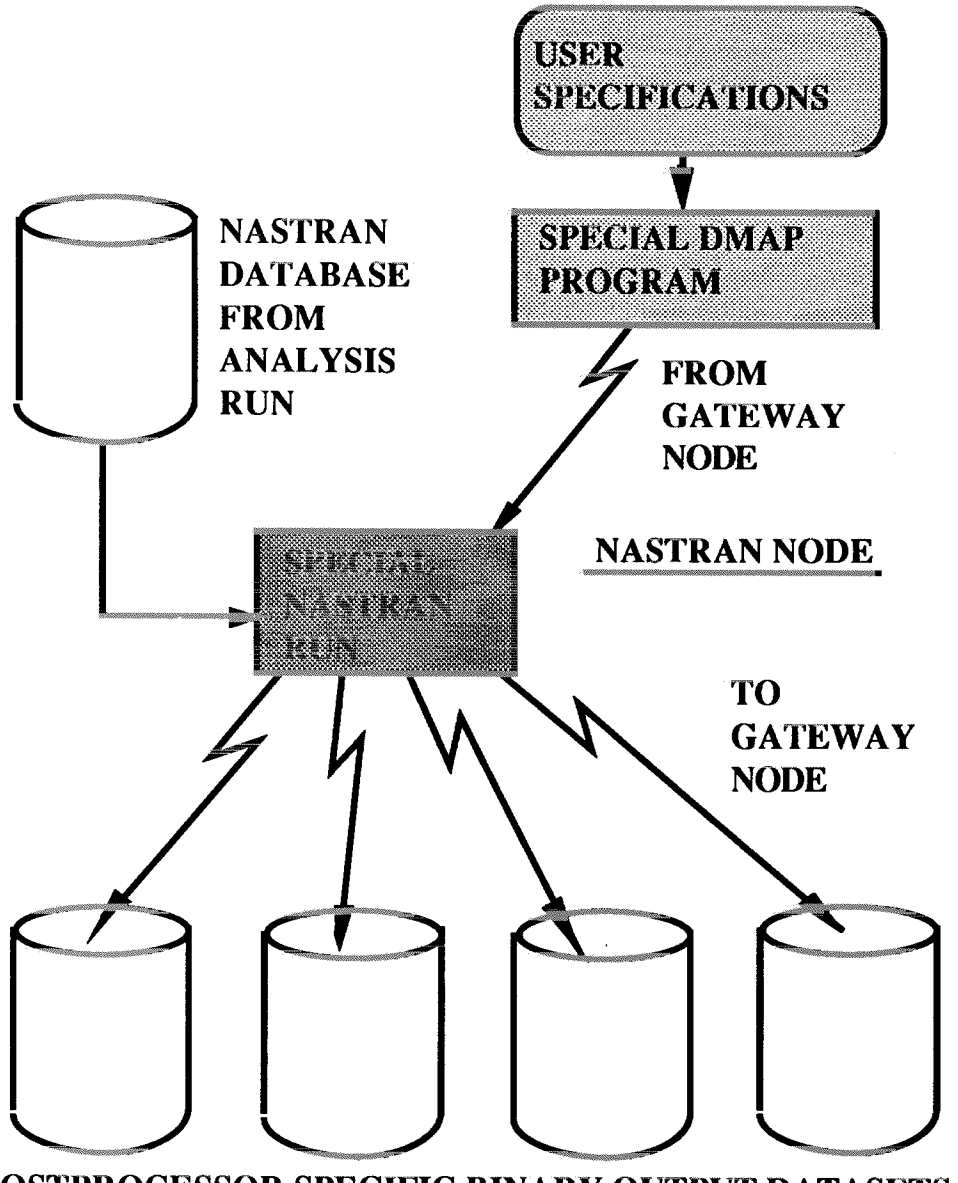

Figure 3: OPTION 3

Option 4 - Add the capability to the custom ODYSSEY DMAP to output several binary output datasets, one for each postprocessing program. The current binary dataset used by ODYSSEY would not be modified, but several additional OUTPUT2 and OUTPUT4 DMAP statements would be added to write the appropriate binary output to different datasets. The ODYSSEY JCL[7] procedures would manage these additional datasets.

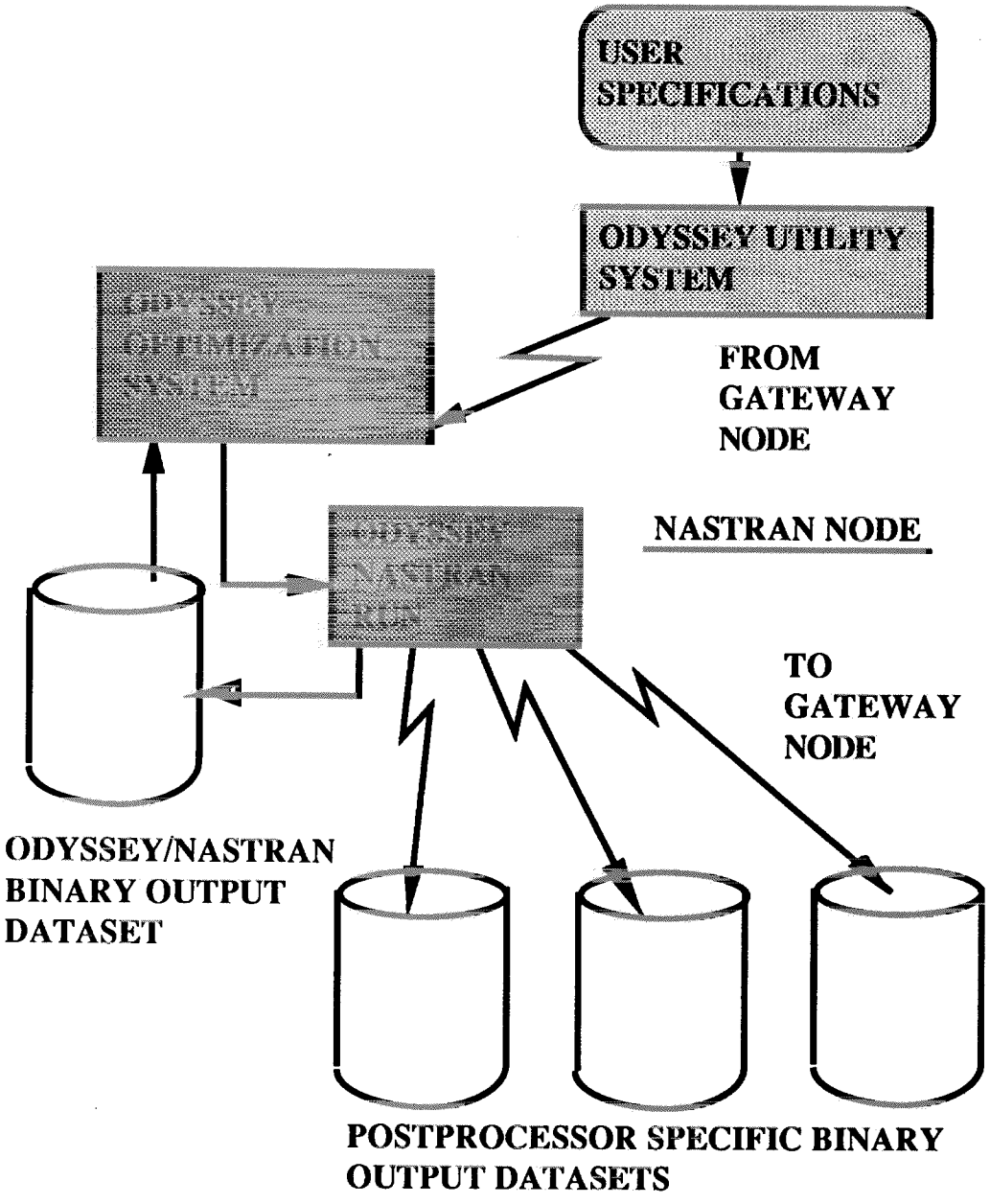

Figure 4: OPTION 4

Each of the above options worked effectively. The problem was to select the best option for any given situation. The best option for the current situation might not be the best option for someone else, or for us at a different point in time. The major criteria was to determine how much work would be required to implement each option, and how much potential disruption would be caused to the current code environment.

The environment the selected system must work in must be understood. Most "production" engineering work at General Motors is done through the the EDS Information Processing Center at Auburn Hills, Michigan. MSC/NASTRAN is run on an IBM3090 with a vector facility. The various operating divisions use other 3090s in the complex as gateways to the MSC/NASTRAN machine node. Most of the users sign on the gateway nodes using the IBM 3270 time sharing network. MSC/NASTRAN jobs are set up on the gateway nodes and submitted to the MSC/NASTRAN machine. All appropriate output is returned to the gateway node using data transfer utilities. ODYSSEY/NASTRAN currently uses MSC/NASTRAN version 65E.

Option 4 had the advantage of producing binary output datasets directly usable by postprocessing programs as soon as a specific ODYSSEY/NASTRAN run was completed. The disadvantage was the added complexity and the maintenance problems with the DMAP program. The logistics of moving several binary datasets from the MSC/NASTRAN machine node to the gateway nodes where the users were was another problem.This required the modification of several procedures and the addition of several control options to the current system. A major concern was that with so many binary output datasets the whole system would become very confusing. This approach was rejected.

Option 3 had the advantage of letting the ODYSSEY DMAP write into the binary output only those blocks needed by ODYSSEY. As long as the MSC/NASTRAN database remained available, any binary output for a specific postprocessor could be created. A disadvantage to this approach was the complexity of making special MSC/NASTRAN runs, particularly where the user might be attached to a machine node that did not have MSC/NASTRAN available. A complex system of JCL and CLIST/Panels[8] needed to be created to submit the special MSC/NASTRAN run(s) required and to return the binary output files to the local system. MSC/NASTRAN is changing the format of their database between version 65 and version 66. Any special DMAP programs written for this system would shortly require significant conversion. This approach was rejected.

Option 2 had the advantage of using a DMAP that would require little conversion between MSC/NASTRAN version 65 and version 66. Another advantage was that the user had direct control over the binary output dataset. This option required writing more data blocks to the binary output datasets, creating some increase in the initial MSC/NASTRAN analysis processing time. Unlike the MSC/NASTRAN database, this dataset is usually returned to its local gateway system. This approach also required a special setup and MSC/NASTRAN run control system similar to option 3. This was considered a distinct disadvantage, so the approach was rejected.

Option 1 had the advantage of restructuring the MSC/NASTRAN binary output on the local machine node. The system ran on-line through TSO[9], avoiding the need to write an extensive, special MSC/NASTRAN run system. The FORTRAN programs required to read data blocks were available in ODYSSEY and were easily incorporated into the system. The major disadvantage was that all data blocks needed for postprocessing had to be written on the binary output dataset during the analysis run. This extra output requirement caused an increase in processing time. However, this option produced the most flexible system with a minimum amount of programming effort. It was selected and became known as NBOP for (MSC/) NASTRAN Binary Output Processor.

Once option 1 was selected, other requirements had to be included in the final design. The NBOP system had to be easy to use. It had to be a practical, automatic facility for commonly used postprocessing programs. It also had to provide a flexible system for restructuring MSC/NASTRAN binary output datasets. A project proposal was developed and customer approval was obtained.

Building the binary output read and block extract routines was the first NBOP function to be developed. The major reason for writing the NBOP system was to process the ODYSSEY/NASTRAN binary output. Therefore, we utilized routines written for ODYSSEY that read binary output. Additional thought lead to the realization that, although it was a simple matter to read and output data blocks, a flexible way to drive the extract, rename and write process needed to be developed. A simple input dataset that specifies the extract, rename and write process was devised. It is called the extraction command file.

The new utility using the extraction command file was called UXTRACT. This program option reads a binary output file and writes a new binary file based on the specifications in the extraction command file (see figure 5). The extraction command input consists of a list describing the data block order in the target output file. Each line in the extraction command file contains three fields. The first field is the data block sequence number on the input file. The second field contains the block name on the input file. An optional third field specifies the block name on the target output file. An entry in the third field implies that the block will be renamed. The UXTRACT utility follows the list in the extraction command file, extracts the specified block, optionally renames it and writes it to the output dataset.

```
 +- 1st field
 |    +- 2nd field
 |    |      +- 3rd field
 V    V      V
+-+-------+-------
****************************** TOP OF DATA ***********************************
01EQEXIN
02EST
03CASECC
05OENER   ONRGY1
08PHIG    UGVS
***************************** BOTTOM OF DATA *********************************
```

Figure 5: EXTRACTION COMMAND DATASET LISTING

A discussion of the ODYSSEY/NASTRAN binary output dataset will help clarify the ideas used in the rest of this paper. The ODYSSEY DMAP writes several data blocks to the binary output dataset that are subsequently read by ODYSSEY routines. The block order on this dataset is a direct reflection of the subcase structure created by ODYSSEY optimization requirements. An example of the blocks on a typical ODYSSEY/NASTRAN binary output follows:

```
CASECC        Initial setup
EQEXIN        data blocks
CSTM, etc.

LAMA          1st modal subcase
PHIG, etc.    modal search path

LAMA          2nd modal subcase
PHIG, etc..   modal search path

UGV           All static and inertia
OES1          relief output
OEF1, etc.    static search path
```

There is an obvious pattern. A modal subcase output always starts with a LAMA table. If there is no LAMA table, assume that no modal analysis is requested in the run. Each modal subcase output starts with a LAMA data block and ends with the next LAMA data block. The modal subcase output ends with the UGV matrix. The static subcase output starts with UGV matrix and ends with the end of file mark.

To build a system that was easy to maintain and was flexible, we needed to create a way of automatically generating the extraction command file for different applications. This requirement lead to the development of the application filter option of the NBOP program. Application filters were designed to automate the creation of binary files used for input to specific postprocessing programs. The filter option uses an application specific filter file to answers the following questions:

What data blocks does the application require?
What data blocks are optional?
What data blocks need to be renamed?
What name is used for a renamed data block?
In what order are the blocks placed?

The filter file is used as input to the NBOP filter option. The NBOP filter uses the NBOP scan option to obtain the order and names of the data blocks on the input binary dataset. The specifications in the application filter file cause the filter option to create an extraction specification file. This file is input into the UXTRACT NBOP option. If a required block is not found, the process is stopped, and the user is informed that the desired postprocessor can not be used. Most

applications have two filter files. One controls the static and inertia relief search path, and the other controls the modal search path.

The filter file (see figure 6) is a list of data block name records. Each line or record of the file consists of three fields. The first field gives the block name to be extracted from the input binary dataset. The second field contains the name to be used for that block on the output binary dataset. A blank in the second field means no data block name change is needed. The third field specifies whether the data block is required or optional. The order the data blocks are listed in the first field specifies the data block order on the output binary dataset. The filter file uses the convention that any field with a "$" in column one is treated as a comment. This convention allows documentation to be included along with the block name specifications.

```
******************************** TOP OF DATA ***********************************
$
$            **** NORMAL MODE APPLICATION FILTER
$                   CONVERT SOL 3 TO SOL 63
$                       NOTE: R = REQUIRED
$                             O = OPTIONAL
  'CASECC'    ,            ,R
  'EQEXIN'    , 'EQEXINS' ,R
  'BGPDT'     , 'BGPDTS'  ,R
  'ECT'       , 'ECTS'    ,R
  'MI'        ,           ,R
  'LAMA'      ,           ,R
  'PHIG'      , 'UGVS'    ,R
*******************************BOTTOM OF DATA ********************************
```

Figure 6: EXAMPLE FILTER FILE

An analysis of potential postprocessing applications showed that some data blocks needed to be added to the ODYSSEY/NASTRAN binary output dataset. The addition of these data blocks gives ODYSSEY/NASTRAN compete postprocessing flexibility. The table below describes the data blocks added to the standard output from ODYSSEY/NASTRAN:

CASE	Case control information
ECT	Element connectivity table
EST	Element summary table
PHIGBS	Modal grid point displacement in basic coordinate
MI	Modal mass matrix
UGVBS	Grid point displacement matrix in basic coordinates
OST	Strain values output table
OQG	Forces on SPCs output table

Note: the production of some of these blocks is controlled by output requests in MSC/NASTRAN CASE CONTROL.

Another problem was encountered while trying to make the ODYSSEY/NASTRAN output compatible with GM standards. GM standards were originally developed with programs that read the MSC/NASTRAN checkpoint tape. There was a block available in that system called BULKDATA. The current GM standard production MSC/NASTRAN batch JCL procedure used a special program to create the standard binary output dataset with the BULKDATA block on it. This binary dataset is passed to MSC/NASTRAN, and the output from MSC/NASTRAN is appended after the BULKDATA block. To build a postprocessor dataset compatible with complete GM standards, the BULKDATA block was needed on our binary dataset.

We obtained a copy of a utility program[10] to create the BULKDATA block. It was decided to make this utility available as an option. This option added the BULKDATA block to an existing binary dataset.

During testing, other utility programs were added to our requirement specifications. A simple scan program option was created. It listed the data block names on the binary output dataset. A program option to list the contents of the case control block, CASECC, was built. These programs gave users the ability to easily determine the contents of binary datasets. Because large scale IBM computer operating systems do not handle the variable, blocked, spanned record format of the MSC/NASTRAN binary output very effectively, many of the IBM supplied list and edit utilities would not work with these datasets. Therefore, specific listing utilities became a necessity.

The decision was made to incorporate the final NBOP System into the ODYSSEY UTILITIES SYSTEM. The ODYSSEY UTILITIES SYSTEM (see figure 7) is a collection of ISPF/Dialog Manager[11] panels, programs and CLISTs that expedites running ODYSSEY. The ODYSSEY/NASTRAN postprocessing utility (see figure 8) was added to the ODYSSEY UTILITIES menu.

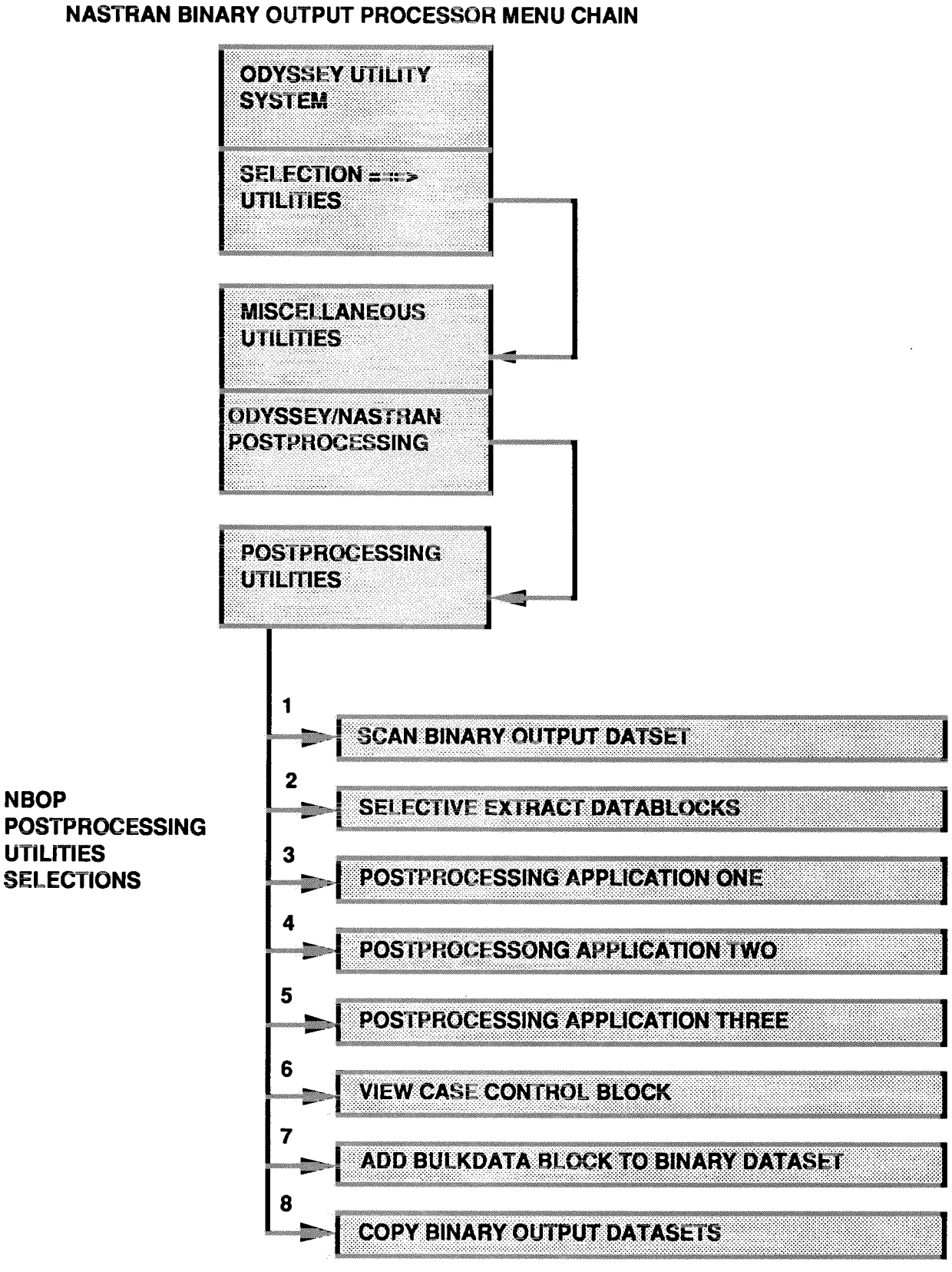

Figure 7: NBOP MENU CHAIN

The ODYSSEY/NASTRAN postprocessing utility has several options.

```
============================ ODYSSEY UTILITIES SYSTEM ============================

                  ODYSSEY/NASTRAN POSTPROCESSING UTILITIES

01/12/90                                                                    20:18
================================================================================

SELECT ===>

                1 SCAN ...........SCAN A NASTRAN BINARY OUTPUT DATASET
                2 XTRACT .........SELECTIVELY EXTRACT OUTPUT DATA BLOCKS
                3 XXXXXX..........FIRST POSTPROCESSING APPLICATION
                4 YYYYYY .........SECOND POSTPROCESSING APPLICATION
                5 ZZZZZZ .........THIRD POSTPROCESSING APPLICATION
                6 CASECC .........VIEW THE NASTRAN BINARY OUTPUT CASE CONTROL
                7 BULKDATA .......APPEND BULKDATA TO A NASTRAN BINARY OUTPUT
                8 COPY ...........COPY A NASTRAN BINARY OUTPUT DATASET


--------------------------------------------------------------------------------
PF3 (END) = RETURN    PF1 = HELP                                PANEL:ODYVSP01
```

Figure 8: ODYSSEY/NASTRAN POSTPROCESSING UTILITIES MAIN MENU EXAMPLE

SCAN, the first postprocessor utility menu option, produces a listing of the data blocks on the specific binary output dataset (see figure 9).

```
****************************** TOP OF DATA ***********************************
 SEQ# BLK TYPE      BLOCK NAME

   1   TABLE     NAME=EQEXIN                                ERROR RETURN=       0
   2   TABLE     NAME=BGPOT                                 ERROR RETURN=       0
   3   TABLE     NAME=CASECC                                ERROR RETURN=       0
   4   TABLE     NAME=LAMA                                  ERROR RETURN=       0
   5   MATRIX    NAME=PHIG    NCOLS=    18 NROWS=  1692    ERROR RETURN=       0
   6   MATRIX    NAME=UGV     NCOLS=     6 NROWS=  1692    ERROR RETURN=       0
   7   TABLE     NAME=OUGV1                                 ERROR RETURN=       0
   8   TABLE     NAME=OEF1                                  ERROR RETURN=       0
   9   TABLE     NAME=OES1                                  ERROR RETURN=       0
****************************** BOTTOM OF DATA ********************************
```

Figure 9: SCAN OUTPUT EXAMPLE

The second postprocessing utility menu option, XTRACT (see figure 10), allows the user to run the UXTRACT utility using his own extraction commands. This XTRACT entry also has another option called AXTRACT. AXTRACT automatically takes ODYSSEY/NASTRAN output and creates binary output datasets that look like they came from standard MSC/NASTRAN analysis solution sequences using GM standard ALTERs. AXTRACT reads the ODYSSEY/NASTRAN binary output and determines the number of modal and static subcases. A special application filter system produces an extraction command file which is input into UXTRACT to produce the appropriate number of analysis binary output datasets.

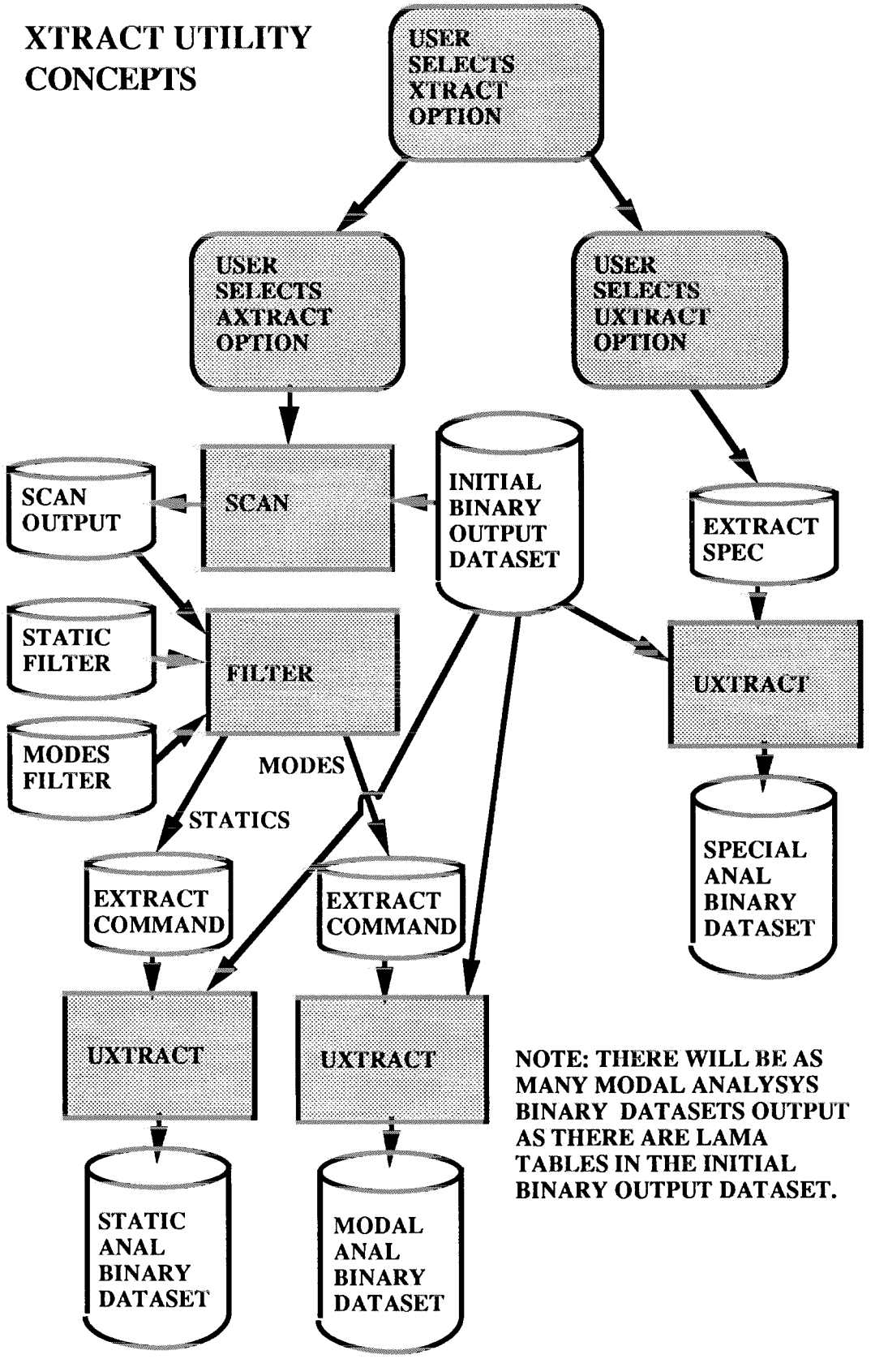

Figure 10: XTRACT UTILITY CONCEPTS

The 3rd, 4th and 5th postprocessing menu options are specifically requested postprocessing applications (see figure 11). Every effort was made to make these custom postprocessing options as simple to use as possible. The applications are structured in a similar way. An application-specific filter file is used with the filter system to create extraction commands. New binary output datasets are then built for input into the selected application. In many instances, the application or preapplication programs have been incorporated into the option CLIST. As more postprocessing requests are received, we expect to add more specific applications. One of the real advantages of the NBOP system organization is the ease with which postprocessing applications can be added.

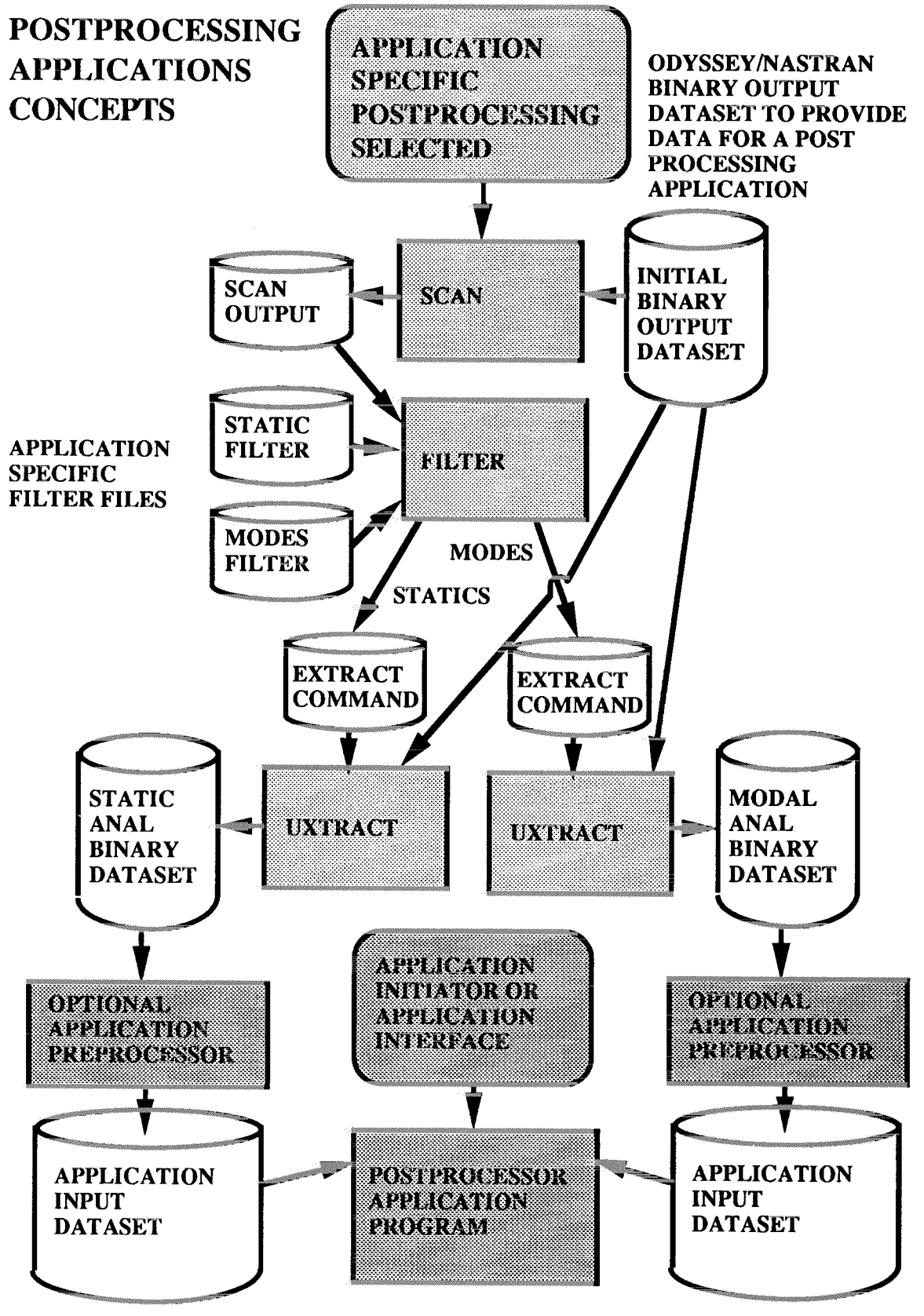

Figure 11: POSTPROCESSING APPLICATIONS CONCEPTS

Option 6 of the postprocessing menu utility reads and displays the contents of the CASECC block (see figure 12). This is a great aid in identifying unknown binary output datasets.

```
******************************** TOP OF DATA ***********************************

                        SUMMARY OF THE ANALYSIS SUBCASES

 SC#                          TITLE/SUBTITLE/LABEL                    OTHER INFO

      BAR 7 ODYSSEY TEST PROBLEM

  1   FREQUENCY ANALYSYS TEST                                  SPC SET  =     1
                                                               LOAD SET =     0
                                                               MPC SET  =     0
                                                               EIGN SET =     1

  2   LOADS AT GRID POINT 3                                    SPC SET  =     1
                                                               LOAD SET =     2
                                                               MPC SET  =     0

  3   LOADS AT GRID POINT 4                                    SPC SET  =     1
                                                               LOAD SET =     3
                                                               MPC SET  =     0

******************************** BOTTOM OF DATA ********************************
```

Figure 12: CASECC UTILITY EXAMPLE OUTPUT

Option 7 of the postprocessing utility menu adds the BULKDATA block to an existing binary output dataset. The data for the BULKDATA block comes from the MSC/NASTRAN input deck specified on the option request panel.

Option 8 of the postprocessing utility menu is a simple utility to copy binary output datasets. As mentioned before, on IBM machines the binary output dataset has a record format of VBS (variable, blocked, spanned). Most of the standard copy utilities available through TSO will not copy VBS datasets. This copy utility option was included as a user convenience.

The NBOP system has pleased many ODYSSEY users by allowing them to utilize postprocessing programs they have used frequently with standard MSC/NASTRAN solution sequences. The automatic nature of NBOP has made using postprocessors with ODYSSEY as easy or easier than using these programs with standard MSC/NASTRAN binary output. The facilities available in the NBOP utilities are general. Any MSC/NASTRAN binary output may be manipulated. Consequently, NBOP is being increasingly utilized by general MSC/NASTRAN users.

Acknowledgements:

We would like to thank the General Motors Analysis Methods Group at Current Product Engineering for development and maintenance of GM Standards. This project was much easier to accomplish because of these standards. Special thanks to Bob Hegman for his programming talent and the historical perspectives he provided on the development of GM standards.

Richard Wigginton, served as the primary consultant on this project. Most of the programming and creative design was done by Andrew LeBlanc. Vijay Vasani was responsible for the quality of the production system through testing and had a large influence on the user interface design.

Special thanks to Cheryl Wouden for her help in proof-reading and editing this document.

MAJOR REFERENCES:

MSC/NASTRAN PROGRAMMER'S MANUAL,
The MacNeal-Schwiendler Corporation, Los Angeles, California, 1982.

MSC/NASTRAN USER'S MANUAL,
The MacNeal-Schwiendler Corporation, LosAngeles,California, 1985.

NUMBERED NOTES AND REFERENCES:

[1] MSC/NASTRAN is a registered trademark and service trademark of The MacNeal-Schwendler Corporation. NASTRAN is a registered trademark of the National Aeronautics and Space Administration (NASA).

[2] MSC/NASTRAN is a proprietary and enhanced version of NASTRAN developed and maintained by the MacNeal-Schwendler Corporation.

.[3] Direct Matrix Abstraction Programming, The data block oriented language used by MSC/NASTRAN to specify problem solutions.

[4] ALTERs are the MSC/NASTRAN system that allows editing standard DMAP solutions.

[5] A.K. Gupta, R.T. Wigginton, C.J. Wouden, J.F. Yang, M.E. Botkin and R.V. Lust, "ODYSSEY: A Structural Optimization Program Using MSC/NASTRAN, MSC/NASTRAN World User Conference, Los Angeles, California, 1988.

[6] ODYSSEY/NASTRAN implies an MSC/NASTRAN execution using the ODYSSEY custom DMAP program that provides combined analysis capability.

[7] JCL, Job Control Language, The procedural specification language used on large scale IBM mainframe computers to set up and run programs, usually in batch execution.

[8] CLISTS - Procedural specification language used to set up and run programs through TSO(see 9). Panels are screen menus and data entry formats used on IBM 3270 display tube networks to access TSO applications.

[9] TSO - Time Sharing Option, the time sharing system available on large scale IBM mainframe computers.

[10] Original program attributed to Bob Hegman at GM Engineering Staff.

[11] ISPF/Dialog Manager - An IBM program product available to expedite application development in a 3270 network environment. ISPF is currently Interactive System Productivity Facility with Dialog Manager providing specific facilities for integration of user written panels, CLISTs and Programs.