Using MSC/MVISION to Compare MSC/NASTRAN Results with Flight Test Data

Kirsten Husak
Software Engineer/Systems Administrator
for Flight Sciences
Kirsten_Husak@readwo.com

Raytheon Systems Company
Waco, Texas

Abstract

At Raytheon Systems Company in Waco, Texas, a MSC/MVISION database has been developed which allows the comparison of flight test data with MSC/NASTRAN results output. A model can be opened in MSC/PATRAN, and strain gauge results generated during a test flight viewed directly on the model. The MSC/NASTRAN strain output can be applied to the model and contrasted with test flight strains in the MSC/PATRAN graphical user interface. If the results are reconciled within a predetermined degree of tolerance, it will then be assumed that the model is accurate. A model predicting the behavior of the aircraft after it has been modified can then be assumed to be accurate also. After the aircraft has been modified, more flight test data will be collected, which can then be compared with the post-modification predictions to aid in support of FAA certification of the modified aircraft. MSC/MVISION will provide a user-transparent interface between MSC/PATRAN and the raw flight test data, allowing engineers to view the accuracy of their models with real-time strain measurements within the MSC/PATRAN graphical environment. Because Raytheon Waco specializes in aircraft modifications, the concentration in this project has been on the comparison of flight test data with airframe finite element models, but the application of this methodology for comparison of strains with models can be used in any industry which employs modeling and testing procedures.

Introduction

When using MSC/MVISION, it is important to remember that the use of a database to manage information requires an entire process. The first step is an extensive one of planning, then the process moves on to development, and finally implementation can begin.

Planning for this integration of flight test data with model results was carried out in conjunction with the MacNeal-Schwendler Corporation Grapevine employees Gerry Norvell (Senior Application Engineer) and John Parady (Application Engineer), and with Ajit Kundu and Michael Farley of the Methods and Finite Element Group of Raytheon Systems Company Waco. The planning process continued from December 1998 through February of 1999, and involved telephone consultation as well as several meetings, moving from a general statement of goals through several refinements which brought an ever clearer vision of the type of development required for the desired end result, to the point of including trial development for clarification of ideas and requirements.

Development began in February of 1999. At this point requirements were clear, and included a MSC/MVISION databank which would contain all the relevant real-time strain gauge data, pcl code to add forms to the MSC/PATRAN environment, and dpi code to give the MSC/PATRAN database access to the databank.

Implementation currently includes a populated MSC/MVISION databank containing the flight test data. Pcl code is in place which brings up forms in MSC/PATRAN to place markers on the finite element model showing the location of the strain gauges, their type, and/or their measured value. The gauges can be filtered by location, type, or number. The markers have a variety of shapes and colors which allows a complete graphical view of gauge placement on the model, allowing comparison of results with actual value. This comparison was achieved by use of dpi to access the MSC/MVISION databank and retrieve information about real-time flight test data within the MSC/PATRAN environment. Forms are in place for the comparison of model results with selected strain gauges or ranges of strain gauges both in tabular format (the MSC/PATRAN spreadsheet) and in graphical format.

Planning -- the First Step in the Process

Planning began with upper-level discussions of what the engineer truly needed from this comparison of real-time data with model data. A flowchart of the process was created, showing in general terms the steps involved in the whole process (see figure 1). The first statement of goals came from an in-depth discussion at Raytheon in Waco as to the needs of the engineer who will be the end-user of the data. The conclusions of this discussion were: there are two sets of data involved: MSC/NASTRAN data consisting of ascii F06 files and binary OP2 files (which are read into MSC/PATRAN to view results), and flight test data which consists of strain gauge measurements from actual flights during which a variety of maneuvers were performed. What the end-user would need was a comparison of the unmodified model results with real-time data from flight tests of the unmodified aircraft, in order to determine (within some degree of tolerance) whether the model represents true conditions. If the results do reconcile, it will be assumed that modeling techniques are accurate in prediction of the behavior of the aircraft after modification. Post-modification, similar comparisons can be made in support of FAA certification for the modified aircraft.

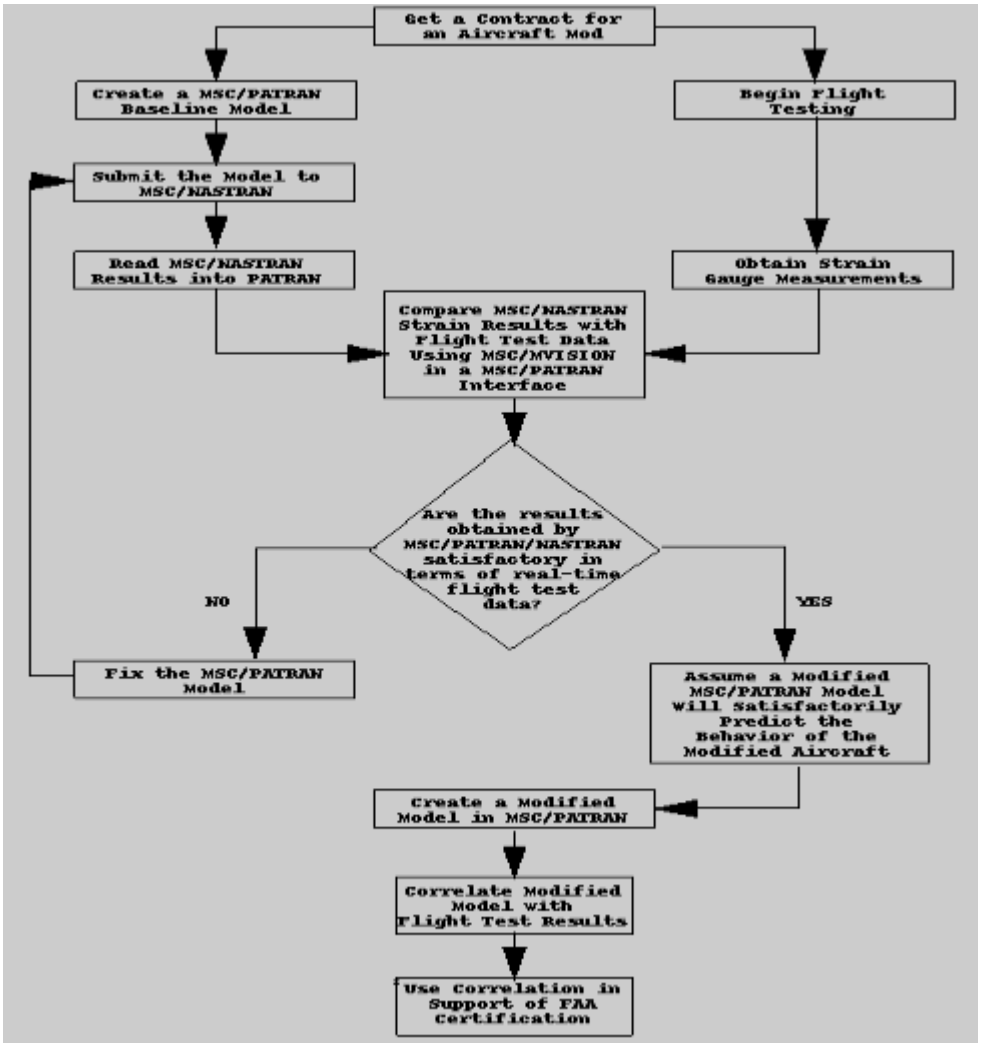

Figure 1 – High-Level Process Flow Chart

In a meeting with the MSC representatives December 10th of 1998, this plan was further refined. Action items such as retrieval of raw flight test data and the XYZ locations of strain gauges were assigned. The end goal was restated: an

engineer will open a model within MSC/PATRAN. He will send the model through a MSC/NASTRAN run, then read the results into MSC/PATRAN. He will then perform some simple operation which will generate a correlation report by accessing the MSC/MVISION databank.

At this point, discussions were still higher-level, revolving around concepts and refining the end-purpose. At the next meeting with MSC, planning became more specific. A trial schema for the databank was developed (see figure 2) consisting of five hierarchy levels.

```
$
$  /xlv0/people/keh/flight_test/mvis/ftest.def
$  Date : 1999-02-15
$  The header on the database is M/VISION Created by keh on 1999-02-15
$
HIERARCHY ADD VERSION INSTRUMENT FLIGHT RUN SOURCE
ATTRIBUTE ADD VERSION_NAME CHAR 72 1 "" "Version" -0- -0-
ATTRIBUTE ADD AC_TYPE           CHAR 72 1 "" "Aircraft Type" -0- -0-
RELATION ADD VERSION VERSION_NAME AC_TYPE

ATTRIBUTE ADD INSTRUMENT_NUMBER CHAR 72 1 ""    "Instrument ID" -0- -0-
ATTRIBUTE ADD INSTRUMENT_TYPE   CHAR 72 1 ""    "Instrument Type" -0- -0-
ATTRIBUTE ADD STRN_GAUGE_LEG    CHAR 4  1 ""    "Strain Gauge Leg" -0- -0-
ATTRIBUTE ADD INSTRUMENT_LOC    CHAR 72 1 ""    "Instrument Location" -0- -0-
ATTRIBUTE ADD LOC_X             REAL 1  1 "in." "X Location of Instrument" -0- -0-
ATTRIBUTE ADD LOC_Y             REAL 1  1 "in." "Y Location of Instrument" -0- -0-
ATTRIBUTE ADD LOC_Z             REAL 1  1 "in." "Z Location of Instrument" -0- -0-
ATTRIBUTE ADD INSTRUMENT_ORIENT CHAR 72 1 ""    "Instrument Orientation" -0- -0-
RELATION ADD INSTRUMENT INSTRUMENT_NUMBER INSTRUMENT_TYPE
RELATION ADD INSTRUMENT STRN_GAUGE_LEG
RELATION ADD INSTRUMENT INSTRUMENT_LOC LOC_X LOC_Y LOC_Z
RELATION ADD INSTRUMENT INSTRUMENT_ORIENT

ATTRIBUTE ADD FLIGHT_NUMBER     INTEGER 1  1 "" "Flight Number" -0- -0-
ATTRIBUTE ADD FLIGHT_DATE       CHAR    72 1 "" "Flight Date" -0- -0-
RELATION ADD FLIGHT FLIGHT_NUMBER FLIGHT_DATE

ATTRIBUTE ADD RUN_NUMBER        INTEGER 1  1 "" "Run Number" -0- -0-
ATTRIBUTE ADD RUN_DESCRIP       CHAR    72 1 "" "Run Description" -0- -0-
ATTRIBUTE ADD AVG_VAL           REAL    1  1 "" "Average Value" -0- -0-
ATTRIBUTE ADD MIN_VAL           REAL    1  1 "" "Minimum Value" -0- -0-
ATTRIBUTE ADD MAX_VAL           REAL    1  1 "" "Maximum Value" -0- -0-
ATTRIBUTE ADD STD_DEV_VAL       REAL    1  1 "" "Standard Deviation" -0- -0-
ATTRIBUTE ADD AVG_DEV_VAL       REAL    1  1 "" "Average Deviation" -0- -0-
RELATION ADD RUN RUN_NUMBER RUN_DESCRIP
RELATION ADD RUN AVG_VAL MIN_VAL MAX_VAL STD_DEV_VAL AVG_DEV_VAL

ATTRIBUTE ADD BOGUS     CHAR 32  1 "" "Place Holder" -0- -0-
RELATION ADD SOURCE BOGUS

ATTRIBUTE ADD STRNvsTIME REAL 3 0       "sec.;micro strain" "Time;Strain"
RELATION ADD STRNvsTIME STRNvsTIME

ATTRIBUTE ADD ACCELvsTIME REAL 3 0      "sec.;in./sec^2" "Time;Acceleration"
RELATION ADD ACCELvsTIME ACCELvsTIME
```

Figure 2 -- Trial Schema

At this point, it was decided to begin loading sample data in order to determine what refinements would be needed, both to the schema and to the end-user requirements.

Because the data was in tabular format, the spreadsheet tool was ideal for populating the databank. After a few attempts at using the data in an unmodified format, it was decided that the data must be transposed in order to 'put' the data into MVISION. This is because, while the raw data was in columnar records (one column per strain gauge), MSC/MVISION requires that data be loaded in rows rather than columns.

The data was 'read' into a location many rows down on the spreadsheet, both to leave room for the transposed data and to keep end-users from becoming confused. The 'polyline' function was used to take the data which represented time ticks and plot it against gauge readings, until reaching the end of the data block. Once the polylines were in place, strain gauge titles were copied and pasted above the corresponding polylines. Because statistical properties would be necessary for the end-user to make meaningful comparisons with raw data, the MSC/MVISION functions 'avg', 'stddev', 'min', 'max', and 'avgdev' were performed for each column (see figure 3).

INSTRUMENT_NAME	SG185	SG186	SG187	SG188	SG189
AVG	11.10413	-9.906654	-36.01045	-24.81835	-37.05826
STD_DEV	1.089178	0.8693379	1.013731	1.296935	3.328823
MIN	9.13941	-14.4511	-40.3007	-27.8482	-50.3702
MAX	13.9152	-7.39354	-33.2212	-21.1982	-24.5264
COUNT	170	170	170	170	170
STRNvsTIME	<polyline>	<polyline>	<polyline>	<polyline>	<polyline>

Figure 3 – Data Obtained from Raw Data

With the polylines and statistical measurements in place, in columnar format, the data was ready to be transposed. In fact the raw data was now irrelevant to the transpose, because the data to be 'put' into MSC/MVISION consisted only of the statistics, the polylines, and the instrument ID (in this case a strain gage ID). For this, an external function 'block_transpose' was used. These external functions come packaged with MSC/MVISION and consist of fortran or c code compiled into an MVISION function library. The function is called by this command: =exfunc("mvfunc","block_transpose"<C94:BO100>) (ex_func tells MVISION the function is external, mvfunc is the name of the function library, block_transpose is the function to be used from within that library, and C94:BO100 is the block to transpose).

With the data in the desired format, additional information such as version name, aircraft type, flight number, run number, flight date, and instrument type could be added. In this case this data was the same for each strain gauge because the raw data in the spreadsheet was all obtained during the same flight. A 'put' was performed and population of the trail databank was begun, the databank now containing not the raw data but rather the polylines and statistical values obtained from the raw data.

While the above appears to fall into the category of development rather than planning, some idea of the shape of the data and the capabilities with which the databank could be imbued was necessary to the continuing planning process. With actual data loaded into the trial databank, the statement of purpose was yet again refined:

> Statistics will be calculated in MVISION and "set in stone" (no access by end-user) to ensure data consistency. Any other functions which may be required will also be performed and recorded in MSC/MVISION (such as VonMises stress, etc.). Within the PATRAN environment, the engineer performing the correlation will use a form where he will 1) choose a node or set of nodes from the model which he wishes to compare to strain gauges in the same location (he must determine the degree of tolerance he wishes to use). 2) The engineer will request a strain gauge or set of strain gauges in the same location and MSC/MVISION will (transparently to the engineer) perform this query and provide him with the locations of the strain gauges, pinpoint them graphically on the model, and supply him with a plot and statistics for the strain gauges. 3) The engineer will decide which value he will use for correlation to the MSC/PATRAN/NASTRAN results within the model (for example the average). 4) The MSC/MVISION data will be superimposed on the PATRAN data in plot format and extracted in such a way that it can be used as part of a stress report.

As often happens, in the course of development some stumbling blocks were discovered which required modifications to this statement, but the spirit of the plan was maintained throughout development and implementation.

Development -- the Second Step in the Process

There were essentially three directions of effort during the development process: refinement of the MSC/MVISION databank, establishment of pcl functions to provide forms and actions within MSC/PATRAN, and development of dpi code to allow the pcl forms to access the databank.

Developing the Databank

As more experience was gained with the data, the schema was modified several times until it reached its current form (figure 4):

```
$
$  /xlv0/people/keh/flight_test/mvis/strnhi.def
$  Date : 1999-02-15
$  The header on the database is M/VISION Created by keh on 1999-02-15
$
HIERARCHY ADD VERSION INSTRUMENT FLIGHT RUN SOURCE
ATTRIBUTE ADD VERSION_NAME    CHAR 72 1 "" "Version" -0- -0-
ATTRIBUTE ADD AC_TYPE           CHAR 72 1 "" "Aircraft Type" -0- -0-
RELATION ADD VERSION VERSION_NAME AC_TYPE

ATTRIBUTE ADD INSTRUMENT_NUMBER CHAR 72 1 ""    "Instrument ID" -0- -0-
ATTRIBUTE ADD INSTRUMENT_TYPE   CHAR 72 1 ""    "Instrument Type" -0- -0-
ATTRIBUTE ADD STRN_GAUGE_LEG    CHAR 4  1 ""    "Strain Gage Leg" -0- -0-
ATTRIBUTE ADD INSTRUMENT_LOC    CHAR 72 1 ""    "Instrument Location" -0- -0-
ATTRIBUTE ADD LOC_X             REAL 1  1 "in." "X Location of Instrument" -0- -0-
ATTRIBUTE ADD LOC_Y             REAL 1  1 "in." "Y Location of Instrument" -0- -0-
ATTRIBUTE ADD LOC_Z             REAL 1  1 "in." "Z Location of Instrument" -0- -0-
ATTRIBUTE ADD INSTRUMENT_ORIENT CHAR 72 1 ""    "Instrument Orientation" -0- -0-
RELATION ADD INSTRUMENT INSTRUMENT_NUMBER INSTRUMENT_TYPE
RELATION ADD INSTRUMENT STRN_GAUGE_LEG
RELATION ADD INSTRUMENT INSTRUMENT_LOC LOC_X LOC_Y LOC_Z
RELATION ADD INSTRUMENT INSTRUMENT_ORIENT

ATTRIBUTE ADD FLIGHT_NUMBER     INTEGER 1  1 "" "Flight Number" -0- -0-
ATTRIBUTE ADD FLIGHT_DATE       CHAR    72 1 "" "Flight Date" -0- -0-
RELATION ADD FLIGHT FLIGHT_NUMBER FLIGHT_DATE

ATTRIBUTE ADD RUN_NUMBER        INTEGER 1  1 "" "Run Number" -0- -0-
ATTRIBUTE ADD RUN_DESCRIP       CHAR    72 1 "" "Run Date" -0- -0-
ATTRIBUTE ADD AVG_VAL           REAL    1  1 "" "Average Value" -0- -0-
ATTRIBUTE ADD MIN_VAL           REAL    1  1 "" "Minimum Value" -0- -0-
ATTRIBUTE ADD MAX_VAL           REAL    1  1 "" "Maximum Value" -0- -0-
ATTRIBUTE ADD STD_DEV_VAL       REAL    1  1 "" "Standard Deviation" -0- -0-
ATTRIBUTE ADD AVG_DEV_VAL       REAL    1  1 "" "Average Deviation" -0- -0-
ATTRIBUTE ADD NUM_PTS           INTEGER 1  1 "" "Number of Time Inc" -0- -0-
RELATION ADD RUN RUN_NUMBER RUN_DESCRIP
RELATION ADD RUN AVG_VAL MIN_VAL MAX_VAL STD_DEV_VAL AVG_DEV_VAL
RELATION ADD RUN NUM_PTS

ATTRIBUTE ADD BOGUS     CHAR 32  1 "" "Place Holder" -0- -0-
RELATION ADD SOURCE BOGUS

ATTRIBUTE ADD STRNvsTIME REAL 3 0       "sec.;micro strain" "Time;Strain"
RELATION ADD STRNvsTIME STRNvsTIME

ATTRIBUTE ADD ACCELvsTIME REAL 3 0      "sec.;in./sec^2" "Time;Acceleration"
RELATION ADD ACCELvsTIME ACCELvsTIME
```

Figure 4 -- Current Schema

The hierarchy consists of five levels, VERSION (unmodified or modified model), INSTRUMENT (in our case a strain gauge), FLIGHT, RUN (there may be several runs during a flight as the pilot performs several maneuvers), and SOURCE (a place-holder required by MSC/MVISION). Attributes of the various levels of the hierarchy are self-explanatory, and are added to the relations in such a manner as to add efficiency when sorting and querying the databank.

A template for loading data was completed with the use of a session file and an external function, each called one time to prepare the spreadsheet template for loading data. A file stats.ses runs through the columns of data creating the polylines and the statistics, assuming a finite maximum number of strain gauges. The external function column_id.F calculates the boundaries of the block to be transposed and of the rows to be 'put' into the databank and stores the values in cells in the spreadsheet. An additional spreadsheet was created to load the XYZ locations of all the strain gauges (see figure 5).

INSTRUMENT_NUMB>	LOC_X	LOC_Y	LOC_Z	INSTRUMENT_ORIE>	QUERY	COMMAND
SG1	1070	-7.97	391.31	T1	INSTRUMENT_NUMB>	:=modify("strnhi.des">
SG10	1070	-4.498	91.07	T1	INSTRUMENT_NUMBER	like 'SG10'
SG11	1150	-9	88	T1	INSTRUMENT_NUMBER	like 'SG11'
SG110	1770	-4.76	358.66	T1	INSTRUMENT_NUMBER	like 'SG110'
SG111	1770	-23.67	356.54	T1	INSTRUMENT_NUMBER	like 'SG111'
SG112	1770	23.67	356.54	T1	INSTRUMENT_NUMBER	like 'SG112'

Figure 5 -- XYZ Location Spreadsheet

With the spreadsheet template complete, a combination of files were written to automate data loading. A session file (load.ses) creates a file named block.dat, to which it writes the information contained in the cells calculated by the spreadsheet external function column_id. Load.ses then makes a call to compile the pcl program get_data.pcl (!!INPUT get_data.pcl). Load.ses then executes the pcl function, which creates yet another session file using the data from block.dat (see figure 6).

```
$3.2
$ created from automated input
MvSpreadsheetCommand(
"I5=``:=ex_func(""gnfunc"",""block_transpose""(AA14:CM20))``" )
MvSpreadsheetCommand( "C1=``=put(""strnhi.des"",C4:O4,C5:O69)``")
```

Figure 6 -- Session File do_transpose.ses Created by PCL Function

Control then returns to the session file load.ses which runs the pcl-created session file do_transpose.ses. As you see in figure 6, this session file performs the actual transpose within the spreadsheet and then executes the 'put' into the MSC/MVISION database.

Developing the PCL

The pcl code produces the forms and the actions performed in forms which are seen by the end-user within MSC/PATRAN. The Makefile (stripped of standard calls and compilations shows what pcl files are being used to create forms (see figure 7).

```
# Name of the target pcl library
PclLibrary = ./readwo_tool.plb
.
.
.
PclObjects = \
        marker.pob\
        attribute.pob\
        filter.pob\
        select_data.pob\
        correlate.pob\
        select_results.pob\
        sprdsht.pob\
        user.pob
```

Figure 7 -- Extract from the pcl Makefile

The marker code defines the Marker Form (which applies the strain gauge markers to the model in their XYZ locations). The attribute form allows a choice of the appearance of the markers. The filter allows gauges to be divided by name, by type, by XYZ range, etc. Select data is used by both marker and correlate; it performs a select function on the databank. Select results is used by the correlate form, to compare the gauge measurements with model results, and the user pob displays menus. These forms will be shown in the section on implementation.

Developing the DPI

What the dpi (database programmatic interface) needed to do was perform a search through the databank's hierarchy as specified by the user on the form within MSC/PATRAN. During any interaction with the databank, MSC/PATRAN spawns dpi which queries the databank and redirects the output to a file. When the dpi has completed execution, control is returned to MSC/PATRAN and the results are displayed.

The crucial dpi in our case was "select" access to the MSC/MVISION databank, which specifies a set of attributes and a query, then returns an array of data. The code used was based on an example select program delivered with MSC/MVISION. An expression is created, consisting of a select with parameters table, attribute(s), and query. A data structure is returned, the rows are counted, and then printed row by row. This output is redirected to standard out. Since MSC/PATRAN has already specified that output is redirected to a file, and was suspended until the file had been written and control returned, MSC/PATRAN can now read the file row by row into an array, then display the array of data on the proper form.

Implementation -- Step Three of the Process

Detailed planning has resulted in organized and purposeful development: the databank is ready to be built, and the graphical interface between the databank and the finite element model is ready to be viewed and utilized.

To build the databank a series of session files were written which were used in mvbatchbuilder to automate the population process. These session files repetitively called the (already developed) session file load.ses to bring in all the files containing strain gauge data. An extract from such a session file can be seen in figure 8.

```
$3.2
MvSpreadsheetCommand( "open ""/xlv0/people/keh/flight_test/mvis/gerry.spd""" )
MvSpreadsheetCommand( "B1=``BFT13001.st1``" )
MvSpreadsheetCommand( "F5=1" )
MvExecuteSessionFile( "/xlv0/people/keh/flight_test/mvis/load.ses" )
MvSpreadsheetCommand( "B1=``BFT13001.st2``" )
MvSpreadsheetCommand( "F5=1" )
MvExecuteSessionFile( "/xlv0/people/keh/flight_test/mvis/load.ses" )
.
.
.
MvSpreadsheetCommand( "B1=``BFT13064.st3``" )
MvSpreadsheetCommand( "F5=64" )
MvExecuteSessionFile( "/xlv0/people/keh/flight_test/mvis/load.ses" )
MvSpreadsheetCommand( "B1=``BFT13065.st1``" )
MvSpreadsheetCommand( "F5=65" )
MvExecuteSessionFile( "/xlv0/people/keh/flight_test/mvis/load.ses" )
MvSaveDatabase()
```

Figure 8 -- Extract from a Data Loading Session File

The open and save commands are needed by mvbatchbuilder, first to open the spreadsheet template, and last to ensure a save as mvbatchbuilder times out quickly and does not automatically save the modified databank. Cell B1 receives the name of the file containing the raw data, and cell F5 receives the run number within the flight, then the data is loaded by load.ses. An example session in mvbatchbuilder:

```
MSC/M_MATERIALS_BUILDER 1997.0801 has obtained 75 concurrent license(s) from
FLEXlm  per a request to execute on a SGI IRIX system at 1999-04-27 15:54:34.


Batch Builder Main Menu
1) Open Databank
2) Set Current Databank
3) Close Databank
4) Create Databank
5) Save Databank
6) Save As
7) Change Units
8) Load Databank
9) Dump Databank
10) Create databank Index File
11) Delete Rows
12) Show Databank
13) Modify Schema
14) Spreadsheet
15) Setup
16) Session File
17) Exit
Enter Selection (X to exit menu) >1
1

Open Databank Menu (Current Directory=/xlv0/people/keh/flight_test/mvis)
1) /disk03/msc/mvision32/db
2) strnhi.des
3) testing_strnhi.des
4) Show all files
5) Other
Enter Selection (X to exit menu) >3
3
Info : Opening databank testing_strnhi.des
Info : Done opening databank

Current Databank = testing_strnhi.des
Current Units = default


Batch Builder Main Menu
1) Open Databank
2) Set Current Databank
```

```
3) Close Databank
4) Create Databank
5) Save Databank
6) Save As
7) Change Units
8) Load Databank
9) Dump Databank
10) Create databank Index File
11) Delete Rows
12) Show Databank
13) Modify Schema
14) Spreadsheet
15) Setup
16) Session File
17) Exit
Enter Selection (X to exit menu) >16
16

Session File Menu
1) Execute Session File
2) Convert Old Session File
3) Execute Old Session File
4) Convert CLI file
5) Type in command
6) Exit
Enter Selection (X to exit menu) >1
1

Session File Menu(Current Directory = /xlv0/people/keh/flight_test/mvis)
1) builder.ses.1
2) builder.ses.2
3) builder.ses.3
4) do_transpose.ses
5) lastrun11.ses
6) lastrun13.ses
7) load.ses
8) patran.ses.01
9) patran.ses.02
10) patran.ses.03
11) patran.ses.04
12) run13.ses
13) run9.ses
14) run_stats.ses
15) stats.ses
16) Show all files
17) Other
Enter Selection (X to exit menu) >5
Info : Executing session command (Line 1) : MvSpreadsheetCommand( "open
""/xlv0/people/keh/flight_test/mvis/gerry.spd""" )
Info : Executing session command (Line 2) : MvSpreadsheetCommand(
"B1=``BFT13049.st3``" )
Info : Executing session command (Line 3) : MvSpreadsheetCommand( "F5=49" )
Info : Executing session command (Line 4) : MvExecuteSessionFile(
"/xlv0/people/keh/flight_test/mvis/load.ses" )
Info : Executing session file : /xlv0/people/keh/flight_test/mvis/load.ses
Info : Executing session command (Line 2) : MvSpreadsheetCommand( "write
""block.dat"" ""w"" W31" )
Info : Executing session command (Line 3) : MvSpreadsheetCommand( "write
""block.dat"" ""a"" W35" )
Info : Executing session command (Line 4) : !! INPUT get_data.pcl
Compiling: make_trans_command
Cleared memory function make_trans_command
  Compiled: make_trans_command
Info : Executing session command (Line 5) : make_trans_command()
Info : Executing session command (Line 6) :
MvExecuteSessionFile("do_transpose.ses")
Info : Executing session command (Line 4) : MvSpreadsheetCommand(
"C1=``=put(""strnhi.des"",C4:O4,C5:O252)``")
Info : Putting data from spreadsheet into databank
Info : Summary - Results of Spreadsheet 'PUT'
Info : Added 72 RUN data row(s)
```

```
Info : Added 248 SOURCE data row(s)
Info : Added 78 STRNvsTIME data row(s)
Info : Done putting data from spreadsheet
Info : Done executing session file
Info : Done executing session file
Info : Executing session command (Line 5) : MvSaveDatabase()
```

Once all raw data has been loaded and 'put' into the databank, the implementation of the interface with MSC/PATRAN is ready to be used. A p3epilog file can be used to automatically load the extra plb at startup, adding an extra menu option to the MSC/PATRAN menu bar (see figure 9).

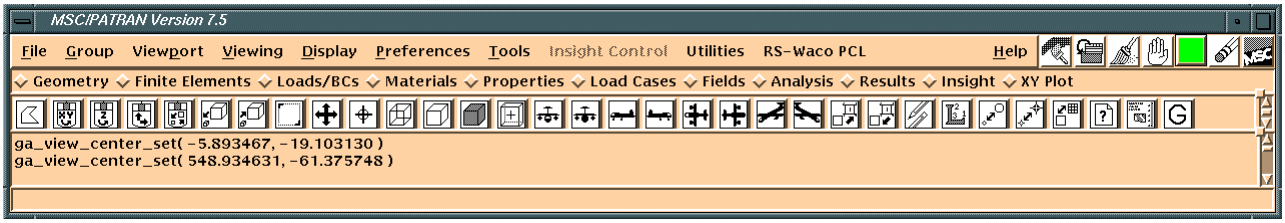

Figure 9 -- Modified MSC/PATRAN Menu Bar

Figure 10 shows the Instrument Marker form, with no data selected.

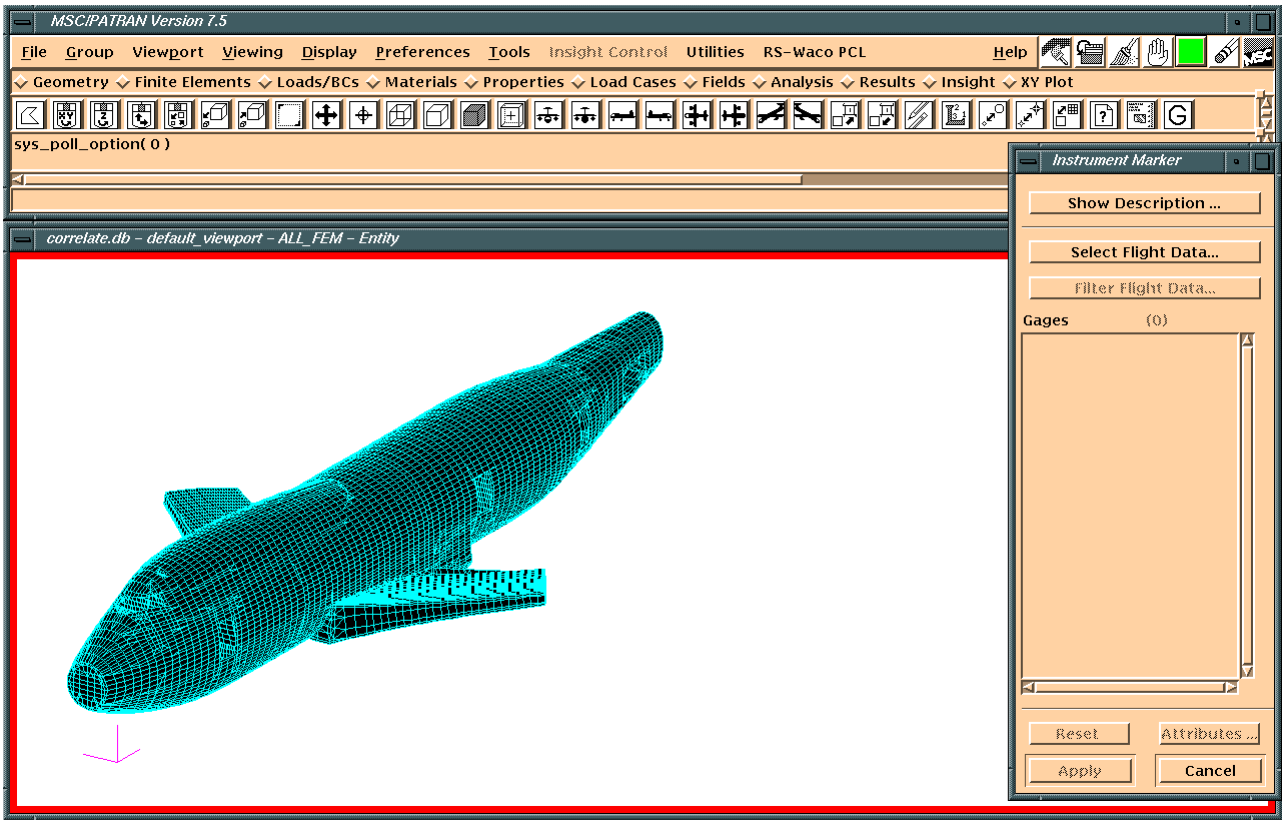

Figure 10 -- Instrument Marker Form Before Data is Selected

Figure 11 adds the MV Selection Tool, showing the databanks available for selection by the end-user.

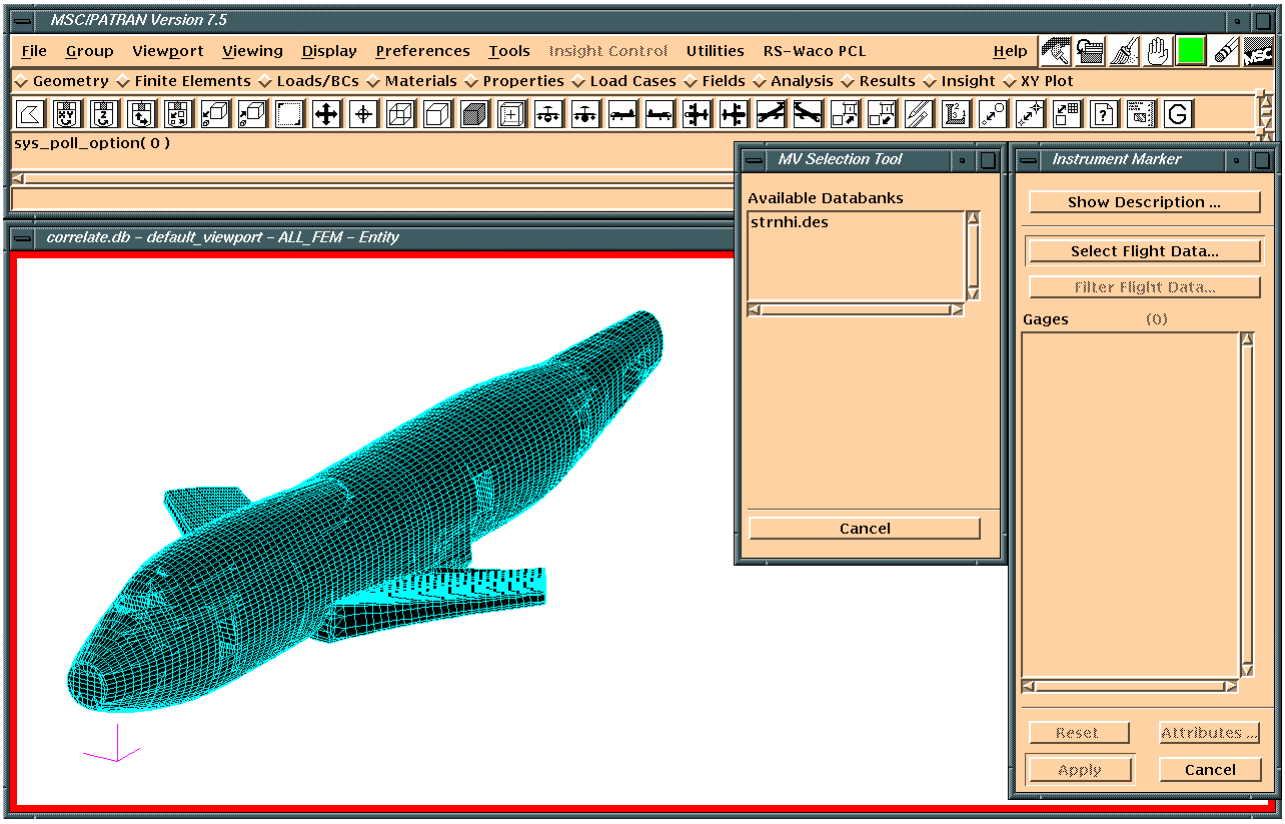

Figure 11 -- MV Selection Tool for Selection of a Databank

In figure 12, a request to the MSC/MVISION databank has been made, and a result returned. In this case the "select" was for the attribute VERSION, with no query attached.

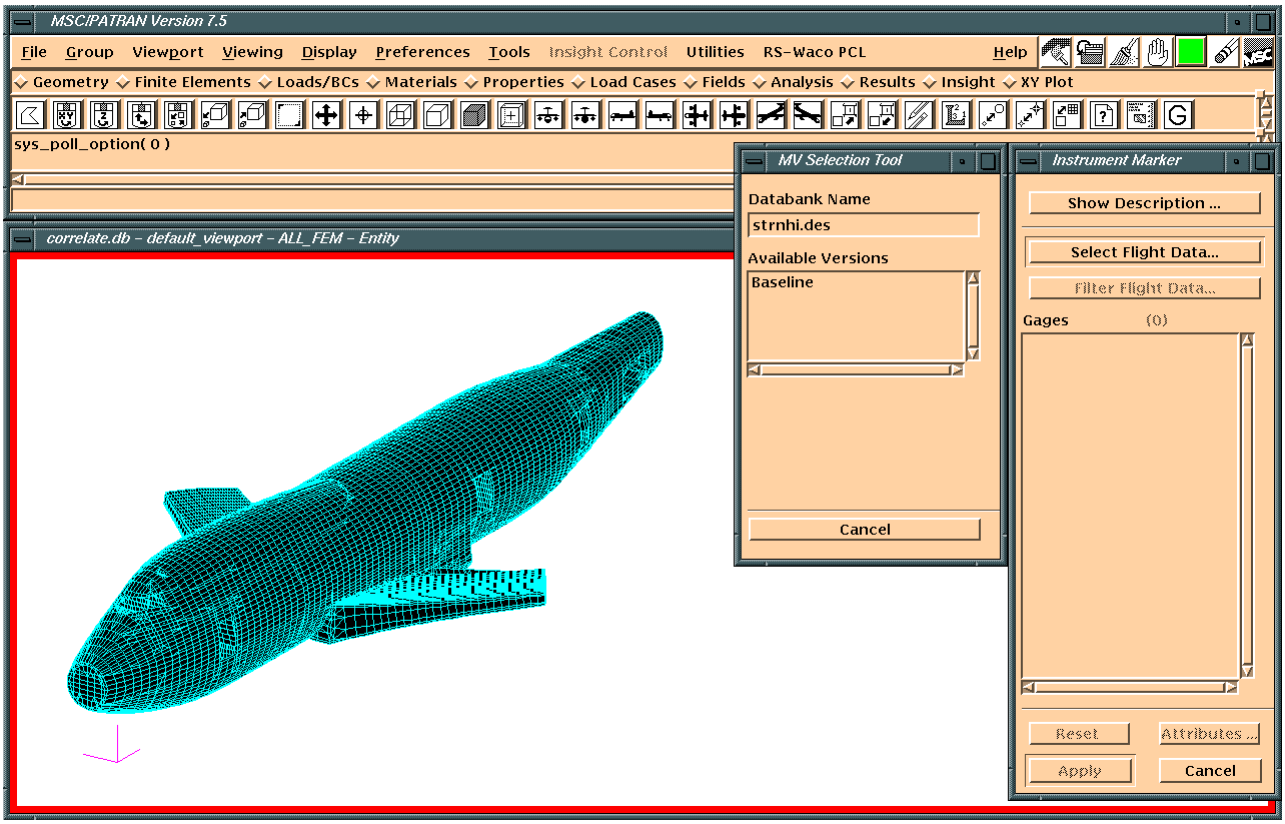

Figure 12 -- First Request to the Databank

In Figure 13, another request has been sent. The "select" was for attribute FLIGHT_NUMBER, with the query VERSION_NAME like Baseline.

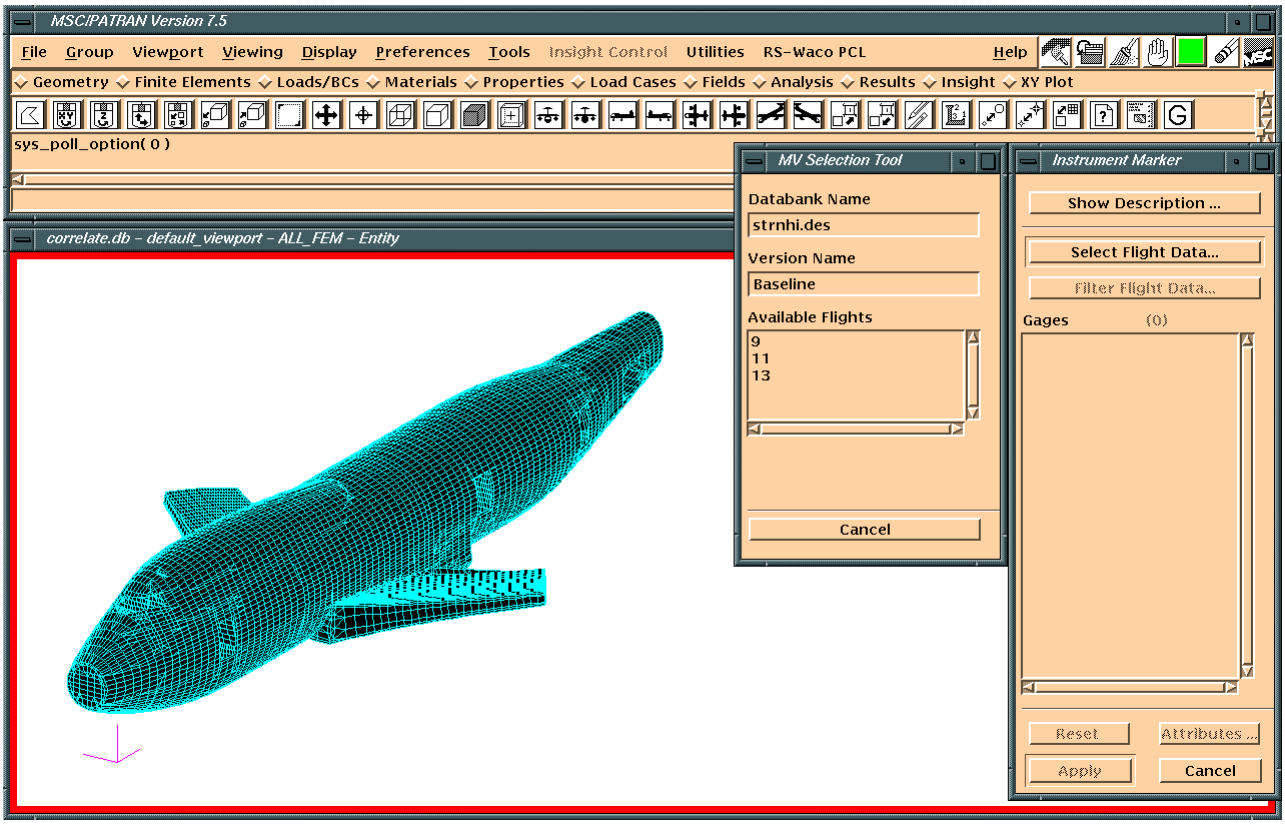

Figure 13 -- Moving Down the Hierarchy

In figure 14, the "select" expands further. Now the attribute is RUN_NUMBER, and the query includes VERSION_NAME like Baseline and FLIGHT_NUMBER = 11.

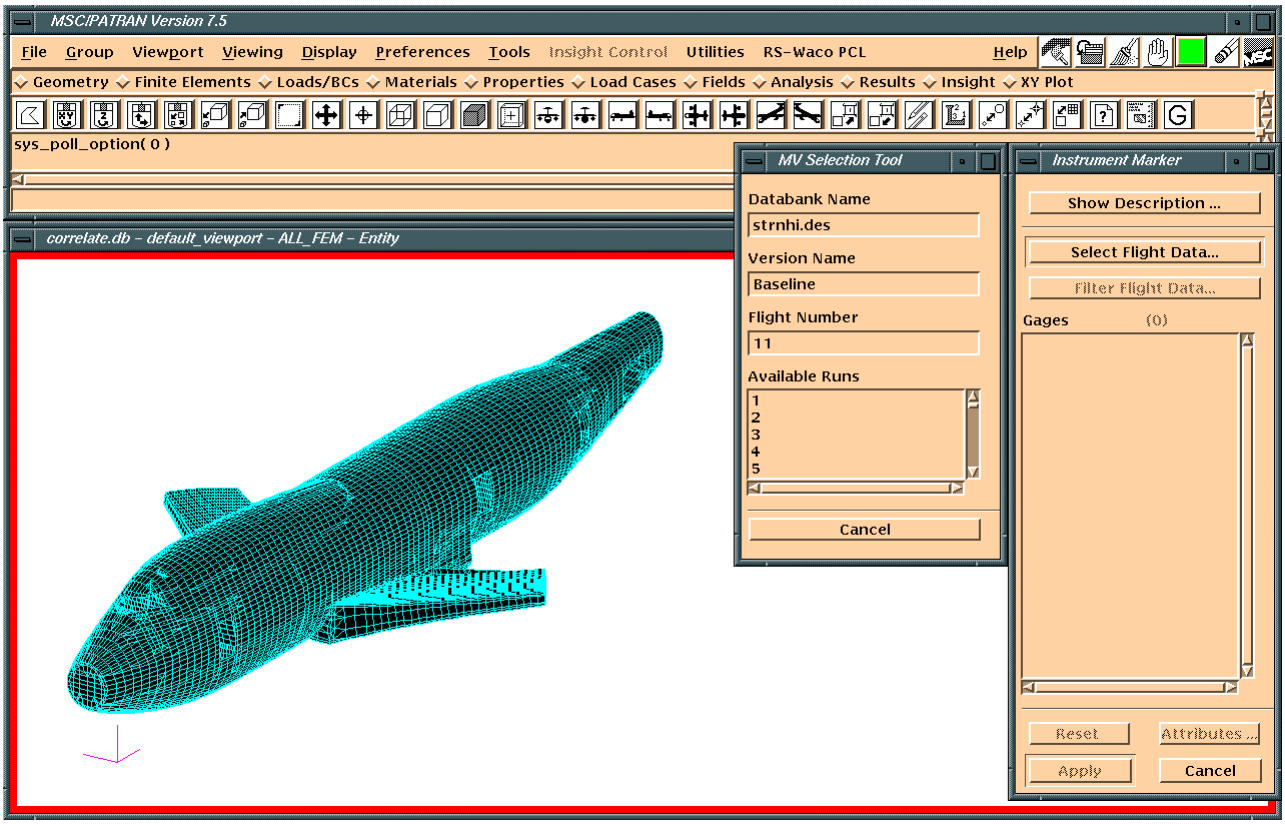

Figure 14 -- Viewing Run Numbers

When a run number is selected, all the strain gauges are listed in the Instrument Marker form. The results returned are all the strain gauge names, types, XYZ locations, descriptions, and average values (see figure 15). All data returned is automatically selected (highlighted) but any one gauge or any subset can be selected if desired.

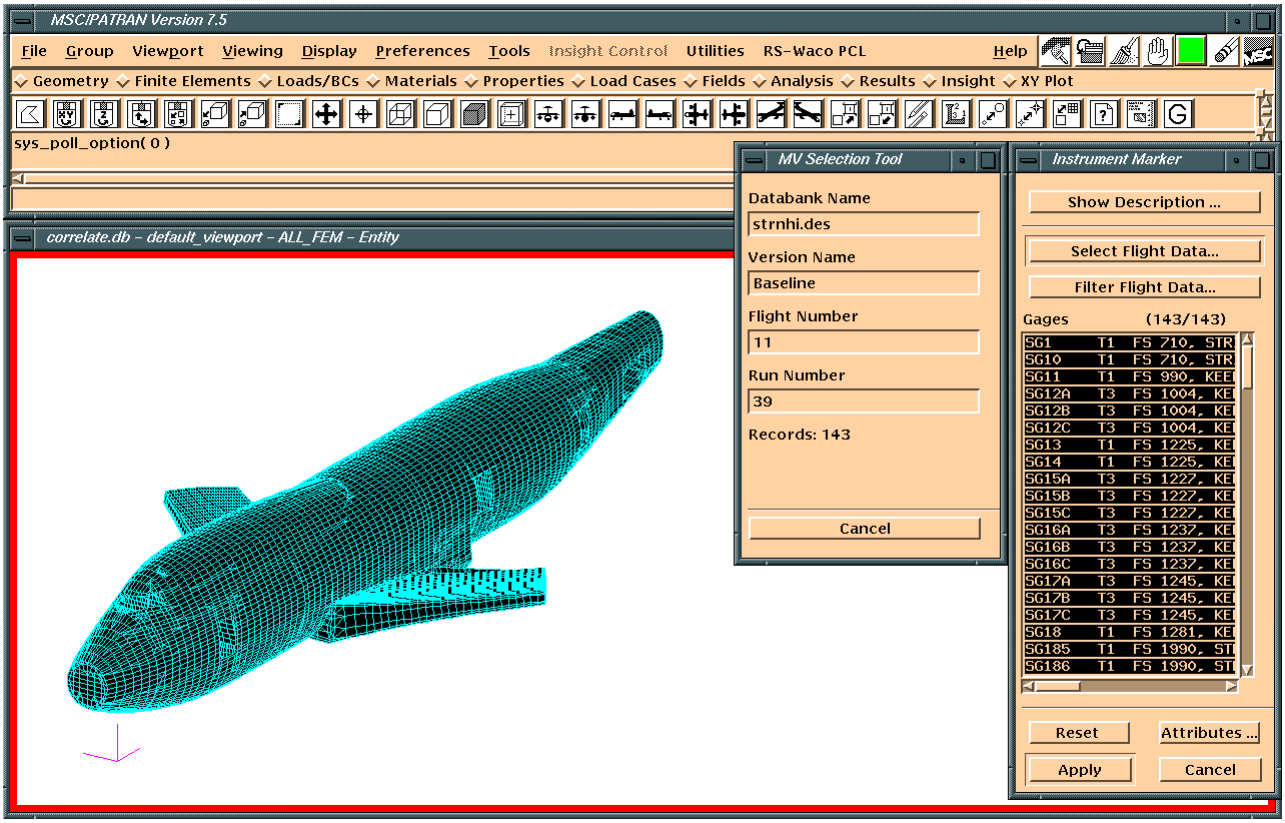

Figure 15 -- Instrument Marker Form Showing Strain Gauge List

Figure 16 shows the four forms reachable through the Instrument Marker form. Gauges can be tagged with their label (SG #), their orientation (T-type), their average value (Reading) or any combination of the three. The gauge list can be filtered by any string or number in their listing, or by X, Y, or Z location.

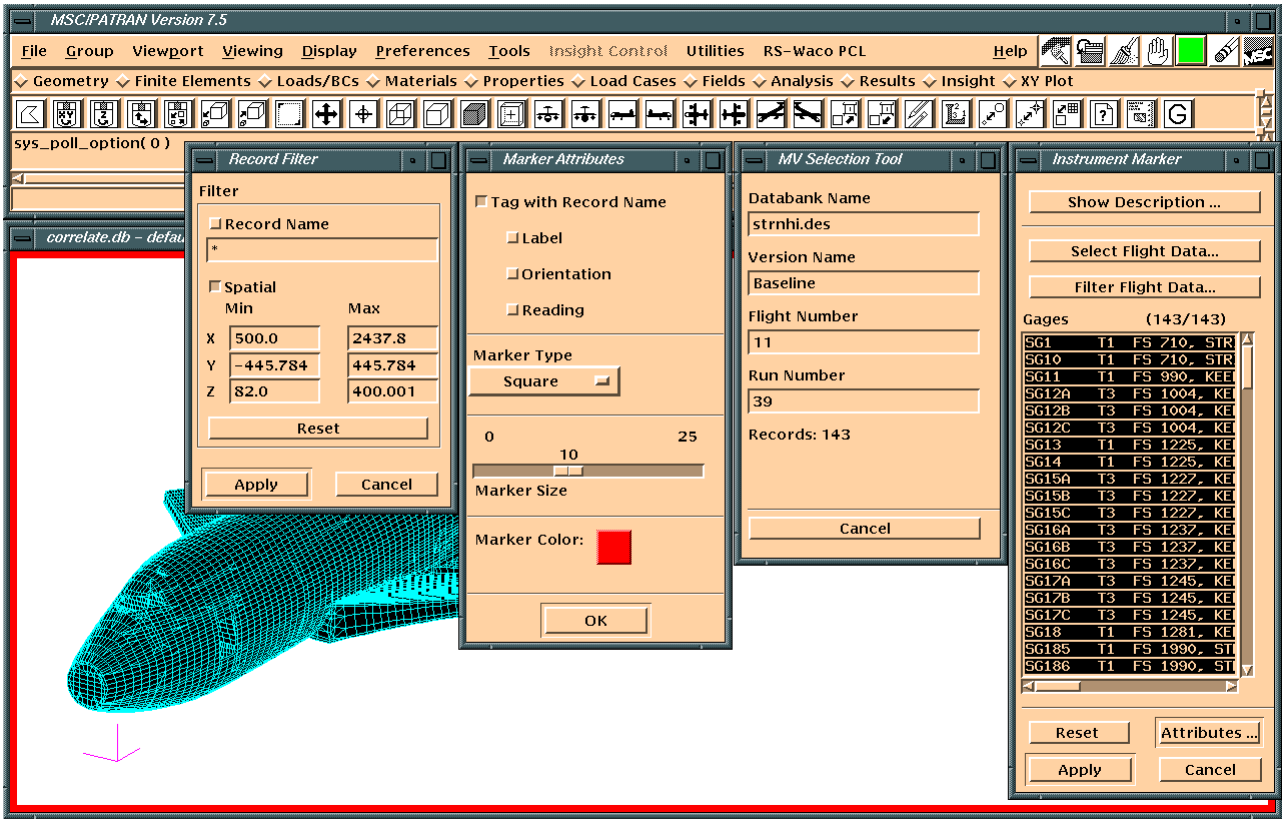

Figure 16 -- All Instrument Marker Sub-Forms

Figure 17 illustrates a filtered set of strain gauges, with different attributes, applied to the model.

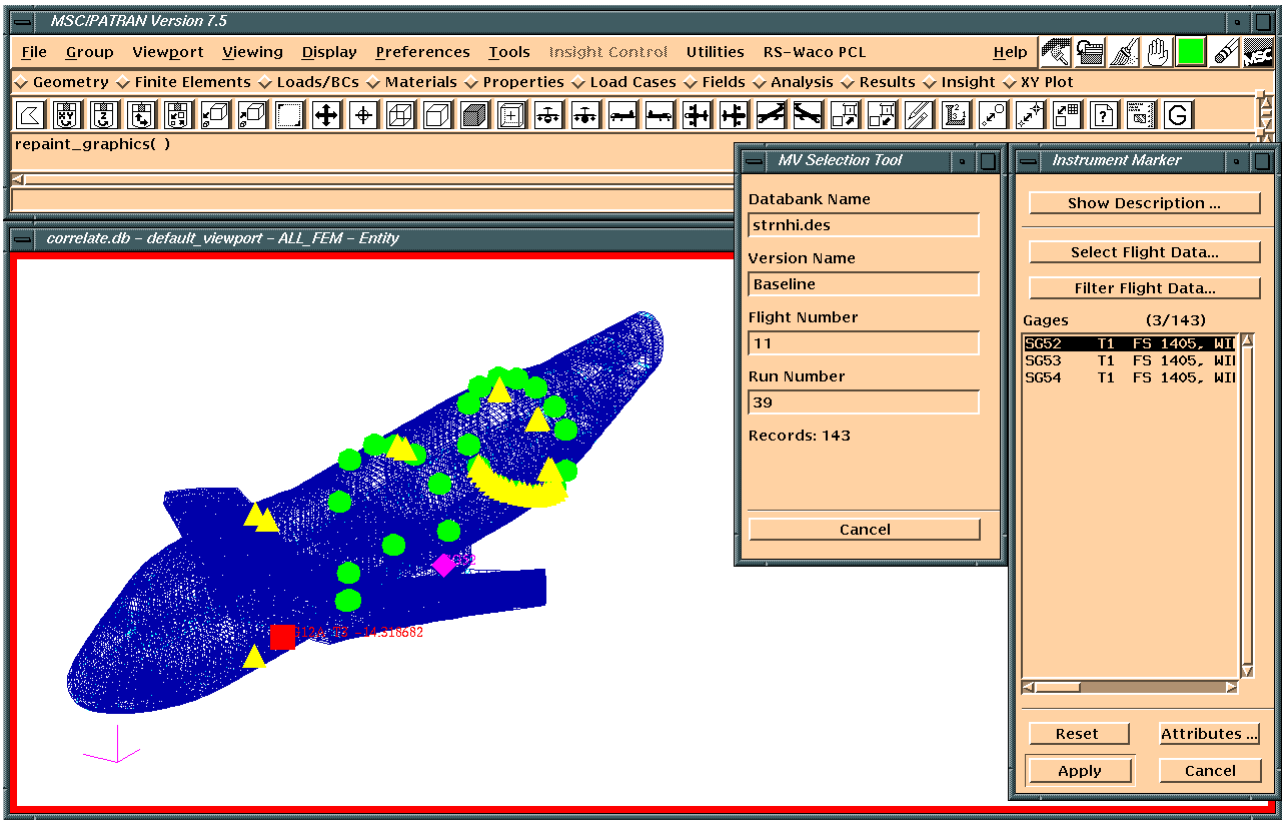

Figure 17 -- Application of the Markers to the Model

The correlation utilities are still in the process of development, but figure 18 shows what their appearance will be. When complete, the end-user will be able to correlate model results data either in a tabular format or graphically.

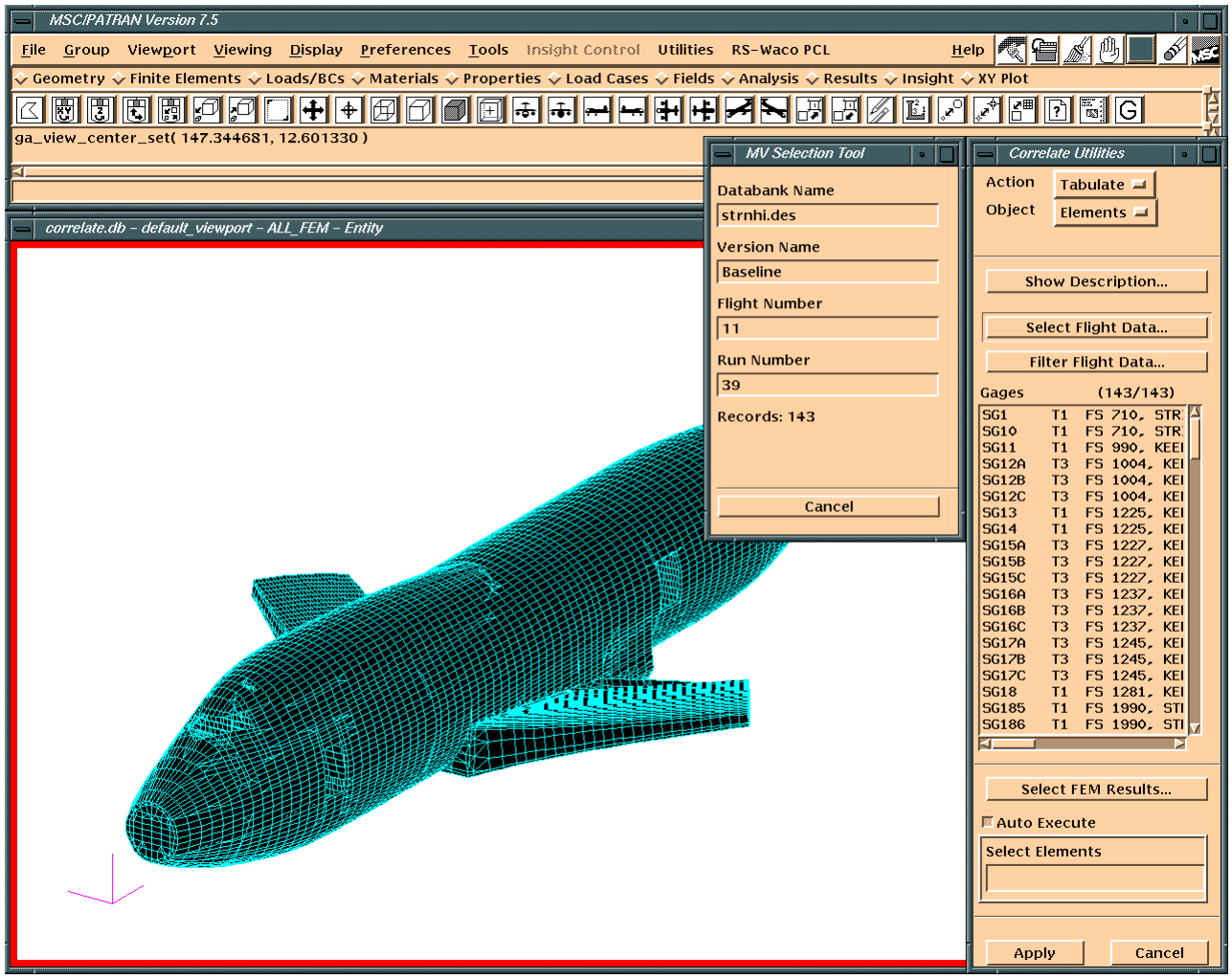

Figure 18 -- Correlate Utilities

Summary

Having planned extensively and clarified the desired end result, development of the MSC/MVISION interface between MSC/PATRAN and raw flight test data was well organized. Now in place is a system by which the engineer can compare MSC/NASTRAN results within the MSC/PATRAN graphical user interface, giving him a clear picture of the position of strain gauges on the aircraft and their measurements for various flight conditions and maneuvers. Graphical and tabular correlation is also in progress, but already the end-user can make sufficient comparison to determine whether the aircraft to be modified is accurately modeled. A comparison of the modified model with data collected after the modification is made will enhance the process of FAA certification for the modified aircraft. This methodology can be used for any model for which test data has been collected, giving engineers detailed indication of the quality of their models.

Acknowledgements

Ajit Kundu and Michael Farley of the Methods and Finite Element Analysis group at Raytheon Systems, Waco, have been very helpful in the development of this project. John Parady, Application Engineer at The MacNeal-Schwendler Corporation in Grapevine, was essential to the pcl programming required for the forms. Without Gerry Norvell (Senior Application Engineer of the same company), and his extensive in-depth knowledge of MSC/MVISION combined with generosity of time and instruction, this project could never have been completed.