Koo, D.; Peak, R.S.; Fulton, R.E. (1999) An Object-Oriented Parser-based Finite Element Analysis Tool Interface,
Invited Paper for Intelligent Systems in Design and Manufacturing II,
SPIE The International Society for Optical Engineering.
Photonics East^R 19-22 Sept. 1999 Hynes Convention Center, Boston, Mass

An Object-Oriented Parser-based Finite Element Analysis Tool Interface

Donald Koo, Russell S. Peak, Robert E. Fulton

Georgia Institute of Technology
CALS Technology Center
Engineering Information System Lab
http://eislab.gatech.edu/

ABSTRACT

To better integrate engineering design and analysis, the multi-representation architecture (MRA) and related methodology have been developed to represent the information transformations between CAD and CAE models [1, 2]. The MRA consists of four representations for increased modularity and flexibility. As one of the representations, solution method models (SMMs) are object-oriented wrappings of tool-specific inputs and outputs that enable highly automated operation of general purpose analysis tools. Outer contexts in the MRA create SMMs from product data and map SMM results back into product-specific terms.

This paper discusses the post-processing aspects of SMMs for finite element analysis (FEA) tools. In this scenario, an SMM dispatches a job to an FEA tool, runs the tool, retrieves results, and stores them in the SMM internal database, which is based on a STEP EXPRESS-G schema. Higher-level context objects in the MRA can then query SMM objects to extract summary results such as stress extrema. These concepts were implemented in an analysis integration toolkit, *DaiTools*™, for two FEA tools (ANSYS and CADAS) with the object-oriented T-gen translator generator tool. This automation process is illustrated with circuit board solder joint analysis and warpage analysis examples. Results show this approach provides effective manipulation and query capabilities that enable enhanced analysis tool integration.

Keywords: CAE-CAD Integration, Object-Oriented, FEA, ANSYS, STEP, EXPRESS-G, MRA, SMM

1. INTRODUCTION

Technological advances in computing technology have made a variety of tools available for solving complicated problems. One significant capability is the finite element analysis (FEA) tool. Currently, there are number of FEA packages available. They are either bundled with solid modeling tools or distributed as stand-alone packages. When an FEA tool is bundled with solid modeling tools, it is generally used to validate designs or to optimize designs. Stand-alone packages generally have more capabilities and thus are used for more thorough analyses. One problem with FEA software is that it can be quite complicated to use. Engineers who have some knowledge of finite element theory are needed to use them and interpret the results. Also it can be time consuming to develop a valid analyzable model. And lastly, these software packages are generally too expensive for small and medium-sized enterprises (SMEs).

2. DAITOOLS™ AND THE MULTI-REPRESENTATION ARCHITECTURE

DaiTools™ is an Internet-based software tool developed by the Engineering Information Systems Lab (EIS Lab) at the Georgia Institute of Technology. One application has been providing analysis services such as product-specific FEA to SMEs through the Internet [3, 4]. Data exchange on the Internet is performed by the means of standard formats like STEP. After a STEP file is uploaded to the server, *DaiTools*™ creates a FEA model based on the STEP product data and runs the FEA analysis program which may be located on the same server or on another machine accessible to the server. Since typically only extrema results are needed, there is an option for returning them over the Internet instead of transferring large result files. This paper focuses on how the output produced by a specific FEA program is handled in an object-oriented manner by the *DaiTools*™ software[1].

DaiTools™ employs a design-analysis integration strategy called the Multi-Representation Architecture (MRA) [1] (Figure 1) to associate design models with analysis models. In the MRA, product-specific analysis idealizations are captured in the form of reusable product model-based analysis models (PBAMs). Analysis building block (ABBs) are product-independent analysis objects with high content of engineering semantics. Solution method models (SMMs) are spawned from ABBs. SMMs are method specific solution objects. SMMs wrap analysis tools such as FEA programs to allow automation of 'routine analysis processes' [6]. *DaiTools*™, as one implementation of MRA concepts, integrates CAD information (e.g. geometry) along with other information such as a material database to construct an analyzable product model (APM). The necessary data can then be extracted to build analysis models such as finite element models.

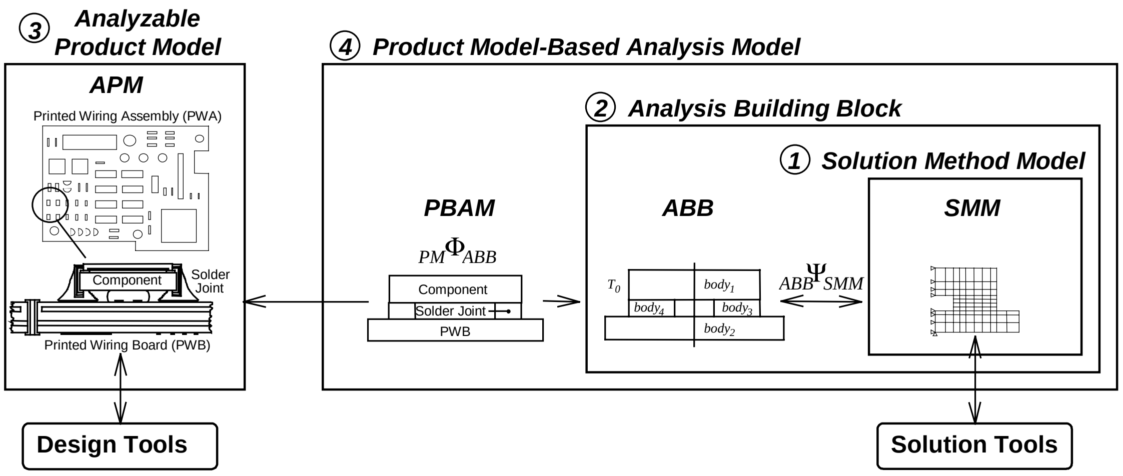


Figure 1. Multi-Representation Architecture (MRA) for Analysis Integration

3. CAD-CAE INTEGRATION

Analysis programs often are designed as user-controlled applications and therefore have little capability to be used automatically by other programs. Most programs offer command line operation with the ability to process optional input arguments. The outputs that are produced, in both graphical-user-interface (GUI) and batch mode, generally consist of a database file and text files. The database files are often program-specific and typically restricted to reuse by the programs that produced them. The text outputs are intended to give the user information concerning the executed job (e.g. error, log). Usually, in order to access the analysis results for post-processing, the user is required to access the database file by running the program and open the data through GUIs. Also, more actions via GUIs are required to carefully analyze the result.

In order to integrate a FEA program with outer context tools like *DaiTools*™, the FEA program can be run in batch mode. But running the FEA program in batch mode limits user interaction in that any unplanned actions are disallowed, because the outer context automatically creates the inputs and uses the outputs. Without a GUI, in batch mode the user is unable to see the solution result directly and therefore an alternate way of processing the FEA data is needed. Since methods for direct interaction with the database file are generally not provided, processing the output file generated by the FEA program in a separate program is often the only integration strategy. Most FEA programs have some capabilities to produce text files of node and element data. However, such files are intended for human use in that they contain descriptive text

[1] This paper primarily describes implementation with *DaiTools*™, a first generation analysis integration toolkit implemented in Smalltalk [3]. *XaiTools*™ is a second-generation Java-based implementation of extended MRA concepts [5].

along with the data. And lastly, because the data is in plain numerical text, there is no way to visualize the result or make associations among scattered data.

The end product of the post-processed FEA data is required to have communication ability with the outer context wrapper tools like the *DaiTools*™. Automated interpretation of the analysis results in product-specific terms is imperative in order to assist designers who do not have knowledge of FEA theory. Such capabilities enable designers and analysts to access FEA results within tools like *DaiTools*™ without having to interpret the data themselves.

While the techniques declared in this paper have been employed with both CADAS [7] and ANSYS [8] FEA tools [1], this paper focuses on more recent work with SMM post-processing for the ANSYS case. The ANSYS FEA program that is used by the *DaiTools*™ software is ANSYS v5.3. ANSYS is a stand alone FEA software and has capabilities to produce output files with some user control as well as command line input operation. *DaiTools*™ software is written in an object-oriented language called Smalltalk in a development environment called VisualWorks v2.5.

ANSYS has the ability to produce a user-defined output file that contains FEA solution data. The typical output file contains system messages such as headers, titles, and notes. Column headings, numbers, and numerical values follow these. Some system messages and column headings cyclically reappear to allow the human reader to keep track of things. Such messages are unnecessary for automated use by outer contexts and only complicate the processes for importing such data. Therefore the system message parts must be cleaned before processing the data file. The resulting file should only have element and node numbers with attributes and solution data along with column headings identifying each number and numerical values. A consistent file format allows for easier transition toward post-processing of the data file.

By parsing the solution data into a database, interpretation of the data for the user can be supported. Scattered data can be combined and sorted into a structured format with relationships established. A series of query commands combined with internal methods can prepare the data to be presented to the user in a predefined way. By making available the outcome of such post-processing for tools like *DaiTools*™, an automation loop, as shown in the Figure 2, can be achieved.

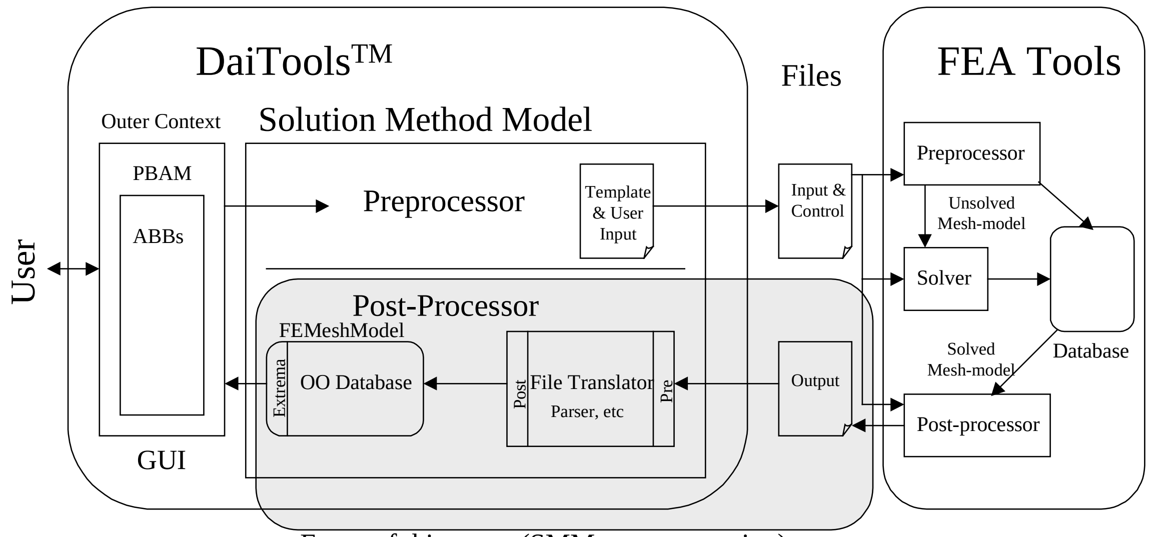


Figure 2. Integration Loop for Highly Automated FEA Tool Usage in an MRA Context

Two ways to model the database for such applications have been investigated here. One is to develop an object-oriented database and the other is to use a commercially available relational database. The object-oriented method was chosen for this application because it is more versatile, and as *DaiTools*™ is an object-oriented program, better compatibility is achieved.

4. INTEGRATION TOOL COMPONENTS

4.1 An Information Model for Basic FEA Mesh Models

The data schema to represent FEA solution data was developed using EXPRESS-G, an object-oriented data modeling language developed for STEP [9]. The resulting data structure for the FEA results was implemented in the VisualWorks Smalltalk environment. The object-oriented data form a communicational object entity that holds attributes and operation methods. Such an object is called a class and the classes for this FEA solution data model are sorted into two categories. The FEA-MeshModels category has five classes and the FEA-Elements category has three application dependent classes. The classes in FEA-MeshModels category are FEMeshModel, FENode, FEGroup, FEMaterial, and FEExtremaSet. The classes in FEA-Elements category are FEElement and its subclasses, PlanarElement and SolidElement[2]. The organization among these classes is given in the Express-G model in Figure 3.

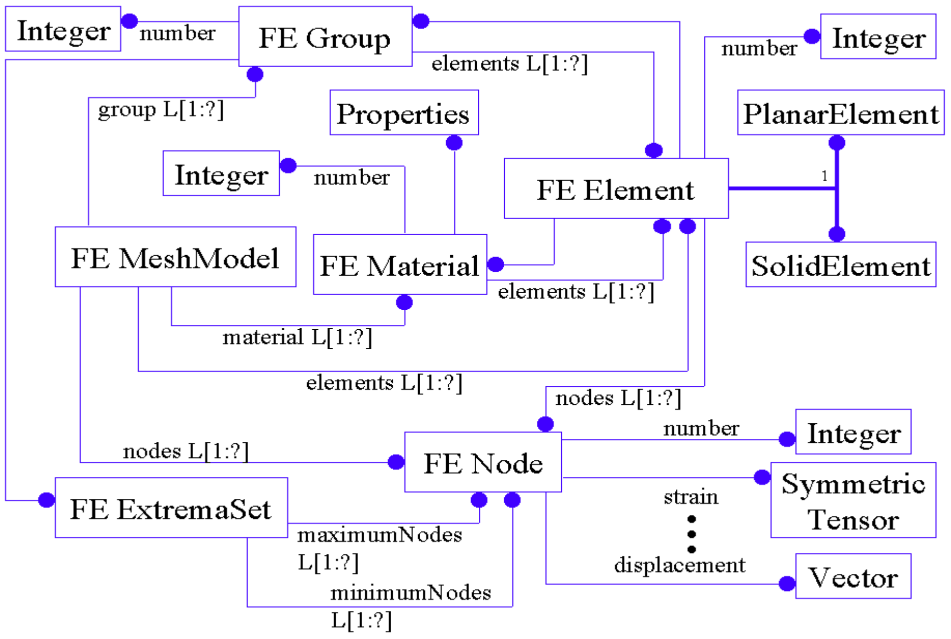


Figure 3. EXPRESS-G schema for Basic FEA MeshModels

All of the output data from the FEA tools are stored within instances of the class FEMeshModel. Nodes, elements, materials, groups, and extrema-sets are attributes of the FEMeshModel. The EXPRESS 'L[1:?]' notation shown in Figure 3 indicates ordered lists, which were implemented in Smalltalk as collections called dictionaries. The keys to the dictionary, such as node number, act as the link to the referring node object. For example, 10 in the dictionary of nodes refers to an instance of a FENode whose node number is 10. As it can be seen from Figure 3, the attributes of FEMeshModel, which are materials, groups, elements, and nodes, are all EXPRESS lists. Also, such relationships were implemented with an inverse capability. When an object such as an element is part of a material object, the material object stores the element in its element attribute list. Concurrently, the element object's material attribute points back to the material to form the inverse of the relation.

The element entities (node, element, and material) have attributes like 'node number' as indicated in Figure 3. The FEGroup is a predefined grouping of elements of interest. It can be sub-assemblies of a part or just a section of a part with heterogeneous materials. The function of FEGroup is to prepare the results to be presented to the user. Customarily searched results such as maximum and minimum values are retrieved

[2] In EXPRESS-G notation, bold lines indicated the superclass-subclass (is-a) relation, while regular lines depict the entity-attribute (has-a) relation.

and stored for use in outer contexts (e.g. a PBAM that needs to know extreme solder joint shear stress). As not all FEA tools identify such extrema, this procedure is performed in FEExtremaSet, which is part of each FEGroup. The FEExtremaSet object contains a dictionary of maximum and minimum nodes for every dimension of every field. The FENode contains nodal data in an array format. Depending on the field, the array format is either vector (x, y, z) or symmetric tensor (x, y, z, xy, yz, xz). The nodal data is used to determine extrema-set data in SMM post-processing.

4.2 Format of ANSYS Output

To eliminate system messages and headers printed in the ANSYS output files, a set of available post-processing commands form the ANSYS POST1 section that is appended to the ANSYS input job file. Typical ANSYS data output format code is listed in Figure 4. These POST1 commands are needed to create an output format that can be automatically processed by SMMs.

```
ESEL, MAT, 3
/HEADER,off,off,off,off,on,off
/OUTPUT,sj-deformation-3D,OUT
*vwrite,'3D','SOLID45'
(2a,3x,8a)
elist,all,,,0,0
PRNSOL,EPEL,COMP
PRNSOL,S,COMP
PRNSOL,DOF
/OUTPUT
```

Figure 4. ANSYS POST1 output commands to aid SMM results processing

The first line selects the material of interest. ANSYS v5.3 can output data for only one material type at a time. If more than one material is selected, ANSYS will insert a warning message at the material transition point. Such insertion of messages requires additional coding for the program that will read in the output file. Another way to handle multi material models is to create separate files for each material. The second option was chosen in this case. The first line selects the material type. The '/HEADER' line eliminates all the header information intended for human readers that SMMs do not need (except the column headings). The '/OUTPUT' line creates an output file according to the specified filename with the .OUT extension. The '*vwrite' line inserts elemental information that is not automatically generated. The next line is a format specification for the inserted elemental information. The 'elist' line creates a list of elements with information such as material number and node numbers. The 'PRNSOL' command line creates a list of nodal solutions for the specified field where EPEL, S, and DOF are strain, stress, and displacement respectively. Another '/OUTPUT' signals the end of the file. These commands produce output like that in the lower right quadrant of Figure 5.

4.3 Importing FEA Results via a T-gen-created File Translator

In order to create finite element objects from the ANSYS output for use in SMMs, a tool called T-gen was used to create a File Translator (Figure 2), which includes a scanner, a parser, and a builder. T-gen is a general-purpose object-oriented tool for the automatic generation of string-to-object translators [10]. It is written in Smalltalk for the VisualWorks environment. Finite element objects are created and populated by processing an ANSYS output file in the translator created by T-gen.

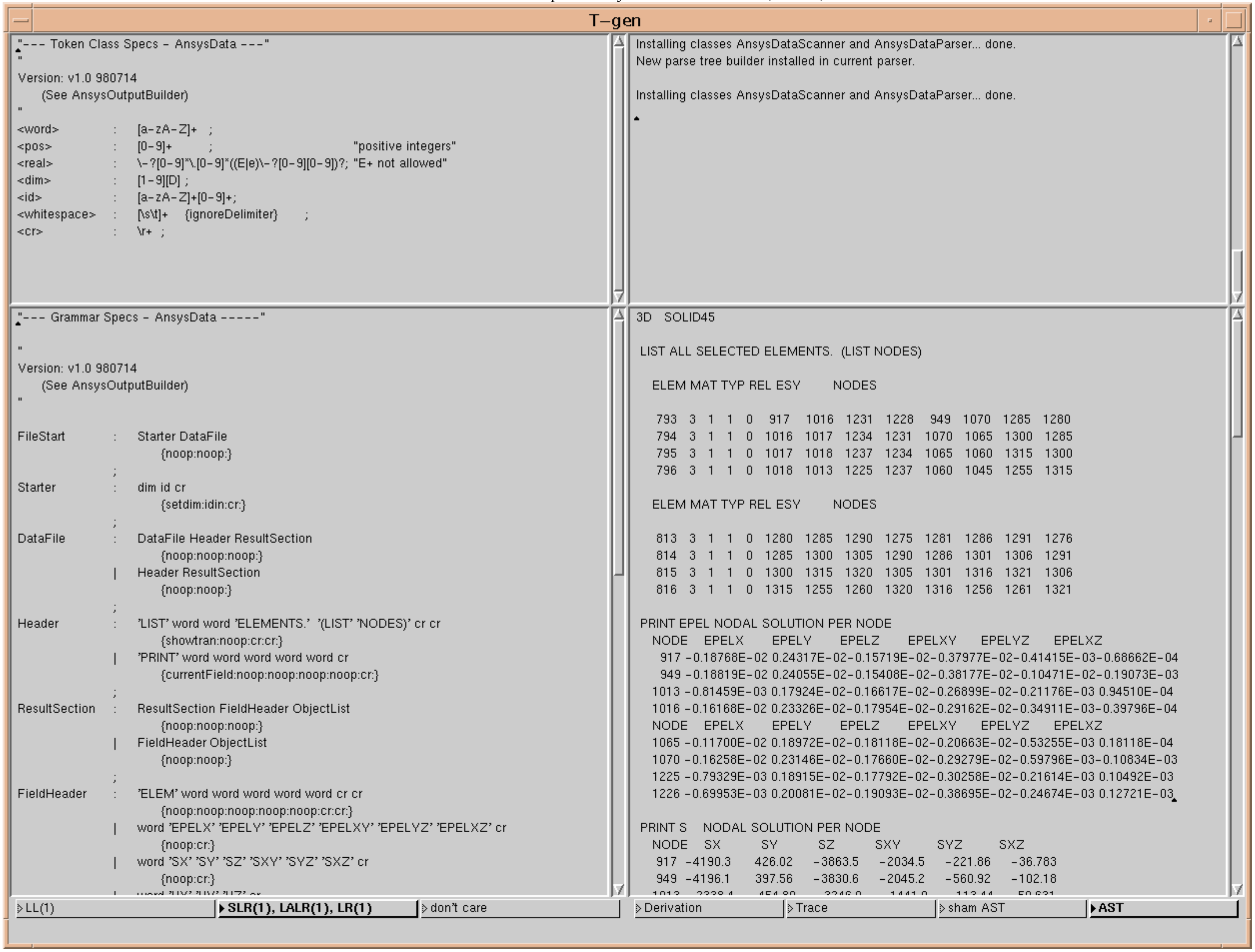
To parse ANSYS output data into Smalltalk objects, a T-gen translator is created. First, tokens are specified in the Token-Class-Specs section (upper left-hand pane) of the T-gen window (Figure 5) according to data types found in the ANSYS output such as numbers and characters. The contexts that match the tokens are recognized by the scanner and then passed to the parser. The sequence of the how the text files are scanned and parsed is determined in the construction of a context-free-grammar (CFG). The grammar tree for the ANSYS output file is constructed according to the output data format specified in the Grammar-Specs section (lower left-hand pane of Figure 5). The grammar also uses non-terminals to direct the flow of the scanning process and directives to perform operations on the scanned strings. For each non-terminal on the left-hand side, the flow rule is given on the right-hand side composed of terminals (tokens and literal strings) and non-terminals. The directives, which are required for every terminal and non-terminal on the right-hand side, have production methods associated with them. The collection of directives and their production methods make up the builder object. The production methods operate on string entities read by the scanner and convert them into finite element objects and populate their attributes. These methods, which were encoded in Smalltalk, place the generated objects as attributes of the FEMeshModel, which has its own methods to post-process the data once it is parsed. After specifying the tokens and CFG, T-gen generates and installs a 'scanner', which reads and passes the recognized strings of the input text, and a 'parser', which checks the validity of tokens and operates on them. To test the translator, sample data can be entered into the input section (lower right-hand pane) of the T-gen window. Each line of the test input is scanned by the corresponding grammar expressions. When the translator hits a new line with different grammatical context, the non-terminals direct the translator to the next grammar expression according to predefined order. For repeated input lines, the grammar simulates looping by calling back its own grammar expression.

A Parse-Tree-Builder class called AnsysDataBuilder was constructed using Smalltalk. It includes the collection of production methods specified for the directives of the T-gen grammar spec, and it utilizes the Ansys-specific scanner and parser mentioned above. It also contains other methods that support the parsing process. The production method names used as directives in the T-gen grammar are sorted under the 'production processing' protocol. These methods perform preliminary operations such as creating a data object to store the ANSYS data. They have other methods to assist the production, most of which are specified under the 'production support'. The resulting finite element objects are post-processed to extract maximums and minimums by methods specified under the 'pre/postprocessing'.

As the ANSYS output data is being scanned, the builder executes the following operations to parse the data. First, it creates a FEMeshModel instance, which will act as the root object. The first line of the sample data has '3D' and 'SOLID45' (Figure 5, lower right pane). These are stored in the builder for determining element type. Then the headers and column heading are ignored, because the builder knows by the grammar specification that the first set of data is a list of elements. As the elements are encountered, an instance of the class FEElement is created for every element. The dimension that was stored, '3D' in this case, is used to determine if the element is a SolidElement or a PlanarElement. The material number is temporarily stored in the FEElement as material number and group number (unless specified) for future post-processing. Then the FENode instances are created according to the scanned node numbers. These nodes are gathered in the dictionary and placed as the 'nodes' attribute of the FEElement. Simultaneously, as the FEElement and FENode are created, the master lists that contain all of the elements and nodes for the whole finite element model are created as dictionaries and placed of as 'elements' and 'nodes' attributes of the FEMeshModel. Since some of the nodes are shared by the elements, the existence of a scanned node is checked against the master list to avoid duplicating FENode instances.

After all the elements and nodes are created, the nodal values for specified fields such as strain, stress, and displacement are parsed. In order to identify which list belongs to which field, the field identifier such as 'EPEL' (strain), 'S' (stress), and 'DOF' (displacement) from list headings is stored in the builder attribute called 'currentField'. The 'currentField' value is replaced whenever another field is encountered. For example, while the 'currentField' has 'S' as its value, the scanned numbers are parsed as the stress values of respective nodes.

Following the completion of parsing, translator post-processing takes place (Figure 2). During translator post-processing, FEMaterial, FEGroup, and FEExtremaSet are created. When post-processing is initiated, the FEElement instances are bundled according to their material number and group number and placed in FEMaterial and FEGroup as dictionaries. The FEMaterial and FEGroup instances replace the material number and group number in FEElement instance to establish object links. The creation of a FEGroup instance triggers the creation of an associated FEExtremaSet instance. To create FEExtremaSet instances, nodal values of every field in every dimension are compared against one another to extract nodes with maximum and minimum values. These extracted maximum and minimum nodes are sorted in dictionaries to complete translation post-processing.



```
"--- Token Class Specs - AnsysData ---"
"
Version: v1.0 980714
     (See AnsysOutputBuilder)
"
<word>         :   [a-zA-Z]+   ;
<pos>          :   [0-9]+        ;                          "positive integers"
<real>         :   \-?[0-9]*\.[0-9]*((E|e)\-?[0-9][0-9])?; "E+ not allowed"
<dim>          :   [1-9][D] ;
<id>           :   [a-zA-Z]+[0-9]+;
<whitespace>   :   [\s\t]+    {ignoreDelimiter}      ;
<cr>           :   \r+  ;
```

```
Installing classes AnsysDataScanner and AnsysDataParser... done.
New parse tree builder installed in current parser.

Installing classes AnsysDataScanner and AnsysDataParser... done.
```

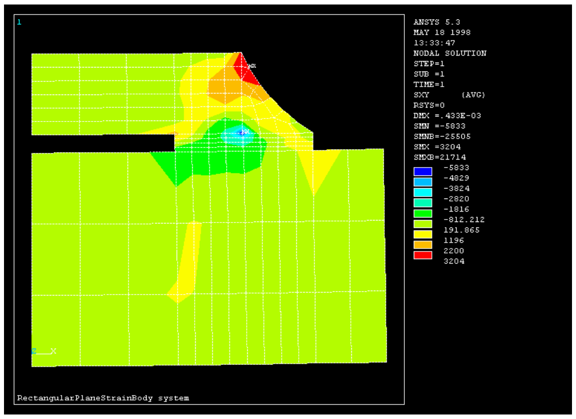
```
"--- Grammar Specs - AnsysData -----"

"
Version: v1.0 980714
     (See AnsysOutputBuilder)
"

FileStart      :   Starter DataFile
                        {noop:noop:}
               ;
Starter        :   dim id cr
                        {setdim:idin:cr:}
               ;
DataFile       :   DataFile Header ResultSection
                        {noop:noop:noop:}
               |   Header ResultSection
                        {noop:noop:}
               ;
Header         :   'LIST' word word 'ELEMENTS.'  '(LIST' 'NODES)' cr cr
                        {showtran:noop:cr:cr:}
               |   'PRINT' word word word word word cr
                        {currentField:noop:noop:noop:noop:cr:}
               ;
ResultSection  :   ResultSection FieldHeader ObjectList
                        {noop:noop:noop:}
               |   FieldHeader ObjectList
                        {noop:noop:}
               ;
FieldHeader    :   'ELEM' word word word word word cr cr
                        {noop:noop:noop:noop:noop:cr:cr:}
               |   word 'EPELX' 'EPELY' 'EPELZ' 'EPELXY' 'EPELYZ' 'EPELXZ' cr
                        {noop:cr:}
               |   word 'SX' 'SY' 'SZ' 'SXY' 'SYZ' 'SXZ' cr
                        {noop:cr:}
```

```
3D   SOLID45

 LIST ALL SELECTED ELEMENTS.  (LIST NODES)

   ELEM MAT TYP REL ESY        NODES

    793   3   1   1   0    917    1016   1231   1228    949   1070   1285   1280
    794   3   1   1   0   1016   1017   1234   1231   1070   1065   1300   1285
    795   3   1   1   0   1017   1018   1237   1234   1065   1060   1315   1300
    796   3   1   1   0   1018   1013   1225   1237   1060   1045   1255   1315

   ELEM MAT TYP REL ESY        NODES

    813   3   1   1   0   1280   1285   1290   1275   1281   1286   1291   1276
    814   3   1   1   0   1285   1300   1305   1290   1286   1301   1306   1291
    815   3   1   1   0   1300   1315   1320   1305   1301   1316   1321   1306
    816   3   1   1   0   1315   1255   1260   1320   1316   1256   1261   1321

 PRINT EPEL NODAL SOLUTION PER NODE
   NODE    EPELX        EPELY        EPELZ        EPELXY       EPELYZ       EPELXZ
    917 -0.18768E-02 0.24317E-02-0.15719E-02-0.37977E-02-0.41415E-03-0.68662E-04
    949 -0.18819E-02 0.24055E-02-0.15408E-02-0.38177E-02-0.10471E-02-0.19073E-03
   1013 -0.81459E-03 0.17924E-02-0.16617E-02-0.26899E-02-0.21176E-03 0.94510E-04
   1016 -0.16168E-02 0.23326E-02-0.17954E-02-0.29162E-02-0.34911E-03-0.39796E-04
   NODE    EPELX        EPELY        EPELZ        EPELXY       EPELYZ       EPELXZ
   1065 -0.11700E-02 0.18972E-02-0.18118E-02-0.20663E-02-0.53255E-03 0.18118E-04
   1070 -0.16258E-02 0.23146E-02-0.17660E-02-0.29279E-02-0.59796E-03-0.10834E-03
   1225 -0.79329E-03 0.18915E-02-0.17792E-02-0.30258E-02-0.21614E-03 0.10492E-03
   1226 -0.69953E-03 0.20081E-02-0.19093E-02-0.38695E-02-0.24674E-03 0.12721E-03

 PRINT S    NODAL SOLUTION PER NODE
   NODE    SX         SY         SZ         SXY         SYZ         SXZ
    917  -4190.3     426.02    -3863.5    -2034.5    -221.86    -36.783
    949  -4196.1     397.56    -3830.6    -2045.2    -560.92    -102.18
```

Figure 5. Creating an ANSYS SMM File Translator in T-gen *(clockwise from top left: Token Specs, Transcript, Test Input Data, Grammar Specs)*

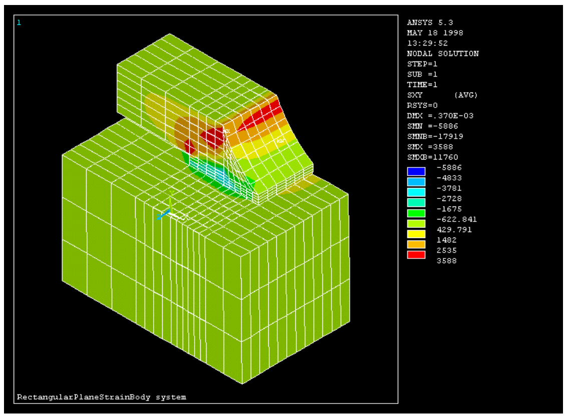
5. TEST CASES

Three examples have been tested for the object-oriented method. They are 2D-solder-joint-deformation-model, 3D-solder-joint-deformation-model, and 2D-pwb-warpage-model PBAMs with ABBs that generate ANSYS SMMs. Within these PBAM/ABB contexts, the resulting ANSYS SMMs create input files that are sent to the ANSYS FEA tool in batch mode (Figure 2). Upon completion of the FEA processes, the FEA tool produces the output files. The context SMM instance then calls the T-gen-created File Translator to parse the output values into the *DaiTools*™ environment. After FEA post-processed results such as the ones shown graphically in Figure 6, Figure 8, and Figure 10 are imported, SMMs extract critical values such as maximum and minimum stress and strain values. The context ABBs/PBAMs then transform these FEA-specific terms into product specific-terms and present them via GUIs such as Figure 7, Figure 9, and Figure 11. The GUIs shown in Figure 7 and Figure 9 are from the same catalogue of solder joint deformation analysis modules (PBAMs). Figure 11 is a catalogue of PWB warpage analysis modules. Other aspects of outer context post-processing included functions such as calculating overall PWB deformation, PWB warpage ratio, and percent difference versus allowables. These results are also displayed and provided as criteria by which the user can evaluate the product design.



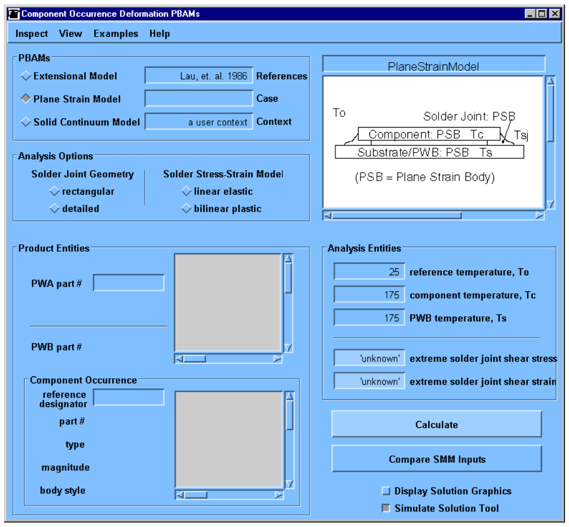
```
ANSYS 5.3
MAY 18 1998
13:33:47
NODAL SOLUTION
STEP=1
SUB =1
TIME=1
SXY       (AVG)
RSYS=0
DMX =.433E-03
SMN =-5833
SMNB=-25505
SMX =3204
SMXB=21714
-5833
-4829
-3824
-2820
-1816
-812.212
191.865
1196
2200
3204
RectangularPlaneStrainBody system
```

Figure 6. 2D-solder-joint-deformation model



```
ANSYS 5.3
MAY 18 1998
13:29:52
NODAL SOLUTION
STEP=1
SUB =1
TIME=1
SXY       (AVG)
RSYS=0
DMX =.370E-03
SMN =-5886
SMNB=-17919
SMX =3588
SMXB=11760
-5886
-4833
-3781
-2728
-1675
-622.841
429.791
1482
2535
3588
RectangularPlaneStrainBody system
```

Figure 8. 3D-solder-joint-deformation model



PBAMs
- Extensional Model — Lau, et. al. 1986 — References
- [✓ Plane Strain Model] — Case
- Solid Continuum Model — a user context — Context

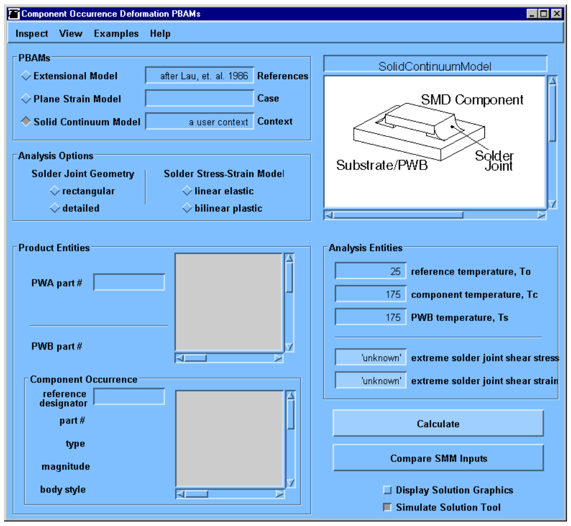
Analysis Options
- Solder Joint Geometry: rectangular / detailed
- Solder Stress-Strain Model: linear elastic / bilinear plastic

Product Entities
- PWA part #
- PWB part #
- Component Occurrence: reference designator, part #, type, magnitude, body style

Analysis Entities
- 25 reference temperature, To
- 175 component temperature, Tc
- 175 PWB temperature, Ts
- 'unknown' extreme solder joint shear stress
- 'unknown' extreme solder joint shear strain

- Display Solution Graphics
- [✓ Simulate Solution Tool]

Figure 7. *DaiTools*™ GUI for 2D-solder-joint model



PBAMs
- Extensional Model — after Lau, et. al. 1986 — References
- Plane Strain Model — Case
- [✓ Solid Continuum Model] — a user context — Context

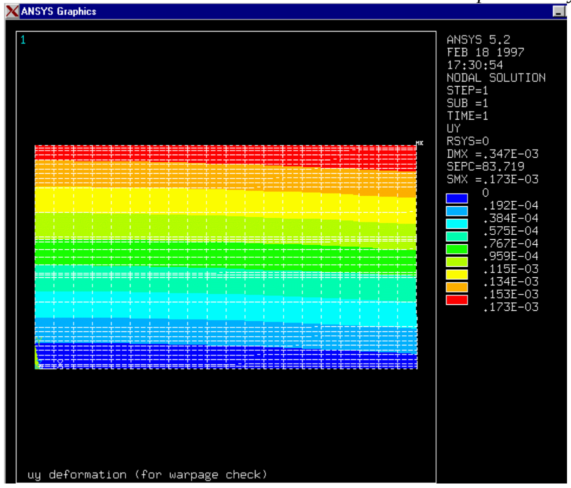
Analysis Options
- Solder Joint Geometry: rectangular / detailed
- Solder Stress-Strain Model: linear elastic / bilinear plastic

Product Entities
- PWA part #
- PWB part #
- Component Occurrence: reference designator, part #, type, magnitude, body style

Analysis Entities
- 25 reference temperature, To
- 175 component temperature, Tc
- 175 PWB temperature, Ts
- 'unknown' extreme solder joint shear stress
- 'unknown' extreme solder joint shear strain

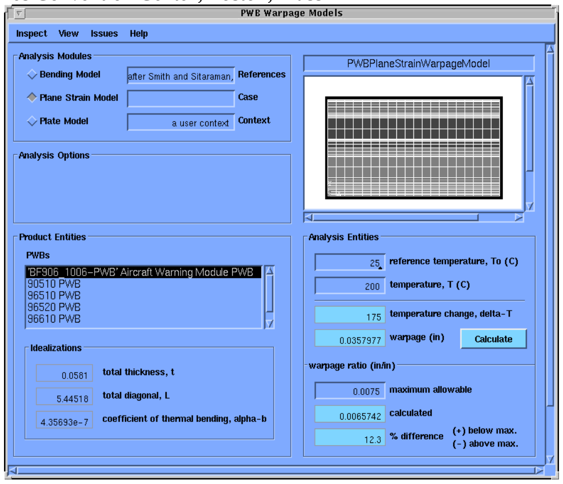
- Display Solution Graphics
- [✓ Simulate Solution Tool]

Figure 9. *DaiTools*™ GUI for 3D-solder-joint-model



```
ANSYS 5.2
FEB 18 1997
17:30:54
NODAL SOLUTION
STEP=1
SUB =1
TIME=1
UY
RSYS=0
DMX =.347E-03
SEPC=83.719
SMX =.173E-03
      0
🟦     .192E-04
🟦     .384E-04
🟦     .575E-04
🟩     .767E-04
🟩     .959E-04
🟩     .115E-03
🟨     .134E-03
🟧     .153E-03
🟥     .173E-03
```

Figure 10. 2D-pwb-warpage-model



Analysis Modules

◇ Bending Model	after Smith and Sitaraman,	References
◆ Plane Strain Model		Case
◇ Plate Model	a user context	Context

Product Entities

PWBs

'BF906_1006–PWB' Aircraft Warning Module PWB
90510 PWB
96510 PWB
96520 PWB
96610 PWB

Idealizations

0.0581	total thickness, t
5.44518	total diagonal, L
4.35693e-7	coefficient of thermal bending, alpha-b

Analysis Entities

25	reference temperature, To (C)
200	temperature, T (C)
175	temperature change, delta-T
0.0357977	warpage (in)

warpage ratio (in/in)

0.0075	maximum allowable
0.0065742	calculated
12.3	% difference (+) below max. (−) above max.

Figure 11. *DatiTools*™ GUI for PWB warpage model

The successful post-processing operation exhibited in the test cases verified feasibility of highly automated analysis tool integration. As the user operates from semantically rich analysis module GUIs like Figure 11, no direct user interaction is needed to create FEA models and run the FEA tools. No user interaction was required also in the loop from the analysis tool back to these product-specific GUIs. Finally, note that the same ANSYS SMM class is usable by all three of these examples, as well as by other analysis modules.

6. Discussion

The SMM post-processing loop (Figure 2) essentially starts during the design stage of the analysis integration tool. It affects the design of extensions for the object-oriented database based on what query functions need to be implemented. The extensions such as query object functions are determined by what information the outer context needs to display in the analysis module catalog GUIs (e.g. stress, strain, and deformation extrema as in Figure 7, Figure 9, and Figure 11). Usually, they are data needed for evaluation criterion embedded in the GUI. Another part of SMM post-processing must be considered during design of the FEA job template (used to create the input and control file per Figure 2). The FEA pre-processing model can be designed with groups, etc. to facilitate obtaining results in specific regions of the model. In the FEA post-processing control section, commands must be utilized to create a formatted output file compatible with the File Translator, which will import the file for outer context use. Commands to extract needed data from specific regions may also be specified. Thus the SMM post-processing loop is pulled by analysis results needs.

Two approaches of SMM post-processing interface to solution tool were investigated. One is an **extensive approach** where it imports all of node and element data for specified regions. This approach is suited for FEA tools with limited post-processing capabilities. Although ANSYS provides functions like extracting extrema, this approach was investigated to take less capable applications into consideration. Another approach is a **minimal approach** where the SMM utilizes post-processing functions provided by the FEA application. With ANSYS, only extrema can be extracted and imported. This approach has been implemented in the next generation tool called *XaiTools*™ with ANSYS v5.4 as the solution tool. *XaiTools*™ is a Java-based tool and it can be expanded to implement the extensive approach with tools like JLex [11]. While the minimal approach can speed up processing time, the extensive approach provides greater data handling flexibility. The extensive approach minimizes possible hindrances caused by limited capabilities in FEA applications. Also it leaves room for more outer context post-processing if needed. This shows that it is sometimes best to implement evaluation capabilities in an outer context post-processing tools rather than have it output from analysis tools, like FEA applications, using their capabilities.

The use of an object-oriented approach showed more potential for semantically rich data processing in the extensive approach. By forming the data into objects, more dynamic data processing was enabled. Unlike other types of representations, objects capture both data and methods to execute functions on their data. Objects also have capabilities to communicate with other objects. Therefore, instead of issuing queries every time sets of data are needed, query objects (such as FE ExtremaSet shown in Figure 3) can be created to store references to the needed data.

The two FEA tool interaction methods that were evaluated are **full batch mode** and **hybrid mode**. Full batch mode supports little or no user interaction. In hybrid (or relaxed) mode, the SMM controls tool input and output but allows user interaction in between. The major advantage of hybrid mode is that users are not restricted by predefined control of outer contexts. The user can take advantage of the FEA tool post-processing functions for graphical and detailed processing, and then return control to the SMM for use in outer contexts. The advantages and disadvantages of the two modes are compared in Table 1.

	Full Batch Mode	**Hybrid Mode**
User Interaction	Little or none	Allowed while tool is in operation
• **FEA tool GUI**	Not displayed	Displayed
• **Use FEA tool functions**	Not allowed	Allowed during post-processing
• **Results Graphic**	Typically not shown	Displayed on GUI
• **Graphics saving option**	Postscript (ANSYS)	Tiff, VRML, Postscript (ANSYS)
Iteration for optimization	Fast	Tedious
Speed	Fast	Slow
Tool knowledge	Not required	Required
Security Control	High	Low

Table 1. Comparison of Full Batch Mode and Hybrid Mode

System security is usually a concern when running an Internet based tool and when usage by non-experts is needed. In hybrid mode, users have access to the full functionality of the FEA tool. This gives the user an opportunity to perform functions that are irrelevant or potentially misleading to the *DaiTools*™ outer context (either accidentally or intentionally). In contrast, when FEA tools like ANSYS are initiated by SMMs in batch mode, user interaction with the tool can be completely eliminated. This prevents analysis tool abuse such as running jobs other than the one dispatched by the outer context. In batch mode, only the job sent by *DaiTools*™ is executed.

One drawback to complete integration of FEA tools like ANSYS is that all the functions available via interactive GUIs may not be available programmatically via batch or hybrid usage. This can be especially true for graphics functions. By running jobs in batch mode, no interactive graphics are displayed. Typically, when using FEA, the user expects to see graphic results to see trends in the analysis results. Although ANSYS has capabilities to produce a variety of picture files in interactive mode, only the PostScript format is supported in batch mode. General programming languages do not readily support displaying PostScript images. They require a graphics library to display picture files. Libraries for other formats like GIF and JPEG are more widely available than PostScript library. Thus in order to display the images within the *DaiTools*™, an image converter needs to be integrated. Another option would be to wrap an image-viewing program to display the images.

To deal with various vendor-specific tools, typically vendor-specific SMMs are needed. While the same object-oriented database structure (FEMeshModel in Figure 2) can typically be reused, the File Translator most often must be tool vendor-specific. Because different CAE tools support different output formats, a translator has to be configured for each of those formats. The advantage of using an object-oriented language for the translator is that even if the grammars are changed, much of the builder code can be reused via inheritance. Therefore it is advantageous to develop robust builder code and change only the grammar to accommodate other CAE tools. Such are the problems with integrating multiple vendor-specific tools, which suggests a need for standards to help the outer context to be less vendor-specific. Table 2 shows candidate standards that would ease outer context usage if supported by CAE tool vendors.

Aspect		Potential Technology
Tool input format (preprocessor model)		STEP AP209
Tool control		CORBA, APIs, Java
Tool output format (solved mesh-model)	: data	STEP AP209, XML
	: graphics	VRML, JPEG

Table 2. Suggested standards to help tool integration with outer contexts

Figure 2 illustrates that CAE tool integration is achieved by means of file exchange. This part of the automation loop can benefit from standards such as STEP AP209 [12]. In this study, three files, ANSYS (template, input, and output) are utilized for tool communication. Constructing multiple vendor-specific translators for tool outputs should be minimized by use of such standards.

In another approach, a commercial relational database manager, ORACLE/SQL, was used to implement the EXPRESS-G data structure shown in Figure 3. This approach was not as natural as with an object-oriented language like Smalltalk or Java due to the object nature of EXPRESS-G models.

7. Concluding Remarks

This paper presents an integration loop for the automation scenario shown in Figure 2. It focuses on solution method model (SMM) post-processing aspects and discusses how detailed FEA results can be translated into objects for usage in higher-level design contexts. A distinction of this paper is how such outer-contexts control FEA tool operation in a highly automated manner as opposed to the typical user-controlled approach. An ANSYS SMM is described along with usage examples for solder joint deformation and PWB warpage. Results indicate that advantages of this approach include:

- Highly automated usage of results in outer contexts like product design
- Reusability by different types of analysis models
- Consistent information interface to FEA tools from diverse vendors
- Integration with other higher-level contexts like optimization

The integration of ANSYS into *DaiTools*™ successfully demonstrated a complete process automation loop. Based on this and other examples, complete analysis tool integration using a wrapping approach is possible for other CAE tools. With this approach tedious analysis tasks can be automated, particularly in the context of product-specific analysis modules, thus reducing design time and increasing process consistency.

Acknowledgements

This work was funded in part in the ProAM project, entitled Product Data-Driven Analysis in a Missile Supply Chain [4]. The U. S. Department of Defense Joint Electronic Commerce Program Office (JECPO) sponsored this effort with the Army Aviation and Missile Command (AMCOM) as primary stakeholder. Under subcontract to Concurrent Technologies Corp. through the Atlanta Electronic Commerce Resource Center (AECRC), ProAM focused on improving missile electronics through advanced engineering analysis integration and delivery via media like http://u-engineer.com/. This Georgia Tech-led effort has addressed barriers to small & medium-sized enterprise (SME) analysis of product physical behavior with the involvement of Circuit Express and S^3, two SMEs in the AMCOM supply chain. The authors wish to acknowledge their colleagues and ProAM team members for help and feedback in this effort, with special thanks to Miyako Wilson and Andrew Scholand.

REFERENCES[3]

1. Peak, R.S., R.E. Fulton, I. Nishigaki, N. Okamoto, *Integrating Engineering Design and Analysis Using a Multi-Representation Approach.* Engineering with Computers, 1998. **Volume 14, Number 2.**: p. 93-114.
2. Peak, R.S., A.J. Scholand, D.R. Tamburini, R.E. Fulton, *Towards the Routinization of Engineering Analysis to Support Product Design.* Intl. J. Computer Applications in Technology, 1999. **Vol. 12, No. 1** (Invited Paper for Special Issue: Advanced Product Data Management Supporting Product Life-Cycle Activities) p. 1-15.
3. Peak, R.S., R.E. Fulton, and S.K. Sitaraman, *Thermomechanical CAD/CAE Integration in the TIGER PWA Toolset*, in *Advances in Electronic Packaging-1997*, E. Suhir and et al., Editors. 1997: Kohala Coast, Hawaii. p. 957-962.
4. *ProAM*, 1999, EIS Lab, CALS Technology Center, Georgia Institute of Technology.
5. *XaiTools^{TM}*, 1999, EIS Lab, CALS Technology Center, Georgia Institute of Technology.
6. Peak, R.S., A.J. Scholand, and R.E. Fulton, *On the Routinization of Analysis for Physical Design*, in *Application of CAE/CAD to Electronic Systems*, D. Agonafer and et al., Editors. 1996: Atlanta. p. 73-82.
7. *CADAS Ver. P4 User's Manual*, 1993, Hitachi Ltd.: Hitachi-shi, Japan.
8. *ANSYS User's Guide*, 1990, Swanson Analysis Systems Inc.: Houston PA.
9. ISO 10303-11, *Industrial Automation Systems and Integration - Product Data Representation and Exchange - Part 11: Description Methods: The EXPRESS Language Reference Manual*, 1994, International Organization for Standardization.
10. Graver, J.O., *T-gen User's Guide*, 1992, Computer and Information Science, University of Florida, Gainesville, FL.
11. Elliot, B., *JLex: A Lexical Analyzer Generator for Java*, 1997: www.cs.princeton.edu/~appel/modern/java/JLex/manual.html.
12. ISO 10303-209, *Industrial Automation Systems and Integration - Product Data Representation and Exchange - Part 209, Application Protocol: Composite and metallic structural analysis and related design*, . 1996, International Organization for Standardization: Geneva, Switzerland.

[3] Some references and project information are available on-line at http://eislab.gatech.edu/