Analysis Reactions and Design Results

(Based on LPM500, Version from January 27, 2000)
by C. Eastman

Purpose:

Reactions are an important output from a structural analysis application. They are aggregated and returned as design result or resistances for later use, for example to detail joint connections. In this tutorial I review first how reactions are represented within an analysis model. Then we also show how they may be associated with design_parts in an assembly_design.

Reactions in an Analysis Model:

The reactions generated by an analysis application are associated with the nodes or elements upon which they apply. An Express-G diagram in Figure One shows the entities used to associate reactions with the parts of an analysis model. Two examples can be traced; reactions applying to a node and those applying to an element.

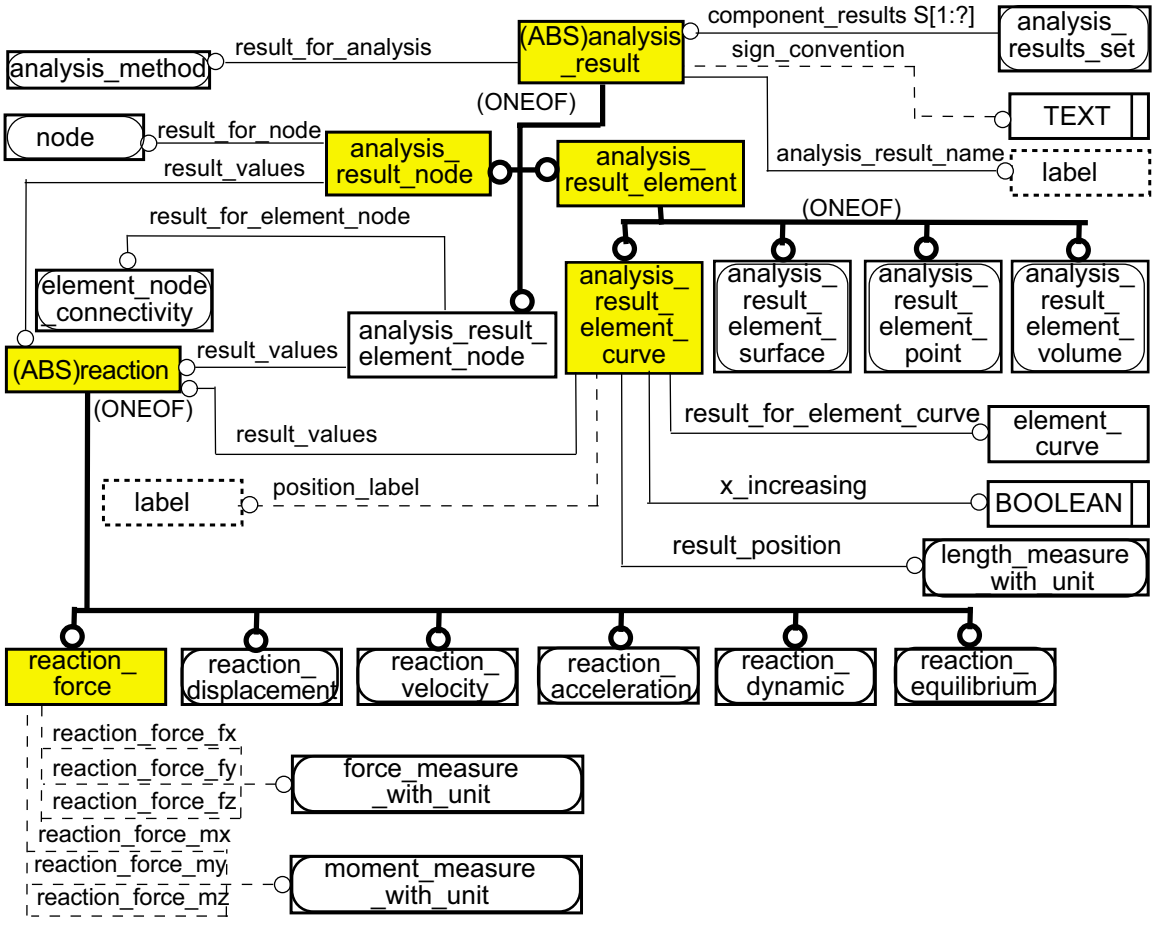

Figure One: Express-G diagram of Entities associating reactions with nodes and elements.

For nodes, the Entities analysis_result_node inherits analysis_result. Flattened with all its inherited attributes, it is defined as:

```
ENTITY analysis_result_node
      (analysis_result:
      analysis_result_name : label;
      sign_convention : OPTIONAL text;
      results_for_analysis : analysis_method;
UNIQUE
      URA1 : analysis_result_name;
      );
 result_for_node : node;
 result_values : reaction;
END_ENTITY;
```

An example Part 021 file might be:

```
#250=ANALYSIS_RESULT_NODE('test one',$,#422,#372,#240);
#422=ANALYSIS_METHOD_STATIC('1st Order Stiffness Matrix
Method',$,.ELASTIC_1ST_ORDER.);
#372=NODE('node_9',#3709,#3699,#3774);
```

We see that analysis_result provides identifiers and analysis method. The analysis_result_ node provides a node and the corresponding reaction. Each node will have a corresponding load_case (not shown in the figure) from which the reactions were derived. If a reaction is at the point where an element meets a node, the Entity analysis_result_ element_node is used. The reaction instance, #240, is described later.

For an element, the corresponding Entity is one of the subtypes of analysis_result_element. Here, I use analysis_result_element_curve. Flattened with all its inheritances, it takes the form:

```
ENTITY analysis_result_element_curve
      (analysis_result_element:
            (analysis_result:
            analysis_result_name : label;
            sign_convention : OPTIONAL text;
            results_for_analysis : analysis_method;
      UNIQUE
            URA1 : analysis_result_name;
            );
      );
 result_for_element_curve : element_curve;
 x_increasing : BOOLEAN;
 result_values : reaction;
 result_position : length_measure_with_unit;
 position_label : OPTIONAL label;
END_ENTITY;
```

The element's behavior, depicted by analysis_result_element_curve, inherits the identifiers and analysis method from analysis_result. Analysis_result_element provides no additional structures. Analysis_result_element_curve provides the reaction, the corresponding element_curve identifying the reaction location, a result_position defining the point along the curve where the reaction applies, a Boolean for direction sense, and an optional label. It is

assumed that the referenced element_curve is a descriptor of an element in the analysis_ model. Other subtypes of elements may be referenced also.

After an analysis is executed, the reactions are assigned to the various elements or nodes whose behavior they depict. These associations can then be transferred and used by later stage processes.

The reaction is an abstract supertype that may be a reaction_force, reaction_displacement, reaction_velocity and so forth. When flattened, the reaction_force, the most common type of force, has the form:

```
ENTITY reaction_force
      (reaction:
      );
      reaction_force_fx : OPTIONAL force_measure_with_unit;
      reaction_force_fy : OPTIONAL force_measure_with_unit;
      reaction_force_fz : OPTIONAL force_measure_with_unit;
      reaction_force_mx : OPTIONAL moment_measure_with_unit;
      reaction_force_my : OPTIONAL moment_measure_with_unit;
      reaction_force_mz : OPTIONAL moment_measure_with_unit;
WHERE
      WRR13 : EXISTS (reaction_force_fx) OR
            EXISTS (reaction_force_fy) OR
            EXISTS (reaction_force_fz) OR
            EXISTS (reaction_force_mx) OR
            EXISTS (reaction_force_my) OR
            EXISTS (reaction_force_mz);
END_ENTITY;
```

Thus the reactions are simply the data defining the type of reaction it is. Similarly flattened, reaction_displacement is defined as:

```
ENTITY reaction_displacement
      (reaction:
       );
      reaction_displacement_dx : OPTIONAL length_measure_with_unit;
      reaction_displacement_dy : OPTIONAL length_measure_with_unit;
      reaction_displacement_dz : OPTIONAL length_measure_with_unit;
      reaction_displacement_rx : OPTIONAL
plane_angle_measure_with_unit;
      reaction_displacement_ry : OPTIONAL
plane_angle_measure_with_unit;
      reaction_displacement_rz : OPTIONAL
plane_angle_measure_with_unit;
WHERE
      WRR9 : EXISTS (reaction_displacement_dx) OR
            EXISTS (reaction_displacement_dy) OR
            EXISTS (reaction_displacement_dz) OR
            EXISTS (reaction_displacement_rx) OR
            EXISTS (reaction_displacement_ry) OR
            EXISTS (reaction_displacement_rz);
END_ENTITY;
```

An example segment of a Part 021 file for a reaction_displacement might be:

```
#240=REACTION_DISPLACEMENT(#184,#184,#184,#219,#219,#219);
```

```
#184=LENGTH_MEASURE_WITH_UNIT(LENGTH_MEASURE(0.),#762);
#219=PLANE_ANGLE_MEASURE_WITH_UNIT(PLANE_ANGLE_MEASURE(0.),#418);
#762=(LENGTH_UNIT()NAMED_UNIT(*)SI_UNIT(.MILLI.,.METRE.));
#418=(NAMED_UNIT(*)PLANE_ANGLE_UNIT()SI_UNIT($,.RADIAN.));
```

Reaction Sets:

Members and connections are designed in response to combinations of loading conditions, where each loading condition generates a set of reactions, one for each structural component. CIS/2 provides a range of means to group reactions into such sets, as basic results, as combined results or as envelope or maximum and minimum results. The multiple results are captured in analysis_ results_set, which points to multiple analysis_ result, as shown in Figure Two.

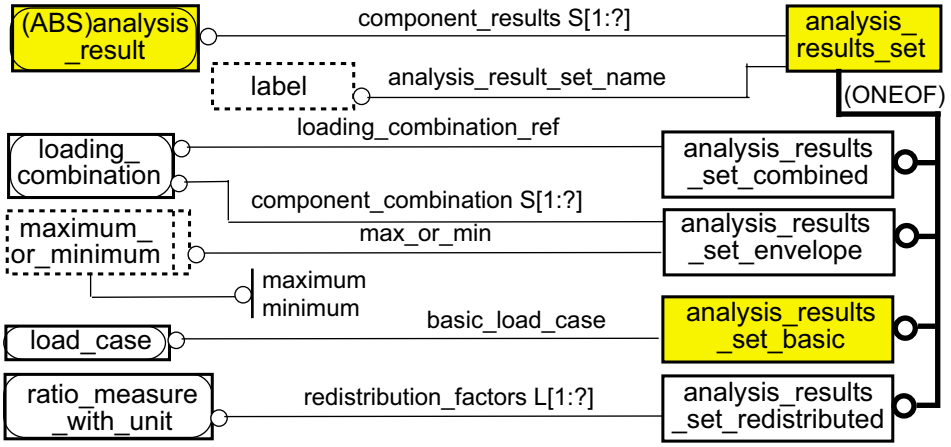

Figure Two: analysis_result_set provides multiple ways to group analysis_results.

The corresponding Express Entities, flattened to designate their actual attributes for the reaction_set_basic and reaction_set_combined are:

```
ENTITY analysis_results_set_basic
      (analysis_results_set:
          results_set_name : label;
          component_results : SET [1:?] OF analysis_result;
      );
      basic_load_case : load_case;
END_ENTITY;

ENTITY analysis_results_set_combined
SUBTYPE OF (analysis_results_set:
          results_set_name : label;
          component_results : SET [1:?] OF analysis_result;
      );
      loading_combination_ref : loading_combination;
END_ENTITY;
```

Analysis_results_set_basic defines the type of loading condition applied to an analysis_ result_set, using a single load_case. The other sets define load cases that deal with loading combinations, or possibly maximal and minimal loading conditions. See the tutorial on analysis loads for more detail.

Associating Analysis Elements with Design Parts:

When an analysis model is created, with its elements and nodes, it was derived from an assembly_design model or assembly_manufacturing model. In some cases, the analysis application may generate both a design model and analysis model. In each of these cases, they correspondence between the elements and design_parts or located_parts needs to be defined. This provides the basis for identifying the association of analysis results with design or manufacturing members.

The correspondence must be recorded by the application that initially generates the analytical elements in combination with the design_parts.. Each element should be linked to its corresponding design_part. This correspondence should be recorded using element_mapping, shown in Figure Three.

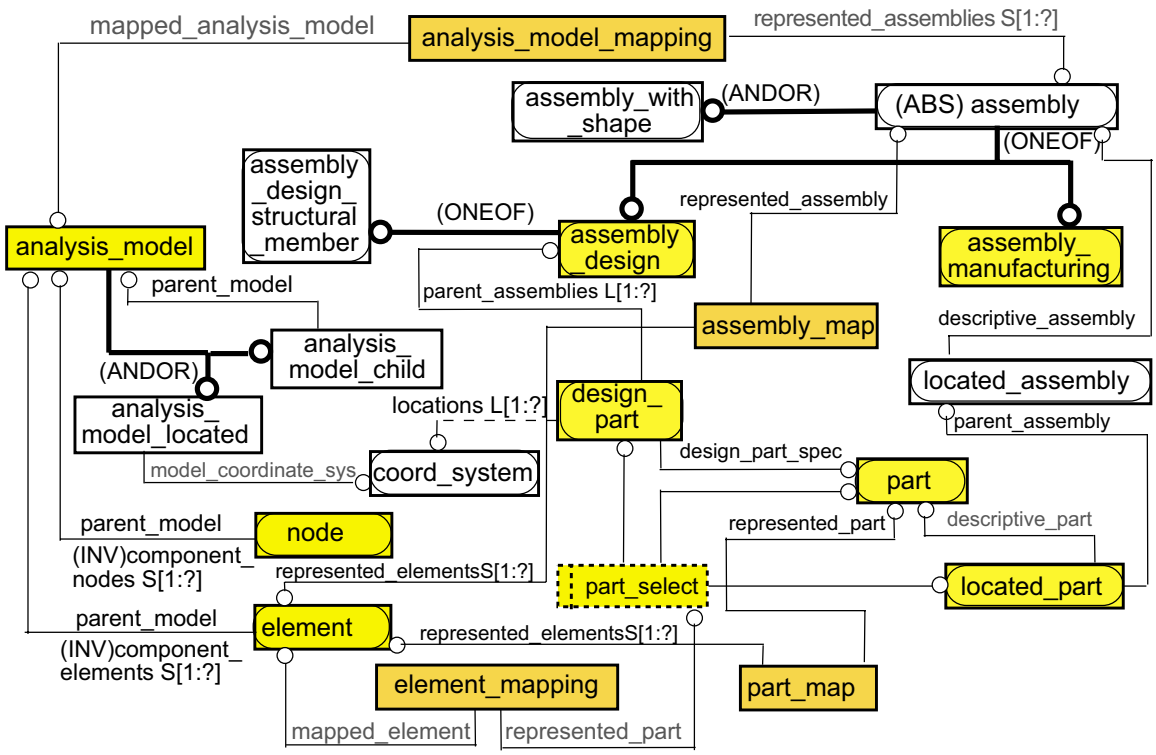

Figure Three: The mapping relations between the three different types of models in CIS/2.

Each design_part has a location and references a part, (which may be singular or shared). Element_mapping has the Express definition:

```
TYPE part_select
 = SELECT
      (part, design_part, located_part);
END_TYPE;

ENTITY element_mapping;
      mapped_element : element;
      represented_part : part_select;
END_ENTITY;
```

In cases where an element is represented by an assembly, assembly_map can be used instead.

Associating Analysis Results with A Design Model

After the mapping relation between elements and design_parts has been defined, and all the reactions defined for the various load_cases, then the resistances can be defined for each design member, connection, etc. Design_result and its various subclasses allow definition of the resistances that a component must satisfy. In addition, an optional link to be made to analysis_result_set, as shown in Figure Four.

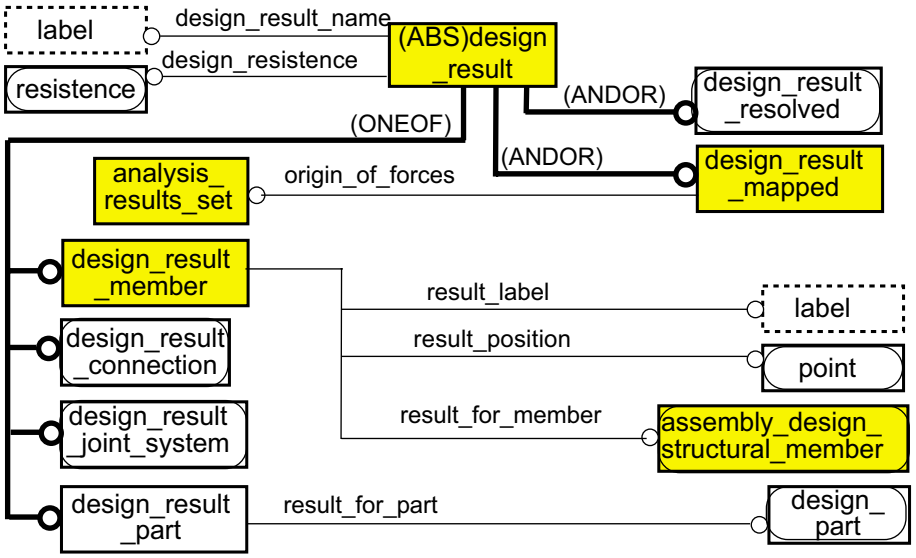

Figure Four: Design_result defines a set of reactions associated with a design element.

Design_result provides the means to define the overall requirements of a component, defined in terms of its required resistance, located at various points along the design member, and to optionally associate the set of reactions the component responds to. Design_result_member, for example, when flattened, has the following definition:

```
ENTITY design_result_member
 (design_result:
     design_result_name : label;
     design_resistance : resistance;
  );
      result_for_member : assembly_design_structural_member;
      result_position : point;
      position_label : label;
END_ENTITY;
```

As defined here, designed_result_member depicts the required behavior capability of the member at one location. Multiple design_result_members may need to be associated with a member to define its overall behavior. As these associations are made, the elements making up, for example, the assembly_design_structural_member can be linked using assembly_map. Assembly_map facilitates going back and forth between elements and their reactions, and the design_result_member and resistances associated with assembly_design_structural_ member. If design_result_member also inherits the optional design_result_mapped, then it also includes an analysis_result_set that reference the range of reactions defining how the

component behaves, as analyzed. These can be used to identify what conditions determined the required resistance.

Resistance:

The Entities for describing the overall performance requirements of a structural component are the resistances and its subclasses. These are shown in Figure Five. Resistance is an Abstract class with five attributes: a resistance_type, an optional description, a resistance_factor that is a scalar multiplier of the moments and forces, which should be 1.0 or greater, and two enumerated values, designating whether the resistance has been specified as local or global and whether the requirements are based on elastic or plastic limits. These general properties are refined as one of three subclasses: axial, shear or bending resistances.

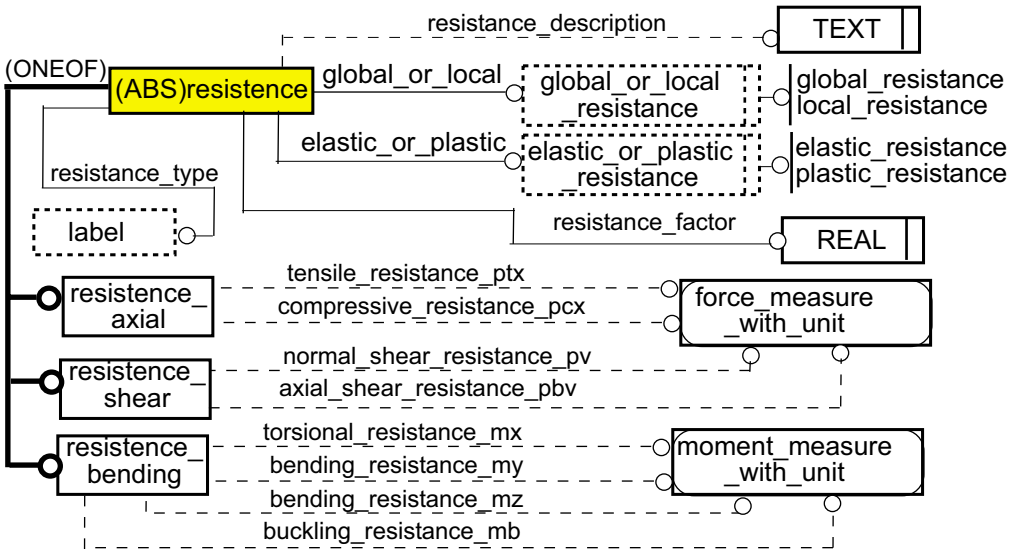

Figure Five: Resistance and its subclasses.

An application must generate the design result, for example for each member and connection. Most typically this will be the analysis application.

Currently, it appears that no manufacturing components may reference or by referenced by resistances.