1996 ASME Intl. Mech. Engr. Congress and Exposition
Application of CAE/CAD to Electronic Systems, EEP-18, 33-46
Atlanta, GA. November 1996

POPULATING PRODUCT DATA FOR ENGINEERING ANALYSIS WITH APPLICATIONS TO PRINTED WIRING ASSEMBLIES

Diego R. Tamburini, Russell S. Peak, Robert E. Fulton
Engineering Information Systems Laboratory
School of Mechanical Engineering
Georgia Institute of Technology
Atlanta, GA

ABSTRACT

The engineering data for products such as Printed Wiring Assemblies (PWAs) is a complex aggregation of heterogeneous information generated by a variety of design tools. At certain stages of the product development cycle this information is used to perform engineering analyses in order to validate the design against several criteria.

This paper addresses the problem of populating product data from several heterogeneous sources in order to support the informational needs of given engineering analyses. This is usually a tedious and error-prone process, which involves accessing the data—often manually—from different repositories, stored in different formats and in different locations. After the data is retrieved, it must be integrated in a way that is meaningful to the analysis models employed. This integration is unique in that much of the data representing the design of the complete product is not used at all in the analysis, while the data that is used has to undergo significant transformation and/or idealization before being fed into the analysis models.

This work proposes an object-oriented architecture to automate the data retrieval and population process. It generates a semantically meaningful set of data from which the analysis applications can extract the information they need more easily. The emerging Standard for the Exchange of Product Model Data (STEP) provides the neutral mechanisms needed by this architecture for describing and exchanging product data. The design and analysis of Printed Wiring Assemblies was chosen to test the concepts developed in this work. In this example, the goal is to assemble PWA product data created by several E/MCAD tools into a single integrated schema that supports the informational needs of specific thermomechanical analyses.

NOMENCLATURE

Γ	Product-Analysis Transformation
α_C	component's coefficient of thermal expansion
ΔL_c	component's length change
aat	Analysis-Analysis Transformation
ABB	Analysis Building Block
al	Associativity Linkage
AOPD	Analysis-Oriented Product Database
AOPM	Analysis-Oriented Product Model
AP	Application Protocol
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
DAI	Design-Analysis Integration
E/CAD	Electrical Computer-Aided Design
ISO	International Standards Organization
L_C	component length
LCCC	Leadless Ceramic Chip Carrier
MDL	Mapping Definition Language
MRA	Multi-Representation Architecture
pat	Product-Analysis Transformation
PBAM	Product Model-Based Analysis Model
PCA	Printed Circuit Assembly
PWA	Printed Wiring Assembly
PWB	Printed Wiring Board
sm	Semantic Mapping
STEP	Standard for the Exchange of Product Data
T_C	component's temperature
T_O	reference temperature

1 INTRODUCTION

In today's product design process, a significant gap typically remains between detailed computer aided design (CAD) models and computer aided engineering (CAE) models. While many aspects of engineering analysis are computation-intensive, we believe design-analysis integration (DAI) is primarily an information-intensive problem that requires engineering information management solutions. This section overviews our DAI approach and identifies the focus of this paper. Section 2 describes the general characteristics of product data to support analysis. Section 3 provides an overview the proposed Analysis-Oriented Product Model Population Technique, as well as a simple example case to illustrate it. Section 4 shows an example of how this populated Analysis-Oriented Product Model is used

by an analysis application. Section 5 discusses the advantages of the technique and future work, and Section 6 summarizes the paper.

This work is being conducted under the DARPA-sponsored TIGER project (Team Integrated Electronic Response) (TIGER Team 1995), in which Georgia Tech is participating jointly with the Atlanta Electronic Commerce Resource Center, Boeing, South Carolina Research Authority, International TechneGroup Incorporated, Arthur D. Little and Holaday Circuits. The task of the Engineering Information Lab at Georgia Tech for this project is to provide thermomechanical analysis capabilities to analyze a Printed-Wiring Assembly that was designed using several E/CAD systems.

1.1. The Multi-Representation Architecture

Linking design and analysis models is fundamentally different than typical data exchange tasks in that it requires **heterogeneous transformations** - transforming one or more types of information into a *different* type of information, typically in different formats (Peak et al. 1996). The integration challenge is further complicated in that a given type of product can have numerous types of analysis models that varying in discipline, resolution, application, and complexity (Peak 1993; Peak et al. 1995).

We believe this diversity makes the gap between design and analysis models too large for a single integration bridge. Hence, (Peak et al. 1995) developed the **multi-representation architecture** (MRA) with intermediate representations as stepping stones to achieve a flexible, modular integration of commercial design and analysis tools (Figure 1). Solution method models (SMMs) are object-oriented wrappers around solution tools (e.g., FEA systems) that utilize an agent-based framework to obtain analysis results in a highly automated manner. Analysis building blocks (ABBs) are a representation of engineering analysis concepts with high semantic content. ABBs generate SMMs based on solution technique-specific considerations such as symmetry and mesh density. Product models (PMs) represent the design-oriented details and provide a common stepping stone to design tools. Finally, product model-based analysis models (PBAMs) are templates that explicitly represent the heterogeneous associativity between analysis models (i.e., ABBs) and design models (i.e., PMs).

PBAMs can be used to create catalogs of ready-to-use analysis modules for applications such as solder joint deformation and fatigue, PWB warpage, and plated-through hole deformation (Peak 1993; Peak and Fulton 1993; Peak et al. 1996). (Cimtalay et al. 1996) have demonstrated initial usage to achieve modular optimization.

1.2. Focus of this paper

The work on the MRA architecture described above focused on developing a mechanism to extract and transform data from an integrated product database in order to perform some engineering analyses. As described in (Peak et al. 1995), the PBAMs use the idealizations supported by this product model to perform the analyses. However, the assumption was made that this integrated product database was already available, without going into further details on how it was populated with the data generated by the different analysis tools. This is precisely the focus of this paper: to complement previous work on the MRA architecture by proposing a mechanism to assemble data from several heterogeneous sources in order to populate an Analysis-Oriented Product Database that supports engineering analysis effectively. We also elaborate more on how idealizations are described and implemented in this product model.

2. CHARACTERISTICS OF PRODUCT DATA TO SUPPORT ANALYSIS

The primary goal of engineering design is to obtain a complete, unambiguous, manufacturable description of a given product. During the development of this product, designers use a wide variety of software systems that generate data describing different aspects of the product. The result is a complex aggregation of heterogeneous information that is very large and detailed. At certain stages of the product development cycle this information is used to perform engineering analyses in order to validate the

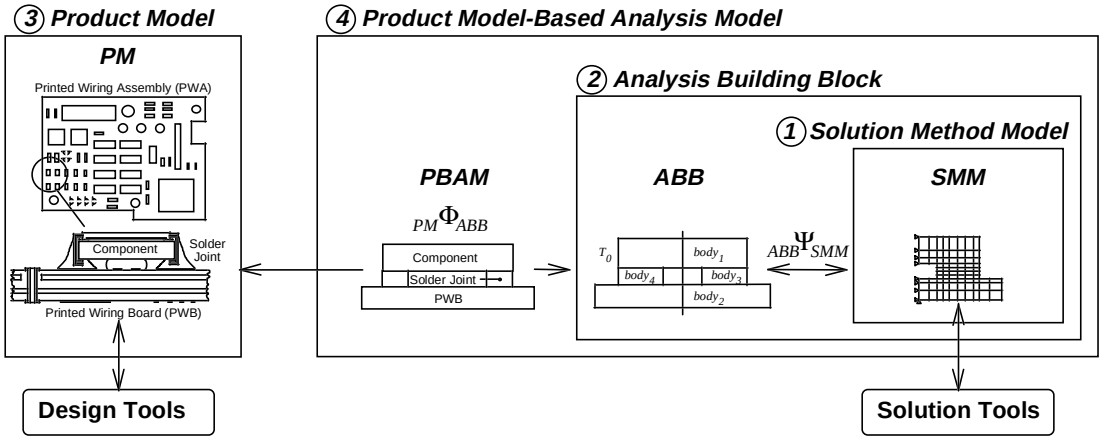


Figure 1: The Multi-Representation Architecture for Design-Analysis Integration

design against several criteria. Although there is a large number of computer aided engineering tools available, the current status is that typically two software tools are not compatible in such a way that they can exchange data directly - without cumbersome (manual or semi-automatic) transformation (Kemper and Moerkotte 1994). In many cases, the data needed as input for the analysis models is even retrieved manually and re-inputted in some other computer application for analysis. Some raw design information must undergo significant transformation and/or idealization before being fed into the analysis models on which the analysis applications are based (Armstrong 1994; Shephard et al. 1990). This is usually a tedious, slow, and error-prone process that illustrates the much dreaded "islands of automation".

A more ideal scenario would be one in which there is a single, analysis-oriented repository that contains all the information needed for the analyses in a ready-to-use form, isolating the analyst from the intricacies of specific file formats and data structures (Urban et al. 1993), unfamiliar entity and attribute names, and complex data transformations.

For this purpose, an integrated analysis-oriented data model should provide mechanisms to support:

- ***Heterogeneous sources of data:*** Data needed for analysis comes from different heterogeneous sources. Design-analysis integration will only be achieved if we can provide a view of the informational world that conceals much of the locational and structural properties of the data (Kemper and Moerkotte 1994). A good example illustrating the different sources of data needed for thermomechanical analysis of PWAs can be found in (Zhou 1996). Another example is the analysis of PWB bending during manufacturing; such an analysis would need data about the PWB itself - created with an E/CAD tool - as well as data about the manufacturing process (process temperatures, forces, etc.) - created by a process/factory definition tool.
- ***Reusable idealizations:*** The product model must support idealizations that relate detailed, design-oriented attributes with simplified, analysis-oriented attributes for usage by potentially many analysis applications[1] (Peak et al. 1995). For example, a common analysis attribute used for different thermal bending analysis models for PWBs is the diagonal length of the board. This length is an idealization that is obtained from the outline of the board, and its computation may be quite complex (specially if the edges of the board are not straight lines). Therefore, it would be better to have the value of this attribute readily available in the analysis-oriented product model, instead of requiring each analysis application that uses this attribute to calculate it.
- ***Data synthesis:*** Synthesis is the opposite of idealization; during synthesis we assign values to product variables based on the results of an analysis. This process is complicated by the fact that the set of product data is richer than the one of analysis data, and therefore there may be the need to add information in order to synthesize data. Additionally, product-analysis transformations that have a closed-form solution in one direction (for instance, a relation of the form $A1 = \Gamma(P1, P2)$, where A1 is an analysis variable, and P1 and P2 are two product variables), may not have one if, for example, we need to solve for one of the product variables, say P1.
- ***Unavailable Analysis Data:*** Design data often must be complemented with additional data in order to perform an analysis. Some analysis may need very specific information that is not being supplied by any of the design tools or that is not readily available in any form. Examples of this type of information are detailed electrical component information (internal materials or dimensions), temperature-dependent material properties, etc.
- ***Simplifications:*** Much of the data representing the design of the complete product is not used at all in the analysis (Morris et al. 1992). An example of this is geometry; analyses rarely need all the detailed information used by geometric modelers to represent the geometry of the product. Analysis models normally use a simplified version of the real geometry. In order to be efficient, the analysis-oriented product model must contain only the minimum amount of information needed by the analysis models.
- ***Data complexity:*** Engineering analyses tend to be "information-hungry". However, they normally demand a large number of types of data - complicatedly interconnected - as opposed to a large number of instances of each type of data.

3. THE ANALYSIS-ORIENTED PRODUCT MODEL POPULATION TECHNIQUE

3.1. Overview of the Technique

As introduced above, the goal of the technique presented in this paper is to populate an Analysis-Oriented Product Model (AOPM) with data coming from different design sources in order to support the information requirements of several analysis models. We will refer to the *populated* AOPM as the Analysis-Oriented Product Database (AOPD). The AOPM is a single, integrated, abstracted view of the design-oriented product data that is more appropriate for engineering analysis. By "more appropriate" we mean:

- It contains entities whose names, attributes and structure are more suitable for use by analysis models.
- It contains mostly data that would be used by typical analysis models, which is a subset of all the data generated by the design tools.
- More importantly, the AOPM supports *idealizations* of the data. As described previously, idealizations are essential to engineering analysis. Reusability is one advantage of including idealizations in the AOPM, since many analysis models may use the same common idealizations.

[1] We may also think of idealizations as attributes of the product beyond those needed to give a complete and unambiguous description to *manufacture* this product.

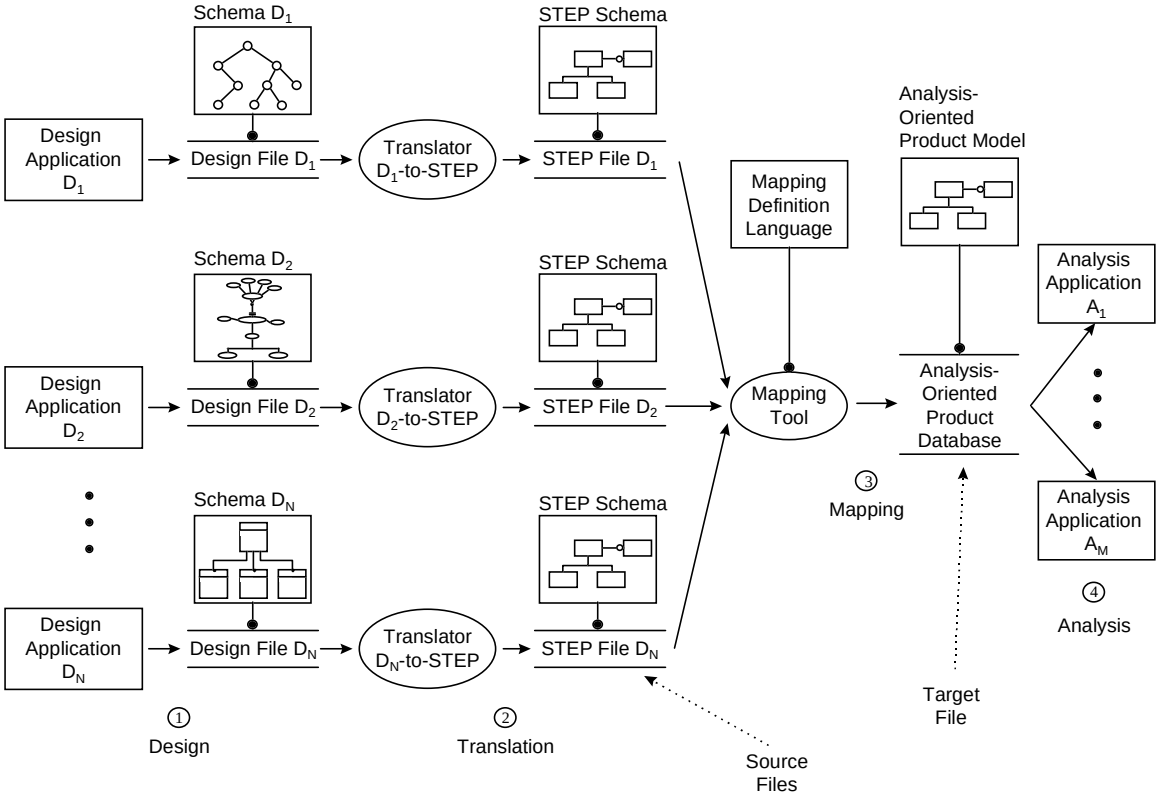


Figure 2: Overview of the Technique

A brief description of the proposed technique is presented next, followed by a simple test case to explain it with more detail. Figure 2 shows the overall data flow of the technique, as well as the schemas that define the structure of the data in each step. From left to right we have:

1. ***Design:*** The process begins when data about different aspects of the product is generated by the design applications (D_1 to D_N). Examples of design applications are geometric modelers, PWA design tools and Database Management Systems. Each application writes its data in a file (or set of files) using its own data models and structures which, in general, can vary widely from one application to another (e.g., hierarchical, relational and object-oriented data models as shown). In addition, the data can be stored in plain ASCII or binary database files.

2. ***Translation:*** An X-to-STEP[2] translator (where X is the proprietary data format of the application) translates the data in each of these files to STEP. The vendors of the design applications will ideally develop these translators in order to make their tools standard-compliant. During this phase, we *homogenize the data models*, that is, translate the data that conforms to each tool's data model into STEP, the common data model (for example, a table in a relational database may be translated to multiple instances of an EXPRESS entity). The product of the translation is a STEP (Part 21) file[3]. Ideally, these STEP files will conform to one of the STEP Application Protocols[4] such as AP210 or AP203, but they could conform to

[2] **STEP** (ISO-10303) (ISO 10303-1 1994) was chosen as the neutral representation of product data for our technique because it provides neutral mechanisms for describing and exchanging product data, and because there are several tools available (Spooner 1993; Spooner and Hardwick 1993) to develop applications that create and manipulate STEP databases. The standard also defines a textual conceptual schema language, called EXPRESS (ISO 10303-11 1994; Schenck and Wilson 1994), used to specify the normative part of all the information models in STEP. EXPRESS is both human-readable and computer-processable.

[3] ***A STEP file*** (also known as Part 21 file, STEP Physical Exchange File or STEP data file) (ISO 10303-21 1994) is a text file that contains instances of entities corresponding to a given EXPRESS schema. The STEP file is what actually gets exchanged among applications, assuming that the applications are cognizant of the associated schema.

[4] A STEP ***Application Protocol*** is a representation of product information for one or more applications. For example, AP210 ("Printed Circuit Assembly Product Design Data") (ISO DIS 10303-210 1993) describes the structure of the data needed to product a manufacturable description of a PCA, and AP203 ("Configuration Controlled Design") (ISO 10303-203 1994) specifies the structures for the exchange between application systems of configuration controlled

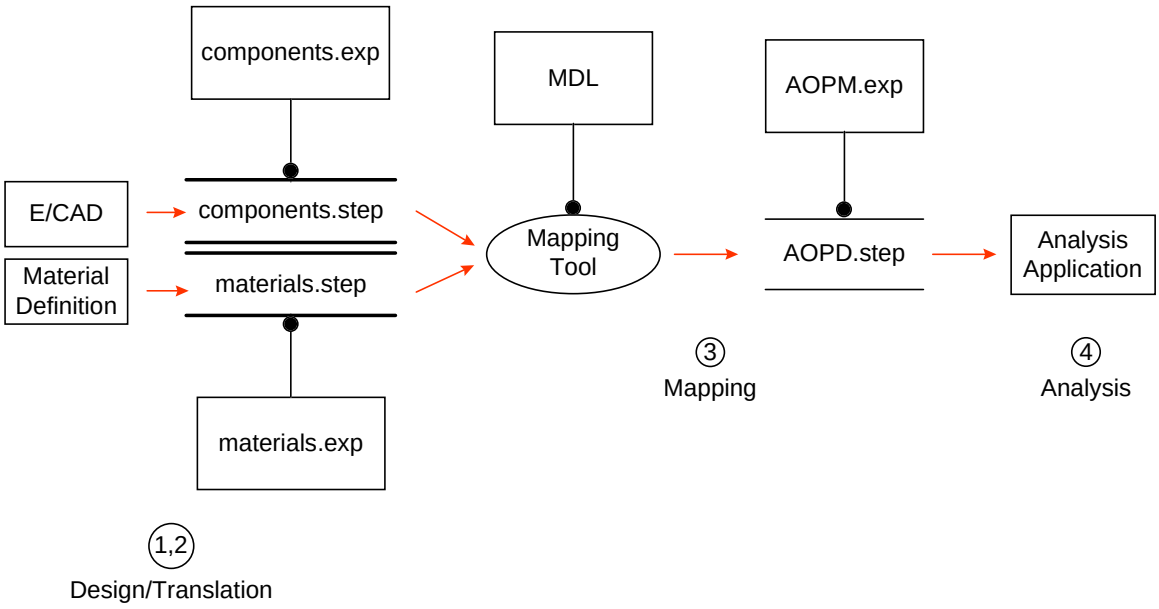


Figure 3: AOPD Population Example Case

any other (non-standard) schema described in EXPRESS. Relying on a standard schema allows integrating design applications from several vendors, as long as they provide a mechanism to generate a STEP-compliant file. From now on, we will refer to these STEP files as the "source data files" (because, as we will see in the next point, they will be the source files for the mappings).

3. ***Mapping:*** As described in the previous point, the translators generate data files expresses in a common information modeling language. However, the information these data files contain will still be, in general, both structurally and semantically very different from one to another. The goal of the mapping stage is to *homogenize the data structures* and generate a single data file (the AOPD or "target data file") that conforms to an integrated schema (the AOPM). In STEP terms, this could mean integrating data that spans more than one Application Protocol. This process is similar to the "semantic mapping" of the EXPRESS-Driven Data Conversion methodology described in (Gadient and Hines 1994). In order to generate this AOPD, we define the mappings between entities in the source data files and entities in the target data file using a Mapping Definition Language (MDL). The mappings are materialized by a mapping tool that reads the source files and the MDL definition of the mappings and generates the target file as its output. EXPRESS-V[5] (Hardwick 1994; Spooner et al. 1995) was selected as the MDL for this work. EXPRESS-V is an extension to EXPRESS that allows an information modeler to write code that describes how one set of EXPRESSS entities are mapped into another set of entities. The final product of this stage is a single, integrated, Analysis-Oriented Product Database (AOPD) containing the data needed for analysis purposes.

4. ***Analysis:*** Finally, the analysis applications read the data they need from the AOPD. It is here where the work presented in this paper is linked to the previous MRA work. PBAMS of the MRA use the data and the idealizations supported in the AOPM to perform the analyses.

3.2. Example Case

In this section we will illustrate the technique just described with the simple example shown in Figure 3. We will show how the data from the two STEP files is mapped into an integrated AOPD (whose structure is defined by the AOPM) and how a few idealizations are described and implemented. In the following section we will show how this resulting AOPD is used by a simple analysis application that we selected for this example.

In the same order in which we presented the technique we have:

1. ***Design***: For this example, we will assume that there are two design applications: one used to define the electrical components and their geometry (we will refer to this design application which as the "E/CAD Tool") and the other to populate a database of material properties (the "Material Definition Tool").

2. ***Translation***: After the translation from the proprietary format of the design applications to STEP is completed we have two STEP files: `components_data.step` (Figure 4, defined by the EXPRESS schema shown in Figure 5)[6], which contains the

three-dimensional product definition data of mechanical parts and assemblies. It is expected that several hundred APs may be developed to support the many industrial applications that STEP is expected to serve.

[5] Other existing MDLs are EXPRESS-M, EXPRESS-C, and KIF. EXPRESS-M and EXPRESS-V are currently merging into EXPRESS-X, which is in the early stages of the ISO standardization process. The intention is to include this mapping language as one of the parts of the STEP standard.

[6] In a STEP file, each line in the DATA section (indicated by a # sign followed by a sequential number, an equal sign, the name of the entity

design data generated by the E/CAD tool, and `materials_data.step` (Figure 6, defined by the EXPRESS schema shown in Figure 7), which contains the data generated by the Materials Definition Tool.

3. ***Mapping***: The goal is to populate the schema `aopm.step` shown in Figure 8 with data stored in the files `components_data.step` and `materials_data.step` (Figures 4 and 6, respectively). In order to do this, we must define the mappings that take place between the source data and the target data. A graphical description of these mappings is shown in Figure 9, along with their lexical description using EXPRESS-V in Figure 10. The resulting integrated STEP file (`aopd.step`) is given in Figure 11. For example, see how an instance of `SM_RESISTOR` (instance #10 in `components_data.step`, Figure 4) is combined with an instance of `LINEAR_MATERIAL` that contains the material properties of alumina (instance #10 in `materials_data.step`, Figure 6) to generate an instance of `RESISTOR` (instance #40 in `aopd.step`, Figure 11).

```
ISO-10303-21;
HEADER;
/*----------------------------------------
 * Exchange File generated by ST-DEVELOPER v1.4
 * Conforms to ISO 10303-21
 */
FILE_DESCRIPTION (('components_data'), '1');
FILE_NAME ('components_data', '1996-07-01T11:33:39-04:00',
(''), (''), 'ST-DEVELOPER v1.4', '', '');
FILE_SCHEMA (('COMPONENTS'));
ENDSEC;

DATA;
#10 = SM_RESISTOR ('RES100', '1206 Surface-Mount
     Resistor', 0.125, 0.063, 0.02, 300., 'Sn-Pb',
     'Alumina', 'Glass', 235., 'RuO2');
#20 = SM_LCCC ('LCC100', 'Leadless Ceramic Chip Carrier',
     1., 0.7, 0.2, 350., 'Sn-Pb', 'Ceramic', 'Glass', 2,
     20 );
ENDSEC;
END-ISO-10303-21;
```

Figure 4: Source STEP file D_1 (`components_data.step`)

and a list of values) represents an instance of an entity defined in a corresponding EXPRESS schema. For example, line #10 in Figure 4 is an instance of `SM_RESISTOR`. The values of the attributes are given in the same order they were defined in the EXPRESS schema. For example, the first value of instance #10 (`'RES100'`) is the `product_id` of the `SM_RESISTOR`. If the value of an attribute is itself an instance of another entity, the number of the latter will appear in the place of the attribute. A `$` sign represents a NULL value. For a complete description of STEP files see (ISO 10303-21 1994).

```
SCHEMA components;

ENTITY component
  SUPERTYPE OF ( ONEOF ( SM_resistor , SM_LCCC ) );
   product_id    : id;
   description   : STRING;
   body_length   : distance_measure;
   body_width    : distance_measure;
   body_height   : distance_measure;
   power_rating  : power_measure;
   lead_material : material_name;
   base_material : material_name;
   protective_film_material : material_name;
END_ENTITY;

ENTITY SM_resistor
  SUBTYPE OF ( component );
   resistance    : resistance_measure;
   resistive_element_material : material_name;
END_ENTITY;

ENTITY SM_LCCC
  SUBTYPE OF ( component );
   number_of_lead_sides  : positive_integer;
   leads_per_side                : positive_integer;
END_ENTITY;

END_SCHEMA (* components *);
```

Figure 5: Schema `components.exp` for Source STEP file D_1

```
ISO-10303-21;
HEADER;
/*----------------------------------------
 * Exchange File generated by ST-DEVELOPER v1.4
 * Conforms to ISO 10303-21
 */
FILE_DESCRIPTION (('materials_data'), '1');
FILE_NAME ('materials_data', '1996-07-01T12:13:37-04:00',
(''), (''), 'ST-DEVELOPER v1.4', '', '');
FILE_SCHEMA (('MATERIALS'));
ENDSEC;

DATA;
#10 = LINEAR_MATERIAL ('Alumina', $ , 0.0000067, $ , $ , $
     , $ );
#20 = LINEAR_MATERIAL ('Ceramic', $ , 0.0000003, $ , $ , $
     , $ );
ENDSEC;
END-ISO-10303-21;
```

Figure 6: Source STEP file D_2 (`materials_data.step`)

```
SCHEMA materials;

ENTITY linear_material;
   name                   : STRING;
   young_modulus          : REAL;
   coef_thermal_expansion : REAL;
   shear_modulus          : REAL;
   yield_stress           : REAL;
   ultimate_stress        : REAL;
   poissons_ratio                 : REAL;
END_ENTITY;

END_SCHEMA (* materials *);
```

Figure 7: Schema `materials.exp` for Source STEP file D_2

```
SCHEMA aopm;

(* Inter-Schema References *)
REFERENCE FROM analysis_models_schema;
REFERENCE FROM support_schema;

(* Entity Definitions *)
ENTITY electrical_component
  ABSTRACT SUPERTYPE OF( ONEOF ( discrete_component ,
integrated_component ) );
   part_number : id;
   description : STRING;
   package : electrical_package;
   (* Idealized Attributes *)
   primary_structural_material : solid_material;
END_ENTITY;

ENTITY discrete_component
  ABSTRACT SUPERTYPE OF( resistor )
  SUBTYPE OF( electrical_component );
   magnitude : positive_real;
   tolerance : positive_real;
   power_rating : positive_real;
END_ENTITY;

ENTITY resistor
 SUBTYPE OF ( discrete_component );
   base_material : solid_material;
 WHERE
   pat1 : primary_structural_material = base_material;
END_ENTITY;

ENTITY integrated_component
 ABSTRACT SUPERTYPE OF( ONEOF ( integrated_circuit ) )
 SUBTYPE OF( electrical_component );
   case_material : solid_material;
END_ENTITY;

ENTITY integrated_circuit
 SUBTYPE OF( integrated_component );
 WHERE
   pat1 : primary_structural_material = case_material;
END_ENTITY;

ENTITY electrical_package
ABSTRACT SUPERTYPE OF( surface_mount_package );

(* Idealized Attributes *)
   bounding_box_length    : positive_length_measure;
   bounding_box_width     : positive_length_measure;
   bounding_box_height    : positive_length_measure;
   inter_solder_joint_distance :
positive_length_measure;
END_ENTITY;

ENTITY surface_mount_package
 ABSTRACT SUPERTYPE OF( ONEOF ( two_lead_component ,
                                chip_carrier ) )
 SUBTYPE OF( electrical_package );
   body_length   : positive_length_measure;
   body_width    : positive_length_measure;
   body_height   : positive_length_measure;
 WHERE
   pat1 : bounding_box_length = body_length;
   pat2 : bounding_box_width = body_width;
   pat3 : bounding_box_height = body_height;
END_ENTITY;

ENTITY two_lead_component
 SUBTYPE OF( surface_mount_package );
 WHERE
   pat1 : inter_solder_joint_distance =
SELF\body_length;
END_ENTITY;

ENTITY chip_carrier
  ABSTRACT SUPERTYPE OF( LCCC )
  SUBTYPE OF( surface_mount_package );
 WHERE
   pat1 : inter_solder_joint_distance = SQRT(
          SELF\body_length**2 + SELF\body_width**2);
END_ENTITY;

ENTITY LCCC
  SUBTYPE OF( chip_carrier );
END_ENTITY;

ENTITY solid_material;
   name : STRING;
   associated_linear_elastic_model :
linear_elastic_model;
   associated_nonlinear_plastic_model :
                                  nonlinear_plastic_model
                                  ;
   associated_fatigue_model : fatigue_model;
END_ENTITY;

END_SCHEMA (* aopm *);
```

Figure 8: Analysis-Oriented Product Model Schema (idealized attributes shown in bold)

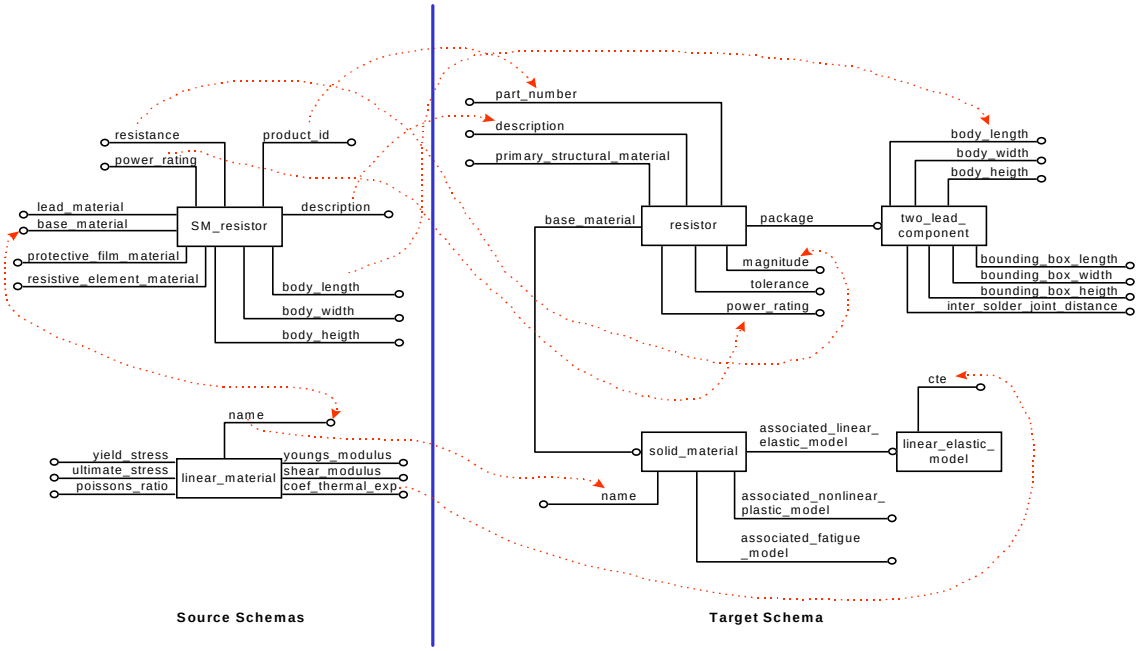


Figure 9: Graphical description of the mappings (partial EXPRESS-G diagrams and mappings for resistor only shown; hierarchies were flatten for simplicity)

```
SCHEMA mapping_schema;

USE FROM components;
USE FROM materials;
USE FROM aopm;

VIEW resistor
FROM( SM_resistor , linear_material )
WHEN ( SM_resistor.base_material = linear_material.name );
VIEW_ASSIGN
   part_number := SM_resistor\SM_component.product_id;
   description := SM_resistor\SM_component.description;

   NEW two_lead_component;
   two_lead_component\surface_mount_package.body_length :=
      SM_resistor\SM_component.body_length;
   two_lead_component\surface_mount_package.body_width :=
      SM_resistor\SM_component.body_width;
   two_lead_component\surface_mount_package.body_height :=
      SM_resistor\SM_component.body_height;

   package := two_lead_component;

   magnitude := SM_resistor.resistance;
   power_rating := SM_resitor.power_rating;

   NEW linear_elastic_model;
   linear_elastic_model.coeff_thermal_expansion :=
      linear_material.coeff_thermal_expansion;

   NEW solid_material;
   solid_material.associated_linear_elastic_model :=
      linear_elastic_model;

   base_material := solid_material;

END_VIEW;
END_SCHEMA;
```

Figure 10: EXPRESS-V of the mappings (only mapping for `resistor` shown)

```
ISO-10303-21;
HEADER;
/*----------------------------------------
 * Exchange File generated by ST-DEVELOPER v1.4
 * Conforms to ISO 10303-21
 */
FILE_DESCRIPTION (('1'), '1');
FILE_NAME ('aopd', '1996-07-06T10:15:46-04:00', (''),
(''), 'ST-DEVELOPER v1.4', '', '');
FILE_SCHEMA (('AOPM',
              'ANALYSIS_MODELS_SCHEMA'));
ENDSEC;

DATA;
#10 = TWO_LEAD_COMPONENT (0., 0., 0., 0., 0.125, 0.063,
     0.02);
#20 = LINEAR_ELASTIC_MODEL (0., 0., 6.7E-06, 0.);
#30 = SOLID_MATERIAL ('Alumina', #20, $, $);
#40 = RESISTOR ('RES100', '1206 Surface-Mount Resistor',
     #10, $, 235., 0., 300., #30);
#50 = LCCC (0., 0., 0., 0., 1., 0.7, 0.2);
#60 = LINEAR_ELASTIC_MODEL (0., 0., 3.E-07, 0.);
#70 = SOLID_MATERIAL ('Ceramic', #60, $, $);
#80 = INTEGRATED_CIRCUIT ('LCC100', 'Leadless Ceramic Chip
     Carrier', #50, $, #70);
ENDSEC;
END-ISO-10303-21;
```

Figure 11: Resulting Analysis-Oriented Product Data File (`aopd.step`) (null values of idealized attributes shown in bold)

4. Analysis: Section 4 describes one analysis application that can use the data stored in the AOPD and the idealizations supported by the AOPM: a simple, formula-based PBAM that calculates the elongation of a component due to temperature changes.

3.3. Supported Idealizations

We will now focus our attention on the idealizations supported by the AOPM (shown in bold in Figure 8):

- **`electrical_component.primary_structural_material`:** An electrical component such as a surface mount resistor can have several layers of different materials (Figure 12). For analysis purposes, we may want to select one of them as the primary structural material and assume that it is the only material composing the component. For resistors, we will consider the base material to be the primary structural material, and for chip carriers, the case material.

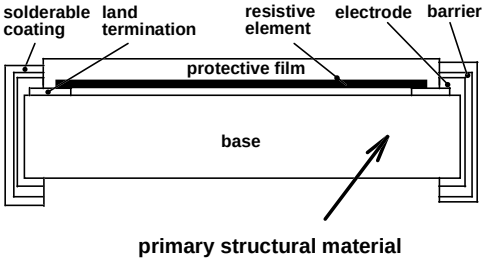


Figure 12: Layers of Materials in a Surface Mount Resistor

- **`electrical_package.inter_solder_joint_distance`:** This is a geometric idealization commonly used by solder-joint fatigue models (Engelmaier 1983; Engelmaier 1989). As shown in Figure 13, for a multi-terminal component such an LCCC (Leadless Ceramic Chip Carrier) it is the diagonal distance viewed from the top, while for a two-terminal component such as a surface-mount resistor, it is just its total length. Here the advantage of using an object-oriented approach becomes more evident since we can take advantage of polymorphism to redefine such attributes depending on the type of object, as done for the attribute `inter_solder_joint_distance` in the entities `two_lead_component` and `chip_carrier` in Figure 8 (WHERE rules '`pat1`' of each entity).

- **`electrical_package.bounding_box_length/height/width`:** Certain analyses need the dimensions of an imaginary rectangular geometry that encloses the component rather than its detailed geometry. Since the two components in this example are already rectangular, the values of these idealized attributes are directly the total dimensions of the components. However, if there was, for example, a cylindrical component, its radius could be idealized as the height and width of the bounding box.

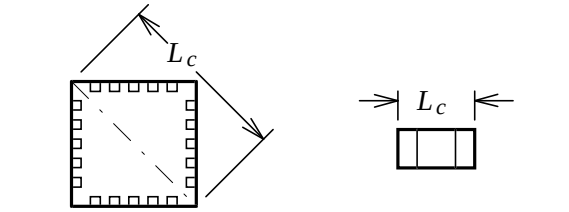


Figure 13: Inter-Solder Joint Distance for Multi-Terminal and Two-Terminal components.

The values of the idealized attributes for this test case are obtained by simple assignments of values of design attributes (e.g., `primary_structural_material = base_material`) or by simple calculations (e.g., `inter_solder_joint_distance` = SQRT(`body_length`^{2} + `body_width`^{2})). In other cases, however, the computation of a value for an idealized attribute may not be as straightforward (recall the example of the diagonal length of the PWB mentioned in the previous section). Considering this, and the fact that these values are needed only when an analysis model asks for them, we choose not to calculate them during the mapping phase in order to avoid wasting resources in unnecessary complex calculations. Notice that in the EXPRESS-V definition of the mapping (Figure 11) these attributes do not get mapped. Also notice that in the AOPD (Figure 11), the values for the idealized attributes are either NULL (indicated by a dollar sign in EXPRESS) or initially set to zero.

3.4. Implementation of the Idealizations

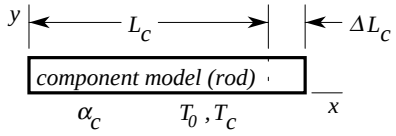
The relations (also known as **product-analysis transformations**) needed to obtain the values of the idealized attributes described above are *defined* in the AOPM as WHERE rules (see Figure 8). For example, in order to calculate the value of the idealized attribute `inter_solder_joint_distance` for a chip carrier, there is a WHERE rule in the `chip_carrier` entity (rule "`pat1`") that defines this distance to be equal to the square root of the sum of the squares of the length and the width.

The EXPRESS definitions of entities of the AOPM and the idealizations they support are *implemented* in a programming language (C++ was used for this work) so that they can be used by the PBAMs. Each entity in the AOPM is implemented as a class, and the attributes of this entity as class variables of this class. The protocol of this class consists of member functions to access and update the values of the attributes of the entity as well as member functions that implement the WHERE rules and allow us to get the values of the idealized attributes.

At this point, the objective of the technique has been reached; in one hand we have the integrated AOPD file (generated with the mapping), and on the other hand we have a library of C++ classes that provide an interface to the AOPD and that augment it by providing means to obtain the values of idealized attributes on demand. In the next section, we provide an example of how this populated AOPM is actually used for analysis.

4. AOPM USAGE EXAMPLE

The following is an example showing how the entities `resistor` and `LCCC` defined in the AOPM are used by a simple analysis application. This analysis application uses some their attributes (mapped and idealized) to calculate the elongation of these components due to a change in temperature (Figure 14). The component is modeled as a one-dimensional rod having a length L_C, a coefficient of thermal expansion α_C and subjected to a temperature T_C. The change in length ΔL_c is caused by the difference between the reference temperature (T_O) and the temperature of the component (T_C).



$$\Delta L_C = \alpha_C (T_C - T_O)\, L_C$$

Figure 14: Analysis Model for the Test Case

This analysis model is represented primarily using two entities: a PBAM called `component_extensional_model` and an ABB called `elementary_rod`. When represented using EXPRESS (Figure 15), the ABB becomes an entity, its analysis variables the attributes of this entity, and the decomposed formulas (or analysis-analysis transformations) WHERE rules (e.g., aat1 and aat2 of entity `elementary_rod` in Figure 15).

```
ENTITY elementary_deformable_body
  ABSTRACT SUPERTYPE OF ( elementary_rod )
  SUBTYPE OF( analysis_primitive );
   associated_temperature : temperature;
   temperature_change     : temperature;
   reference_temperature  : temperature;
   material_model                 : stress_strain_model;
  WHERE
   aat1 : temperature_change = associated_temperature -
          reference_temperature;
END_ENTITY;

ENTITY elementary_rod
  SUBTYPE OF ( elementary_deformable_body );

   (* Analysis Variables *)
   associated_area       : area;
   undeformed_length     : positive_length_measure;
   total_elongation      : length_measure;
   associated_strain     : strain;
   associated_force      : force;
   (* Semantically Mapped Variables *)
   youngs_modulus        : REAL;
   cte           : REAL;
   (* Subsystems *)
   material_model        :
linear_elastic_model;(*redeclaration*)
  WHERE
   (* Analysis-Analysis Transformations *)
   aat1 : total_elongation =
          associated_strain*undeformed_length;
   aat2 : associated_strain =
          associated_force/(youngs_modulus *
          associated_area) + cte*temperature_change;
   (* Semantic Mappings *)
   sm1 : youngs_modulus = material_model.youngs_modulus;
   sm2 : cte = material_model.cte;
   (* Subsystem Conditions *)
   (* none extra *)
END_ENTITY;

(* Entity Defs:Product Model-Based Analysis Models *)

ENTITY component_extensional_model
  SUBTYPE OF ( product_model_based_analysis_model );

   (* Product Variables *)
   component : electrical_component;
   (* Analysis Variables *)
   (* none extra *)
   (* Semantically Mapped Variables *)
   undeformed_length     : positive_length_measure;
   associated_cte                : cte;
   reference_temperature : temperature;
   associated_temperature: temperature;
   temperature_change    : temperature;
   total_elongation      : length_measure;
   associated_strain     : strain;
   (* Subsystems *)
   deformation_model     : elementary_rod;
WHERE
   (* Associativity Linkages *)
   al1 : deformation_model.undeformed_length =
         SELF\component.package.bounding_box_length;
   al2 : deformation_model.cte =
         SELF\component.primary_structural_material.assoc
         iated_linear_elastic_model.cte;
   (* Analysis-Analysis Transformations *)
   (* none extra *)
   (* Semantic Mappings *)
   sm1 : undeformed_length =
         deformation_model.undeformed_length;
   sm2 : associated_cte = deformation_model.cte;
   sm3 : reference_temperature =
         deformation_model.reference_temperature;
   sm4 : associated_temperature =
         deformation_model.associated_temperature;
   sm5 : temperature_change =
         deformation_model.temperature_change;
   sm6 : total_elongation =
         deformation_model.total_elongation;
   sm7 : associated_strain =
         deformation_model.associated_strain;
   (* Subsystem Conditions *)
   (* none extra *)
END_ENTITY;
```

Figure 15: elementary_rod and component_extensional_model EXPRESS descriptions (idealized attributes in bold)

The constraint schematic in Figure 16 shows how the PBAM wraps the ABB and defines the linkages between the attributes of a design object (`component`) and the attributes of the ABB (for a complete description of the constraint schematics notation see (Peak 1993), or refer to Figure 17 for the basic notation). This PBAM also provides the interface needed to input analysis variables (e.g., reference temperature) and get analysis results (e.g., total elongation).

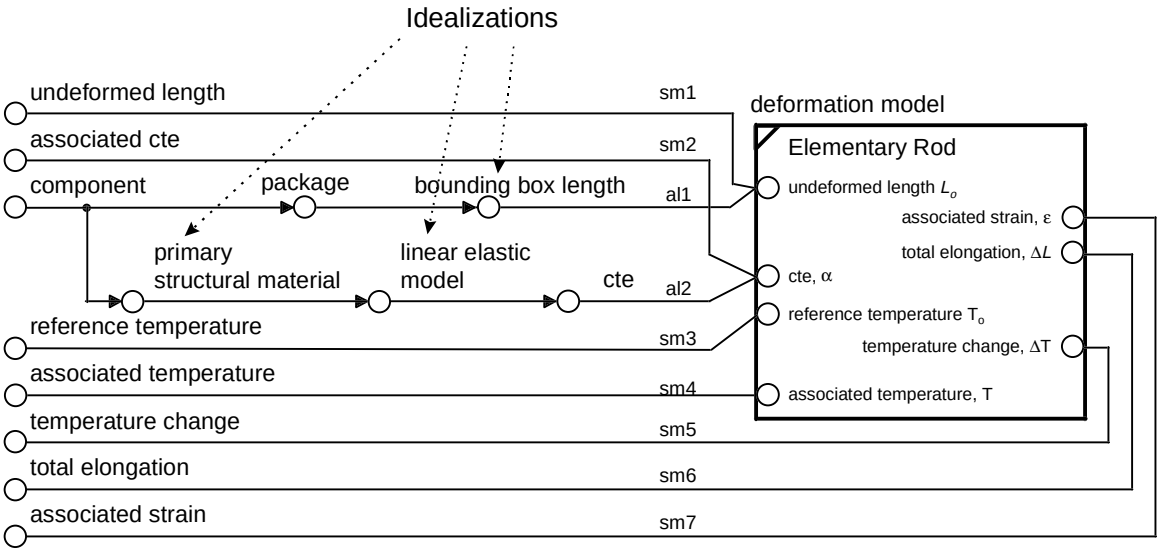


Figure 16: Component Extensional Model Constraint Schematic

Two important types of linkages are shown in this constraint schematics: *Semantic Mappings* (sm1-7), which link attributes of the PBAM with attributes of the embedded ABB, and *Associativity Linkages* (al1-2), which link attributes of the product with attributes of the ABB. Associativity Linkages will usually occur between idealized attributes of the product model and attributes of the ABB (as it happens in the figure in the "al1" and "al2" linkages). This is precisely where the analysis applications "use" the idealizations supported by the AOPM. Semantic Mappings and Associativity Linkages are described as WHERE rules in the EXPRESS description of the PBAM `component_extensional_model` (Figure 15).

As also shown in Figure 15 (in bold), this particular PBAM uses the idealizations `electrical_package.bounding_box_length` (in the WHERE rule "al1") and `electrical_component.primary_structural_material` (in the WHERE rule "al2"). Notice that the AOPM supports more idealizations than those needed by this analysis application. The more complicated PBAMs referenced in the introduction utilize similar idealizations to drive both formula- and finite element-based analyses.

Figure 18 is an "instance view" (Peak et al. 1995) of the PBAM, depicting the usage of an instance of the `component_extensional_model` class, with specific inputs and resulting intermediate values and outputs. This PBAM has been implemented as a C++ class whose protocol provides member functions to input the component, the reference temperature, the associated temperature (analysis inputs) and to obtain the temperature change, the total elongation and the associated strain (analysis results). When an instance of the PBAM is created, an instance of `elementary_rod` is also created and assigned as the value of the attribute `deformation_model` of the PBAM. Upon creation, the attributes of `elementary_rod` are initialized and the analysis results sent back to the PBAM according to the rules defined in the semantic mappings and the associativity linkages. Therefore, *"using"* a PBAM means sending messages to an instance of the PBAM class in order to set analysis variables and get analysis results. For example, as shown in Figure 18, we can input an instance of `component` (RES100), the reference temperature (20°C), the associated temperature (120°C) and get the temperature change (100°C), the total elongation (0.00213 mm) and the associated strain (6.7E-4). Intermediate values shown are "Alumina" as the primary structural material of the component, 6.7E-6 (mm/mm)/°C as its coefficient of thermal expansion, and

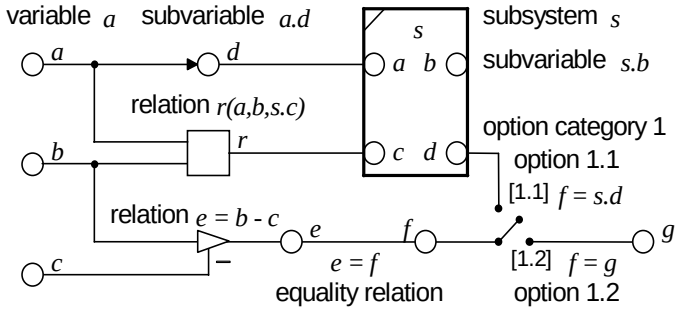


Figure 17: Constraint Schematics Notation

3.175 mm as the length of the bounding box of the component's package.

5. DISCUSSION

5.1. Need for intermediate representations

The technique presented introduces several intermediate representations and transformations of product data intended to bridge the gap between design and analysis. The main drive of this technique is to provide maximum flexibility and minimize the number of customized translators needed. For this purpose, we have stressed modularity and adopted a neutral product representation such as STEP.

The reason for these intermediate representations and transformations may not be obvious at first, but a closer look will reveal the advantages of implementing this kind of modular technique. Perhaps the best way to understand the importance of each intermediate representation is by examining what would the consequences be of eliminating it. Let us begin by eliminating both the intermediate STEP and AOPM representations (Figure 2). Analysis applications would have to read their data directly from design files D_1 to D_N. These files have, in general, data structures and file formats that differ widely. Therefore, we would need vendor-to-vendor translators to convert the data in the design files into a format that can be understood by the analysis applications. If we replaced any application with another from a different vendor, we would need a different translator for each application with which it exchanges data. Additionally, idealizations and simplifications would have to be performed within the code of each analysis application, making it more difficult to reuse those idealizations and simplifications that are common to more than one analysis.

Suppose now that we only eliminate the AOPM. Analysis applications would now have to read their data directly from STEP files D_1 to D_N. By introducing the intermediate STEP representation we are eliminating the need of specific vendor-to-vendor translators (as long as the design applications have translators to a common STEP representation). However, these STEP files will conform, in general, to STEP Application Protocols, which could define the structure of the data in a way that may not be the most convenient for the analysis applications at hand. This could make the analysis code unnecessarily complicated and difficult to maintain, and idealizations and simplifications would still have to be performed within the code of each analysis application. If there was a change in any of the STEP schemas (or a new schema was added to the picture), all the analysis applications would have to be modified to accommodate this change.

Finally, let us consider the case when we eliminate the intermediate STEP representation. In this case the mapping tool would have to map the data directly from the design files D_1 to D_N in order to populate the AOPM. This mapping is complicated by the fact that the design files have different data structures and formats. Although now the analysis applications are isolated from changes in the design applications, the mapping is not. The mapping code will have to be modified if there is any change in the design applications or in the formats of their files.

5.2. Advantages of the technique

In addition to the advantages presented throughout the paper and to solving the problems presented in the previous subsection, this technique has the following advantages:

- If the idealizations needed by a given analysis application

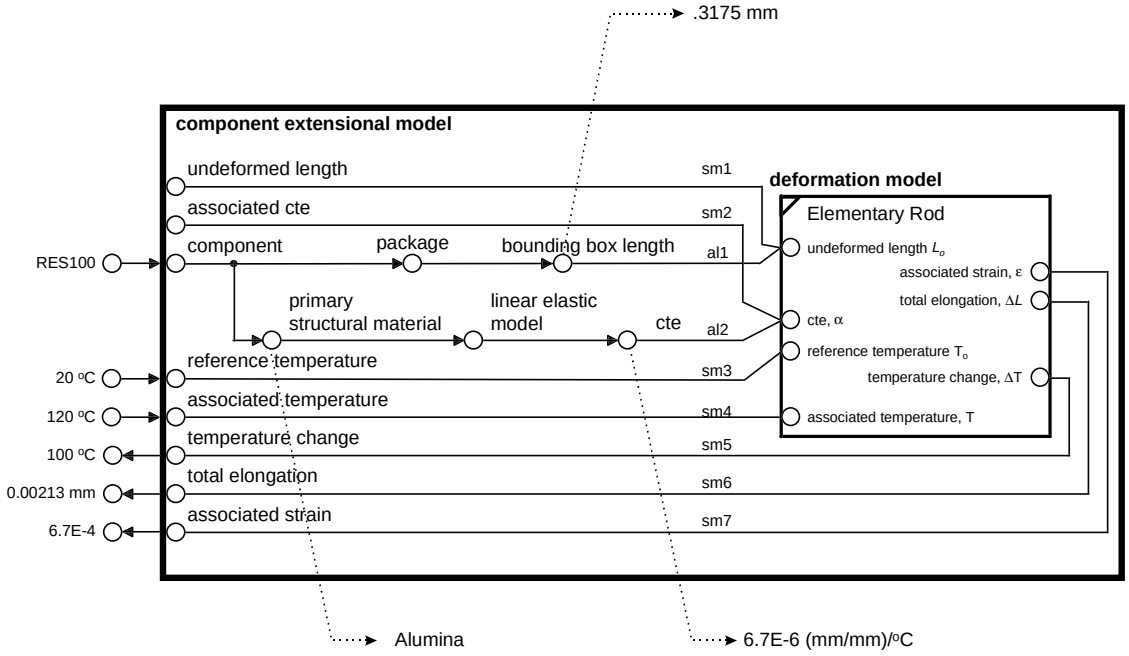


Figure 18: PBAM Schematic Instance View

are already supported by the AOPM, we may not have to add anything new to the AOPM in order to integrate this analysis application with the others. As more analysis applications are integrated, more reusable idealizations will be identified and incorporated to the AOPM.

- The technique provides a means to describe product information not supported by any existing Application Protocol. Since an Application Protocol is, in essence, an EXPRESS schema, we are can always create a new (non-standard) schema for the non-supported data (as long as we have a way to populate it with data) and integrate it with the rest of the schemas.
- Once the analysts have agreed with the common AOPM that will support their information needs, the analysis applications and the mapping tool may be developed simultaneously.

5.3. Future direction

The following are areas related to this work that we consider need future attention:

- Use actual STEP Application Protocols such as AP210 (for the definition of the PWA and its components) and AP203 (for the description of their geometry) as the schemas for the source data (for this work, we created two small EXPRESS schemas for the source data: `components.exp` and `materials.exp`). This would test the technique under more realistic conditions.
- Provide support for synthesis as well as for idealizations.
- Provide a mechanism to add missing data (data that is not populated by any design tool) that is needed for analysis. This is specially important when the missing data is precisely a key attribute to perform the mapping.
- Experiment with more complicated idealizations (such as total board thickness, diagonal board length, PWB warpage properties, etc.).

6. SUMMARY

In this paper we proposed an approach to assemble product data from several heterogeneous sources in order to support engineering analysis. We called this approach the Analysis-Oriented Product Model Population Technique. This technique consisted of mapping the data generated by the different design tools to populate an analysis-oriented view called the Analysis-Oriented Product Model (AOPM). The *populated* AOPM was called Analysis-Oriented Product Database (AOPD). The mapping rules were described using the EXPRESS-V mapping language. The AOPM provides support for the information needs of the analysis applications, including support for engineering idealizations. A simple example case was used to illustrate the technique. In this example, we generated an integrated AOPD of electrical components and material properties from separate design files. We also showed how some simple idealizations were supported and implemented. Finally, we provided an example of how entities and idealizations defined in the AOPM were used by a simple analysis application that calculates the elongation of the electrical components due to changes in temperature. By integrating this technique with the MRA architecture developed in previous research here at Georgia Tech, we provide a methodology to bridge the gap between design and analysis.

7. REFERENCES

Armstrong, C. G. (1994). Modelling Requirements for Finite-Element Analysis. Computer-Aided Design. **26:** 573-578.

Cimtalay, S., R. S. Peak, et al. (1996). "Optimization of Solder Joint Fatigue Life Using Product Model-Based Analysis Models." ASME Winter Annual Meeting **to appear**.

Engelmaier, W. (1983). "Fatigue Life of Leadless Chip Carrier Solder Joints During Power Cycling." IEEE Transactions on Components, Hybrids, and Manufacturing Technology **3**(6): 232-237.

Engelmaier, W. (1989). Thermal-Mechanical Effects. Electronics Material Handbook. M. L. Minges. Materials Park, OH, ASM International. **Volume 1 - Packaging:** 740-753.

Gadient, A. J. and L. E. Hines (1994). EXPRESS Driven Data Conversion. North Charleston, SC, South Carolina Research Authority.

Hardwick, M. (1994). Towards Integrated Product Databases Using Views. Troy, NY, Design and Manufacturing Institute, Rensselaer Polytechnic Institute (Report No. 94003).

ISO 10303-1 (1994). Industrial Automation Systems and Integration - Product Data Representation and Exchange - Part 1: Overview and Fundamental Principles. Geneva, Switzerland, International Organization for Standardization.

ISO 10303-11 (1994). Industrial Automation Systems and Integration - Product Data Representation and Exchange - Part 11: Description Methods: The EXPRESS Language Reference Manual, International Organization for Standardization.

ISO 10303-21 (1994). Industrial Automation Systems and Integration - Product Data Representation and Exchange - Part 21: Implementation methods: Clear text encoding of the exchange structure, International Organization for Standardization.

ISO 10303-203 (1994). Industrial Automation Systems and Integration - Product Data Representation and Exchange - Part 203, Application Protocol: Configuration Controlled Design. Geneva, Switzerland, International Organization for Standardization.

ISO DIS 10303-210 (1993). Industrial Automation Systems and Integration - Product Data Representation and Exchange - Part 210, Application Protocol: Printed Circuit Assembly Product Design Data. Geneva, Switzerland, International Organization for Standardization.

Kemper, A. H. and G. Moerkotte (1994). Object-oriented Database Management : Applications in Engineering and Computer Science. Englewood Cliffs, NJ, Prentice Hall.

Morris, K. C., M. J. Mitchell, et al. (1992). Database Management Systems in Engineering. Gaithersburg, MD, National Institute of Standards and Technology (Report No. NISTIR 4987).

Peak, R., S. (1993). Product-Based Analytical Models (PBAMs): A New Representation of Engineering Analysis Models

(Doctoral Thesis). School of Mechanical Engineering. Atlanta, Georgia Institute of Technology.

Peak, R. S. and R. E. Fulton (1993). "Automating Routine Analysis in Electronic Packaging Using Product Model-Based Analytical Models (PBAMs), Part II: Solder Joint Fatigue Case Studies." ASME Winter Annual Meeting **93-WA/EEP-24**.

Peak, R. S., R. E. Fulton, et al. (1995). "Integrating Engineering Design and Analysis Using a Multi-Representation Approach." Engineering with Computers **to appear**.

Peak, R. S., A. Scholand, et al. (1996). "On the Routinization of Analysis for Physical Design." 1996 ASME International Mechanical Engineering Congress and Exposition **to appear**.

Schenck, D. and P. Wilson (1994). Information Modeling the EXPRESS Way. New York, NY, Oxford University Press.

Shephard, M. S., E. V. Korngold, et al. (1990). Design Systems Supporting Engineering Idealizations. Geometric Modeling for Product Engineering. M. J. Wozny, J. U. Turner and K. Preiss. North-Holland, Elsevier Science Publishers**:** 279-299.

Spooner, D. (1993). Creating a Product Database with ROSE. Troy, NY, Design and Manufacturing Institute, Rensselaer Polytechnic Institute (Report No. 93054).

Spooner, D. and M. Hardwick (1993). Using Persistent Object Technology to Support Concurrent Engineering Systems. Concurrent Engineering: Methodology and Applications. P. Gu and A. Kusiak. New York, Elsevier Science**:** 205-234.

Spooner, D., M. Hardwick, et al. (1995). The EXPRESS-V Language Manual (http://www.rdrc.rpi.edu). Troy, NY, Laboratory for Industrial Information Infrastructure, Rensselaer Polytechnic Institute.

TIGER Team (1995). Technical Development Plan for the TIGER Program. South Carolina Research Authority, N. Charleston, SC., prepared for the Department of Defense Advanced Research Projects Agency (Document Number: TR002.01.00).

Urban, S. D., J. J. Shah, et al. (1993). Engineering Data Management: Achieving Integration Through Database Technology. Computing & Control Engineering Journal**:** 119-126.

Zhou, W. X. (1996). Modularized & Parametric Modeling Methodology for Concurrent Mechanical Design of Electronic Packaging (Doctoral Thesis). School of Mechanical Engineering. Atlanta, Georgia Institute of Technology**:** 180.