Integrating Engineering Design and Analysis Using a Multi-Representation Approach[†]

Russell S. Peak [1,‡], **Robert E. Fulton** [2], **Ichirou Nishigaki** [3], **and Noriaki Okamoto** [4]

[1] Visiting Researcher, [3] Researcher, and [4] Chief Researcher
Hitachi Ltd., Mechanical Engineering Research Laboratory
502 Kandatsu, Tsuchiura, Ibaraki 300, Japan

[2] Professor
Georgia Institute of Technology, School of Mechanical Engineering
Atlanta, Georgia, 30332-0405, USA

Abstract With the present gap between CAD and CAE, designers are often hindered in their efforts to explore design alternatives and ensure product robustness. This paper describes the multi-representation architecture (MRA) - a design-analysis integration strategy that views CAD-CAE integration as an information-intensive mapping between design models and analysis models. The MRA divides this mapping into subproblems using four information representations: solution method models (SMMs), analysis building blocks (ABBs), product models (PMs), and product model-based analysis models (PBAMs). A key distinction is the explicit representation of design-analysis associativity as PM-ABB idealization linkages that are contained in PBAMs.

The MRA achieves flexibility by supporting different solution tools and design tools, and by accommodating analysis models of diverse discipline, complexity and solution method. Object and constraint graph techniques provide modularity and rich semantics.

Priority has been given to the class of problems termed *routine analysis* - the regular use of established analysis models in product design. Representative solder joint fatigue case studies demonstrate that the MRA enables highly automated routine analysis for mixed formula-based and finite element-based models. Accordingly, one can employ the MRA and associated methodology to create specialized CAE tools that utilize both design information and general purpose solution tools.

Keywords

- CAD-CAE integration
- design-analysis associativity
- idealization
- routine analysis
- constraint schematic
- multi-representation architecture (MRA)
- solution method model (SMM)
- analysis building block (ABB)
- product model (PM)
- product model-based analysis model (PBAM)

Nomenclature

MRA	multi-representation architecture
SMM	solution method model
ABB	analysis building block
PM	product model
PBAM	product model-based analysis model
Ψ	ABB-SMM transformation
Γ	idealization relation between design and analysis attributes
Φ	PM-ABB associativity linkage indicating usage of one or more Γ_i

[†] Reference: *Engineering with Computers* (1998) 14: 93-114. Original version submitted March 1995. This document is a February 1999 revision, which includes minor updates and formatting for web delivery.

[‡] Affiliation as of January 1996: Assistant Director, Engineering Information Systems Laboratory, Georgia Institute of Technology, Atlanta, Georgia 30332-0560, USA, *http://eislab.gatech.edu/*

1 Introduction

In today's product design process, a significant gap typically remains between computer aided design (CAD) and computer aided engineering (CAE). In an industry survey, Liker *et al.* [1] confirm this observation and identify 'an iterative and seamless link between CAD and CAE' as one of the 'unfulfilled promises of CAD'.

A recent survey of design-analysis integration practice and research highlights the following needs [2]:

- *General methodologies for automating routine analysis to support product design*: Methodologies are lacking for creating CAE systems that provide designers with product-specific tools while taking advantage of general purpose analysis tools.
- *Representation of design-analysis associativity.* Design-analysis integration requires capturing how a CAE model is related to a CAD model, both for creating the analysis model and for associating analysis results back with the design model.
- *Support for numerous diverse analysis models* for each product type. The same kind of product often has analysis models from a variety of engineering disciplines that involve different solution techniques. Even within the same discipline, analysis models of varying resolution and complexity can exist for the same analysis problem. The unifying factor among these numerous analysis models is the product itself. Hence, the product information used by these analysis models should ideally come from a common source to maintain consistency and support analysis automation.

Other objectives that an ideal design-analysis integration strategy should fulfill are described in the same work, including support for design and analysis tools from different vendors.

This paper describes an architecture aimed at meeting the above needs and discusses the extent of success to date. It overviews the components of this approach and proposes a methodology for creating automated routine analysis tools.

2 Multi-Representation Architecture Overview

Given the above needs, the gap between design and analysis models is considered too large for a single general integration bridge. While many aspects of engineering analysis are computation-intensive, this research views CAD-CAE integration as an information-intensive problem that requires engineering information management solutions. The **multi-representation architecture** (MRA) has been developed to address this problem by placing four information representations as stepping stones between the design and analysis tool extremes. Fig. 1 summarizes this multi-representation architecture using solder joint thermomechanical analysis as an example.

On the right extreme are **solution method models** (SMMs) representing analysis models in

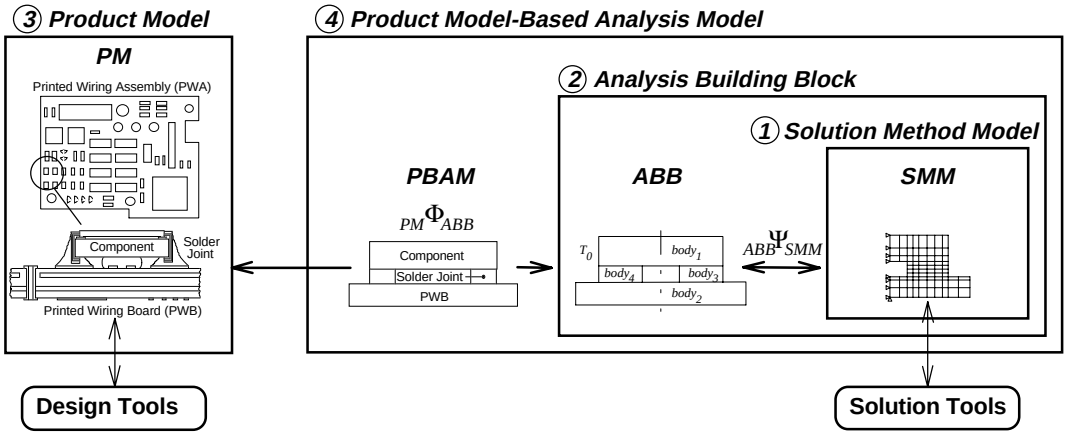


Fig. 1 The Multi-Representation Architecture for Design-Analysis Integration

relatively low-level, solution method-specific form. SMMs combine solution tool inputs, outputs, and control into a single information entity to facilitate automated solution tool access and results retrieval.

Analysis building blocks (ABBs) represent engineering analysis concepts in a manner that is largely independent of product application and solution method. ABBs obtain results by generating SMMs through transformations, ${}_{ABB}\Psi_{SMM}$, that are based on solution method considerations.

Skipping to the left extreme, **product models** (PMs) represent detailed, design-oriented product information. A PM is considered the master description of a product which supplies information to other product life cycle tasks, including engineering analysis and manufacturing. To enable usage by potentially many analysis applications, PMs in the MRA go beyond their traditional role and support **idealizations** that relate detailed, design-oriented attributes with simplified, analysis-oriented attributes.

Finally, **product model-based analysis models** (PBAMs) contain linkages that represent design-analysis associativity between PMs and ABBs, ${}_{PM}\Phi_{ABB}$. These **associativity linkages** indicate the *usage* of idealizations for a particular analysis application.

From the MRA viewpoint, providing solutions to the design-analysis integration problem involves defining these four representations (SMMs, ABBs, PMs, and PBAMs) and two inter-representation mappings (${}_{ABB}\Psi_{SMM}$ and ${}_{PM}\Phi_{ABB}$). The following sections describe these components and how together they provide flexible, modular analysis capabilities to support product design. Solder joint analysis examples from a prototype implementation of the MRA are included. More detailed descriptions of the ABB and PBAM representations and these case studies are available [2, 3, 4, 5].

3 Solution Method Models

In an abstracted view of an engineering solution process (Fig. 2), model data and tool control are input to a solution tool, and results are subsequently output. These solution tools are generally computational software programs such as commercial finite element analysis (FEA) tools, symbolic equation solvers, and proprietary specialized codes. In this paper FEA tools provide specific examples for the concepts discussed.

Fig. 2 Typical Engineering Solution Process

Solution tools in use today typically accept inputs and produce outputs in the form of user actions, computer files and graphics. In the FEA solution process of Fig. 3, for example, the postprocessor tool takes postprocessor control and solved mesh model files as inputs, and produces graphics and files of processed results as outputs. Extrema, such as maximum displacements and stresses, are a type of processed result that often interest product designers.

Note that this FEA solution process involves a sequence of tools that each play a distinct role in the overall solution process. The files consumed and generated for a single analysis problem often have no explicitly stated relationship, except possibly in the notes of the engineering analyst.

3.1 Information Content

With the automation of such solution processes in mind, a **solution method model** is defined to be the information entity that wraps these tool inputs and outputs into a single logical package. In the case of FEA, an SMM is not just the preprocessor input file; it also includes files that control the solution tool, as well as the results themselves. After execution, an SMM instance contains the following information at a minimum:

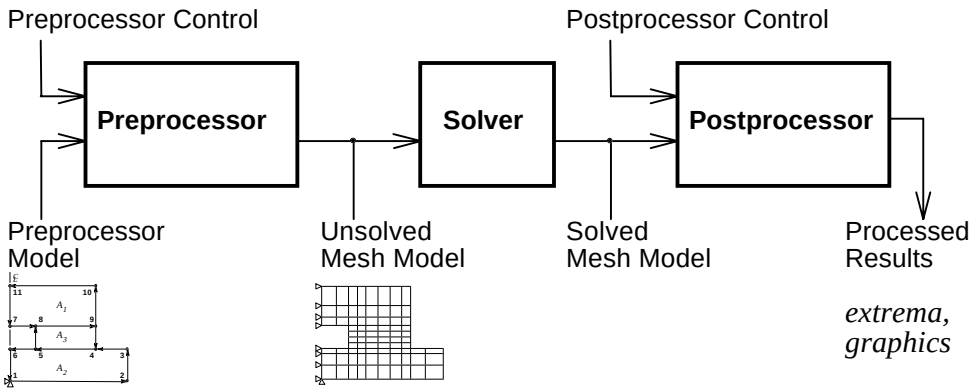


Fig. 3 Typical FEA Process

- *Analysis results* that fulfill the analysis problem purpose.
- *Initial inputs* that produce these results (e.g. preprocessor model, preprocessor control, and postprocessor control).

Thus, an SMM instance need not store all intermediate results, depending on the archival needs and results requirements of the current analysis problem. The log of each tool can be considered an analysis result and included in the SMM instance, if desired, to check for analysis errors and record solution statistics.

3.2 Tool Agents

For flexibility in automating solution processes like Fig. 3, SMMs use **tool agents** that serve as automated tool wrappers. Tool agents perform the following tasks partially depicted in Fig. 4 for the case of FEA:

1. *Determine which solution tool instances to use*. For a given SMM instance, these may reside on different machines.
2. *Provide solution tool inputs.* Create input files based on initial inputs. Transfer these files and intermediate tool outputs to the appropriate solution tool.
3. *Run each tool*. Show tool progress, including graphics output.
4. *Retrieve results.* Load results into the SMM instance, which then parses the results to extract salient information (e.g. stress extrema).

While the SMM instance represents the analysis model itself, the tool agent manages the operation of site-specific tools to solve this model.

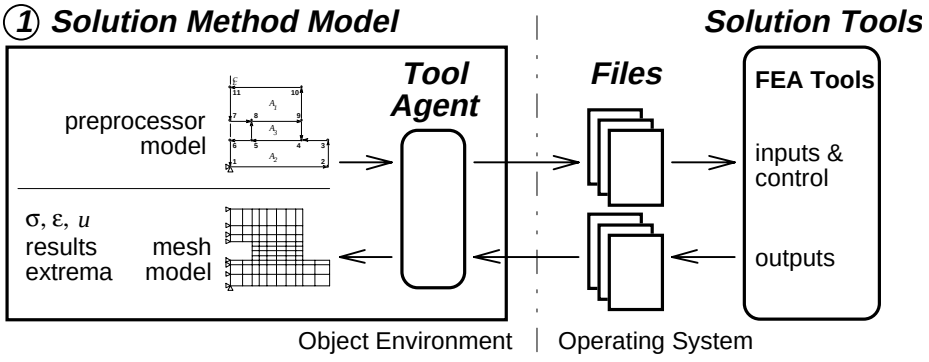


Fig. 4 Automated Tool Operation via SMMs and Tool Agents

3.3 Example SMMs

SMMs for Ansys [6] and Cadas [7] (two representative FEA tools) have been implemented as object classes to demonstrate these concepts. Fig. 5 shows an Ansys SMM screen which closely resembles the analysis process in Fig. 3. The entity that automatically creates an SMM instance and receives its results (known as its **context**) is usually an ABB. In such cases users can employ this kind of screen to simply view the SMM details if desired. Alternately, a user can play the role of SMM context and enter the preprocessor model and postprocessor control commands directly via such screens. He or she can then submit the job for automatic execution and results retrieval.

In the most basic case, a tool agent takes initial inputs from an SMM instance, combines them into a single file, and invokes the solution tool. This situation is possible when all solution functions are invoked as a single program on a single machine. The value of automated tool agents becomes more apparent in complex computing environments in which solution functions are divided among distinct tools located on different machines. A Cadas SMM instance demonstrates this case, for example, with distinct pre- and postprocessor tools running on a high-end graphics workstation, and various solvers available on several high performance computers. In such cases the tool agent automatically selects tools (or follows user selections), places SMM initial inputs into separate files, and sequences file transfers and tool executions on appropriate machines. Intermediate translations between solver and pre/postprocessor formats are automatically performed.

Experience to date indicates FEA solution processes can be automated to a high degree via SMMs and tool agents. Tool agents heavily utilize Unix remote shell and X Windows remote display capabilities to automate file transfers, tool execution, and graphics display. If desired, a user can temporarily suspend these automated actions and manipulate the tools via their standard interactive facilities (e.g. to further examine an automatically created mesh). Because existing solution tools typically have not been developed with wrapping in mind, some of their capabilities are difficult to represent in the

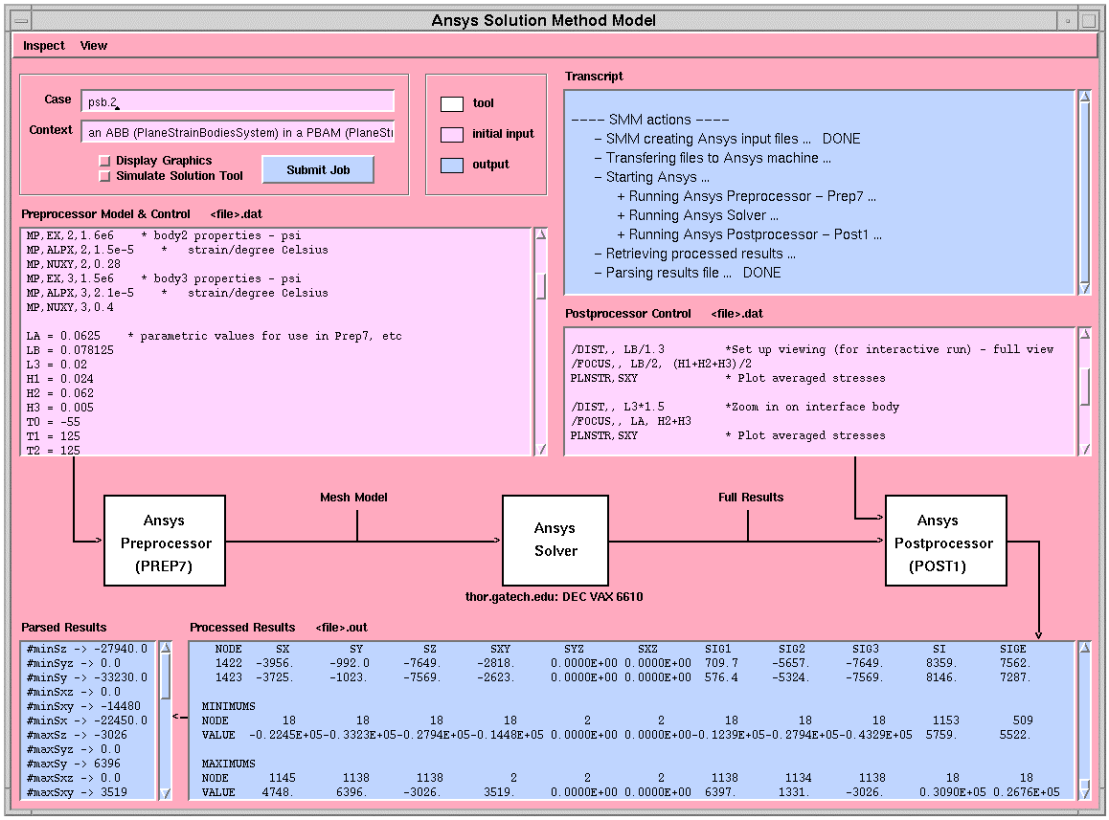


Fig. 5 Prototype Ansys SMM

tool control portion of SMMs. In such cases the user must perform some nonautomatable actions, ranging from pushing a button to entering several commands. Stephens [8] describes engineering tool wrapping strategies in more detail and reports similar experiences.

4 Analysis Building Blocks

Analysis building blocks (ABBs) represent engineering analysis concepts as computable information entities that (a) include engineering semantics, and (b) are largely independent of solution method. Table 1 summarizes categories of ABBs described in this paper. Of the two major types, general purpose ABBs represent analysis concepts independent of a specific product design application, whereas PBAMs explicitly include design-analysis associativity.

Table 1 Classification of Analysis Building Blocks

ABB Type	Description
General Purpose ABB	- ABB with no product associativity
Analysis Primitive	- Basic ABB used in other ABBs
Analysis System	- Container joining primitives and/or other analysis systems
General	- User-defined instance
Specialized	- Predefined template/class
PBAM	- Design-analysis associativity container joining PMs & ABBs
General	- User-defined instance
Specialized	- Predefined template/class

Fig. 6 exemplifies major categories of general purpose ABBs, which have been influenced by the work of Rosenberg and Karnopp [9], Ingrim and Masada [10], STEP Parts 41 and 105 [11], Mashburn and Anderson [12], and Stephens [8]. **Analysis primitives** represent basic engineering concepts as entities containing semantic groupings of analysis attributes and relations. For example, a simple spring primitive includes the attributes force, F, total elongation, ΔL, and spring constant, k, as well as the relation $F = k\Delta L$.

Analysis systems are containers of other general purpose ABBs that have been assembled together to form an engineering analysis model. For example, Fig. 6 shows a cantilever beam system as an assembly of beam, rigid support, and distributed load primitives.

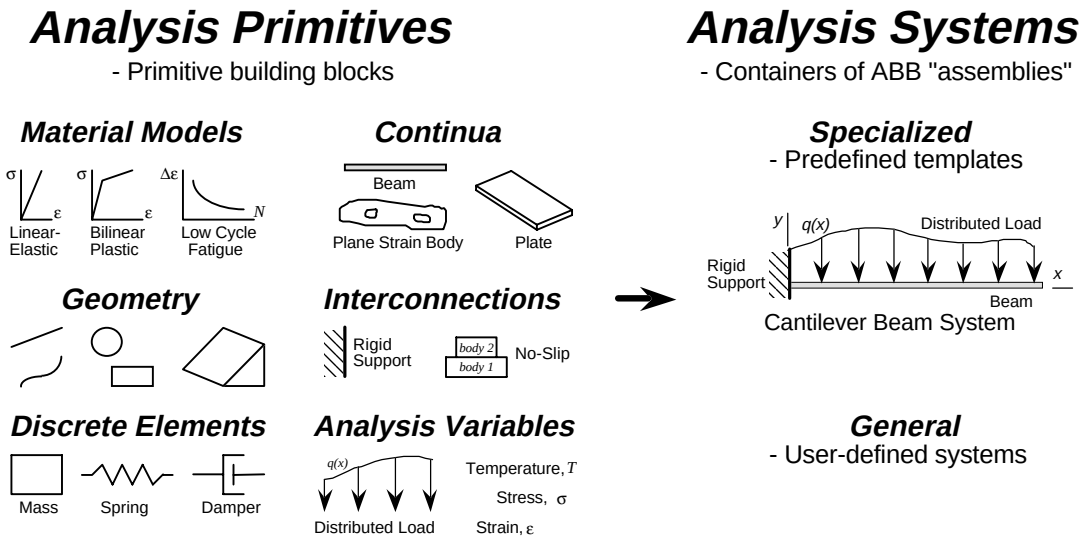


Fig. 6 Categories of General Purpose ABBs

4.1 The ABB Representation

Possessing defined structure and operations, the ABB representation combines constraint graph techniques [13, 14, 15, 16, 17] and object-oriented principles [18, 19, 20] to represent analysis concepts. Other engineering applications of constraints include DC motors by Rinderle and Colburn [21], assembly modeling by Gui and Mäntylä [22] and geometric modeling by Solano and Brunet [23]. Benefits of objects to engineering have been discussed using examples such as column primitives by Forde *et al.* [24], as well as finite elements and matrices by Fenves [25], Filho and Devloo [26], and Lee and Aurora [27].

Within the ABB representation, the **ABB structure** is the information template for defining data structure and relations. A specific type of ABB (e.g. *Linear-Elastic Material Model*) is defined by a populated ABB structure and can be implemented as an object class. Fig. 7 shows graphical and tabular information views of the ABB structure, and Table 2 summarizes the purpose of each view. Just as STEP EXPRESS-G [11] provides a graphical view of a lexical EXPRESS model, these views emphasize subsets

Table 2 ABB Representation Views

Structural View		Description
ABB Structure	-	Information template in lexical form (master view)
Constraint Schematic	-	Graphical view emphasizing relations among ABB variables
Object Relationship Diagram	-	Graphical view emphasizing ABB *part-of* and *is-a* relations. Based on EXPRESS-G [11]
Subsystem *	-	Encapsulated view used in other constraint schematics to show occurrence of ABB
Extended Constraint Graph *	-	Decomposed view of ABB constraint schematic. Helps trace relations among variables
I/O Table *	-	Explicit view of available input/output combinations
Instance View		Description
Graph Instance View *	-	Annotated extended constraint graph showing usage of ABB instance
Schematic Instance View *	-	Similarly annotated constraint schematic
Subsystem Instance View *	-	Similarly annotated subsystem view

*ABBs may have more than one of this type of view.

of a populated ABB structure for easier human comprehension [2]. For example, **object relationship diagrams** emphasize ABB attributes and ABB hierarchies. The ABB structure is the master view, having all the content of the other views but in lexical form. For example, the notation *a.d* indicates *d* is an

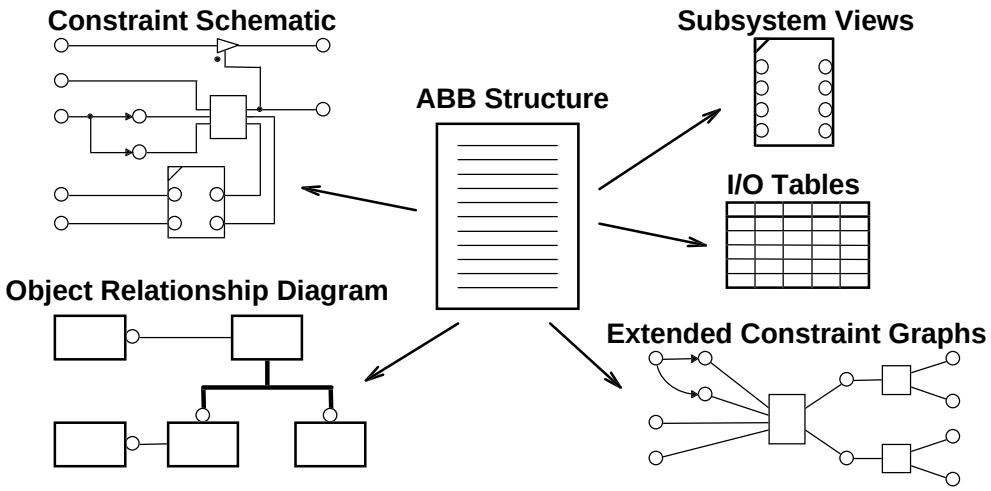


Fig. 7 Structural Views in the ABB Representation

attribute of *a*, denoting the *variable-subvariable* relationship (a.k.a. the *part-of* relationship).

A new notation called **constraint schematics** [2] graphically emphasizes relationships among objects. In this notation (Fig. 8) a variable can be a simple object like a number, or it can be a complex object having attributes that are themselves complex objects. One strength of this notation is depicting relations among complex objects (analogous to electrical schematics showing connections between complex integrated circuits). This capability is achieved by graphically showing *variable-subvariable* relations, by abstracting complex objects as subsystems (analogous to integrated circuits), and by supporting hierarchical nesting of other constraint schematics within these subsystems. An option category indicates alternate subgraphs within a constraint schematic (e.g. the value of *f* depends on option category 1 in Fig. 8, where $f = s.d$ for option 1.1, while $f = g$ for option 1.2).

The ABB structural views are analogous to flow charts in procedural programming, as they are largely independent of a specific computer implementation. Similarly, they aid the development, implementation, documentation and usage of ABBs and PBAMs [2]. Table 2 also summarizes several kinds of **instance views** which depict the usage of an ABB instance by showing specific input and output values.

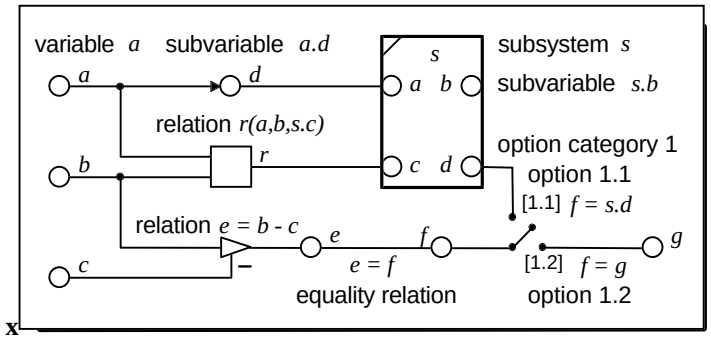


Fig. 8 Basic Constraint Schematic Notation

4.2 Example Analysis Primitive

To illustrate some of these information views, a spring is modeled as a simple analysis primitive called *Elementary Spring* in Fig. 9(a)-9(d). The ABB structure, Fig. 9(a), defines the class of each variable in this
primitive and the relations among these variables. The constraint schematic, Fig. 9(c), graphically depicts these relations, using the convention that the appropriate mathematical operator is shown beside the triangular ternary relation symbol. The figure also includes one possible subsystem view, Fig. 9(d). Fig. 10 shows the usage of an instance of *Elementary Spring* via a schematic instance view, where L and ΔL are outputs calculated from inputs F, k, and L_O.

Elementary Spring

superclass: Discrete Primitive

variables

undeformed length, L_o	*: Distance*
spring constant, k	*: Spring Constant*
start, x_1	*: Point*
end, x_2	*: Point*
length, L	*: Distance*
total elongation, ΔL	*: Distance*
force, F	*: Force*

subsystems

<none>

semantic linkages

<none>

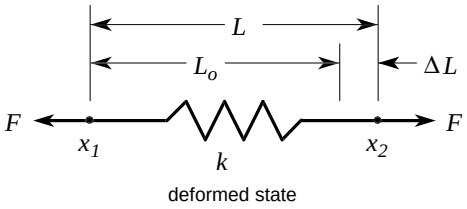
relations

$$L = x_2 - x_1$$

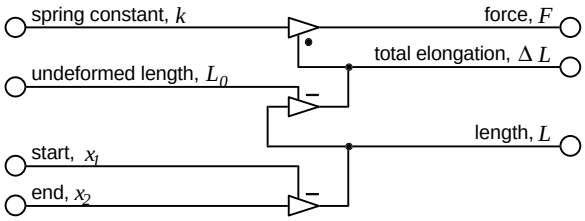
$$\Delta L = L - L_o$$

$$F = k\Delta L$$

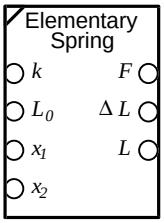
a. ABB Structure



b. Analysis Primitive Figure



c. Constraint Schematic



d. One Subsystem View

Fig. 9 ABB Views of a Spring Primitive

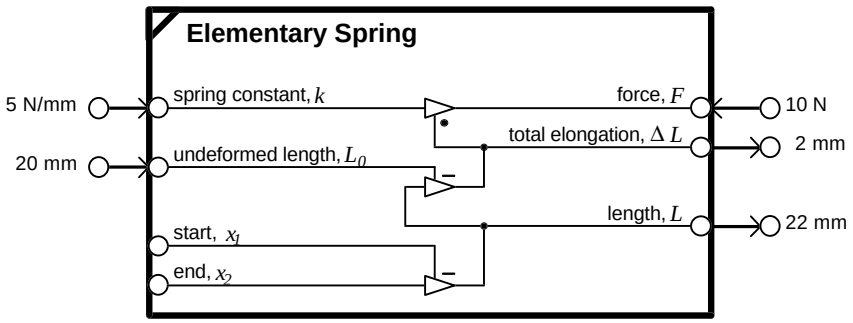
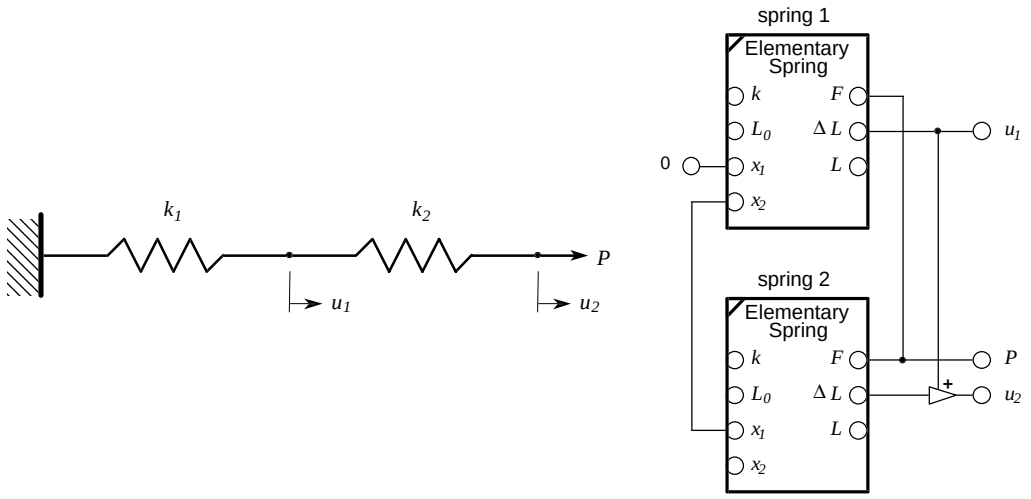


Fig. 10 Example Schematic Instance View



a. Analysis System Figure **b.** Constraint Schematic

Fig. 11 Example Analysis System

4.3 Example Analysis Systems

The usefulness of constraint schematics becomes more apparent when ABBs like the *Elementary Spring* are combined to represent more complex analysis models. The analysis system in Fig. 11 is a simple example constructed by connecting two instances of *Elementary Spring* using force and kinematic boundary conditions (e.g., *spring2.F = P* and *spring1.*x_2 = *spring2.*x_1). The subsystem view in Fig. 9(d) occurs twice in this schematic to indicate the presence of these two spring instances. All the inter-primitive construction is done here using equality relations (Fig. 8) instead of interconnection primitives for simplified illustration. Three system-level variables, P, u_1 and u_2, are defined as functions of subsystem attributes (e.g. u_2 = *spring2.*$\Delta L + u_1$). Semantic mappings also exist so that internal subsystem variables can be accessed by their system level names, (e.g. k_2 = *spring2.k*). A semantic mapping generally is not shown unless the variable name in the system scope differs from what would be expected.

After an instance of this analysis system is created, the internal relations of each spring instance take effect and contribute to the overall system behavior. The constraint schematic view of an analysis system hides primitive-level details, enabling one to concentrate on system-level behavior without losing the effect of internal details. Furthermore, analysis systems can be used as components in other ABBs, and so on, thus characterizing the building block nature of ABBs.

A **specialized analysis system** class is a template representing a set of ABB assemblies with predefined configuration possibilities. For example, the *Plane Strain Bodies System* implementation in Fig. 12 is a specialized analysis system representing a limited class of thermomechanical behavior for symmetric multimaterial structures. Instances of this specialized analysis system class are constructed using instances of the *Plane Strain Body* primitive, which itself has attributes for body geometry and material model defined by instances of other analysis primitives.

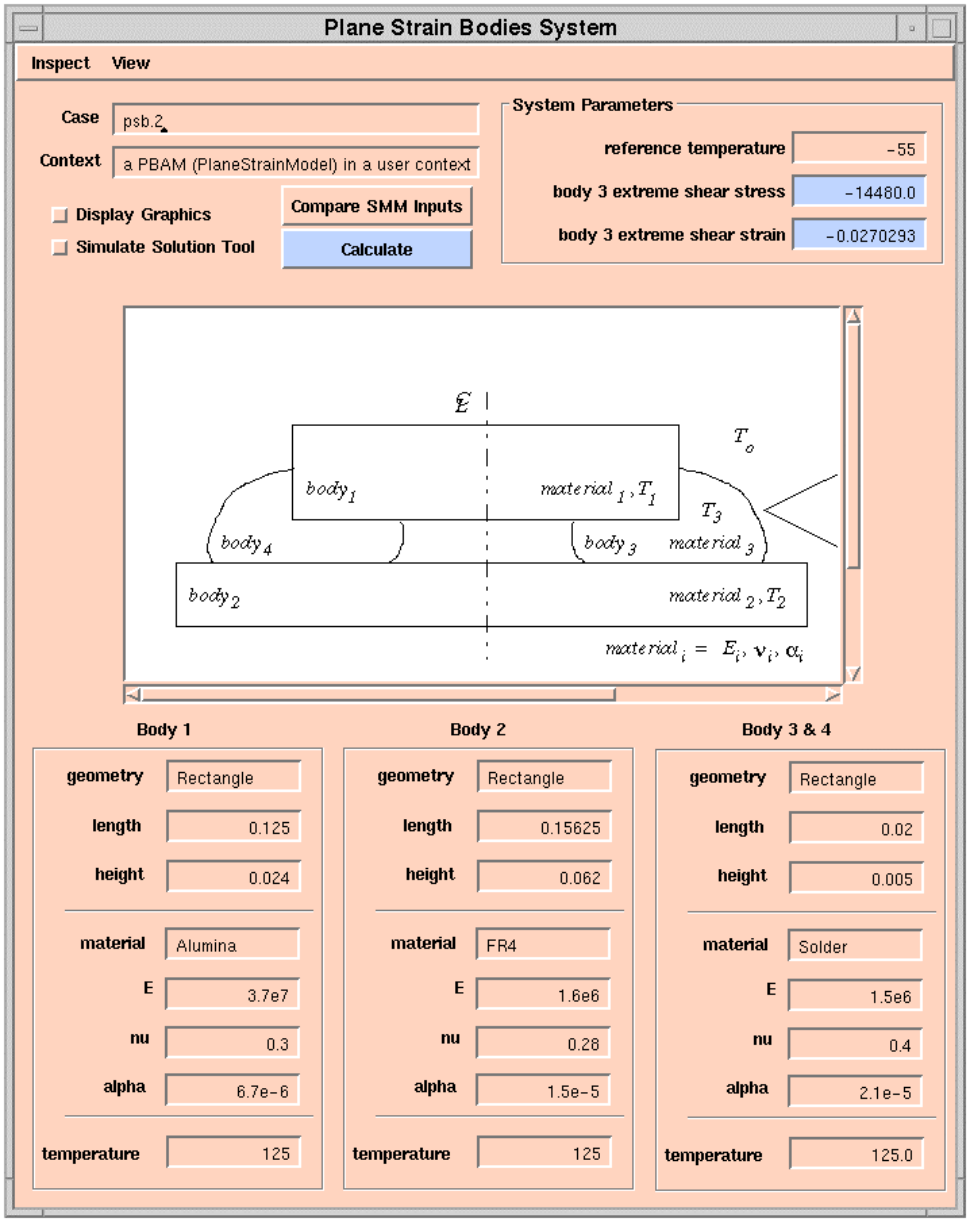


Fig. 12 Implementation of a Specialized Analysis System

As with SMMs, the context of an ABB instance indicates who is directly utilizing it. In this figure the context field indicates a *Plane Strain Model* PBAM for solder joint analysis created this instance. In the MRA a PBAM or another ABB is typically the context for general purpose ABB instances. When users are the context, they can directly input attribute values and interact with analysis system instances via screens such as Fig. 12.

Alternatively, users can create an equivalent instance using a modeling language approach (Fig. 13). Per Smalltalk syntax, *new* signifies the creation of an instance of the indicated class [18]. Attributes are assigned object values in a top-down manner, e.g., such that system $body_1$ is a plane strain body instance whose geometry is a 0.125 x 0.024" rectangle. Material models are assigned by single high level statements indicating the intended model type, rather than by specifying multiple disjoint property values.

```
PlaneStrainBodiesSystem new
    referenceTemperature: -55;
    body1: (PlaneStrainBody new
        geometry: (Rectangle length: 0.125 height: 0.024);
        stressStrainModel: (Alumina asLinearElasticModel);
        temperature: 125);
    body2: (PlaneStrainBody new
        geometry: (Rectangle length: 0.15625 height: 0.062);
        stressStrainModel: (FR4 asLinearElasticModel);
        temperature: 125);
    body3: (PlaneStrainBody new
        geometry: (Rectangle length: 0.020 height: 0.005);
        stressStrainModel: (Solder asLinearElasticModel);
        temperature: 125).
```

Fig. 13 Semantically Rich User Creation of an ABB Instance

This specialized analysis system also supports a predefined variety of body shapes and material models, which translates into predefined topological variations in the underlying constraint graph [2]. Though initially developed for use in solder joint analysis, this specialized analysis system class is defined in product-independent terms, as are all general purpose ABBs. Hence, it can possibly be used as-is or extended to analyze other products with similarly configured multimaterial structures.

4.4 Solving ABBs Using SMMs

After being created as above, ABB instances use transformations, ${}_{ABB}\Psi_{SMM}$, to associate themselves with SMM instances and obtain analysis results. Fig. 14 illustrates this concept for the above specialized analysis system with an FEA-based SMM. The forward form of ${}_{ABB}\Psi_{SMM}$ involves creation of the SMM instance and its initial inputs based on solution method considerations (e.g. creation of the preprocessor model considering symmetry, region decomposition, and mesh density). The inverse form of ${}_{ABB}\Psi_{SMM}$ takes results from the SMM instance and translates them back into terms that are relevant to the ABB instance. For example, this specialized analysis system knows which finite elements in the resulting mesh model are associated with its $body_3$, and thus takes the extreme shear stress among these elements to be the extreme shear stress in $body_3$. The limited scope of a specialized analysis system enables the highly automated execution of this process via modular mappings to parameterized preprocessor models.

Depending on the nature of an ABB, it may be solvable by different solution methods - each of which would have its own associated SMM class (Fig. 15). Because a variety of solution tools exist even for the same solution method (e.g. for FEA: Abaqus, Ansys, Cadas, Nastran, etc.), the same ABB instance can produce a corresponding variety of vendor-specific SMMs (Fig. 16).

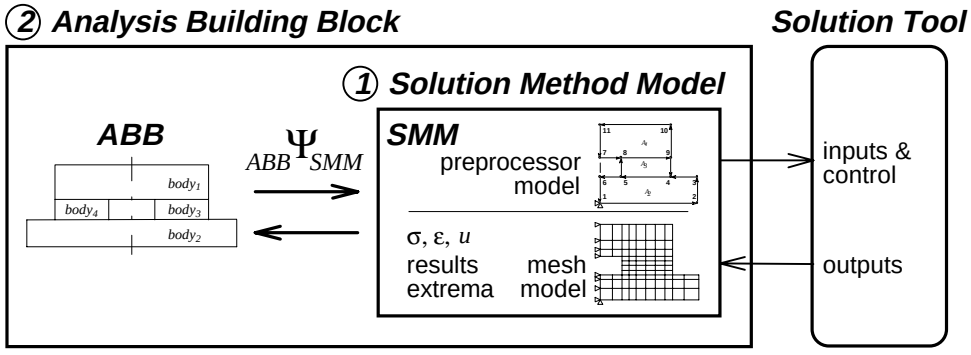


Fig. 14 Obtaining Analysis Results via an SMM

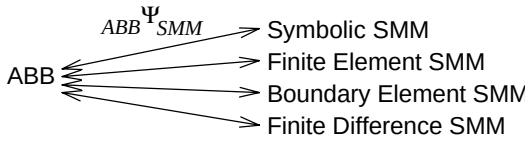


Fig. 15 ABB Support for Diverse Solution Methods

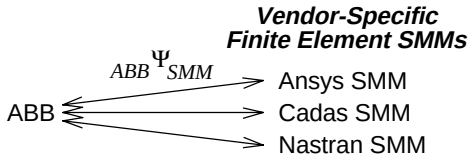


Fig. 16 ABB Support for Vendor-Specific SMMs

Overall, general purpose ABBs serve as product-independent analysis models which contain a higher degree of analysis intent than SMMs (as numerous types of SMM instances can be derived from the same ABB instance). An ABB can be thought of as a semantically rich combined 'pre-preprocessor' and 'post-postprocessor' model for traditional engineering solution tools.

5 Product Models

Much of the CAD/CAE effort in past years has focused on the geometric description of products. Recently attention has been given to **product models** (PMs) to also represent non-geometric product information, including material, assembly information, test specifications, cost, and versioning [28].

In broad terms, a product model is a representation of a product (e.g., a physical system, an assembly, or a part) that contains life cycle information - the information used by all parties associated with a product, including design, analysis, manufacturing, marketing, installation, and repair. Though such an omniscient PM is more a goal at present rather than industrial practice, STEP (STandard for the Exchange of Product model data) is one effort aimed at realizing this goal [11, 29, 30, 31].

PMs support the information needs of design tools by defining a neutral data structure that facilitates access to a common product database (Fig. 17). Tools can share information with increased consistency and reduced redundancy via this common database approach. In terms of the database management three schema architecture [32], the neutral data structure may be used as the conceptual schema in product database management systems. Often, however, it is an external schema supported by the database management systems and design applications, since these systems may have optimized internal schemas [33].

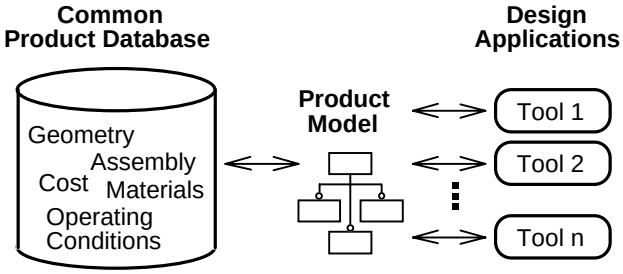


Fig. 17 Common Resource of Design Information

5.1 Purpose in the MRA

In the design-analysis integration context of the MRA, the term product model is used in a narrower sense to mean the representation of *design-oriented* information. This information is loosely defined as a combination of what STEP Part 210 terms the 'manufacturable description' [11] plus what the MRA terms the 'environmental description' - the conditions the product may experience during its life cycle, including manufacturing, storage and operating conditions. Analysis model geometry, connectivity, and material models are generally related to the former, while loads and boundary conditions are related to the latter. From a design verification viewpoint, this restricted definition emphasizes the difference between the independent information that describes a product and the dependent information derived during an analysis of that product. Hence, in this context design geometry and material information are considered part of the PM, while a finite element model is not.

Depending on the design process stage, the exchange of information between PMs and analysis models takes on a different emphasis. In the early stage of design synthesis, analysis models primarily provide inputs which design tools transform into descriptions in the PM. In the later stage of design verification, the principal information flow is reversed as the PM supplies the information needs of analysis models via idealizations. Furthermore, since the PM and analysis models act as both information servers and receivers to some degree in both stages, supporting their *bidirectional* associativity is important.

5.2 Example PWA Product Model

The simplified PWA[1] PM utilized in this research is partially given in Fig. 18 using adapted EXPRESS-G notation. Details of a representative PWA instance are shown in a prototype implementation of this PM (Fig. 19).

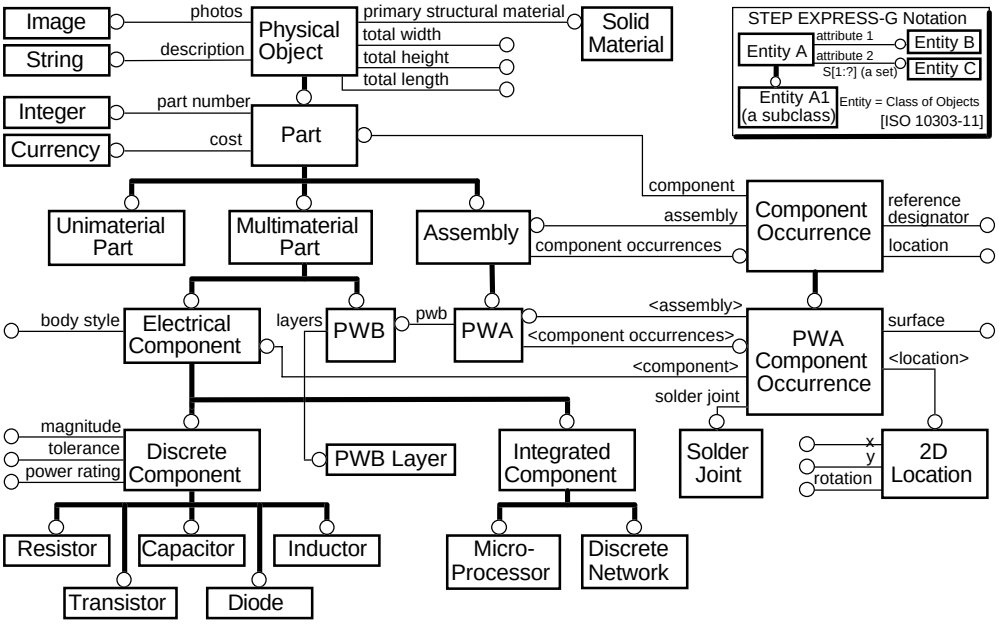


Fig. 18 Partial PWA Product Model

[1] PWA = printed wiring assembly (a circuit board with components)
PWB = printed wiring board (a bare circuit board)

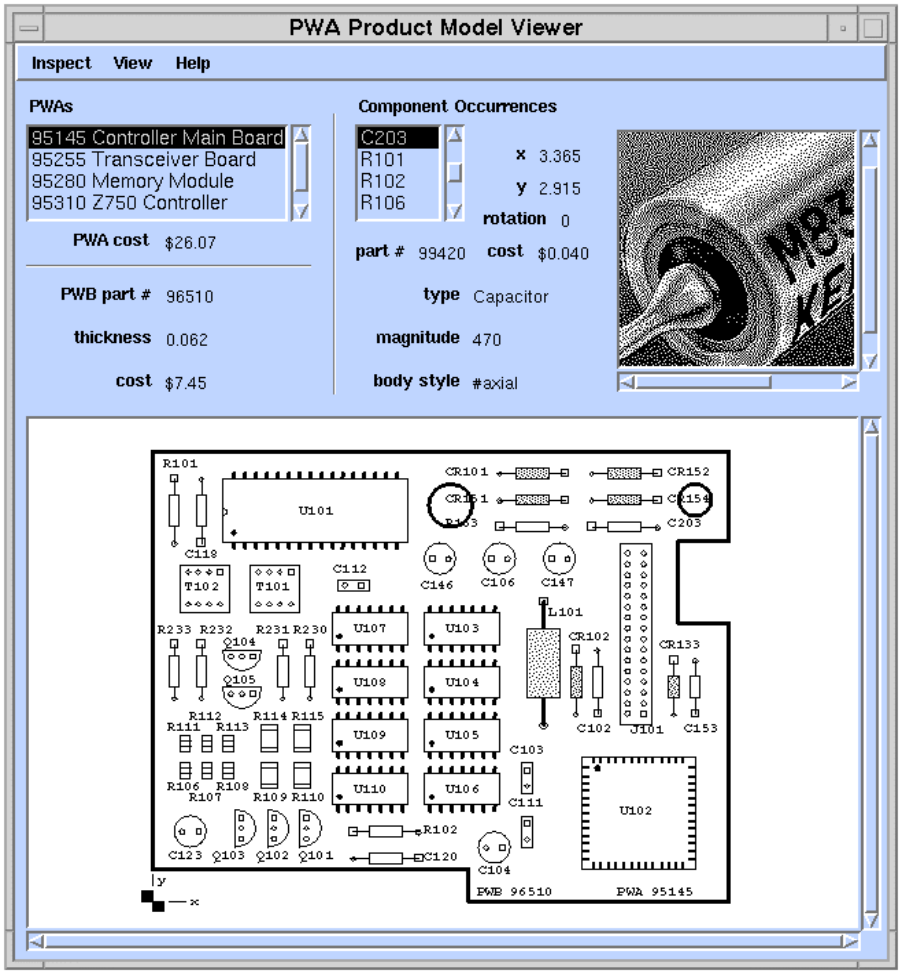


Fig. 19 PWA Product Model Implementation

In this PM, *Physical Object* is an abstract class representing physical objects. It includes a *photos* attribute (a set whose members are of type *Image)*, as well as idealization attributes described below. *Part* is a specialization of this entity that includes *part number* and *cost* in addition to the attributes it inherits from *Physical Object*. PWAs and other products are similarly represented as specialized subclasses of *Part*.

An instance of the class *Component Occurrence*, ω, represents the usage of a part as a component in another part (i.e., in an assembly). The class *PWA Component Occurrence* specializes this concept for PWAs and refers to a component-solder joint-PWB structure, ω_c. The inherited *component* attribute of this class represents an electrical device of a given part number (an *Electrical Component* instance) which may be used multiple times on the same PWA. The unique identifier for a component occurrence is a *reference designator* (e.g., U102) versus a *part number* for a component (e.g., 99120). In Fig. 19 some details of the selected component occurrence C203 are shown, including its x-y position on the PWB and a photo of its component, an axial capacitor.

5.3 Support for Idealizations

The MRA extends the traditional role of a PM beyond providing a "manufacturable description" of a product. An MRA PM also includes relations between these detailed, design-oriented attributes and simplified, analysis-oriented attributes to support the information needs of potentially many analysis

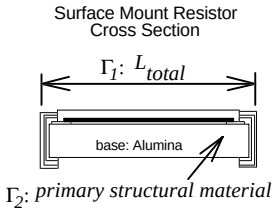


Fig. 20 Example Analysis Idealizations

models. The form of such a relation where an analysis attribute is the output is similar to what has been called an **idealization** [34, 35, 36].
Example idealizations (Fig. 20) include finding total dimensions, Γ_1, and determining the primary structural material in a multimaterial part, Γ_2.

Geometric Simplification: $\quad$ *total length,* $L_{total} = \Gamma_1(part)$ (1)

Composition Idealization: $\quad$ *primary structural material* $= \Gamma_2(part)$ (2)

Another example is a material idealization, Γ_3, which assumes that a physical material behaves in some prescribed manner. Several material models can be associated with the same physical material (e.g. linear-elastic and bilinear-plastic stress-strain models). Which material model is suitable for a given analysis model depends upon such factors as analysis purpose and parameter magnitudes.

Material Idealization: $\quad$ *linear-elastic model* $= \Gamma_3(material)$ (3)

Idealizations in the form of discrete relations and basic formulae have been implemented to date. For example, overall dimensions are typically included as attributes in electrical component databases - an implementation of discrete relations. Alternatively, total dimensions could be calculated from a detailed PM of the component itself if such information is available.

Table 3 shows some values of the discrete composition idealization, Γ_2, in which the primary structural material is stated explicitly based on expert knowledge. These rows are equivalent to expert system rules, e.g. the resistor row says:

if (a part is a resistor with a SMD body style)
then (the primary structural material = the material composing the resistor's base)

Note, however, that in an object implementation of a PM, such rules are more naturally implemented as polymorphic methods or attributes of the various part classes. Inferencing is then done automatically based on the class definition rather than via a distinct inference engine as in a rule-based expert system.

Table 3 Example Composition Idealization, Γ_2

part	body style [2]	primary structural material
Resistor	SMD	base.material
Microprocessor	LCC	case.material
PWB		core.material

[2] SMD = surface mount discrete, LCC = leadless chip carrier

Because each type of product typically requires multiple PBAMs, reusability is an important advantage of including idealizations in the PM representation. In other words, common idealizations like the examples above are used by potentially many types of products in many types of analysis models. Consequently, the PM definition of an idealization does not include associativity with any particular analysis model.

Object-oriented PM implementations readily accommodate the incremental addition of both general and product-specific idealizations by placing the former high in the class hierarchy and the latter lower. Idealizations that are likely to be used only in a particular kind of analysis model may be alternately defined in the associated PBAM.

6 Product Model-Based Analysis Models

Product model-based analysis models (PBAMs) [2] contain design-analysis associativity between PMs and ABBs, ${}_{PM}\Phi_{ABB}$. Individual **associativity linkages**, Φ_i, represent this relationship explicitly and indicate the *usage* of one or more idealizations, Γ_j, to form a particular analysis model. Typically a PBAM connects associativity linkages between a PM and a general purpose ABB to take advantage of the analysis capabilities of the latter. In other words, PBAMs connect PMs to product-independent ABBs in order to solve product-specific analysis problems.

6.1 Example PBAM

Fig. 21 illustrates these concepts via a solder joint analysis example (after Lau *et al.* [37]). Due to the coefficient of thermal expansion mismatch between the PWB and component, the solder joint deforms under thermal loads. The overall goal of this analysis model is to determine the resulting strain, which can later be used to estimate the solder joint fatigue life.

The left side of the figure shows design-related details of the PM entities: a component (in this case, a surface mount resistor), a PWB, solder joints, and epoxy. The assembly of these entities is another PM entity, a *PWA Component Occurrence*, ω_c, described earlier (Fig. 18). On the right, the ABB used as a subsystem is the specialized analysis system, *Plane Strain Bodies System*, presented previously (Fig. 12).

The PBAM, *Component Occurrence Plane Strain Model* (a.k.a. the *Plane Strain Model*), contains associativity linkages, Φ_i, which indicate how the component, PWB, and solder joints are modeled as four homogeneous plane strain bodies in the ABB subsystem. Linkage Φ_1 uses a geometric idealization, Γ_1, to specify a geometric dimension of ABB $body_1$ based on component geometry. Similarly, Φ_2 uses a composition idealization, Γ_2, to specify which component material to consider. It then uses a material idealization, Γ_3, to specify a model of this material for use in $body_1$. This combination of idealizations is typical for multimaterial parts like electrical components. These two linkages can be written textually as follows:

Φ_1: *body1.height,* h_1 *= component.total height,* h_c (4)

Φ_2: *body1.stress-strain model = component.primary structural material.linear-elastic model* (5)

Note that most of the PM details, including the epoxy, are neglected in this analysis model under the assumption that they would not significantly affect the analysis results.

From Table 1, the PBAM representation is a special case of the ABB representation which also includes variables and relations involving PM information. As a result, a specific type of PBAM is defined by its populated PBAM structure - a special form of the ABB structure, which, for example, categorizes variables as either analysis variables or product variables [2]. As such, PBAMs benefit from the defined structure, operations, and views of ABBs described previously. A key point is that design-analysis associativity linkages are represented explicitly in the same way as other relations.

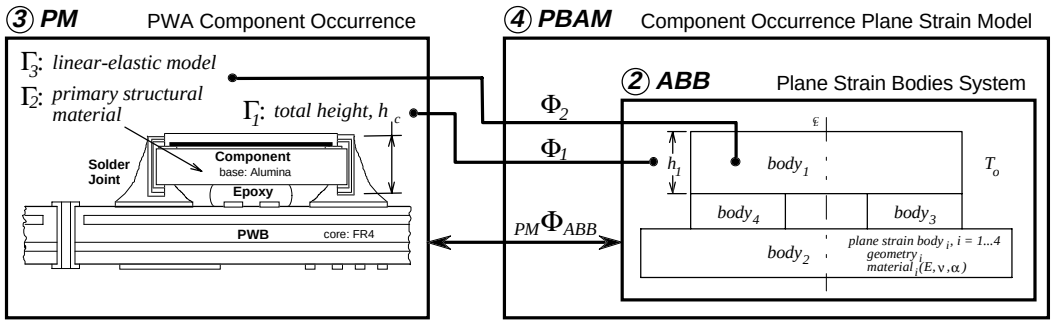


Fig. 21 Design-analysis associativity in a solder joint analysis model

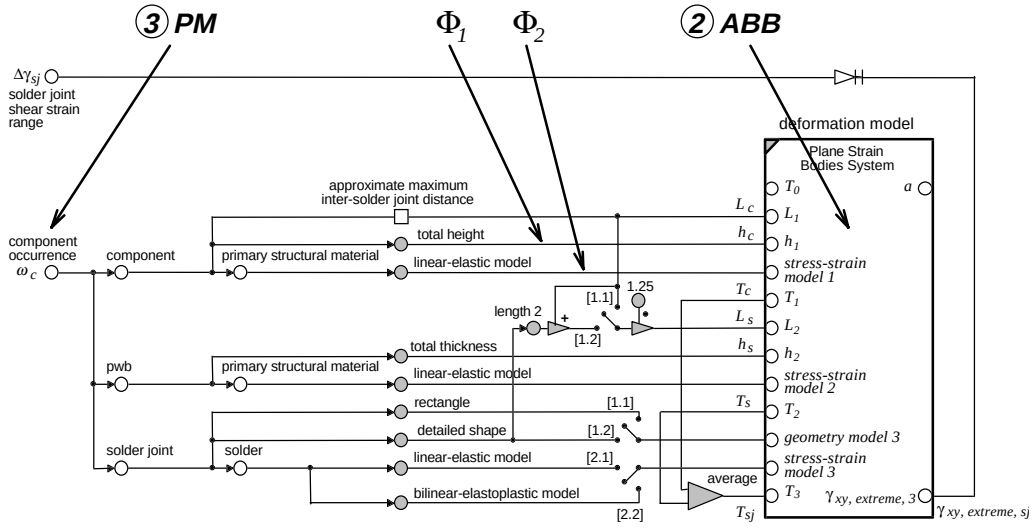


Fig. 22 PBAM Constraint Schematic for *Component Occurrence Plane Strain Model*

For example, Fig. 22 is the constraint schematic of the above PBAM; it is a structured information model of Fig. 21 that specifies all associativity linkages (including examples Φ_1 and Φ_2 from above), as well as product variables, analysis variables, a subsystem, and other relations. Hence, a variety of component occurrences, ω_C, from a variety of PWAs can be used as PM inputs to instances of this PBAM.

PBAMs can also include **analysis options** to support different degrees of idealization. For example, this type of PBAM supports rectangular and detailed solder joint geometry, as indicated in Fig. 22 by option switch positions [1.1] and [1.2], respectively. Likewise, both linear-elastic [2.1] and bilinear plastic [2.2] solder models are supported. Which options are chosen for a PBAM instance typically depends on the intended use of the analysis results and considerations of computational cost versus analysis accuracy.

6.2 Using PBAMs in Routine Analysis

A major focus of the MRA has been automating the class of problems termed **routine analysis** - *the regular use of established analysis models in product design* [2]. As overviewed later, one can use the MRA to develop and implement catalogs of specialized PBAMs that represent routine analysis models as predefined templates. Fig. 23 shows a catalog of solder joint analysis PBAMs based on models by solder joint researchers [38, 39, 40, 37]. These PBAMs can be used for highly automated routine analysis as in the design verification scenario illustrated.

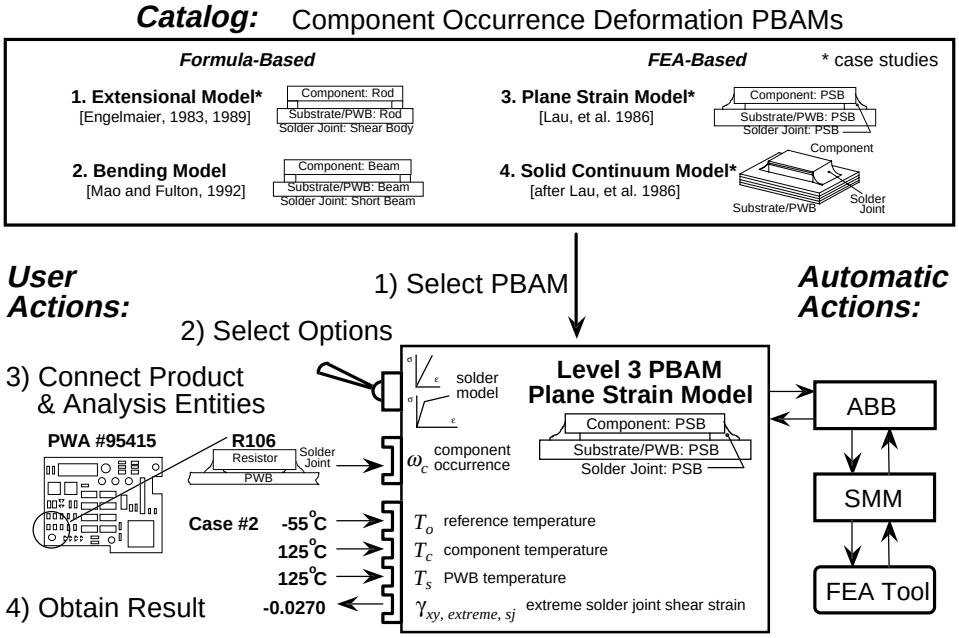


Fig. 23 Using PBAMs in Routine Analysis

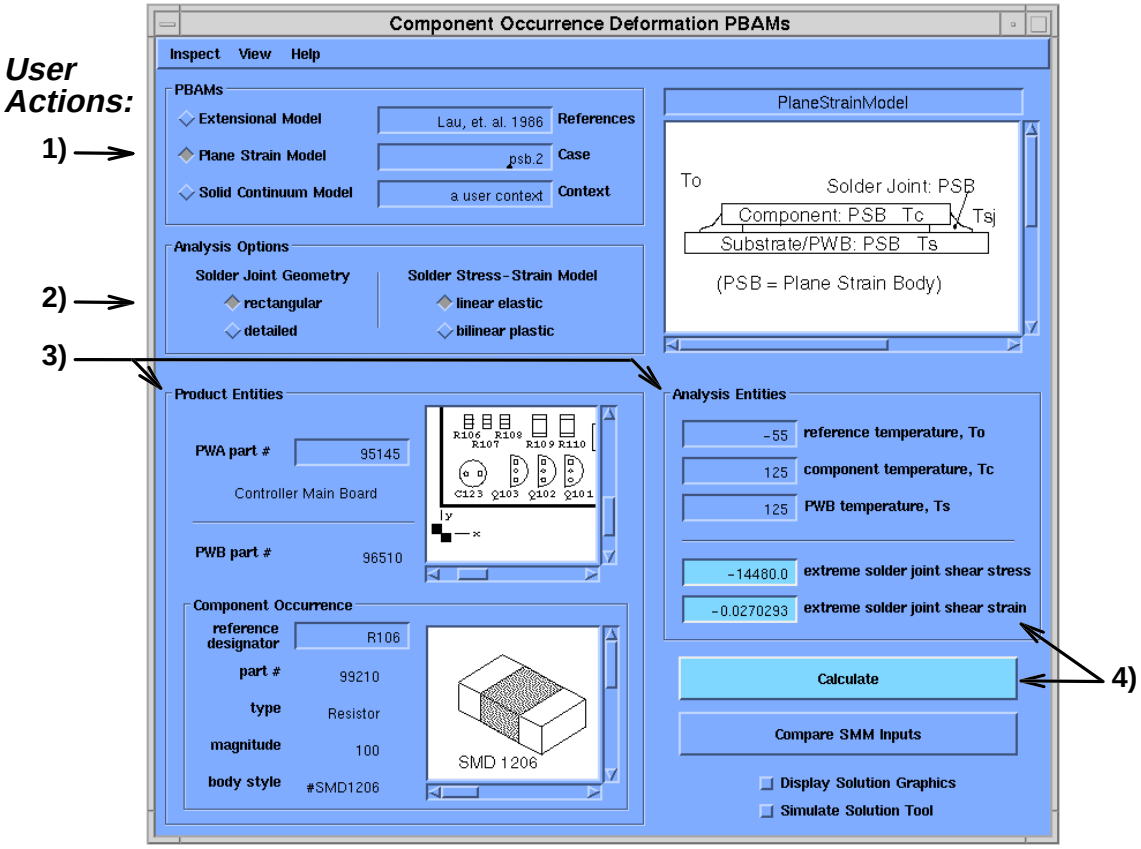
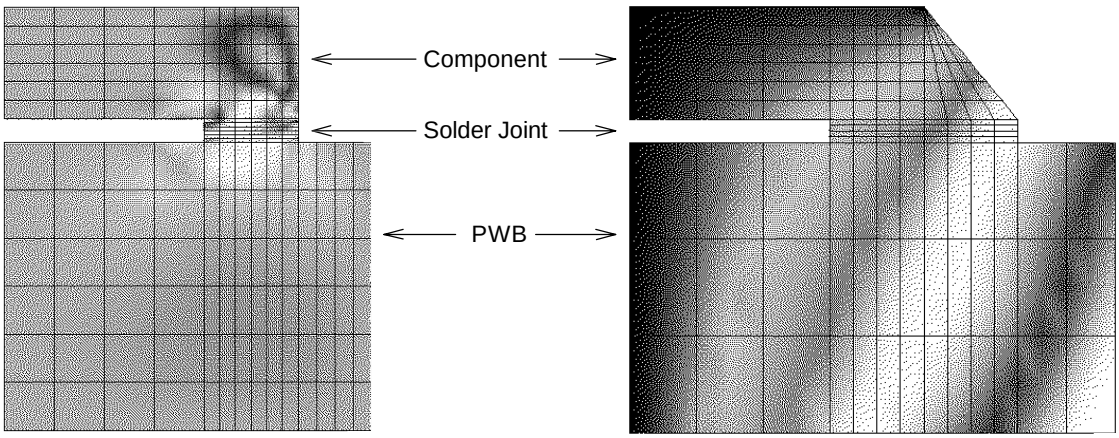


Fig. 24 Implemented Catalog of PBAMs

To check solder joint strains in a PWA, a designer 1) selects a particular type of PBAM, 2) selects analysis options, 3) specifies objects containing product information and analysis boundary conditions as inputs, and 4) requests analysis results as outputs. The following sequence of events then occurs per the MRA technique. An instance of the selected type of PBAM (*Component Occurrence Plane Strain Model* of Fig. 21) is created with these inputs. The PBAM instance creates an ABB instance (a *Plane Strain Bodies Systems* instance) and sets its attributes via associativity linkages. This ABB instance subsequently instantiates an SMM, which in turn calls a tool agent to run the appropriate solution tool (e.g., an Ansys or Cadas tool). The SMM instance receives the results from this analysis tool and passes them back to the ABB instance. Finally, the ABB instance returns the results to the PBAM instance, which puts them in product-specific terms (*extreme solder joint shear strain*) for usage by the designer.

Fig. 24 shows an implementation of this PBAM catalog, where the screen has been used per the preceding scenario. From this screen the user can open and view associated SMM and specialized analysis system instances like those given in Figs. 5 and 12, as this PBAM instance is their root context. Fig. 25 depicts the Cadas FEA shear stress plot (a) for the rectangular solder joint geometry case, as well as the *u* displacement plot (b) for the detailed geometry case. This scenario summarizes how PBAMs bridge the gap between design and analysis models and serve as product-specific front and back ends to general purpose analysis tools.



Shear Stress, τ_{xy}

a. Rectangular Solder Joint

Displacement, *u*

b. Detailed Solder Joint

Fig. 25 Automatically Generated FEA Results

6.3 PBAMs as Building Blocks

Because PBAMs are ABBs, they can be used as subsystems in other PBAMs. Fig. 26 illustrates this concept via a solder joint fatigue PBAM [2] composed of two other PBAMs and a general purpose ABB. The three subsystems shown correspond to three major steps in solder joint fatigue, namely determining thermal loads, solder joint strain, and fatigue life [38, 39].

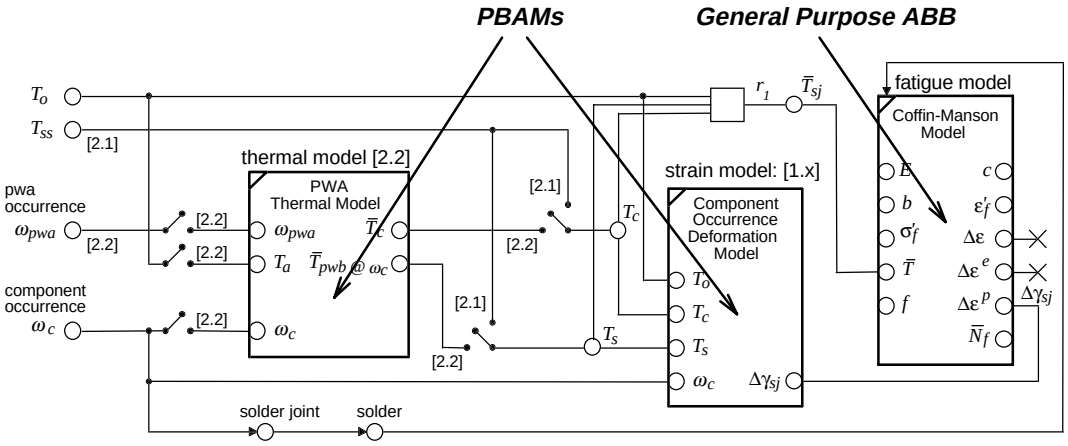


Fig. 26 Solder Joint Fatigue PBAM Constraint Schematic

This PBAM class supports analysis options, [1.x], regarding which deformation model a fatigue PBAM instance should use as its *strain model*. Available options are shown in the catalog of Fig. 23, ranging from a formula-based extensional model [1.1] to a 3-D FEA-based model [1.4]. The choices are subclasses of the abstract class *Component Occurrence Deformation Model*, all of which support the *strain model* subsystem view in Fig. 26 (including the *Plane Strain Model* of Fig. 22). When analysis options involve a common subsystem view such as this, they are known as **subsystem substitution options** [2] and are depicted in constraint schematics using a shorthand notation (instead of including each possible type of subsystem separately with associated option switches).

Even though the deformation PBAM subclasses are internally different, a common subsystem interface is possible because of object polymorphism and encapsulation. All deformation PBAMs accept the same component occurrence object, ω_C, as a connection, but internally each PBAM subclass extracts different information from ω_C via subclass-specific associativity linkages, Φ_i. Predictably, the *Extensional Model* has fewer linkages than the *Plane Strain Model*, while the *Solid Continuum Model* has more. With access to these seamless variations in analysis complexity, users can choose the PBAM with the best cost-accuracy combination for their design needs.

Two thermal load options are also available, namely thermal cycling [2.1] and power cycling [2.2]. An instance using the former option requires no thermal model as it simply equates the component temperature, T_c, and the substrate/PWB temperature, T_s, to a uniform steady state temperature load, T_{ss}. In the latter case T_c and T_s are equated to values determined by the *thermal model* subsystem, which simulates PWA operating conditions.

PBAMs like Fig. 26 are called **complex PBAMs**, while those with only one general purpose ABB subsystem are called **simple PBAMs**. A PBAM is itself an analysis model, albeit a new kind of product-specific one that includes idealization relations along with traditional relations. PBAMs are classified as ABBs due to this fact, coupled with their building block nature and the structure and operations they have in common with other ABBs. However, PBAMs do not have SMM mappings, ${}_{ABB}\Psi_{SMM}$, as general purpose ABBs do. Instead, PBAMs achieve solutions via the general purpose ABBs they ultimately utilize. It follows that complex PBAMs can solve multi-step problems requiring the interaction of diverse solution methods. Given this background, Fig. 1 is actually a special case view of the MRA, where the ABB illustrated is a general purpose ABB and the PBAM is a simple PBAM.

6.4 Other Capabilities

By utilizing constraints, PBAMs can support multidirectional input/output in some cases and aid the design synthesis process [2]. For example, instead of returning component fatigue life, $\overline{N}_f$, as an output, the PBAM in Fig. 26 can accept it as an input and use the *Extensional Model* to calculate the

corresponding solder joint height. Fig. 27 summarizes this and other PBAM capabilities via a pictorial view of the fatigue PBAM and its usage.

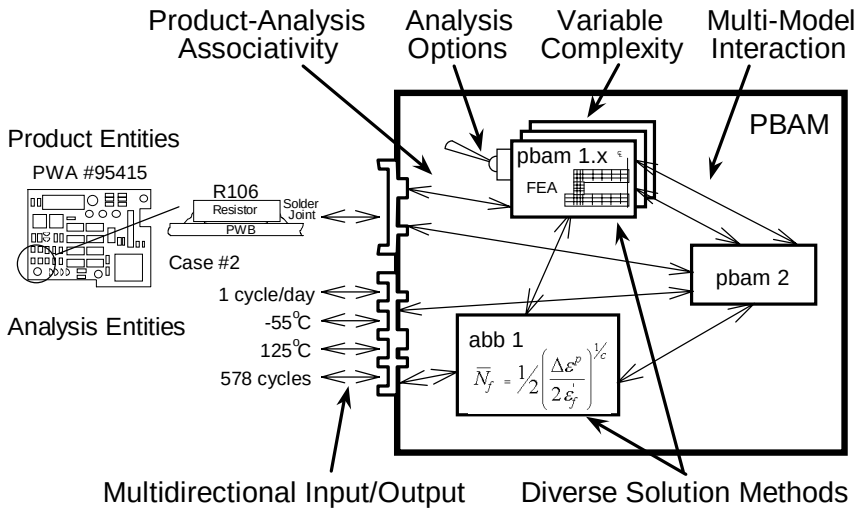


Fig. 27 Summary of PBAM Capabilities

7 Methodology for Routine Analysis Automation

The MRA methodology for automating routine analysis involves two distinct phases: tool creation and tool usage (Fig. 28). During the creation phase, a designer and an analyst first identify which routine analysis models are to be automated. These two people should be familiar with the types of products and analysis models being considered. The analyst then works with a developer (a person familiar with the MRA and associated object and constraint techniques) to develop PBAM classes that represent these analysis models. PBAM development basically means filling in the structural views of the PBAM class that will represent a specific type of analysis model (Fig. 7). This development process may be recursive in that supporting PBAMs, ABBs, SMMs and PM entities must also be developed if they do not already exist.

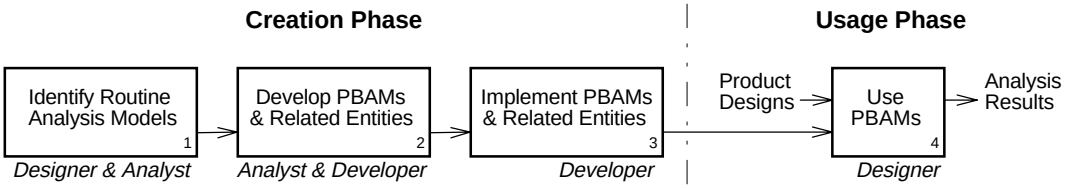


Fig. 28 MRA Methodology for Routine Analysis Automation

Based on the populated structural views, the developer next implements the PBAMs and other new supporting entities in a specific computing environment. Though likely to evolve with broader usage, preliminary development and implementation guidelines are available [2].

After the implementations are validated, the next phase of the MRA methodology is entered in which designers regularly apply the PBAMs to product design, as overviewed earlier. In object-oriented terms, specialized PBAM *classes* are developed and implemented in the creation phase, while *instances* of these classes are employed during the usage phase.

In keeping with object-oriented philosophy, the MRA methodology eases the creation of product-specific analysis tools through the reuse of generic entities where possible (Fig. 29). Many types of tools can be built by adding product-specific entities to the same generic MRA foundation. The present

representative implementation of such a foundation facilitates finite element-based solutions using Ansys and Cadas SMMs, and maintains constraints using the DeltaBlue constraint solver [14]. The solder joint examples in this paper come from the PWA-specific analysis tool built upon this foundation. Overall, the MRA methodology provides a way to incrementally create extendible, modular, product-specific routine analysis tools. Adding a new analysis model can be as simple as adding a new PBAM subclass.

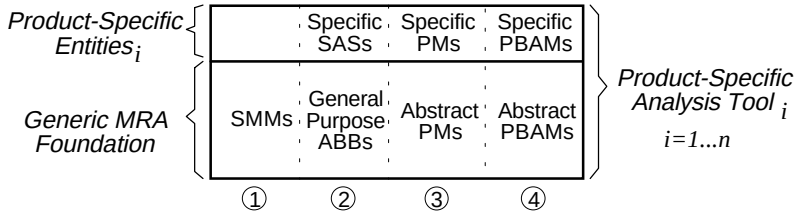


Fig. 29 Building Specialized Analysis Tools on the MRA Foundation

8 Discussion

8.1 Representation Cardinality

Reflecting the diversity supported by the MRA, note that the cardinality ratio between adjacent representations is generally *many:many*. For example, one ABB instance can produce many different kinds of SMM instances. One kind of SMM can have many instances produced by many kinds of ABBs. In this light the role a representation plays in the MRA can be further understood by considering the implications of removing it. If the ABB representation is removed, then each type of PBAM would have to provide its own mappings to each type of SMM, resulting in duplication of effort. Removing PMs would necessitate linkages between each PBAM type and each relevant design tool. The number of required extra linkages multiplies further if more than one representation is removed.

8.2 Specialized Analysis System Pros and Cons

In the current specialized analysis system approach, different analysis system topologies typically necessitate different specialized analysis system classes or different ${}_{ABB}\Psi_{SMM}$ mappings within the same class. Consequently, some specialized analysis systems (SASs) are primarily intended for specific products (Fig. 29). Still, a SAS class is beneficial as a logical place for capturing specialized knowledge in a reusable manner, including recommended mesh density and specialized ${}_{ABB}\Psi_{SMM}$ mappings. In the absence of generalized ${}_{ABB}\Psi_{SMM}$ mappings, SASs provide a modular, extendible, expert systems technique for automating routine analysis. Preliminary results indicate the current ABB structure can also support user-created general analysis system instances (Table 1), but their subsequent solution via general ${}_{ABB}\Psi_{SMM}$ mappings remains an open issue.

8.3 Neutral SMMs

The SMMs examples to date basically wrap input and output files for vendor-specific solution tools. Vendor-independent representations of SMMs could also prove useful if mappings to vendor-specific SMMs were provided. Fig. 30 illustrates this neutral format concept (analogous to IGES for CAD geometry) for the case of finite element SMMs. Other solution methods would require their own neutral representations. With this approach, an FEA-based ABB would need to support only a single mapping, ${}_{ABB}\Psi_{SMM}$, between itself and the neutral finite element SMM, instead of a mapping for each vendor-specific SMM (Fig. 16).

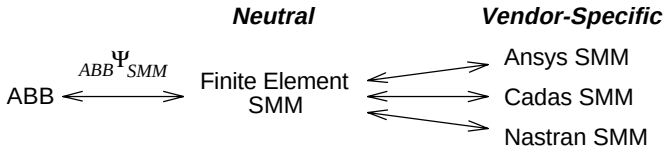


Fig. 30 Simplifying ABB-SMM Mappings via Neutral SMMs

In the case of FEA, Yeh [41] has demonstrated how the draft standard STEP Part 104 [11] can be used as a neutral representation of the mesh model portion of a finite element SMM. However, a standard representation that also includes the preprocessor model (Fig. 3) evidently does not currently exist.

8.4 Other Extensions

Other possible extensions include: a) more sophisticated idealization algorithms that determine characteristics like primary materials based on part geometry, material properties, and phenomenon type; b) better support for design synthesis relations, i.e. the inverse of idealizations, where analysis results are used to modify the PM; and c) better representation of product "environmental descriptions" and their relation to boundary conditions.

9 Summary

The multi-representation architecture (MRA) has been presented as a design-analysis integration strategy with the following characteristics. The MRA:

- Addresses the information-intensive nature of CAD-CAE integration;
- Breaks the design-analysis integration gap into smaller subproblems;
- Flexibly supports different design and analysis methods and tools;
- Is based on modular, reusable information building blocks;
- Defines a methodology for creating specialized, highly automated analysis tools to support product design.

Four representations compose the MRA and together make it a flexible, extendible architecture:

1) Solution Method Model (SMM)
- Packages solution tool inputs, outputs, and control as integrated objects.
- Automates solution tool access and results retrieval via tool agents.

2) Analysis Building Block (ABB)
- Represents analysis concepts using object and constraint graph techniques.
- Has a defined information structure with graphical views (e.g. constraint schematics) to aid ABB development, implementation, documentation and usage.
- Acts as a semantically rich 'pre-preprocessor' and 'post-postprocessor' model. ABB instances create SMM instances based on solution method considerations and receive results after automated solution tool execution.

3) Product Model (PM)
- Represents design aspects of products and enables connections with design tools.
- Supports idealizations usable in numerous analysis models.
- Has possibly many associated PBAMs.

4) Product Model-Based Analysis Model (PBAM)
- Contains linkages explicitly representing design-analysis associativity, indicating the usage of idealizations.
- Special case of the ABB representation, utilizing the same information views.

- Creates analysis models from ABBs and automatically supplies PM data as inputs.
- Represents routine analysis models as automated, predefined templates.
- Supports interaction of analysis models of varying complexity and solution method.
- Enables parametric design studies via multidirectional input/output (in some cases).

To date attention has been primarily given to the representation of design-analysis associativity via PBAMs, the inclusion of idealizations in PMs, and the automation of routine analysis using the MRA. PBAMs in particular and the MRA in general have been evaluated using PWA solder joint fatigue as a representative routine analysis case study. A "starter set" of ABBs and representative SMM classes have been developed to support this application. Results show that specialized PBAMs enable highly automated routine analysis and uniformly represent analysis models containing a mixture of both formula-based and finite element-based relations.

Acknowledgments

This research was initiated under funding from the Georgia Tech Manufacturing Research Center and its industrial sponsors: DEC, Ford, IBM, Motorola and the U. S. Army Missile Command. Investigation has continued both at Georgia Tech and Hitachi. We gratefully acknowledge the following people for their helpful comments and cooperation with regard to this particular paper: Norimasa Chiba, Kiyomi Morizumi, Andrew Scholand, Srivatsa Shamanna, Diego Tamburini, Michihiro Watanabe, and Takashi Yokohari.

References

1. Liker, J.; Fleischer, M.; Arnsdorf, D. (1992) Fulfilling the Promises of CAD. *Sloan Management Review* (Spring) 74-86
2. Peak, R. S. (1993) *Product Model-Based Analytical Models (PBAMs): A New Representation of Engineering Analysis Models*, Doctoral Thesis, Georgia Institute of Technology, Atlanta
3. Peak, R. S.; Fulton, R. E. (1994) A Multi-Representation Approach to CAD/CAE Integration: Research Overview. *Rapid Thermomechanical Design of Electronic Products in a Flexible Integrated Enterprise, Interim Report*, Fulton, R. E.; Ume, C.; et al., Advanced Electronic Packaging Lab., Prj. MS-93-03, Manufacturing Research Center, Georgia Tech, Atlanta, 22-27
4. Peak, R. S.; Fulton, R. E. (1993) Automating Routine Analysis in Electronic Packaging Using Product Model-Based Analytical Models (PBAMs), Part I: PBAM Overview. Paper 93-WA/EEP-23, ASME Winter Annual Meeting, New Orleans
5. Peak, R. S.; Fulton, R. E. (1993) Automating Routine Analysis in Electronic Packaging Using Product Model-Based Analytical Models (PBAMs), Part II: Solder Joint Fatigue Case Studies. Paper 93-WA/EEP-24, ASME Winter Annual Meeting, New Orleans
6. *ANSYS User's Guide* (1990) Swanson Analysis Systems Inc., Houston PA.
7. *CADAS Ver. P4 User's Manual* (1993) Hitachi Ltd., Hitachi-shi, Japan (in Japanese)
8. Stephens, E. R. (1993) *LEGEND: Laboratory for the Generation, Evaluation, and Navigation of Design*, Doctoral Thesis, Georgia Institute of Technology, Atlanta
9. Rosenberg, R. C; Karnopp, D. C. (1983) *Introduction to Physical System Dynamics*, McGraw Hill, New York
10. Ingrim, M. E.; Masada, G. Y. (1991) Extended Bond Graph Notation. *ASME J. Dynamics Systems, Measurements, and Control*, 113 (March) 113-17
11. ISO 10303-x, Industrial Auto. Sys. - Exchange of Prod. Model Data (STEP)
    - 10303-1, Part 1: Overview
    - 10303-11, Part 11: The EXPRESS Language
    - 10303-41, Part 41: Fundamentals of Product Description and Support
    - 10303-104, Part 104: Finite Element Analysis
    - 10303-105, Part 105: Kinematics
    - 10303-210, AP 210: Printed Circuit Assembly Product Design Data
12. Mashburn, T. A.; Anderson, D. C. (1991) An Extensible Computer Environment for Modeling and Analysis in Mechanical Design. *Proc. ASME Computers in Engineering Conf., Vol. 1*, 127-35.

13. Leler, W. (1988) *Constraint Programming Languages - Their Specification and Generation*, Addison-Wesley, Reading MA, USA
14. Freeman-Benson, B. N.; Maloney, J.; Borning, A. (1990) An Incremental Constraint Solver. *Comm. ACM*, 3(1) 54-63
15. Borning, A.; Freeman-Benson, B.; Maloney, J.; Wilson, M. (1991) Constraint Hierarchies and Their Applications. *1991 IEEE COMPCON Spring*, 388-93
16. Kumar, V. (1992) Algorithms for Constraint Satisfaction Problems: A Survey. *AI Magazine*, 13, 1, 32-44
17. Sannella, M. (1994) *Constraint Satisfaction and Debugging for Interactive User Interfaces*, Doctoral Thesis, University of Washington, Seattle. Available as Dept. of Comp. Sci. & Engineering TR 94-09-10
18. LaLonde, W.; Pugh, J. (1990) *Inside Smalltalk - Vol. I*, Prentice Hall, Englewood Cliffs NJ
19. Wirfs-Brock, R.; Wilkerson, B.; Wiener, L. (1990) *Designing Object-Oriented Software*, Prentice Hall, Englewood Cliffs, NJ
20. Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F.; Lorensen, W. (1991) *Object-Oriented Modeling and Design*, Prentice Hall, Englewood Cliffs, NJ
21. Rinderle, J. R.; Colburn, E. R. (1990) Design Relations. *Proc. ASME Design Theory & Methodology Conf.*, 267-72
22. Gui, J-K.; Mäntylä, M. (1994) Functional Understanding of Assembly Modeling. *Comp.-Aided Design*, 26, 6, 435-51
23. Solano, L.; Brunet, P. (1994) Constructive Constraint-based Model for Parametric CAD Systems. *Comp.-Aided Design*, 26, 8, 614-621
24. Forde, B. W. R.; Russell, A. D.; Stiemer, S. F. (1989) Object-Oriented Knowledge Frameworks. *Engineering with Computers*, 5, 79-89
25. Fenves, G. L. (1990) Object-Oriented Programming for Engineering Software Development. *Engineering with Computers*, 6, 1-15
26. Filho, J. S. R. A; Devloo, P. R. B. (1991) Object-Oriented Programming in Scientific Computations: The Beginning of a New Era. *Engineering Computations*, 8, 81-87
27. Lee, H.; Arora, J. (1991) Object-Oriented Programming for Engineering Applications. *Engineering with Computers*, 7, 225-35
28. Eastman, C. M.; Fereshetian, N. (1994) Information Models for Use in Product Design: A Comparison. *Comp.-Aided Design*, 26, 7, 551-572
29. Owen, J. (1993) *STEP: An Introduction*, Information Geometers, Winchester
30. Schenck, D.; Wilson, P. (1994) *Information Modeling the EXPRESS Way*, Oxford University Press
31. Curran, L. (1994) STEP Bridges Way to Better Product Modeling. *Mach. Design* (March 7) 137-42
32. Bray, O. H. (1988) *Computer Integrated Manufacturing - The Data Management Strategy*, Digital Press, Bedford, MA
33. Rangan, R. M. (1992) An Object Oriented Dictionary Based CAD/CAM Data Exchange Environment. Paper 92-WA/EDB-4, ASME Winter Annual Meeting, Anaheim, CA
34. Wentorf, R.; Budhiraja, A.; Collar, R. R.; Shephard, M. S.; Baehmann, P. L. (1992) Two Prototypes Testing the Use of an Expert System in the Control of Structural Analysis Idealizations. *Expert Systems for Scientific Computing*, (Houstis, E. N., et al., Editors), North-Holland, Amsterdam, 283-326
35. Finn, D. P. (1993) A Physical Modeling Assistant for the Preliminary Stages of Finite Element Analysis. *AI EDAM*, 7, 4, 275-286
36. Armstrong, C. G. (1994) Modelling Requirements for Finite-Element Analysis. *Comp.-Aided Design*, 26, 7, 573-578
37. Lau, J. H.; Rice, D. W.; Avery, P. A. (1986) Nonlinear Analysis of Surface Mount Solder Joint Fatigue. *Proc. IEEE CHMT Intl. Electronic Mfg. Tech. Symp.*, San Francisco, 173-84
38. Engelmaier, W. (1983) Fatigue Life of Leadless Chip Carrier Solder Joints During Power Cycling. *IEEE Trans. on Components, Hybrids, and Mfg. Tech.*, CHMT-6, 3, 232-37
39. Engelmaier, W. (1989) Thermal-Mechanical Effects. *Electronics Material Handbook. Vol 1 -Packaging*, (Minges, M. L., Editor), ASM Int'l., Materials Park OH, 740-53
40. Mao, J.; Fulton, R. E. (1992) Thermal Fatigue Reliability of the Solder Joints of Leadless Chip Components. *An Integrated Approach to Printed Wiring Board Design: Thermal Mechanical Behavior and Engineering Information Integration, Final Report: June 1991 - Sept. 1992*, (Fulton, R. E.; Ume, C.; et al., Editor), Mfg. Research Center - Advanced Electronic Packaging Lab, Georgia Tech, Atlanta
41. Yeh, C. P. (1992) *An Integrated Information Framework for Multidisciplinary PWB Design*, Doctoral Thesis, Georgia Institute of Technology, Atlanta