KNOWLEDGE COMPOSITION METHODOLOGY FOR EFFECTIVE ANALYSIS PROBLEM FORMULATION IN SIMULATION-BASED DESIGN

A Dissertation Presented to
The Academic Faculty

by

Manas Bajaj

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Mechanical Engineering

Georgia Institute of Technology
December, 2008

Knowledge Composition Methodology for Effective Analysis Problem Formulation in Simulation-based Design

Dr. Christiaan J. J. Paredis
Committee Co-Chair
Mechanical Engineering
Georgia Institute of Technology

Dr. Russell S. Peak
Committee Co-Chair
Product & Systems Lifecycle
Management Center
Georgia Institute of Technology

Dr. David Rosen
Mechanical Engineering
Georgia Institute of Technology

Dr. David McDowell
Mechanical Engineering
Georgia Institute of Technology

Dr. Charles Eastman
College of Architecture &
College of Computing
Georgia Institute of Technology

Dr. Steven J. Fenves
National Institute of Standards and
Technology

Date of Approval: November 11, 2008

DEDICATION

To my parents (Urmil and Indra) for their love, patience, and belief in me

To my wife (Mansi) for her love and companionship

To my family members for their boundless affection and unconditional pampering

PREFACE

The motivation for this research distills from the countless days and nights that I have spent in writing, debugging, and modifying computer code for creating and adapting finite element analysis models for design variations. In particular, I have spent a significant time over the last few years developing a production-ready software application for formulating FEA models for computing thermo-mechanical behavior of electronics artifacts, such as printed wiring boards and assemblies. The underlying design model-to-behavior model transformations in this application were realized using Java-based methods. As soon as I started testing this application with production-level design models gathered from several electronic design and manufacturing organizations, the hardships were apparent. It became extremely difficult to adapt the transformations to variations in design models and to maintain consistency of idealizations embodied in the application. Attempts to incorporate different fidelities of idealizations made matters worse. For production-ready deployment of this application, it was necessary that analysts have complete (and yet simple) control of the underlying idealizations and transformations. Without direct control of the source code, this was impossible. From discussions with several colleagues, conference presentations and publications, and interactions with designers and analysts across several organizations (NASA, Rockwell Collins, Lockheed Martin and Boeing, to name a few), it was apparent that this was in principle their story as well.

In this dissertation, I have made an initial attempt at researching and developing an approach that could alleviate some of the more painful conceptual problems experienced in “our combined hardships”. In particular, I find the application of graph transformations to model formulation for variable topology problems as a new application area emergent from this research. It is my hope that this dissertation provides a meaningful step towards a seamless interface between design and analysis activities, and a significant (though small) cornerstone for variable topology problems in general.

ACKNOWLEDGEMENTS

In the traditional *gurukul* system of education in India, it is believed that education is a process where students and the master (*guru*) can have unhindered, countless debates on topics. It is believed that this is essential for students to internalize the concepts and learn for themselves, as opposed to plainly accepting and memorizing the teachings of *gurus*. I have been a strong proponent of this method, where good ideas need to stand the test of time and critique. I will take this opportunity to thank my advisors Dr. Russell Peak and Dr. Chris Paredis for providing a *gurukul* environment during my PhD. As traditional *gurus*, they were always open to hear my thoughts, evaluate them objectively, and help me think for myself. Though there were several instances when I was shortsighted or overly conservative, this process has helped me grow as an independent thinker. It was the optimal combination of independent learning and strong guidance that a student may have.

I would also like to take this opportunity to thank members of my PhD reading committee—Dr. David Rosen, Dr. David McDowell, Dr. Charles Eastman, and Dr. Steve Fenves—for their valuable time and patience. Their comments and critique have sometimes broken my existing “shell of thoughts” and have encouraged me to expand my research ideas further. I have learnt tremendously from them, either directly by taking their classes or working on projects supervised by them, or in-directly by reading their published work. I strongly believe that it is an honor to have a committee of this stature.

I would like to especially thank my former advisor, Late Dr. Robert E. Fulton who encouraged me to join the PhD program in Mechanical Engineering at Georgia Tech. It was an honor to work with him. It was only much later after he passed away that I realized the value of his mere presence. He would have been proud to see this day, to see the last of his students (or *Fultonians* as they are known) ready to graduate this long and enriching journey. I am sure I have his best wishes with me. May God bless his departed soul!

I would like to specially thank the following project leads and organizations for sponsoring my research:

- Kevin Brady and John Messina at the Electronics and Electrical Engineering Laboratory (EEEL) at the National Institute of Standards and Technology (NIST)
- Steve Fenves, Ram Sriram, and Steve Ray at the Manufacturing Engineering Laboratory (MEL) at the National Institute of Standards and Technology (NIST)
- Thomas Thurman, Jim Lorenz, and Jack Harris at Rockwell Collins
- Sandy Friedenthal at Lockheed Martin Corporation

I am indebted to the opportunity created by the research projects sponsored by these organizations. Without their financial and technical support, it would not have been possible to perform this research.

I take this opportunity to specially thank members of the PDES Inc. Electromechanical team—Thomas Thurman, Greg Smith, Lothar Klein, Giedrius Liutkus, Viktoras Kovaliovas, Mike Benda, Stephen Waterbury, and Mike Dickerson. I would also like to thank the OMG SysML standards working group, especially Sanford Friedenthal and Roger Burkhart for their expertise and encouragement; and the VIATRA team—Daniel Varro and his colleagues at the Budapest University of Technology and Economics in Hungary.

What good is research life without the camaraderie of fellow researchers and trips to coffee shops? I would like to thank the following members of the Engineering Information Systems Lab (now Modeling and Simulation Lab) for their camaraderie—Sai Zeng, Injoong Kim, Greg Mocko, Nsikan Udoyen, Mohammed Saadat, Robbie Ludlow, Andy Scott, and Bennett Wilson. I would also like to thank the following members of the Systems Realization Lab for their camaraderie—Jitesh Panchal, Tommy Johnson, Rich Malak, Jonathan Jobe, Jay Ling, Benay Sager, Jamal Wilson, Carolyn Seepersad, Hae-Jin Choi, and Matthias Messer. I also thank Dr. Dirk Zwemer and Dr. Selcuk Cimtalay at InterCAX (where I work full-time) for their help and support, especially handling the extra work during the last few weeks of my dissertation journey.

My research journey at Georgia Tech was made lively and exciting by a large group of friends. I would like to thank each one of them. Without them, it would have been an “all work, no play” journey that would have exhausted me long ago. Space and time do not permit me to mention each name but to name a few in particular—Kunal Singh, Arnab Choudhury, Rajesh Luharuka, Anandita Seth, Rithika Naidu, Sakethraman Mahalingam, Sriram Kanvah, Jitesh and Pinky Panchal, Saurabh Marda, Tejpal Singh, Raman and Pooja Baijal, Anandraj and Deepti Sengupta, Lokendra and Meghna Chauhan, Vivek and Alina Mishra, Abhinav Saxena, Lalit Bohra, Apurva Mohan, Dheeraj Reddy, Vipul Saini, and Vishal and Tarang Gupta. I thank each of you for the lively evenings, late night movies, and gatherings over several cups of coffee and tea. It would have been impossible for me to accomplish this goal without your camaraderie.

I would also like to thank my family members for their boundless affection and unconditional pampering over several years, especially my *mamus, mamis, and massis* (maternal uncles and aunts). I thank all my cousin brothers and sisters for the love and affection they have showered on me. I often bask in sweet memories of my summer vacations at my maternal place. I thank my late grandparents (paternal and maternal) for their blessings.

In the end, I would like to deeply thank those who shaped my life and inspired me to dream beyond the obvious. My father (Indra) and my mother (Urmil) have been the foundation stone of my life. Words cannot fathom the sacrifices they have made for me to be where I am. This thesis is dedicated to my parents and their dreams. With the grace of God, last year I found a person who has changed my life for good. I found my true love in my gorgeous and sweet wife Mansi. Her love and companionship, patience with my thesis, and countless culinary delights have made this journey possible for me. I do owe her our first real vacation that has been postponed long due to this dissertation.

Lastly, I would like to thank God for holding me strong and for powering me through this journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

AMTL	Artifact Model Transformation Library
B1 model	Artifact Behavior Meta-Model (e.g. Core Behavior Model) See section 7.4 for detailed description of B1-B5 models
B2 model	Analysis-specific Behavior Meta-Model
B3 model	VTMB Artifact Behavior Meta-Model
B4 model	FTMB Artifact Behavior Model Structure
B5 model	FTMB Artifact Behavior Model Instance
BDD	Block Definition Diagram (SysML)
BGA	Ball Grid Array Electronics Chip Package
BMFS	Behavior Model Formulation Specifications
BMFM	Behavior Model Formulation Method
CAD/E	Computer-Aided Design / Engineering
CBM	Core Behavior Model
CPM2	Core Product Model version 2 (NISTIR 7185) (Fenves 2004)
CPM2_xKCM	CPM2 extended by the Knowledge Composition Methodology
D1 model	Artifact Meta-Model (e.g. CPM2_xKCM) See section 6.2 for detailed description of D1-D5 models
D2 model	Application-specific Artifact Meta-Model
D3 model	VTMB Artifact-specific Meta-Model
D4 model	FTMB Artifact Model Structure
D5 model	FTMB Artifact Model Instance
FEA/M	Finite Element Analysis / Method
FTMB	Fixed Topology Multi-Body
IBD	Internal Block Diagram (SysML)
KCM	Knowledge Composition Methodology
PCA/B, or	Printed Circuit Assembly/Board, or Printed Wiring Assembly/Board

PWA/B	
SBD	Simulation-Based Design
SysML	Systems Modeling Language (www.omgsysml.org)
VTMB	Variable Topology Multi-Body

SUMMARY

In simulation-based design, a key challenge is to formulate and solve analysis problems efficiently to evaluate a large variety of design alternatives. The solution of analysis problems has benefited from advancements in commercial off-the-shelf math solvers and computational capabilities. However, the formulation of analysis problems is often a costly and laborious process. Traditional simulation templates used for representing analysis problems are typically brittle with respect to variations in artifact topology and the idealization decisions taken by analysts. These templates often require manual updates and "re-wiring" of the analysis knowledge embodied in them. This makes the use of traditional simulation templates ineffective for multi-disciplinary design and optimization problems.

Based on these issues, this dissertation defines a special class of problems known as variable topology multi-body (VTMB) problems that characterizes the types of variations seen in design-analysis interoperability. This research thus primarily answers the following question:

How can we improve the effectiveness of the analysis problem formulation process for VTMB problems?

The knowledge composition methodology (KCM) presented in this dissertation answers this question by addressing the following research gaps: (1) the lack of formalization of the knowledge used by analysts in formulating simulation templates, and (2) the inability to leverage this knowledge to define model composition methods for formulating simulation templates. KCM overcomes these gaps by providing: (1) formal representation of analysis knowledge as modular, reusable, analyst-intelligible building blocks, (2) graph transformation-based methods to automatically compose simulation templates from these building blocks based on analyst idealization decisions, and (3) meta-models for representing advanced simulation templates—VTMB design models, analysis models, and the idealization relationships between them.

Applications of the KCM to thermo-mechanical analysis of multi-stratum printed wiring boards and multi-component chip packages demonstrate its effectiveness—

handling VTMB and idealization variations with significantly enhanced formulation efficiency (from several hours in existing methods to few minutes).

In addition to enhancing the effectiveness of analysis problem formulation, the KCM is envisioned to provide a foundational approach to model formulation for generalized variable topology problems.

Chapter 1 : INTRODUCTION

In today's dynamic product realization environments driven by functionality, time-to-market and cost-to-develop, it is often economically advantageous for engineers to create virtual prototypes of a system (Pratt 1995) and verify design alternatives by means of simulations. Here, simulation refers to the use of computational models to analyze and evaluate the behavior of an engineering system. Simulations enable designers and analysts to predict and optimize system performance during the design process, thereby reducing the number of design cycles, cycle time, costly reworks, and improving system quality. This approach of using simulations as the primary means of analysis and evaluation of system alternatives is commonly known as *simulation-based design* (SBD)[a] (Fenves, Choi et al. 2003; Shephard, Beall et al. 2004; NSF 2006). Simulation-based design bridges the knowledge and methodologies of engineering domains, such as mechanical, aerospace, electrical, and civil, with those of mathematical and computational sciences, thus providing integrated techniques for predicting system behavior and optimizing system designs (NSF 2006). Figure 1.1 depicts the scope of simulation-based design in a model of the design process (Gero 1990).

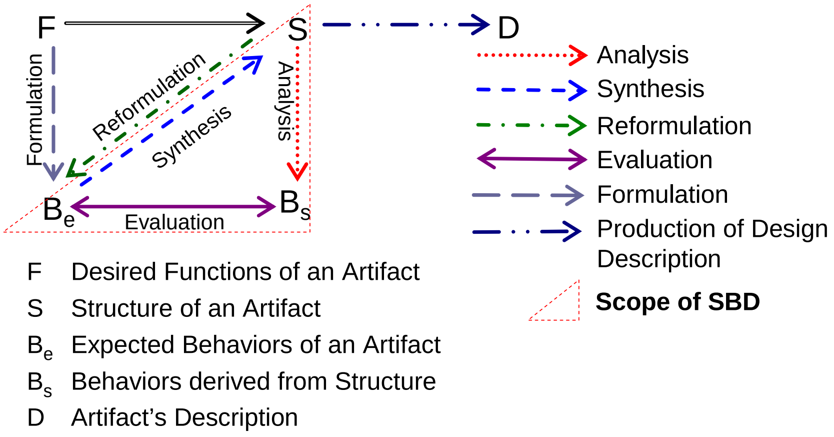

F
S
D
Formulation
Reformulation
Synthesis
Analysis
Be
Bs
Evaluation
Analysis
Synthesis
Reformulation
Evaluation
Formulation
Production of Design Description
Scope of SBD
F Desired Functions of an Artifact
S Structure of an Artifact
Be Expected Behaviors of an Artifact
Bs Behaviors derived from Structure
D Artifact's Description

Figure 1.1: Scope of simulation-based design in a model of the design process (Gero 1990)

[a] Simulation-driven design, Simulation-based engineering science, Analysis-based design, and Analysis-driven design are also widely used similar phrases.

Simulation-based design involves three key stages of a design process—synthesis, analysis, and evaluation. Typically during a design process, a set of desired functions (F) is transformed to a design description (D) to be used for downstream product lifecycle processes, such as manufacturing. Except during catalog lookup, the direct transformation F→D is not available. Hence, designers generate an artifact's structure (S) which is then transformed to its design description (D). The transformation F→S—an alternate statement of the design problem—is achieved in the following manner in the design process: (a) During *formulation*, the desired functions of an artifact are transformed to expected behaviors (B_e); (b) Then during *synthesis*, different alternatives of an artifact's structure (S) are generated based on its expected behaviors (B_e); (c) During *analysis*, the behaviors of each alternative of an artifact's structure are determined (B_s); (d) Then during *evaluation*, the expected behaviors (B_e) are compared with behaviors derived from an artifact's structure (B_s) for each alterative of the structure. Evaluation is used to narrow down on a set of alternatives. Sometimes when a structure is analyzed, its behavior can be a useful superset of expected behaviors. In such a case, the set of desired functions is accordingly extended and this is known as *reformulation*.

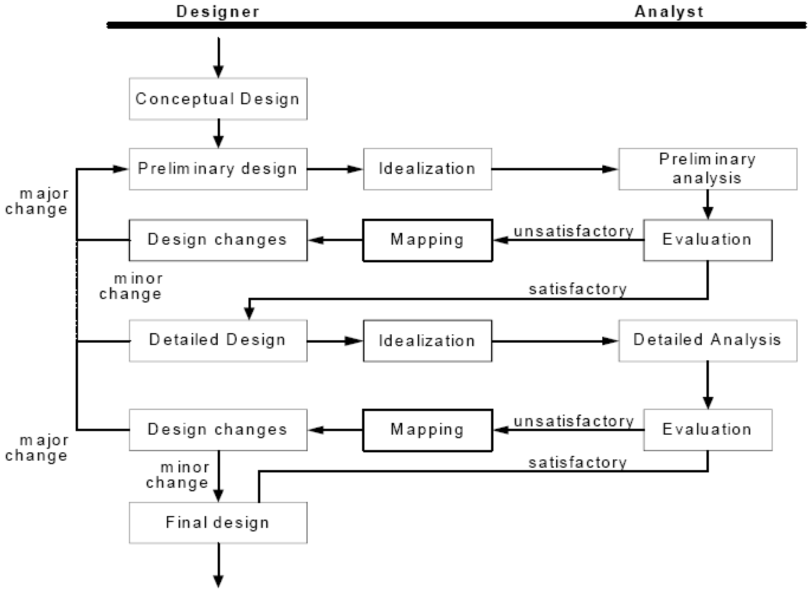


Figure 1.2: Integrated functional and spatial design through design phases (Fenves, Choi et al. 2003)

Simulation-based design is an iterative and collaborative process involving designers and analysts, and spanning all design phases. Figure 1.2 illustrates an integrated functional and spatial design scenario (Fenves, Choi et al. 2003) representing the iterative and collaborative nature of simulation-based design. In a given design phase, designers synthesize alternative forms of an artifact that are represented as design models. For a particular type of analysis, (i) design alternatives are idealized in the context of the analysis, (ii) a particular set of behaviors are computed using simulation, and (iii) the simulation results are evaluated against requirements. The evaluation results from a family of analyses are then used for selecting the best-in-class alternatives for the next design phase or mapped to generate new design alternatives for the current phase. This collaborative process is realized by means of models. Alternative forms of an artifact are represented as design models that are then idealized and enriched with analysis oriented information for creating analysis models—also known as behavior models in the context of this dissertation.

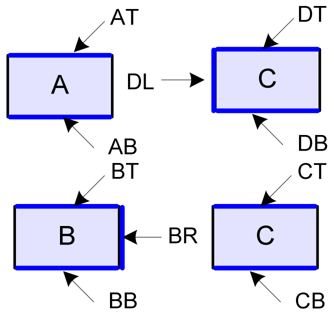
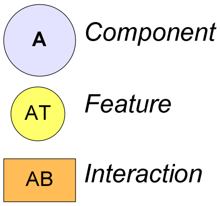
To reuse the knowledge associated with analyzing design alternatives, and to automate the analysis and evaluation process, designers and analysts create simulation templates—models that relate an artifact's design parameters to its behavior parameters. Design parameters are abstracted from design models and behavior parameters are abstracted from behavior models. In essence, a simulation template provides a structure for relating design models and behavior models, and provides a template for model-based communication between designers and analysts. In an automated analysis and evaluation process, for each design alternative: (a) the values of design parameters are input to the simulation templates, (b) the values of behavior parameters are computed as outputs of the simulation templates, and (c) the values of behavior parameters are evaluated against requirements. At each design stage, this process is typically repeated for a set of design alternatives using several simulation templates, and the best-in-class alternatives are selected for the next design phase. If simulation templates are defined with stepping stone models between design models and behavior models, and the idealization relationships between these models are inherently non-causal, then simulation templates may also be used to compute the "preferred" values of design parameters from given values of behavior parameters (Peak and Fulton 1994; Peak, Burkhart et al. 2007).

In design optimization problems that aim to select the best-in-class alternative(s), simulation templates are used for computing behavior parameters that *directly or indirectly participate in the objective function*. For a particular alternative in the design space exploration path, simulation templates are used for computing behavior parameters that are then used to evaluate the objective function. In general, simulation templates provide an efficient approach for *routine analysis* (including complex and coupled simulations) *and optimization problems* today, such as multi-scale, multi-body, and multi-disciplinary analysis and optimization problems.

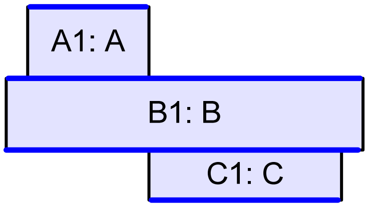
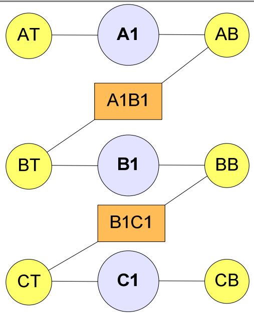
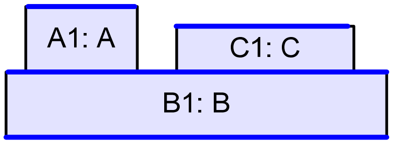
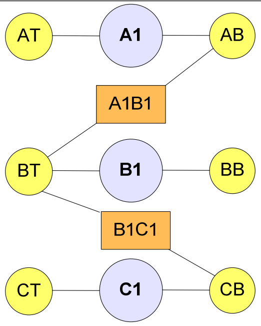
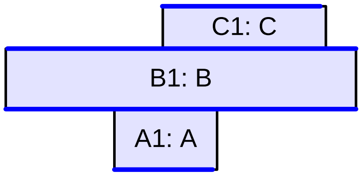
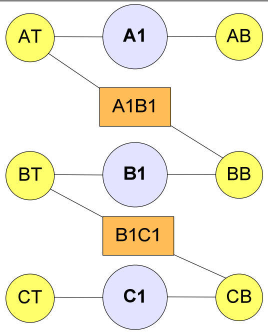
Considering the time and effort required to create simulation templates, it is economically preferable that a given set of simulation templates be reused for computing behavior parameters for all feasible design alternatives. However, *simulation templates are generally brittle to changes in the assembly system topology of design alternatives*. As an example, with variations in the configuration of components in an assembly or variations in the number of components (including features and interactions), simulation templates have to be manually updated. With variations in the types of components, features, and interactions among components, analysts not only have to provide the idealization intent for these new types of design objects but also manually update simulation templates.

Assembly system topology—defined more precisely later in this dissertation—is a collective measure of the number and types of components, their interactions, and component features participating in these interactions in an artifact assembly. In general, simulation templates are not reusable for computing behavior parameters for design alternatives with non-equivalent assembly system topologies.

Figure 1.3 below illustrates simple examples of design alternatives with non-equivalent assembly system topologies. The first column in the figure shows the spatial arrangement of parts in alternative assembly systems, and the second column shows the equivalent graph representations. For a given part A, the top, bottom, left, and right features are referred as AT, AB, AL, and AR respectively. Assembly ABC is a reference design alternative with three components A1, B1, and C1 of part types A, B, and C respectively, arranged in a certain configuration. In assemblies ABC2 and ABC3, the number and types of components are the same but their configurations are different. With respect to ABC1, C1 interacts with the top feature of B1 in ABC2; and with respect to ABC2, A1 interacts with the bottom feature of B1 in assembly ABC3. In assembly ArBC, the number, types, and configuration of components is the same as in ABC1, ABC2, and ABC3 but the interaction between A1 and B1 is changed—A1 can roll along the top surface of B1. In assembly ABCD, a new component D1 (of new type D) is added with respect to ABC.

Parts and Features	***Graph representation***
	

Assembly System Configurations

ABC1 	
ABC2  *Baseline*: ABC1 *Change*: C1 moved to top of B1	
ABC3  *Baseline*: ABC2 *Change*: A1 moved to bottom of B1	

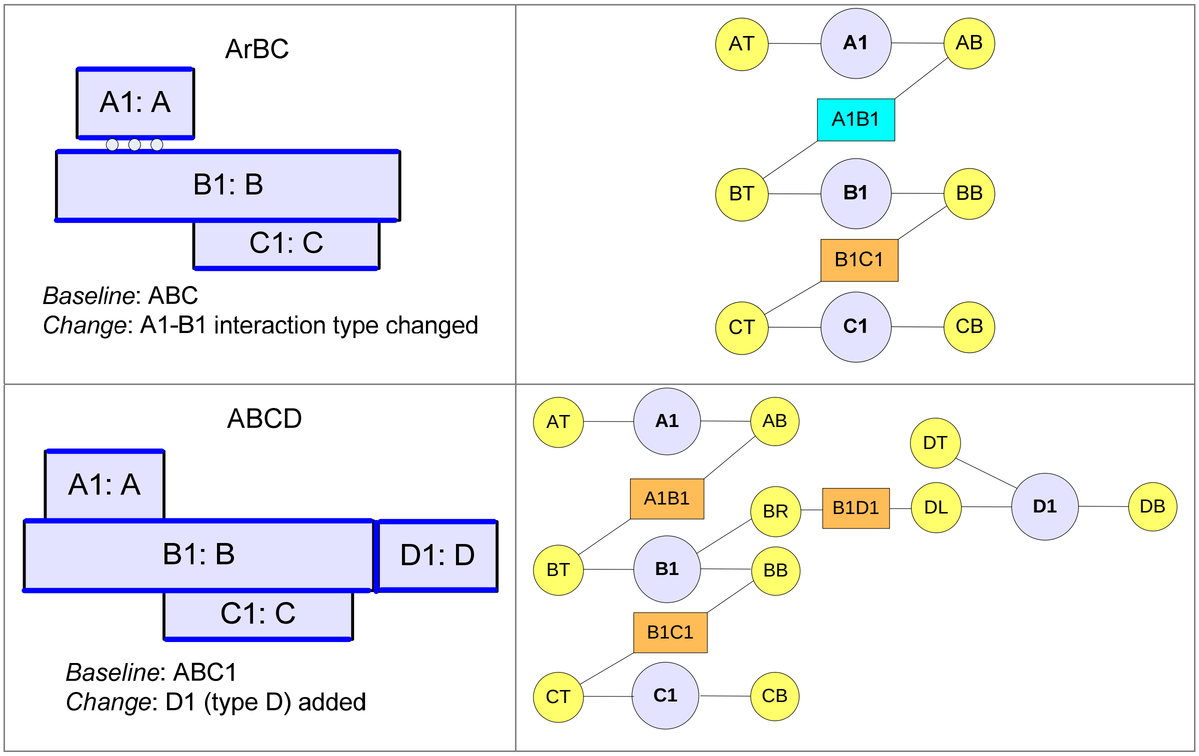


Figure 1.3: Examples of design alternatives with non-equivalent assembly system topologies

Any such changes in the number or types of components, interactions, or features participating in the interactions alter the assembly system topology of artifact alternatives. These changes are also reflected in changes in the topologies of equivalent graph representations. A more precise and formal graphical representation of assembly system topology is presented later in this dissertation.

The idealization relationships between design parameters and behavior parameters in a simulation template are typically based on assumptions about the number, type, or the configuration of components and their interactions in an artifact assembly. With changes in the assembly system topology, idealization relationships embodied in simulation templates may need to be modified or extended. Idealization relationships “implicitly” represented as parameterized scripts can typically handle only a subset of topology changes. For example, changes in the number of components can be handled with assumptions on the nature and type of interactions and configuration of components. Such scripts are commonly used today to create behavior models from design models, such as the case when automatically generating FEA models in commercial tools such as ABAQUS and ANSYS.

Based on the concept of assembly system topology, a special class of analysis problems known as **Variable Topology Multi-Body (VTMB) Problems** is defined in this dissertation. *VTMB Problems are a class of problems where the assembly system topology of design alternatives changes.* In the context of simulation-based design VTMB problems affect simulation templates, generally requiring manual updates and “re-wiring” of relationships between design parameters and behavior parameters in simulation templates. The brittleness of simulation templates to VTMB problems makes their reuse even more difficult for multi-disciplinary design optimization problems where the number of idealization relationships and behavior parameters per simulation template and the number of simulation templates are generally larger as compared to optimization problems concerning a single discipline. In general, the lack of robustness of simulation templates to VTMB problems jeopardizes their efficacy for multi-scale, multi-body, and multi-disciplinary analysis and optimization problems.

In addition to assembly system topology variations among design alternatives, changes in idealization decisions taken by analysts also cause changes in simulation templates. Generally, these changes involve manual “re-wiring” of the idealization relationships embodied in simulation templates. This is economically infeasible, especially in cases when analysts perform trade studies on idealizations, especially for new types of analysis problems and to measure the relative advantages of high-fidelity, time-intensive analyses versus quick, low-fidelity analyses. Even without variations in assembly system topology, changes in the idealizations—such as using shells versus solids, or isotropic versus orthotropic material behavior—involves manually restructuring the relationships between design parameters and behavior parameters in simulation templates.

Broadly, there are two steps in leveraging simulation templates for behavior analysis and design optimization problems as described above. These are: (a) formulation of simulation templates, and (b) execution of simulation templates. The execution of simulation templates has benefited from advancements in computational capabilities and commercial off-the-shelf solvers, such as differential algebraic equation solvers and FEA solvers. However, the formulation of simulation templates is often costly and laborious, especially for VTMB problems and idealization changes.

The lack of effectiveness of simulation templates for performance evaluation and design optimization is primarily due to the: (a) inability to automatically adapt simulation templates to VTMB problems, (b) inability to automatically adapt simulation templates to changes in idealization decisions taken by analysts, and (c) inefficient representation and creation of simulation templates in general. In light of these challenges, the primary research question that this dissertation answers is as follows:

How can we improve the effectiveness of the analysis problem formulation process for VTMB problems?

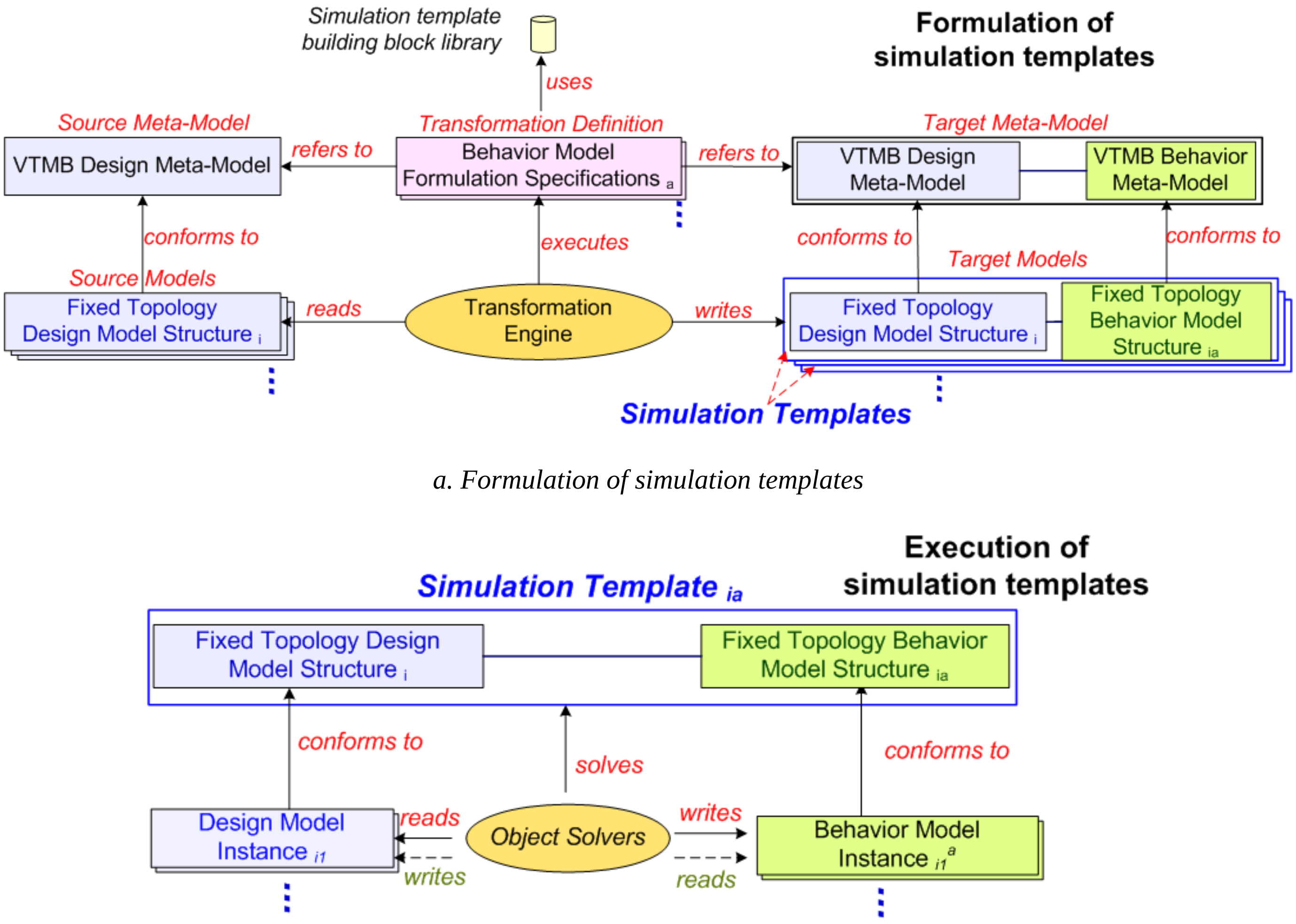
Though simulation templates are brittle to VTMB problems, it is also not pragmatic to create a simulation template that is robust to all types of changes in the assembly system topology of design alternatives. Additionally, changes in idealizations will require manual and costly "re-wiring" of simulation templates. Hence, a holistic and pragmatic solution to this challenge problem is to have the capability to automatically compose simulation templates from idealization decisions taken by analysts. By the virtue of information-rich representation of idealization intent, analysts can create simulation templates that are robust to certain types of assembly system topology changes. With the capability to compose simulation templates from building blocks automatically, analysts can create simulation templates for other types of assembly system topology changes as well as for changes in idealization decisions in an efficient manner.

However, research gaps exist in the current state-of-the-art for achieving this solution. Specifically, these gaps are: (a) the lack of formalization of the knowledge used by analysts in formulating simulation templates, and (b) the inability to leverage this knowledge to define model composition methods for formulating simulation templates. The lack of formalized knowledge is particularly apparent in the direct representation of idealization decisions as mathematical equations and procedural functions in scripts or programs used for creating behavior models, without necessarily representing the idealization intent. This results in simulations templates that are brittle to VTMB problems and idealization changes. If one can formalize the types of idealization decisions taken by analysts, and the conditions for these decisions, one may explicitly represent these decisions at a higher level of abstraction from which mathematical relations or computable scripts may be automatically derived. For efficient formulation of simulation templates, it is also necessary to define model composition methods that can automatically compose simulation templates from reusable building blocks and the idealization decisions taken by analysts. The representation of building blocks requires both static knowledge—What concepts are represented by building blocks?—as well as dynamic knowledge—How are building blocks composed to create simulation templates?

The *Knowledge Composition Methodology (KCM)* presented in this dissertation addresses these research gaps by providing (a) a method to formalize and reuse the knowledge required for creating simulation templates, and (b) model composition methods to automatically compose simulation templates from this formalized knowledge and the idealization decisions taken by analysts. Figure 1.4 illustrates a high-level functional view of the KCM. Figure 1.4a illustrates the formulation of simulation templates using KCM's Behavior Model Formulation Method (BMFM). The BMFM is a model transformation method used for automatically composing simulation templates from fixed topology design model structures based on the idealization decisions taken by analysts. This model transformation method is founded on graph transformations, where fixed topology design model structures and simulation templates are abstracted as source and target graphs respectively, and reusable graph transformation patterns and rules are explicitly scheduled to compose the target graph from the source graph and the idealization decisions. The Behavior Model Formulation Method and its model transformation approach are presented in Chapter 8. An overview of each component in the functional view is described below:

- *VTMB Design Meta-Model* defines the constructs and relationships to represent design alternatives with non-equivalent assembly system topologies for a specific type of artifact, such as printed circuit boards. KCM provides an extended Core Product Model (CPM2_xKCM) based on the Core Product Model originally proposed by Fenves et al. (Fenves 2004) to represent designs and idealized designs of artifacts. CPM2_xKCM is specialized to define a VTMB Design Meta-Model for representing variable topology alternatives for a specific artifact type. CPM2_xKCM is presented in Chapter 6 of this dissertation.

- *Fixed Topology Design Model Structure* represents a set of design alternatives with equivalent assembly system topologies. Several fixed topology design model structures may be defined conforming to a VTMB design meta-model. Examples of fixed topology design model structures are presented in Chapter 6 of this dissertation.


Simulation template building block library
uses
Formulation of simulation templates
Source Meta-Model
VTMB Design Meta-Model
refers to
Transformation Definition
Behavior Model Formulation Specifications a
refers to
Target Meta-Model
VTMB Design Meta-Model
VTMB Behavior Meta-Model
conforms to
executes
conforms to
conforms to
Source Models
Target Models
Fixed Topology Design Model Structure i
reads
Transformation Engine
writes
Fixed Topology Design Model Structure i
Fixed Topology Behavior Model Structure ia
Simulation Templates
Execution of simulation templates
Simulation Template ia
Fixed Topology Design Model Structure i
Fixed Topology Behavior Model Structure ia
conforms to
solves
conforms to
Design Model Instance i1
reads
writes
Object Solvers
writes
reads
Behavior Model Instance i1 a

a. *Formulation of simulation templates*

b. *Execution of simulation templates*

Figure 1.4: Knowledge Composition Methodology – A functional overview

- *VTMB Behavior Meta-Model* defines the constructs and relationships for representing behavior models for design alternatives with non-equivalent assembly system topologies. Together a VTMB design meta-model, a VTMB behavior meta-model, and their relationships provide a meta-model for simulation templates. KCM provides the Core Behavior Model (CBM) as a meta-model for representing behavior models (including relationships with design models). Together CPM2_xKCM and CBM define a comprehensive meta-model for representing simulation templates for design alternatives with non-equivalent assembly system topologies. The Core Behavior Model is presented in Chapter 7 of this dissertation. KCM builds on the MRA simulation template pattern (Peak and Fulton 1994; Peak, Burkhart et al. 2007) to represent simulation templates that are founded on physics-based concepts and independent of a particular solution method or solvers.

- *Fixed Topology Behavior Model Structure* represents a set of behavior models created for a design model structure based on the idealization decisions taken by analysts. Several fixed topology behavior model structures may be formulated for variations in design model structures and idealization decisions. Examples of fixed topology behavior model structures are presented in Chapter 7 of this dissertation.
- *Behavior Model Formulation Specification* (BMFS) embodies the idealization decisions taken by analysts. BMFS provides specifications for the composition of simulation templates.
- *Simulation Template Building Block Library* provides a library of reusable building blocks that are used for automatically composing simulation templates. KCM provides the Analysis Building Block (ABB) Meta-Model that represents the constructs and relationships for key types of building blocks. For different analysis disciplines, building blocks are defined as specializations of the generic building block concepts in the ABB Meta-Model. The library also contains reusable model transformation rules and patterns used by the Behavior Model Formulation Method. The ABB Meta-Model and ABB library are presented in Chapter 7. The dynamic aspects of ABBs that govern how ABBs are composed in an ABB system are represented by graph transformation rules and patterns, and described in Chapter 8.
- *Transformation Engine* is a graph transformation engine that executes the Behavior Model Formulation Specifications to automatically compose simulation templates.

KCM addresses VTMB problems because for different desing model structures—each of which represents a set of design alternatives with equivalent assembly system topologies—behavior model structures and simulation templates can automatically be created for the same Behavior Model Formulation Specifications. Additionally, behavior model structures and simulation templates can also be automatically created for different Behavior Model Formulation Specifications and for a given design model structure.

Figure 1.4b illustrates the execution of simulation templates composed using KCM's Behavior Model Formulation Method. With the availability of a simulation template, object solvers (or solver managers) such as ParaMagic[b], OpenModelica[c], and Mathematica[d] may be used for solving the idealization relationships embodied in simulation templates for design instances that conform to the fixed topology design model structure embodied in simulation templates. As shown in Figure 1.4b, for each design instance the idealization relationships are solved to create a behavior model instance that confirms to the fixed topology behavior model structure embodied in the simulation template. Each behavior model instance can then be solved using different solution methods and solver tools, such as using FEA method and solvers such as ABAQUS or ANSYS. If the idealization relationships embodied in simulation templates are inherently non-causal, such as mathematical equations, then analysts may specify target values of behavior parameters and compute design parameters values (unique or a range) using the same simulation template. Multi-disciplinary design optimization tools—at their backend—can deploy the ability to automatically formulate simulation templates for VTMB problems, and the ability to execute a given simulation template (possibly in multiple directions) for different values of design model (or behavior model) instances. This will provide an effective mechanism to search for the feasible design space that has alternatives that have non-equivalent assembly system topologies.

The primary contribution of the research presented in this dissertation is the Behavior Model Formulation Method that prescribes a graph transformation-based approach for automatically composing simulation templates for (i) variations in assembly system topology of design alternatives, and (ii) variations in idealization decisions taken by analysts. The secondary contributions of this research are the (i) characterization of VTMB problems, (ii) meta-models for representing simulation templates and their building blocks, (iii) graph pattern and transformation rules to manage models that

[b] www.intercax.com/sysml

[c] www.ida.liu.se/labs/pelab/modelica/OpenModelica.html

[d] www.wolfram.com/mathematica

conform to these meta-models, and (iv) an extensible, proof-of-concept library of simulation template building blocks.

The most promising extension of this research lies in the application of KCM's model transformation approach to other types of VTMB problems. Examples of simulation templates that are brittle to topology variations of system alternatives are abound. The concept of assembly system topology, as presented in this research, is defined for systems in general, including systems that may have human and software components. Suggested applications include manufacturing systems, real time embedded systems, and energy generation and distribution networks.

This dissertation consists of three parts:

- Part 1: Problem Definition
- Part 2: Knowledge Composition Methodology
- Part 3: Verification and Validation, Future Work, and Closure

Part 1 lays a platform for framing the research problem, identifying research gaps, and posing research questions. It consists of Chapters 2-4. Chapter 2 presents basic concepts necessary for problem description, followed by a presentation of three foundational perspectives in aspects of simulation-based design relevant to this research, and definition of VTMB problems that form the thrust of this research. It ends by identification of research gaps and their exemplification using an example VTMB problem. Chapter 3 describes related technical work in the context of these research gaps, and Chapter 4 builds on the research gaps and relevant technical background to pose the research questions and present research hypotheses.

Part 2 presents the Knowledge Composition Methodology (KCM), specifically emphasizing aspects of the methodology that address the research gaps identified in Part 1. Part 2 consists of Chapters 5-8. Chapter 5 presents an overview of the KCM and describes its key functional requirements, stakeholders, use cases, and the overall approach. Chapters 6-7 describe the meta-models used for representing simulation templates, and Chapter 8 presents the model transformation methods in KCM.

Part 3 comprises Chapters 9-11. Chapter 9 presents the VTMB test cases including descriptions of simulations templates automatically composed using a proof-of-concept software implementation of KCM's model transformation method. In Chapter

10, a summary of research contributions is presented followed by recommended future work to extend and apply the Knowledge Composition Methodology. In Chapter 11, a summary of the Knowledge Composition Methodology is presented.

This dissertation also includes three appendices. Appendix 1 provides a brief description of basic information modeling concepts used in this dissertation. Appendix 2 provides a summary of OMG Systems Modeling Language (SysML) constructs used in this dissertation; and Appendix 3 provides a brief description of KCM's Generic Properties Meta-Model.

PART 1: PROBLEM DEFINITION

Chapter 2 : PROBLEM DESCRIPTION

This chapter describes the research challenges in formulating simulation templates for VTMB analysis problems. The intent of this description is to characterize simulation template formulation capabilities of existing methods that do not scale to address VTMB challenges, thereby making simulation templates ineffective for multi-disciplinary analysis and optimization problems in particular. First, a set of basic concepts necessary to describe the problem are presented in section 2.1. Then, two key aspects of simulation-based design that are foundational to this research are presented in section 2.2. These aspects establish the need for simulation templates for integrated functional and spatial design. The time and effort required to create simulation templates and the types of changes that result in manually updating simulation templates are discussed in sections 2.2.2.1 and 2.2.2.2. In particular, simulation templates are brittle to a specific type of change—variation in the assembly system topology of design alternatives. Assembly system topology is the main concept used in describing Variable Topology Multi-Body problems in section 2.3. In section 2.4, the primary research question is presented followed by a description of two key research gaps in existing methods for formulating analysis problems.

2.1 Description of basic concepts

In this section, a set of basic concepts necessary to describe simulation templates and VTMB analysis problems are presented.

Idealization is a transformation that relates aspects of a real world system or phenomena to models representing the system or phenomena for the purpose of facilitating mathematical analyses. For example, a linear elastic material behavior is an idealization of the material behavior of an artifact. Similarly, a static force is an idealization of real forces acting on a system. In general, a model (or its aspects) is an idealization of the system or phenomena represented by the model (or its aspects).

An *Artifact* is a distinct subset of a physical product or system (Fenves 2004). An Artifact could be a system itself, such as a specific printed circuit assembly, or any of its sub-systems, such as a printed circuit board used in a printed circuit assembly.

Form[5] represents the physical characteristics of an artifact, such as its shape and material (Fenves 2004). The goal of a design process is to create a form that performs the desired functions.

Function is what an artifact is intended to do (Fenves 2004). An artifact may have several functions, and a function may be performed by several artifacts. A function may be broken down into several sub-functions. Examples of common types of functions are transfer of materials, energy, or information.

Behavior is the response of an artifact to external stimuli (environment). A behavior may be intended—implements an artifact's function, or it may be unintended—doesn't contribute or has adverse effects on an artifact's function. For example, heat generation is an unintended behavior of a microprocessor chip in operation.

A *Behavior Model* represents an idealized subset of behaviors of an artifact in a given environment. The purpose of a behavior model is to answer questions concerning the subset of artifact behaviors that it represents. In the context of this research, a behavior model is formalized as a computable model—one that it can be solved to compute behavior parameters or other measures-of-effectiveness of the artifact. In general, analysts formulate a *Behavior Model Structure* that represents a set of behavior models. Each member of the set is a *Behavior Model Instance* that conforms to a Behavior Model Structure. Structure and instance correspond to the concepts of schema and schema instance (Schenck and Wilson 1994) in information modeling. Like a schema, a behavior model structure represents the parameters and relationships embodied in a behavior

[5] also referred as structure

model. In a behavior model instance, values of some parameters are given while others are computed using the relationships embodied in the structure.

Behavior Model Formulation is a process of designing a behavior model (structure and instances) to compute a set of behavior parameters for a family of artifacts. For example, an analyst would formulate a behavior model to calculate the maximum deformation of a printed circuit board when it is subjected to a thermal load during the assembly process. The formulation of a behavior model consists of the following key steps:

1) *Identifying behavior parameters* to characterize the subset of behaviors that are of interest. For example, the behavior parameters of interest in the PCB deformation problem are the out-of-plane deformation parameter (u_z) and in-plane deformation parameters (u_x and u_y).
2) *Identifying domain theories* that may be used for computing these behavior parameters. Examples of domain theories are Euler's beam theory (Gere and Timoshenko 1997) and Kirchhoff's plate theory (Krauthammer and Ventsel 2001).
3) *Idealizing the artifact and environment* under which behavior parameters are to be computed. For example, in the PCB deformation problem, each stratum of a PCB may be idealized to have homogenous material distribution, the thermal load may be idealized as a uniform temperature increase, and one edge of the PCB may be held fixed as a boundary condition.
4) *Creating a model* that represents the idealized artifact, environment, behavior parameters, and their relationships based on domain theories.

This view of model formulation is in principal also corroborated by (Gruber 1992). This research focuses on solution method- and solver-independent formulation of behavior model structures. A behavior model may be reformulated for specific solution method and solvers. For example, the parameters and relationships in a behavior model may be used to create a solution method-specific system of equations (such as the global stiffness matrix in finite element analysis) that may be solved using a specific solver (such as ABAQUS (Dassault Systemes 2006) for finite element analysis).

Behavior Model Solution is the process of solving the mathematical relationships in a formulated behavior model. During solution, behavior parameters are computed using an appropriate solution method and a solver. Behavior model solution may require reformulations of a behavior model for specific solution methods and solvers.

Simulation is a process of formulating models, solving models, and analyzing results to gain an understanding of a given system (Fishwick 1995). Usually, the term simulation is used when the mathematical relationships in a model do not have an exact or analytical form or they are so complex that it is computationally inefficient to solve them and hence they need to be solved numerically (Law and Kelton 2000). Some examples of systems that are subjects of simulation studies are: products, processes, combination of products and processes, or theoretical systems. In the context of this research, the term simulation is used in a broader sense—includes models that have a closed form solution and those that have to be solved numerically. This research focuses on computer-based simulations (and hence computer-based models) to compute behaviors of artifacts. In the context of this research, the term *simulation model* refers to *behavior model*.

In the context of this research, *behavior simulation* is a process of formulating behavior models, solving them, and evaluating results to gain an understanding of an artifact's behavior under external stimuli. In this dissertation, the terms "simulation" and "behavior simulation" are used interchangeably.

A *Behavior Parameter* is a computable parameter that is used to characterize the behavior of an artifact. The value of a behavior parameter measures the (idealized) behavior of an artifact. Deformation, Stress, and Strain are examples of behavior parameters to measure the structural behavior of an artifact, and Temperature is an example of a behavior parameter to measure the thermal behavior of an artifact.

Analysis is a process of computing the behavior of an artifact from its form. Specifically, the term analysis used here implies *behavior analysis*—computing the behavior of an artifact under external stimuli. Other types of analyses include but are not limited to *requirements analysis*—computing requirements that must be met by an artifact to satisfy the needs of customers, *cost analysis*—computing the cost of producing an artifact given its form.

Evaluation is a process of comparing the behavior(s) of an artifact with the artifact’s function (intended behavior). Specifically, the term evaluation here implies *behavior evaluation.* Other types of evaluation include but are not limited to *requirements evaluation*—checking if an artifact satisfies the requirements, *cost evaluation*—checking if the cost of producing an artifact satisfies budget requirements.

An *Inverse Problem* is a problem where the natural outputs of a behavior model are known but not all the natural inputs to the behavior model are known. The natural outputs of a behavior model are the behavior parameters, and the natural inputs are the artifact’s form, load, and behavior conditions. As an example for static structural analysis of an artifact, the form of the artifact (including geometry and material specifications), boundary conditions, and loads are natural inputs and the deformation, stress, and strain fields are natural outputs (solution). One of the inverse problems for this case would be such that the deformation, stress, and strain fields, loads, boundary conditions, and artifact’s material specifications are inputs to the problem and the form of the artifact is to be determined. Inverse problems are prominent in science and engineering as can be corroborated by a dedicated journal in this topic area (Taylor and Francis (Inverse Problems in Science and Engineering) 2008). The objective of this research is to formulate behavior models such that the relationships between parameters are represented in a non-causal manner (if the relationships are inherently non-causal, such as equations). Non-causal representation of relationships is necessary for solving inverse problems efficiently.

2.2 Aspects of simulation-based design foundational to this research

In this section, two keys aspects of simulation-based design that are foundational to this research are presented. Collectively, these aspects establish a platform for (a) clearly describing the primary question that this research shall answer, and (b) describing the specific research gaps that motivate this research. The context of each aspect is as stated below:

- *Integrated Functional and Spatial Design* aspect establishes that analysis is an activity performed through the design process and requires model-based communication between designers and analysts.
- *Simulation Templates* aspect establishes that simulation templates, patterns, and instances are mechanisms for enabling model-based communication between designers and analysts.

2.2.1 Integrated Functional and Spatial Design

Designers and analysts are key stakeholders in simulation-based design. Figure 2.1 (Fenves 2004) illustrates the integrated functional and spatial design process scenario involving designers and analysts. Designers generate alternative forms of an artifact that are idealized to create analyzable forms for the purpose of analysis and evaluation. The

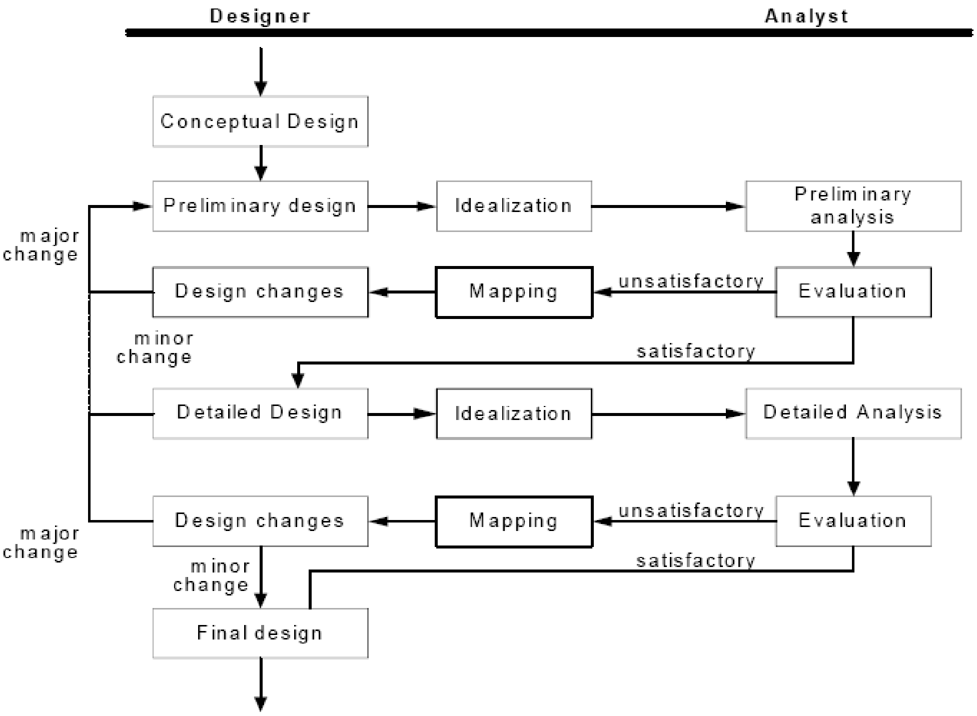


Figure 2.1: Integrated functional and spatial design (Fenves 2004) through design phases

outcome of an analysis and evaluation process is either (i) satisfactory—selecting a set of alternatives that are candidates for the next design phase, or (ii) unsatisfactory—mapping proposed design changes to generate new alternatives. Here, spatial design refers to generating the form of the artifact, and functional design refers to generating an artifact that performs the required functions. The term "integrated" implies that developing the form and function of an artifact are inherently coupled aspects of the design process, and should be performed collaboratively by designers and analysts. It is necessary to view analysis as a process of continuously evaluating an artifact across all design phases. It is necessary that simulation-based design methods enable analyses during conceptual design phases that largely govern the overall product cost and form.

Figure 2.2 illustrates the communication process between designers and analysts for a given design phase by elaborating on the subjects of the idealization and mapping operations illustrated in Figure 2.1. The design of a complex product may require several types of designers and analysts who work collaboratively on their specific aspects. For example, the simulation-based design of an electromechanical product (such as a printed circuit assembly) typically requires the following types of designers and analysts.

- Designers
    - *electronics designers* propose a form to satisfy electronic function,
    - *mechanical designers* propose a form to satisfy mechanical function,
    - *system designers* integrate electrical and mechanical perspectives of a form;
- Analysts
    - *electronic analysts* analyze the electronic behavior of proposed forms,
    - *electromagnetic analysts* analyze the electromagnetic behavior of proposed forms,
    - *thermal analysts* analyze the thermal behavior of the proposed forms, and
    - *structural analysts* analyze the structural behavior of the proposed forms.

In engineering workflows, a single individual may play roles of a designer and an analyst both.

In a given design phase, designers collectively generate several alternatives of an artifact. The nature of analyses to be performed on these alternatives is collectively determined by the following three broad metrics:

- *Type* indicates analysis domain, such as thermal, electromagnetics, and structural.

- *Resolution* indicates the subject of analysis. This could be the artifact system (such as printed circuit assembly), or subsystem (such as a printed circuit board or chip package), or features (such as solder balls and joints).
- *Fidelity* indicates the level of detail incorporated in the behavior model

Based on the nature of a given analysis, the design alternatives are idealized to create analyzable forms. An analyzable form may be used for creating behavior models of different types and fidelities. For example, a structural analyst may use an analyzable form of a PCB to create a 2D (or 3D) thermal (or structural) behavior model.

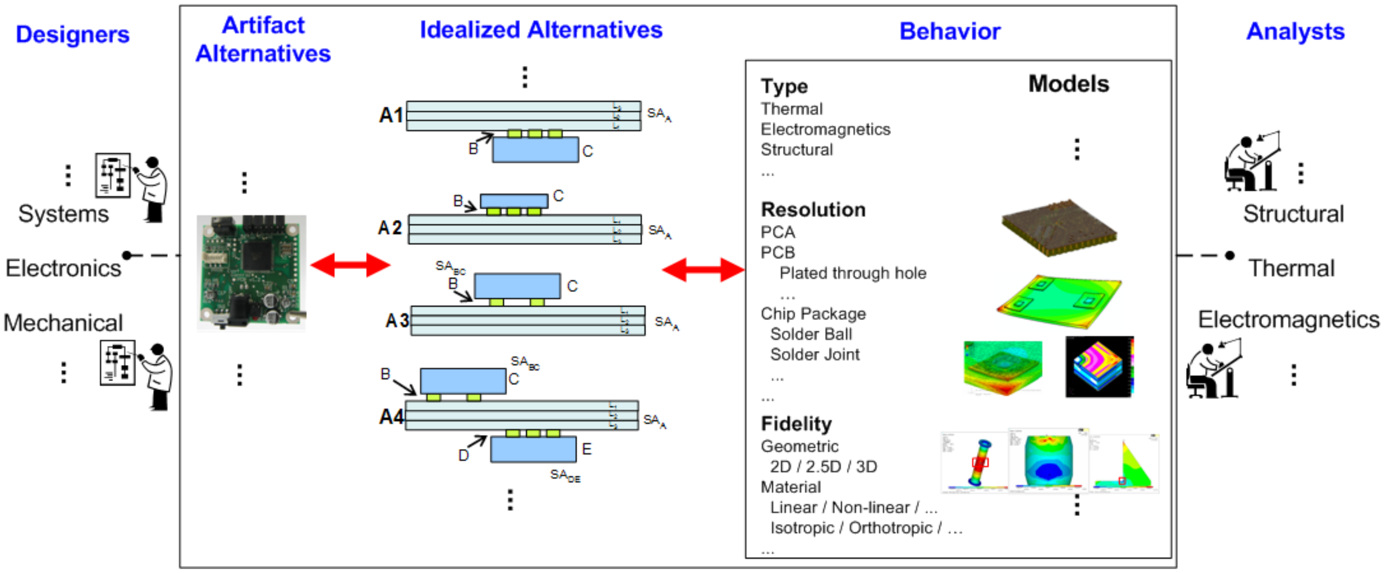


Figure 2.2: Integrated functional and spatial design in a given design phase

As indicated in Figure 2.1 and Figure 2.2, the communication process between designers and analysts is bi-directional. While the designers provide the artifact alternatives to be analyzed, the results of analysis and evaluation performed by analysts are used to propose design changes in the alternatives and to solve inverse problems.

Figure 2.1 and Figure 2.2 also help illustrate the complexity of the communication process between designers and analysts. For a tighter integration between functional and spatial design, it is necessary that such a communication process be founded on model-based templates that provide a mechanism to (a) represent and organize the different types of models exchanged between designers and analysts, (b) represent the fine-grained connections between these models (Peak 2003), and (c) realize the bi-directional flow of information. Such model-based templates can then enable "what-if" trade studies, sensitivity and opportunistic analyses. Simulation templates—described in the next section—contribute towards this objective.

2.2.2 Simulation Templates

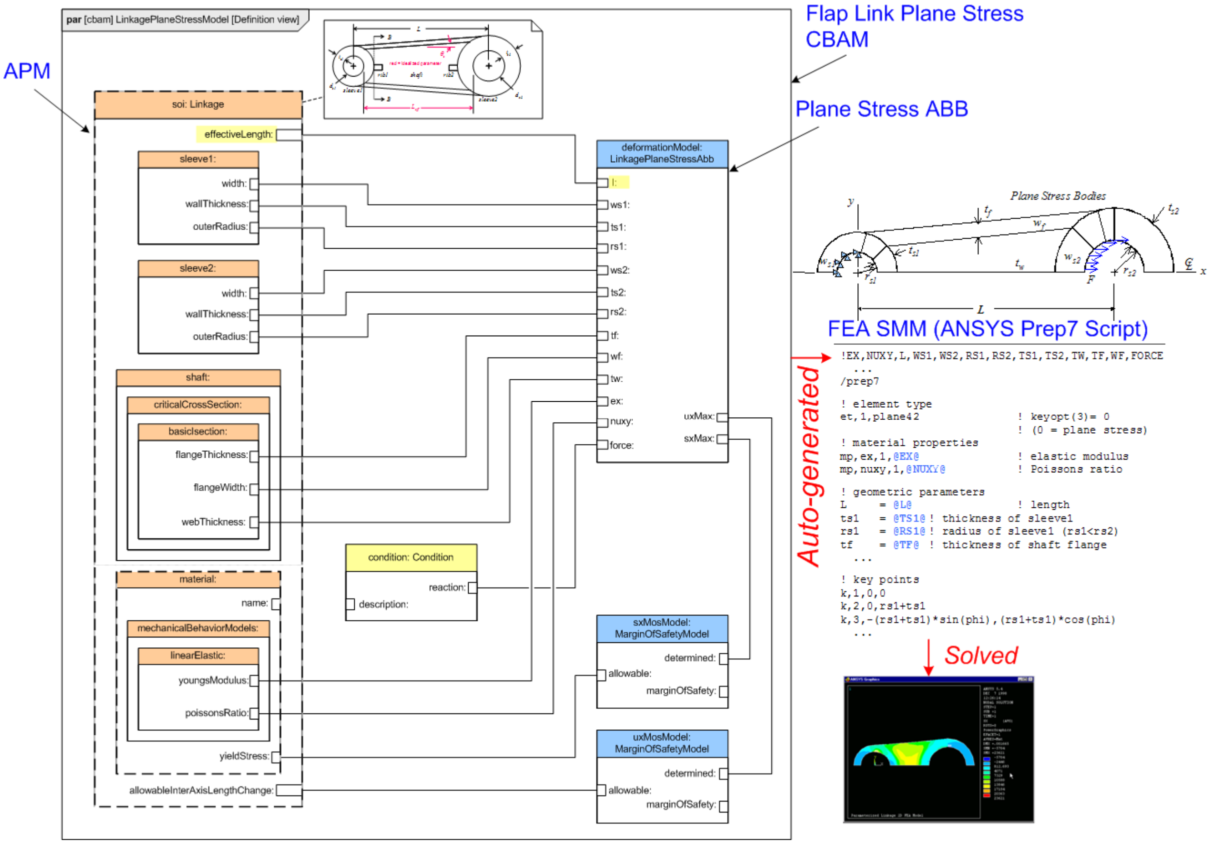
In this section, simulation templates, patterns, and instances are described as enablers for model-based communication between designers and analysts. The effort required for creating simulation templates and the types of changes in design alternatives and analysis specifications that require costly and manual updates to simulation templates are also presented. The brittleness of simulation templates to these types of changes is the challenge problem being addressed by this research. First, the general idea of simulation templates is presented. Then, a specific simulation template pattern relevant to this research is presented. Types of variations in analyses that require manually updating simulation templates are presented in sub-sections 2.2.2.1 and 2.2.2.2. A specific type of variation in design alternatives that affects simulation templates is presented in section 2.3.

A *Simulation Template* is a model structure for formulating and solving a class of simulation models. In the context of this research, simulation templates associate design model structure to behavior model structure, thereby allowing one to compute behavior parameters for different values of design parameters. Thus, simulation templates may be used for formulating and solving all simulation models that conform to the idealization relationships embodied in a simulation template. A simulation template may be categorized as:

- White-box or Black-box: A white-box simulation template exposes the entities, attributes, and relationships that collectively define a simulation template. For a given causality, some attributes are input parameters, some are output parameters, and others may be ancillary or do not contribute to the computation. A black-box simulation template exposes only those attributes that may be inputs or output parameters for the computation process.
- Causal or Non-causal: A causal simulation template has a fixed causality. It consists of a fixed set of input parameters and output parameters. A non-causal simulation template doesn’t have a fixed causality. The causality of the parameters can be changed such that an input parameter for some computations may be an output

parameter for other computations. In such a case, a simulation template can be used for formulating and solving simulation models and also for solving inverse problems. It is to be noted that some relationships between parameters are inherently causal (such as if-else relationships) and hence it is not possible to use simulation templates for all computation directions. An explicit inverse relationship must be defined for a causal relationship to use it for solving inverse problems.

In the context of this research, the key ingredient of a simulation template is the behavior model structure that is associated with the design model structure via idealization relationships. Figure 2.3 illustrates a simulation template that is used for computing the plane-stress behavior of a Flap Link—a mechanical part used in an air frame (Peak, Burkhart et al. 2007). Specifically, it shows a SysML parametric diagram view of the plane-stress deformation model structure whose attributes are connected to the design attributes of the Flap Link part. For example, deformationModel.t is connected to soi.effectiveLength (soi means "system of interest" which is the Flap Link in this case).



```
!EX,NUXY,L,WS1,WS2,RS1,RS2,TS1,TS2,TW,TF,WF,FORCE
  ...
/prep7

! element type
et,1,plane42                    ! keyopt(3)= 0
                                ! (0 = plane stress)
! material properties
mp,ex,1,@EX@                    ! elastic modulus
mp,nuxy,1,@NUXY@                ! Poissons ratio

! geometric parameters
L     = @L@                     ! length
ts1   = @TS1@ ! thickness of sleeve1
rs1   = @RS1@ ! radius of sleeve1 (rs1<rs2)
tf    = @TF@  ! thickness of shaft flange
  ...

! key points
k,1,0,0
k,2,0,rs1+ts1
k,3,-(rs1+ts1)*sin(phi),(rs1+ts1)*cos(phi)
  ...
```

Figure 2.3: Simulation template for computing plane stress behavior of Flap Link part

In this example, the simulation template is used for formulating a behavior model which is then used for auto-generating and solving a finite element analysis model. This template is non-causal in the sense that relationships between the attributes are (a) inherently non-causal, and (b) represented in a non-causal way using SysML binding connectors. Such a template can be used both for formulating and solving a behavior model, and also to compute required design parameters to achieve desired behavior.

Figure 2.4 illustrates both design verification and synthesis scenarios for a simulation template that associates design parameters of the Flap Link part to a linear extensional behavior model. When this simulation template is used in the design verification scenario, the form-related attributes of the Flap Link part and the end forces (condition.reaction) on the part are inputs while the elongation of the part and axial stresses and strains are outputs. When the same simulation template is used in the design synthesis scenario, some form-related attributes of the Flap Link part are outputs and the elongation and end forces are inputs. For example, deformationModel.area is computed in the design synthesis scenario and not an input from soi.Shaft.criticalCrossSection.basic.area. Hence, a simulation template is instantiated for a specific computation. In this case, the simulation template that embodies an extensional behavior model of the Flap Link part is instantiated twice—once for design verification scenario and once for design-synthesis scenario.

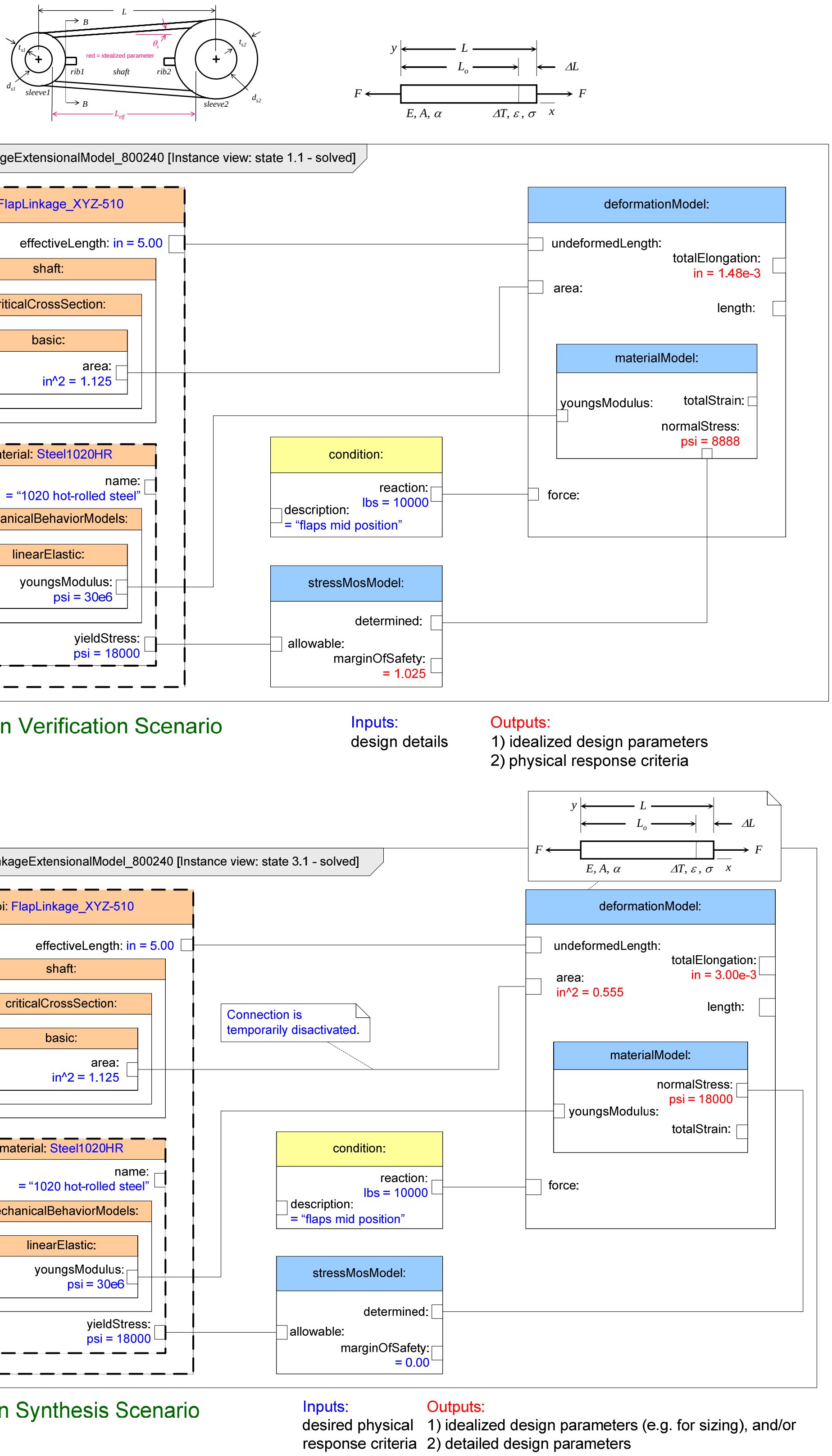


Figure 2.4: Instances of a simulation template that embodies a linear extensional behavior model of Flap Link part

A *Simulation Template Pattern* is a meta-model of a simulation template. It represents the types of models, their attributes, and their inter-relationships in a family of simulation templates. The Multi-Representation Architecture (Peak and Fulton 1994; Peak, Fulton et al. 1998; Peak, Paredis et al. 2005) is a simulation template pattern for creating simulation templates that can provide a foundation for model-based communication between designers and analysts through the design process. The rationale behind the MRA pattern—illustrated in Figure 2.5—is to have modular components that can be reused in different templates. The MRA pattern consists of four stepping stone models:

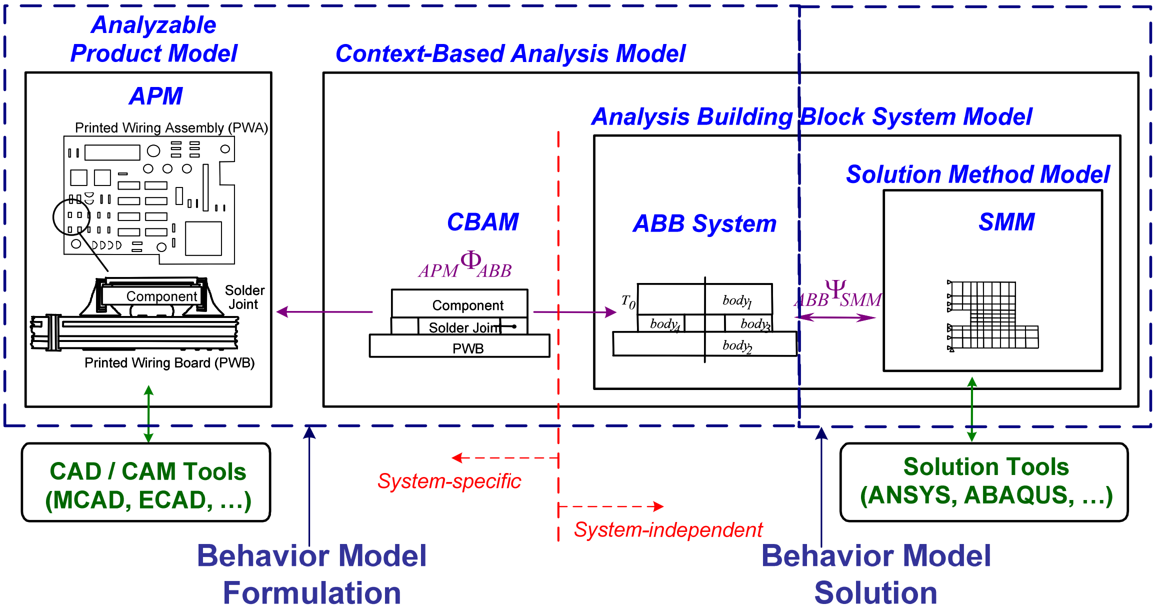


Figure 2.5: Multi-Representation Architecture (Peak, Fulton et al. 1998) — A simulation template pattern showing the behavior model formulation and solution sub-patterns

- *Analyzable Product Model (APM)* represents an idealized design model with additional analysis intents, and is created for a family of analyses.
- *Context–based Analysis Model (CBAM)* represents product-specific simulation templates that capture the relations ($_{APM}\Phi_{ABB}$) between the APM and ABB system model.
- *Analytical Building Block (ABB) System Model* represents a system composed of reusable analysis concepts that encapsulate domain knowledge. These reusable analysis concepts are known as analysis building blocks (ABBs)—for example *linear elastic material behavior* ABB and *point-load* ABB. A behavior model of an artifact is

formalized as a CBAM that includes the ABB system model and its relationships to the APM.

- *Solution Method Model (SMM)*: Represents a solution method-specific behavior model, such as a finite element model.

All the four models above have an explicit structure and may have several instances that conform to this structure. Figure 2.3 and Figure 2.4 illustrate simulation templates that are based on the MRA pattern. In Figure 2.3, the Flap Link plane stress CBAM is shown—relates the Flap Link APM and the plane stress ABB model. The FEA SMM model is auto-generated from this CBAM. In Figure 2.4, the Flap Link linear extension CBAM is shown—relates the Flap Link APM to the linear extension ABB (deformationModel). The SMM—not shown in the figure—is a Mathematica (Wolfram Mathematica 2008) model auto-generated from this CBAM. An APM is created for a family of analyses, and different APMs may be created for analyses of different fidelities—one APM for 2D analyses and one for 3D analysis. The Flap Link APM used in the plane stress CBAM includes the two sleeves and the shaft features of the Flap Link part. But, the Flap Link APM used in the linear extension CBAM includes only the shaft feature. Though not explicitly shown here, the MRA pattern has been extended to include the as-designed (DM) or as-manufacturable product model (MPM)[6] structure (Zwemer, Bajaj et al. 2004) and their relationships to the APM structure.

Figure 2.5 also shows two sub-patterns that are related to behavior model formulation and solution respectively. In the context of this research, the formulation of behavior model (structure or instance) implies the formulation of the CBAM (structure or instance), i.e. the ABB system model (structure or instance) and its relationships to the APM (structure or instance). In effect this is the formulation of a simulation template itself. The solution of a behavior model may require the re-formulation of behavior models as SMMs for specific solution methods and solvers. In this dissertation, the term *simulation template* is used to refer to design-analysis simulation templates based on the MRA pattern. Note that the MRA is a broad and generic pattern and specific simulation templates may instantiate the MRA in entirety or in part.

[6] These DM and MPM are shown explicitly in the formalized MRA pattern included in KCM (Part 2 of this dissertation).

2.2.2.1 Effort in creating simulation templates

The cost-benefit ratio of simulation templates depends on the type of analysis—original, adaptive, or ubiquitous (routine) (Peak 1993; Peak, Scholand et al. 1999; Bajaj 2006). For ubiquitous analysis, simulation templates are created once and reused for different causalities, and different input values for a given causality. In this case, the cost of creating simulation templates is amortized with usage. However for the case of adaptive and original analyses, new simulation templates need to be created or existing simulation templates need to be modified for analysts to perform trade studies on idealizations. This involves manually creating CBAMs by (a) instantiating ABBs from a library, and (b) establishing connections among the ABBs, and from ABB attributes to APM attributes. As an example, the Flap Link plane stress simulation template shown in Figure 2.3 consists of 17 relationships between APM attributes and ABB system attributes. In order to create such a simulation template, usages of APM and ABBs needs to be created in the simulation template (assuming that relevant APM and ABBs already exist), and 17 relationships have to be manually created among APM and ABB system model attributes. This involves significant time and effort on an analyst's part to create and maintain the relationships (a.k.a. associativities) between the models (Peak 2003). The Flap Link is a single part. For complex multi-level assemblies where each component is idealized differently, the number of entities and the number of relationships that need to be created between these entities in a simulation template increases significantly.

This research focuses on automated creation of simulation templates based on specifications provided by analysts. This reduces the cost-benefit ratio of simulation templates esp. for adaptive and original analyses (conditional to the availability of ABBs).

2.2.2.2 Robustness of simulation templates

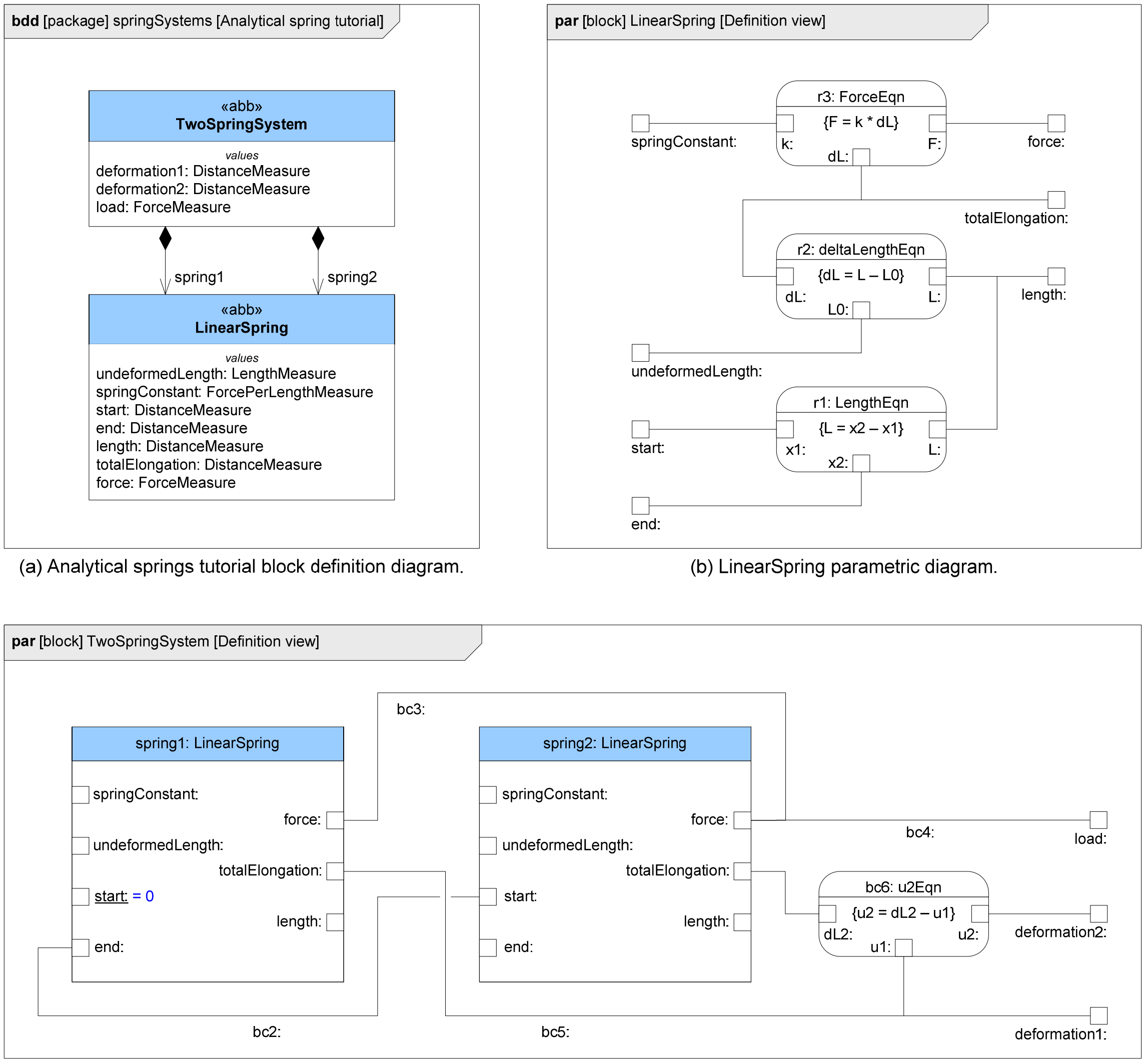
The structure of a simulation template holds for the specific (a) type of analysis for which it is created, (b) family of artifacts for which it is created, and (c) idealizations that are represented in it — MPM-APM type idealizations and APM-ABB type idealizations (CBAM). As an example, the linear extension simulation template shown in Figure 2.4 is created for static linear extension analysis of Flap Link part where the part is idealized as a linear extensional rod. Analysts may use this simulation template with different parameter values and do trade studies with different causalities. If however, there are design alternatives of the Flap Link part that have other analyzable features in addition to the sleeves and the shaft, then these simulation templates have to be manually modified to include entities related to those features and establish relationships between the additional analyzable features and their corresponding ABBs. Also, if the applied forces were not idealized as axial loads but as eccentric loads, then the bending behavior would need to be computed in addition to the extension. This would imply including the ABB for bending behavior in the simulation template and establishing connections to the APM entities. Additionally, if the type of analysis were dynamic and non-linear in the sense that deformations of the Flap Link part were not small but were large enough to change the point of application of applied loads and deform the part substantially such that new features like a surface recess or a crack develop on the part, then the above simulation templates would need substantial modification—finding the right ABBs, include them in the templates, and establishing relationships to the APM. In a nutshell, the structure of a simulation template changes with the following three types of changes in the analysis specifications:

- ***ST_Change_Type_1*: Changes in a simulation template due to changes in assembly system topology of design alternatives**

  This category includes changes in simulation templates due to a change in assembly system topology (defined in section 2.3) of design alternatives and corresponding APMs. These changes are such that they may affect a change in the number of analysis bodies in the ABB system model. Figure 2.6 illustrates a LinearSpring ABB—single spring with linear behavior, and TwoSpringSystem (ABB system)—two springs with linear behavior connected in series. An analyst may use the LinearSpring simulation template for computing the linear behavior of a single spring. If the assembly system changes such that two linear springs are connected in series, then another usage of the LinearSpring ABB must be created in the simulation template and both usages of LinearSpring ABB must be connected to reflect the series connection (such as the end point of one spring is associated with the start point of the other spring).

If a third spring needs to be added in series or parallel to both or either of the springs, then it would imply creating an additional usage of LinearSpring ABB and establishing relationships between this usage and previous two usages to reflect the modified system.

In general, this type of change in assembly system topology helps define a special class of analysis problems known as Variable Topology Multi-Body Analysis problems (defined later in section 2.3). This research specifically focuses on this class of analysis problems.



(a) Analytical springs tutorial block definition diagram.

(b) LinearSpring parametric diagram.

(c) TwoSpringSystem parametric diagram.

Figure 2.6: LinearSpring (ABB) and TwoSpringSystem (ABB system) examples—SysML parametric diagram view

- ***ST_Change_Type_2*: Changes in a simulation template due to changes in the idealization decisions taken by analysts**

This category includes changes in simulation template due to the changes in the idealization decisions taken by analysts. This includes idealization decisions concerning (a) MPM-APM relationship—how an analyzable product model is idealized from a design model or manufacturable product model for a class of analyses, and (b) APM-ABB system—how is the behavior of the analyzable product model idealized.

If MPM-APM idealization decisions are such that they result in a change in the assembly system topology of the assembly system in the APM, then these changes are included in ST_Change_Type_1 category above. However, if the idealization decisions are such that the nature of the relationship(s) between MPM attribute(s) and APM attribute(s) change, then they are included in this category. For example, if the effective length attribute in the Flap Link APM in the plane stress CBAM in Figure 2.3 were to be computed differently the Flap Link design model, then such an idealization change would be included in this category. Changes in the type and (or) fidelity of analyses that the APM is required to support will affect changes in the MPM-APM idealizations.

The APM-ABB system idealization changes result in changes in a simulation template by using different type of ABB for idealizing the behavior of the artifact. These type of idealization changes occur due to changes in the type of analysis and (or) the fidelity of analysis. For example, an analyst may perform a 2D plane stress analysis or a relatively lower fidelity 1D linear extension analysis to compute the axial deformation of the Flap Link part.

This research focuses on automatically generating simulation templates based on the specifications provided by analysts. Changes in the idealization decisions are reflected as changes in the specifications. The updated specifications may then be used to regenerate the simulation templates using methods developed in this research.

- ***ST_Change_Type_3*: Changes in simulation template due to simulated behavior**

This includes changes in simulation template due to non-linear analysis. These changes typically occur when (a) idealized behavior(s) of an artifact affect a change in the assembly system topology of the artifact itself, and/or (b) different set of idealizations need to be applied for different analysis regimes. For example, if the deformation of an assembly is large such that the connection between any two components breaks when simulating the behavior of the assembly, then this should be reflected in the simulation template by deactivating the interaction behavior between the corresponding components (analysis bodies) during the course of simulation. Further, an analyst may select a conditional idealization such that if the deformation is within a specified range, a different set of idealizations are in-effect (a separate set of ABBs in the ABB system) versus if the deformation is outside the specified range. These use cases are distinguished by using the terms static simulation template versus dynamic simulation template.

This research proposes a conceptual approach for handling these types of changes in simulation templates. Note that the brittleness of simulation templates also depends on the manner in which relationships are created between models in a simulation template. For example, if the geometric idealization relationships in a simulation template are represented using a generic scheme such as Affine transformations (Mortenson 1997), then they are more robust to changes in the type of shapes at the input end of the idealization relationship—the structure of the idealization relationship can handle wider varieties of input shapes. However, if a geometric idealization relationship is represented by a set of relationships between the attributes of specific shapes and their features, then they are brittle to changes in the type of shapes that are being idealized. Further, the use of logical relationships (such as IF-THEN relationship) can enhance the robustness of simulation templates to types of changes in ST_Change_Type_3 category. This research is also aimed at developing guidelines for creating robust simulation templates.

2.3 Variable Topology Multi-Body (VTMB) Problems

As described in the previous section, changes in the assembly system topology (AST) of design alternatives result in changes[7] in the structure of simulation templates using these models. In this section, the concept of Assembly System Topology (AST) is defined and illustrated. This dissertation defines a special type of graph construct and corresponding visualization diagram—an Assembly System Topology diagram—to help characterize VTMB problems and visualize and communicate changes in AST. Following the definition of AST and AST diagram, the specific subsets of a simulation template that AST changes impact and the conditions for these changes are presented. Founded upon the concept of AST and AST diagram, a special class of analysis problems, namely *Variable Topology Multi-Body (VTMB) Problems*, is defined in this dissertation. This research is aimed at addressing VTMB problems.

What is Assembly System Topology and how can it be characterized?

Assembly System Topology (AST) is a property of an assembly system that is used to collectively characterize (a) the number and type of components in an assembly, (b) the number and type of interactions between these components, and (c) the number and type of component features that participate in these interactions. Since AST is a collective characteristic, it is easier and pragmatic to compare if two assembly systems have equivalent AST rather than computing an absolute value of AST for an assembly system. The AST of two assembly systems is equivalent if and only if

a) They have the same number of components of each type

b) Each component has the same number and type of features

c) The type and number of interactions between any two features is the same

Let us denote the AST of an assembly system AS_i as AST(AS_i), then we can define the AST Equivalence Relation as follows.

AST Equivalence Relation, *AST_EQ* (denoted as ~), is a binary relation between the AST of two assembly systems AS_i and AS_j that implies that the AST of AS_i and AST of

[7] except when the idealizations ignore the components and interactions involved in the change

AS_j are equivalent. This relation is denoted as: $AST(AS_i) \sim AST(AS_j)$. The AST Equivalence relation is:

- Reflexive: $AST(AS_i) \sim AST(AS_i)$ — implies that the AST of an assembly system AS_i is equivalent to itself.
- Symmetric: If $AST(AS_i) \sim AST(AS_j)$, then $AST(AS_j) \sim AST(AS_i)$ — implies that is the AST of assembly system AS_i is equivalent to the AST of assembly system AS_j, then by definition of AST, the AST of assembly system AS_j is equivalent to the AST of assembly system AS_i.
- Transitive: If $AST(AS_i) \sim AST(AS_j)$ and $AST(AS_j) \sim AST(As_k)$, then $AST(AS_i) \sim AST(AS_k)$ — implies that if the ASTs of assembly systems AS_i and AS_j are equivalent, and the ASTs of assembly systems AS_j and AS_k are equivalent, then by definition of AST, the ASTs of assembly systems AS_i and AS_k are equivalent.

An ***AST Equivalence Set***, *AST_EQ_Set*, is a set of all assembly systems such that the ASTs of any two members of the set, ASi and ASj, are equivalent. Thus, an AST_EQ_Set is defined as: $\forall\ AS_i\ ,\ AS_j \in AST\_EQ\_Set,\ AST(AS_i) \sim AST(AS_j)$

An ***Assembly System Topology Diagram*** (*AST diagram*) is a type of SysML Internal Block Diagram that depicts an assembly system and its components, features of components, and the interactions between components. Hence, the AST diagrams of two assembly systems can be compared to unambiguously decide if have equivalent AST.

Figure 2.7, Figure 2.9, Figure 2.10, and Figure 2.11 help illustrate the concepts of AST and AST diagram. Figure 2.7 shows a set of parts (or bodies)[8,9]—A, B, C and D—using which several assembly configurations are composed. Figure 2.9 illustrates a set of assembly systems with equivalent AST while Figure 2.10 and Figure 2.11 each illustrate a set of assembly systems with non-equivalent AST.

[8] AST can be used to characterize the topology of assemblies at the MPM (and APM) level where the part-component terminology is used, or the ABB system level where analysis body and analysis body system terminology is used.

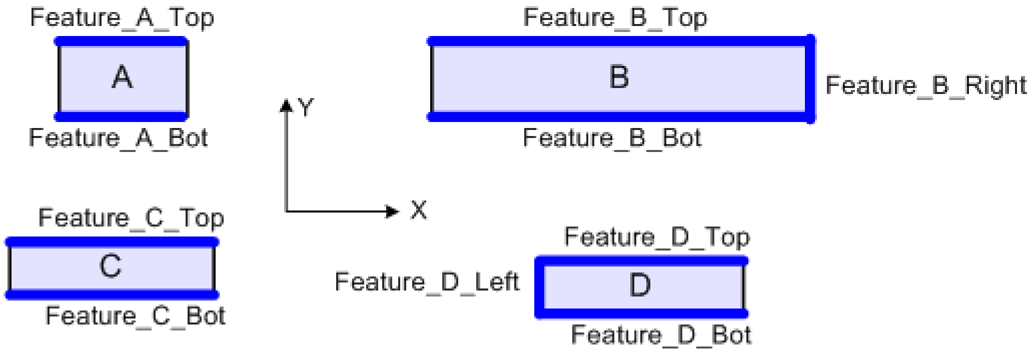


Figure 2.7: Parts[9] *and their features*

The constructs of an AST diagram are illustrated in Figure 2.8 and described below.

- *Assembly system block* is used to represent assembly systems. It is denoted as a SysML block
- *Component block* is used to represent components of an assembly system. Each component block is identified by the name of the component and its type (part name). For example, as shown in Figure 2.8, component_A1 is of type Part_A. A component block is labeled as component_A1: A. It is denoted as a SysML part property and is shown inside its parent assembly system block.
- *Feature block* is used to represent features of assembly components that participate in defining the interaction between components. Each feature block is identified by the name of the feature and its type, and is shown inside the component block corresponding to its parent component. A feature is a part of the component's form that participates in the interactions between the components. For example, Figure 2.8 shows that component_A1 has two features, Feature_A_Top and Feature_A_Bot of type Feature_Type. Features may be typed according to their shape (such as point feature, line feature, or surface feature), their constituent material(s) (such as copper features, solder features), their function (such as electrically conductive feature or electrically non-conductive feature), or other characteristics relevant to tracking them

[9] Name of components and corresponding parts used in this section have the prefixes *component_* and *part_*. For brevity, the prefixes are not shown in the assembly configurations.

in the product realization process. It is denoted as a SysML part property and is shown inside its parent component block.

- *Interaction block* is used to represent interactions between features (and hence components) in an assembly. Each interaction block is identified by its name and type. Interactions are typically typed by their function (such as structural, thermal, or electrical function). For example, in Figure 2.8, component_A1 and component_B1 are glued together and this is represented by the interaction block, A1_B1_Interaction of type Glued_Interaction between Feature_A_Bot (of component_A1) and Feature_B_Top (of component_B1). It is denoted as a SysML part property.

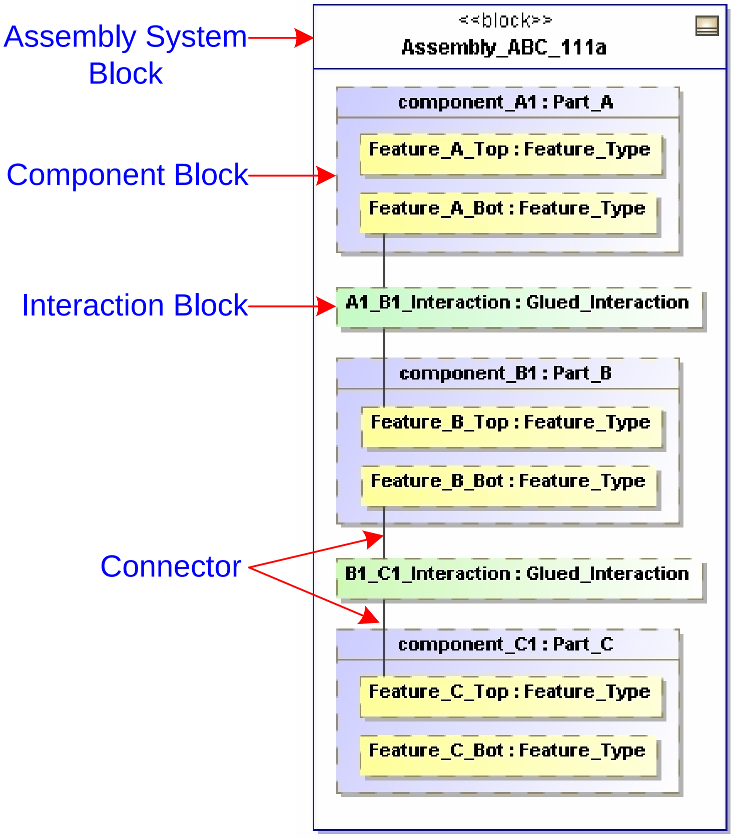

Assembly System Block
Component Block
Interaction Block
Connector
<<block>>
Assembly_ABC_111a
component_A1 : Part_A
Feature_A_Top : Feature_Type
Feature_A_Bot : Feature_Type
A1_B1_Interaction : Glued_Interaction
component_B1 : Part_B
Feature_B_Top : Feature_Type
Feature_B_Bot : Feature_Type
B1_C1_Interaction : Glued_Interaction
component_C1 : Part_C
Feature_C_Top : Feature_Type
Feature_C_Bot : Feature_Type

Figure 2.8: AST diagram constructs

Figure 2.9 shows three assembly systems Assembly_ABC_111a1, Assembly_ABC_111a2, and Assembly_ABC_111a3 that have equivalent ASTs, as illustrated by the AST diagram in the figure. Hence, these assembly systems belong to the same AST Equivalence Set. One may draw a single AST diagram for an AST Equivalence Set since the AST diagrams for all members in the set are isomorphic.

Figure 2.9 also illustrates that changes in the size and shape of components and even the geometric topology of components doesn't necessarily affect the AST of the assembly system. For example, the size of component A1 in Assembly_ABC_111a1 is different than in Assembly_ABC_111a2, and the geometric topology of component A1 in Assembly_ABC_111a3 is different than in other two assembly systems. These changes do not affect the AST diagram, and hence by definition do not affect the AST and simulation templates[10].

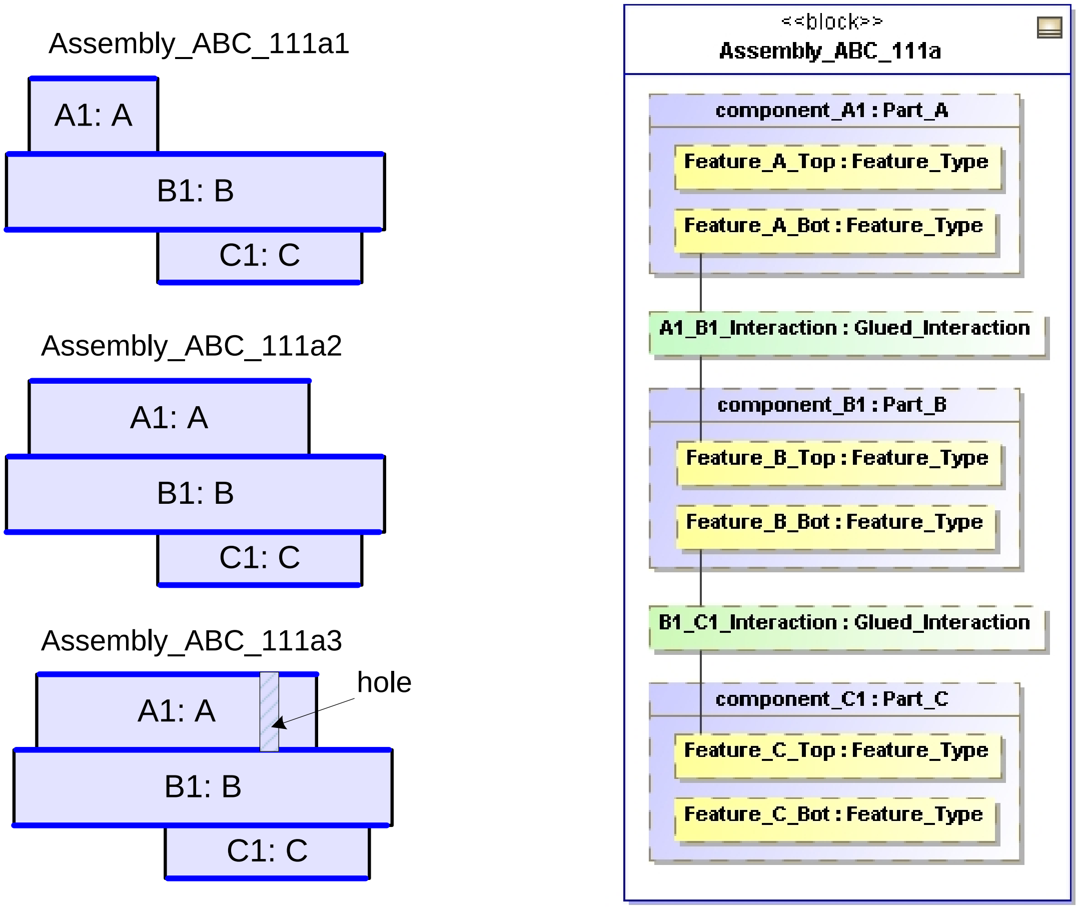

Assembly_ABC_111a1
A1: A
B1: B
C1: C
Assembly_ABC_111a2
A1: A
B1: B
C1: C
Assembly_ABC_111a3
hole
A1: A
B1: B
C1: C
<<block>>
Assembly_ABC_111a
component_A1 : Part_A
Feature_A_Top : Feature_Type
Feature_A_Bot : Feature_Type
A1_B1_Interaction : Glued_Interaction
component_B1 : Part_B
Feature_B_Top : Feature_Type
Feature_B_Bot : Feature_Type
B1_C1_Interaction : Glued_Interaction
component_C1 : Part_C
Feature_C_Top : Feature_Type
Feature_C_Bot : Feature_Type

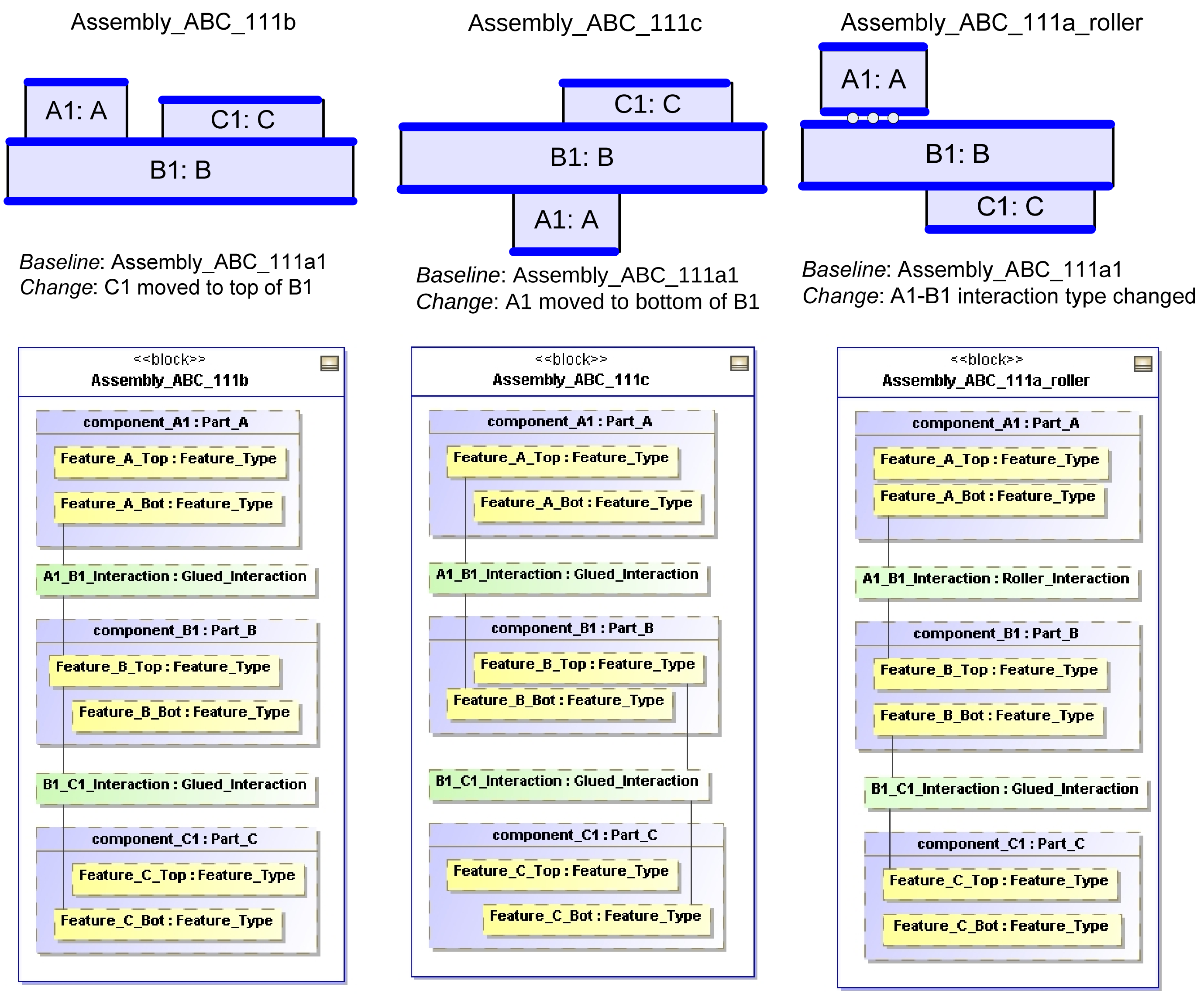
Figure 2.9: Assemblies with equivalent system topologies; ST diagram as SysML IBD

Figure 2.10 illustrates changes in AST due to reconfiguration of existing components. In assembly system Assembly_ABC_111b, component C1 is moved to the

[10] As stated in the previous section, this assumes that an analyst has defined geometric idealization relationships at the object level and not at the attribute level. For example, using Affine transformations for idealizing shapes versus relating attributes of shape by algebraic relationships.

top of component B1 with respect to assembly system Assembly_ABC_111a1. This change in reflected in AST diagram for Assembly_ABC_111b—Feature_B_Top and Feature_C_Bot are associated with B1_C1_Interaction instead of Feature_B_Bot and Feature_C_Top in the AST diagram for Assembly_ABC_111a1 in Figure 2.9. Similarly in assembly system Assembly_ABC_111c, component A1 is moved to the bottom of component B1 with respect to assembly system Assembly_ABC_111a1. The corresponding changes are reflected in the AST diagram of Assembly_ABC_111c. In assembly ABC_111c_roller, the interaction type between components A1 and B1 has changed from glued interaction to roller interaction—A1 and B1 can mutually slide along the interacting surface as opposed to being glued in Assembly_ABC_111a. The changes in the AST diagrams for these three assembly systems with respect to the AST diagram for assembly system Assembly_ABC_111a reflects that the AST of these three assembly systems is (a) not equivalent to the AST of Assembly_ABC_111a, and (b) not equivalent to the AST of each other.

Figure 2.11 illustrates changes in AST due to addition of new components. These changes are reflected in the AST diagram as addition of new component blocks and feature blocks—representing new components and their features, and interaction blocks and connectors—representing new interactions among new and existing components. In Assembly_ABC_211, a new component A2 (usage of part A) is added to the assembly, and in Assembly_ABCD_1111, a new component D1 (usage of part D) is added to the assembly. The AST of these two assembly systems is not equivalent to the AST of Assembly_ABC_111a1, and to the AST of each other.



Assembly_ABC_111b

Baseline: Assembly_ABC_111a1
Change: C1 moved to top of B1

Assembly_ABC_111c

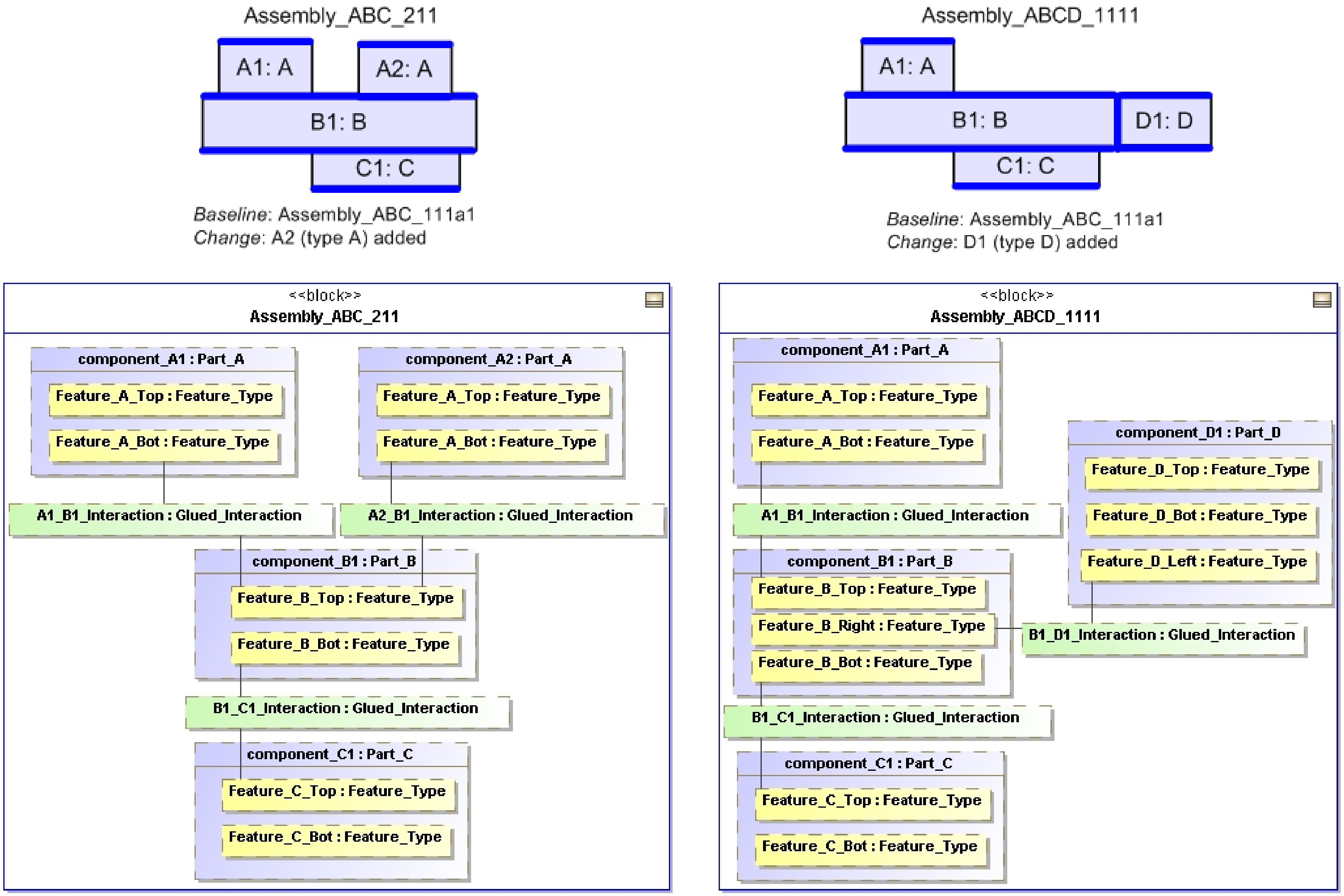
Baseline: Assembly_ABC_111a1
Change: A1 moved to bottom of B1

Assembly_ABC_111a_roller

Baseline: Assembly_ABC_111a1
Change: A1-B1 interaction type changed

A1: A
C1: C
B1: B
C1: C
B1: B
A1: A
A1: A
B1: B
C1: C
<<block>>
Assembly_ABC_111b
component_A1 : Part_A
Feature_A_Top : Feature_Type
Feature_A_Bot : Feature_Type
A1_B1_Interaction : Glued_Interaction
component_B1 : Part_B
Feature_B_Top : Feature_Type
Feature_B_Bot : Feature_Type
B1_C1_Interaction : Glued_Interaction
component_C1 : Part_C
Feature_C_Top : Feature_Type
Feature_C_Bot : Feature_Type
<<block>>
Assembly_ABC_111c
component_A1 : Part_A
Feature_A_Top : Feature_Type
Feature_A_Bot : Feature_Type
A1_B1_Interaction : Glued_Interaction
component_B1 : Part_B
Feature_B_Top : Feature_Type
Feature_B_Bot : Feature_Type
B1_C1_Interaction : Glued_Interaction
component_C1 : Part_C
Feature_C_Top : Feature_Type
Feature_C_Bot : Feature_Type
<<block>>
Assembly_ABC_111a_roller
component_A1 : Part_A
Feature_A_Top : Feature_Type
Feature_A_Bot : Feature_Type
A1_B1_Interaction : Roller_Interaction
component_B1 : Part_B
Feature_B_Top : Feature_Type
Feature_B_Bot : Feature_Type
B1_C1_Interaction : Glued_Interaction
component_C1 : Part_C
Feature_C_Top : Feature_Type
Feature_C_Bot : Feature_Type

Figure 2.10: Change in AST due to reconfiguration (changes in interactions and participating features)


Assembly_ABC_211
A1: A
A2: A
B1: B
C1: C
Assembly_ABCD_1111
A1: A
B1: B
D1: D
C1: C
<<block>>
Assembly_ABC_211
component_A1 : Part_A
Feature_A_Top : Feature_Type
Feature_A_Bot : Feature_Type
component_A2 : Part_A
Feature_A_Top : Feature_Type
Feature_A_Bot : Feature_Type
A1_B1_Interaction : Glued_Interaction
A2_B1_Interaction : Glued_Interaction
component_B1 : Part_B
Feature_B_Top : Feature_Type
Feature_B_Bot : Feature_Type
B1_C1_Interaction : Glued_Interaction
component_C1 : Part_C
Feature_C_Top : Feature_Type
Feature_C_Bot : Feature_Type
<<block>>
Assembly_ABCD_1111
component_A1 : Part_A
Feature_A_Top : Feature_Type
Feature_A_Bot : Feature_Type
component_D1 : Part_D
Feature_D_Top : Feature_Type
Feature_D_Bot : Feature_Type
Feature_D_Left : Feature_Type
A1_B1_Interaction : Glued_Interaction
component_B1 : Part_B
Feature_B_Top : Feature_Type
Feature_B_Right : Feature_Type
Feature_B_Bot : Feature_Type
B1_D1_Interaction : Glued_Interaction
B1_C1_Interaction : Glued_Interaction
component_C1 : Part_C
Feature_C_Top : Feature_Type
Feature_C_Bot : Feature_Type

Baseline: Assembly_ABC_111a1
Change: A2 (type A) added

Baseline: Assembly_ABC_111a1
Change: D1 (type D) added

Figure 2.11: Change in AST due to addition of new components (and hence also addition of new interactions)

The AST of a family of design alternatives may be different from AST of another family of design alternatives. Changes in the AST of design alternatives require changes in simulation templates. For example, addition of new components and interactions require analysts to manually create new entities, parameters, and relationships in simulation templates, or the changes in the type of interaction between components require analysts to re-wire existing relationships between parameters in a simulation template. Specifically, in the MRA simulation template pattern, each stepping stone model consists of a representation of an assembly system. Table 2.1 shows the types of assembly system and components represented in design-analysis models used in MRA simulation template pattern.

Table 2.1: Assembly system and components in design-analysis models used in MRA simulation template pattern

Model in MRA pattern	***Assembly System***	***Components***
Design model (DM) / Manufacturable product model (MPM)	Design assembly / Manufacturable product assembly	Sub-assemblies and parts
Analyzable Product Model	Idealized DM / MPM assembly	Analyzable sub-assemblies and components
ABB System Model	Analysis Body System	Analysis bodies (e.g., plates and shells)
Solution Method Model	Assembly of solvable elements (e.g., meshed assembly in FEA)	Solvable elements (e.g., mesh elements in FEA)

The AST of idealized assembly system in an APM depends on the AST of the design model (or MPM) and the idealization relationships between them. Similarly, the AST of the analysis body system in ABB system model depends on the AST of the idealized assembly in the APM and the idealization relationships between then (${}_{APM}\Phi_{ABB}$). Hence, changes in assembly system topology of design alternatives require updates to simulation templates that are generally done manually.

In this context, ***Variable Topology Multi-Body (VTMB) Problems*** are a class of problems where the assembly system topology of design alternatives varies. In the context of simulation-based design VTMB problems affect simulation templates, generally requiring manual updates and "re-wiring" of parameters and relationships in a template. The acronym VTMB is used instead of the complete phrase *variable topology multi-body* in this dissertation.

Note that variable topology multi-body problems are defined here based on the concept of assembly system topology and not geometric topology. The definition of variable topology presented here is different from highly-coupled variable topology problems defined by Zeng (Zeng, Peak et al. 2008) where changes in the geometric topology of interconnected bodies pose FEA meshing challenges.

2.4 Primary Research Question and Gaps

2.4.1 Primary Research Question

The primary question that this research answers is as follows:

How can we improve the effectiveness of the analysis problem formulation process for VTMB problems?

In this sub-section, three measures of effectiveness of analysis problem formulation are described. These measures provide means to characterize why existing methods are ineffective for formulating analysis problems, and to characterize how methods developed in this research are more effective.

The term “analysis problem formulation” in the primary research question refers to the formulation of simulation templates. A simulation template provides a structure to create a class of behavior models for a class of design models. The value of simulation templates in performing what-if trade studies on design alternatives has been established in the previous sections. The term “process” in the primary research question refers to the way in which simulation templates are created in existing methods.

The term “effectiveness” in the primary research question sums up the core of the research problem. Figure 2.12 below illustrates three measures of effectiveness of analysis problem formulation in the context of this research.

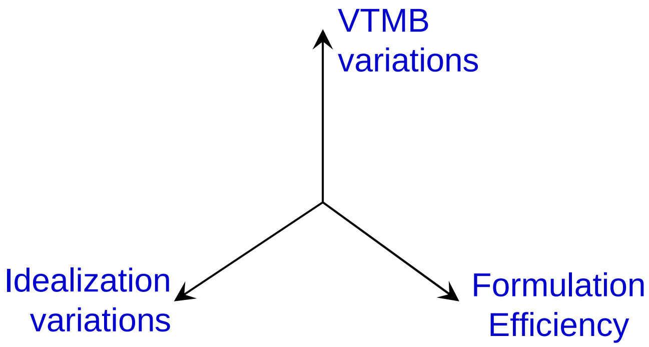


Figure 2.12: Measures of effectiveness of analysis problem formulation

These measures of effectivess are described below.

- *VTMB variations*: This measure-of-effectiveness concerns the ability of analysis problem formulation methods to address VTMB problems. As discussed in previous sections, simulation templates (formulated analysis problems) are generally brittle to variations in assembly system topology of design alternatives. This makes simulation templates ineffective for design optimization problems where they are used for computing parameters that directly or indirectly participate in the objective function.
- *Idealization variations*: This measure-of-effectiveness concerns the ability of analysis problem formulation methods to handle variations in idealization decisions taken by analysts. The idealization decisions taken by analysts are embodied in simulation templates as design and behavior parameters and relationships between these parameters. As discussed in the previous section, for new types of analyses, analysts perform what-if trade studies on idealizations and compare results from different behavior models, such as low-fidelity, easy-to-solve models and high-fidelity, complex-to-solve models.
- *Formulation Efficiency*: This measure-of-effectiveness concerns the ability of analysis problem formulation methods to create simulation templates in an efficient manner. In this dissertation, formulation efficiency is characterized in terms of percentage reduction in time take to formulate simulation templates using new methods (developed in this research) versus current methods. Section 9.5.3.3 describes how the percentage reduction in time is measured.

In the context of this research, the following functional aspects contribute towards increasing formulation efficiency.

1. Automated methods for formulating simulation templates that are based on easy-to-modify analysis specifications and simulation template meta-model.
2. Existence of meta-models for formally representing simulation templates for VTMB problems.
3. Analysis specifications that abstract the idealization decisions taken by analysts from the details of the formulation process. This will allow analysts to change idealization decisions without manually reconfiguring the formulation process.
4. Abstraction of building blocks of simulation templates that can be used for formulating a large class of simulation templates. Each simulation template is used for a class of analysis problems.
5. Methods for formulating simulation templates are modular and extensible to allow usage of different building blocks, such as shape and material behavior, for different types of analysis problems.

In the context of this dissertation, an analysis problem formulation method is highly effective if it scores high on all the three measures of effectiveness. This implies that the analysis problem formulation method is effective if:

- it can be used for creating simulation templates for greater types of design variations, specially VTMB-type variations
- it can be used for creating simulation templates for greater types of idealization variations
- it has a higher formulation efficiency

2.4.2 Research Gaps

The effective formulation of analysis problems using existing methods is hindered by two key research gaps as stated below.

- Lack of formalization of the knowledge used by analysts in formulating simulation templates
- Inability to leverage this knowledge to define model composition methods for formulating simulation templates

In the context of this research, this knowledge refers to the intent of the idealization decisions taken by analysts. Existing methods, such as those based on parameterized scripts for creating behavior models, do not represent the intent of the idealization decisions. At best, these methods are based on an interpretation of this intent in the form of mathematical relations between design parameters and behavior parameters for a particular class of analysis problems. VTMB-type variations or variations in idealization decisions taken by analysts require manual and costly updates to a large set of these parametric relations. If one can formalize the types of idealization decisions taken by analysts and the conditions for these decisions, one may explicitly represent these decisions at a higher level of abstraction from which mathematical relations or computable scripts may be automatically derived.

Efficient formulation of simulation templates also requires model composition methods that can automatically compose simulation templates from reusable building blocks and the idealization decisions taken by analysts. The representation of building blocks requires both static knowledge—what concepts are represented by building blocks—as well as dynamic knowledge—how are building blocks composed to create simulation templates.

2.5 Summary

In this chapter the presentation of integrated functional and spatial design scenario and simulation templates as means to achieve this, provide a platform for this research. The brittleness of simulation templates to VTMB problems and changes in idealization decisions taken by analysts is presented in details. The concept of assembly system topology which is central to the definition and characterization of VTMB problems is defined and illustrated in this chapter. The central theme of the primary research question is the improvement of effectiveness of analysis problem formulation. Variation in design alternatives, idealizations decisions, and efficiency in formulating simulation templates are presented as three key factors contributing to the effectiveness of analysis problem formulation. The lack of effectiveness in formulating analysis problems using existing methods is contributed to two key research gaps: (1) lack of formalization of the knowledge used by analysts in formulating simulation templates, and (2) inability to leverage this knowledge to define model composition methods for formulating simulation templates. In the following chapter, a thorough review of published research, methods, and tools relevant to these gaps is presented. This review provides a refined understanding of these research gaps, and establishes requirements for model formulation methods developed in this research.

Chapter 3 : Related Research

In this chapter, a research survey is presented towards answering the primary research question (section 2.4.1). Past and existing research efforts are described and evaluated in this survey. The purpose of this survey is twofold: (a) categorize models, methods, and ontologies used in diverse applications in these research efforts, (b) elaborate on the lack of existing methods to address the gaps identified in this research, and (c) leverage existing models and methods to address these research gaps. The survey also points to research efforts that have been directed towards similar end goals as this research but for a different class of problems.

Table 3.1: Metrics for categorizing and evaluating related technical work

1	**Design information and knowledge modeling (design meta-model)**
a	Represent conceptual and detailed design models
b	Domain-specific detailed design ontologies
c	Open-standard and non-proprietary ontologies
d	Extensibility
e	Represent associated behavior models
f	Export model structure from design tools (such as ECAD, MCAD tools)
g	Export model instances from design tools
2	**Behavior modeling**
a	Formulating behavior models (solution method and solver-independent)
b	Relationship between design models and behavior models
c	Solution method-, and solver-specific behavior models
d	Behavior model building blocks (and library) & reuse
e	Auto-generate behavior models from building blocks
3	**Simulation templates**
a	Template patterns and templates for trade studies
b	Auto-generate simulation templates and their components
d	Multi-directional solution of simulation templates (and inverse problems)
e	Adapting simulation templates to changes in idealization decisions
f	Ability to address VTMB problems
4	**Model definition and transformation**
a	Declarative representation of models (and their associativities)
b	Declarative representation of model transformations

Table 3.1 above enlists a set of qualitative metrics to categorize and evaluate existing body of research. These metrics account for the research gaps and requirements for efficient analysis problem formulation presented in section 2.4. The research survey is

presented roughly in the order in which the metrics are listed. At the end of this chapter, the results of the survey are summarized.

3.1 Design Information and Knowledge Modeling

With geographically and temporally distributed product realization teams, it is required that next generation product development systems create and exchange information and knowledge across different product lifecycle activities in an information-rich electronic form. Of particular interest to this research is the interoperability and knowledge exchange between design and analysis systems. As a foundation, Fenves et al. (Fenves 2004) have proposed the Core Product Model (CPM2) as a formal representation of an artifact. It is a conceptual meta-model representing a broad range of design concepts including requirements, form, function, behavior, material, physical and functional decompositions, and their inter-relationships. The CPM is targeted to be (a) software vendor solution-independent, (b) open and non-proprietary, (c) simple and generic, (d) extendable, (e) independent of any particular product development process, and (f) applicable through different lifecycle phases. In the context of this research, CPM2 can serve as meta-model to represent an artifact during different design phases (Pahl and Beitz 1996)—from conceptual design models to detailed design models to manufacturable design models.

CPM2 is influenced by the Entity-Relationship data model (Chen 1976), and consists of two key classes, called CommonCoreObject and CommonCoreRelationship (equivalent to *Class* and *AssociationClass* in the Unified Modeling Language (UML)) (Rumbaugh, Jacobson et al. 2004; UML 2 2004). A UML class diagram for CPM2 is show in Figure 3.1. The principal entity in CPM2 is the Artifact—a distinct entity in a product (component, sub-assembly, or assembly). An artifact has properties such as form—physical description of the artifact, function—what an artifact is intended to do, and flow—medium for realizing transfer functions. Form consists of geometry—spatial description of an artifact, and material—physical constituent of an artifact. A feature is a part of an artifact's form that has function(s) associated with it. An artifact satisfies a specification—a collection of customer requirements. The specializations of CommonCoreObject in CPM2 can be related to each other using specializations of

CommonCoreRelationship. For instance, the Usage entity relates the definition of a CommonCoreObject to its usage in a particular context.

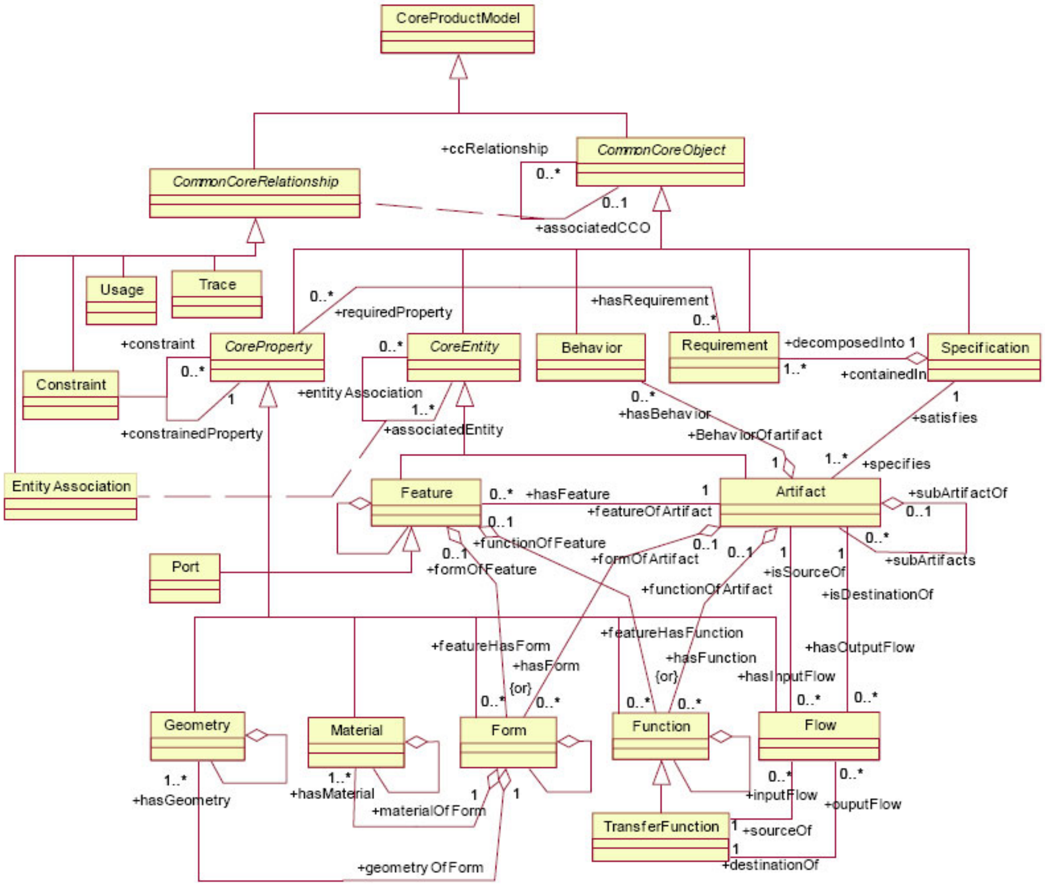


Figure 3.1: Core Product Model version 2 (CPM2) - UML class diagram view

CPM2 allows one to associate behaviors to an artifact, and associate behavior models to a behavior. However, it doesn't specify the structure of this behavior model and the nature of fine-grained associativities between a behavior model and other properties of an artifact, partially so because CPM2 is intended to be open and extensible. One of the target contributions of this research is to augment CPM2 with these representation capabilities. CPM2 also support the use case of representing computed behavior parameters and results of their evaluation against requirements.

As an example of CPM2's intent to represent product information through different lifecycle phases, the cardinality of the Aritfact-Form association reflects that an artifact may have 0 or more forms associated with it. This represents the use case that during conceptual design stages, the form of an artifact may not be available.

It is to be noted that CPM2 represents a conceptual meta-model that can be specialized and extended for different product domains such as aerospace, electronics, and automotive. New domain-specific entities may be added as specializations of existing core entities. Extensions to the Core Product Model, such as the Open Assembly Model (Rachuri, Han et al. 2006), have been developed for specializing different aspects of an artifact.

The ISO 10303 family of standards (STEP) is an extensive set of open standards-based product domain ontologies, such as for mechanical design, electronics, and automotive and cross-domain constructs such as geometry and product configuration control. Though the intent of STEP was to enable exchange of product information across different CAD/CAE/CAM systems, it has matured into a set of modularized ontologies for representing different aspects of product information typically during detailed design and manufacturing phases of the product lifecycle[11]. These modularized ontologies (formally known as STEP modules) are extended and specialized into ontologies for product application domains, such as AP210 (ISO 10303-210 2001) for electronics products, AP203 (ISO 10303-203 2000) for mechanical products, AP214 (ISO 10303-214 2003) for automotive products, and AP215, 216, and 218 for ships(ISO 10303-216 2000; ISO 10303-218 2000; ISO 10303-215 2001). In addition, integrated resources provide concepts that are reusable across several application domains. For example, Part 42 (ISO 10303-42 2000) is a modular ontology for representing geometry- and topology-related aspects of a product and is used across different product domain-specific ontologies (such as AP210 and AP203).

In the context of this research, CPM2 and STEP ontologies are complimentary in the sense that the former provides an organizing principle for product information that is recurrent in different product domains through the lifecycle phases while the latter provides rich formal information models for specific aspects of product information and for different product application domains typically during detailed design and

[11] Part 41 (ISO 10303-41 2000) and AP239 (ISO 10303-239 2000) provide representations for generic product structure and basic product lifecycle information respectively.

manufacturing phases. As an example, the Geometry entity in CPM2 can refer to the constructs in STEP Part 42. For formulating behavior model structures to simulate different types of behaviors of an artifact at different fidelities in different disciplines, it is necessary to have meta-models that (a) represent different aspects of product design, such as form, function, and requirements, and (b) are rich and formal ontologies that may be used to create design information models, which may then be used to create behavior models. Together CPM2 and STEP satisfy these requirements. An example of complex analyses supported by STEP ontologies is provided by (Zwemer, Bajaj et al. 2004; Bajaj, Peak et al. 2006) wherein detailed PCB design information available as STEP AP210 instance model is used to perform high fidelity thermo-mechanical warpage analysis.

In actual industry practice, product design information is typically available via a collection of models, such as CAD models, enterprise databases, and auxiliary models. Each model populates a subset of the design information shown in Figure 3.1, and collectively all models may not populate the all aspects of design information—leading to gap filling tools such as PCB layer stackup editors (Peak, Wilson et al. 2002; PCB Layer Stack Editor (LKSoft) 2008). In general, CAD tools provide a good authoring environment for form- and function-related design information—typically MCAD tools provide detailed 3D form and ECAD tools provide 2D form and electrical function information. There are two broad approaches for using the available design information for analyses:

- *Integrated simulation capabilities with CAD tools*: Most CAD tools provide integrated capabilities for simulating certain types of behaviors of artifacts, based on the form and function-related information authored in these tools. For example, some MCAD tools provide utilities to create finite element models (NX CAE (Siemens PLM)), and ECAD tools provide utilities to create electrical simulation models (Zuken CR-5000 PSpice & HSpice). These utilities can be used to simulate only certain types of behaviors at certain fidelities, and work well as long as all the design information required for simulating behaviors-of-interest is available in these tools, and the behavior models can be solved using specific solvers integrated with these tools. Additionally, cross-version interoperability and long-term retention of design and simulation models has always been a challenge with such an approach. This approach is not extendable to other types of analyses beyond those supported by the integrated simulation capabilities. One may argue that CAD tools provide application programming interfaces (APIs) to extract design information and use it for creating customized behavior models. This approach may alleviate some potential limitations outlined above but it does not increase the set of available design information beyond the models created in CAD tools, and there is limited subset of this information that is accessible via the APIs. Typically, even the extraction of form-related parameterization scheme from CAD tools via their APIs is an open question.

- *Design Integrators*: To enable a wider variety of analyses, and to use customized methods for formulating behavior models, and to allow a combination of CAE solvers to solve them, it is necessary to integrate subsets of design information in a unified non-proprietary standard form. For the purposes of detailed design, STEP ontologies typically satisfy this requirement. Design integrators are tools that may be customized for an enterprise and are used for automatically integrating design information from multiple CAD tools, enterprise databases, and other auxiliary models. As an example, LKSoft design integrator / importer (IDA-STEP (LKSoft) 2008) has been customized for electronics design enterprises to create a unified STEP AP210 model from design information sub-sets, which is then used for enabling multiple fidelities of thermo-mechanical warpage analyses (Zwemer, Bajaj et al. 2004; Bajaj, Peak et al. 2006) and design-for-manufacturability analyses (DFXpert (SFM Technology Inc.)) of PCBs. This approach makes a greater sub-set of design information available for complex multi-fidelity analyses. Also, the existence of rich open standard and non-proprietary STEP models enables long term design information retention and reuse.

In general, the industry practice is to use both approaches depending on the types of analyses being performed and the design information required to support them. However, for the purpose of this research, the latter approach is preferred as it provides for a greater subset of design information that is required to support a wider variety of analyses.

3.2 Behavior Modeling

In this section, research related to formulating behavior model structure, analysis knowledge representation and reuse is presented. Prior to investigating existing methods for formulating behavior models, a taxonomy of behavior models is presented.

3.2.1 Types of behavior models

Figure 3.2 illustrates a taxonomy of behavior models as a SysML block definition diagram (SysML 2007). Behavior models may be classified in many different ways depending upon the perspective. In Figure 3.2, each perspective is represented as a SysML Viewpoint, and the classification of behavior models in that perspective is contained in a SysML View. In essence, a viewpoint provides the context for specialization and a view—confirming to this viewpoint—contains the specialization tree. Each view has an abstract block (italicized name) which is the parent (class) for all specializations in that view.

This approach for categorizing behavior model is extensible in the sense that other viewpoints and views may be added and further specializations of behavior models in each view may be created. A brief explanation of each viewpoint and confirming views is provided below:

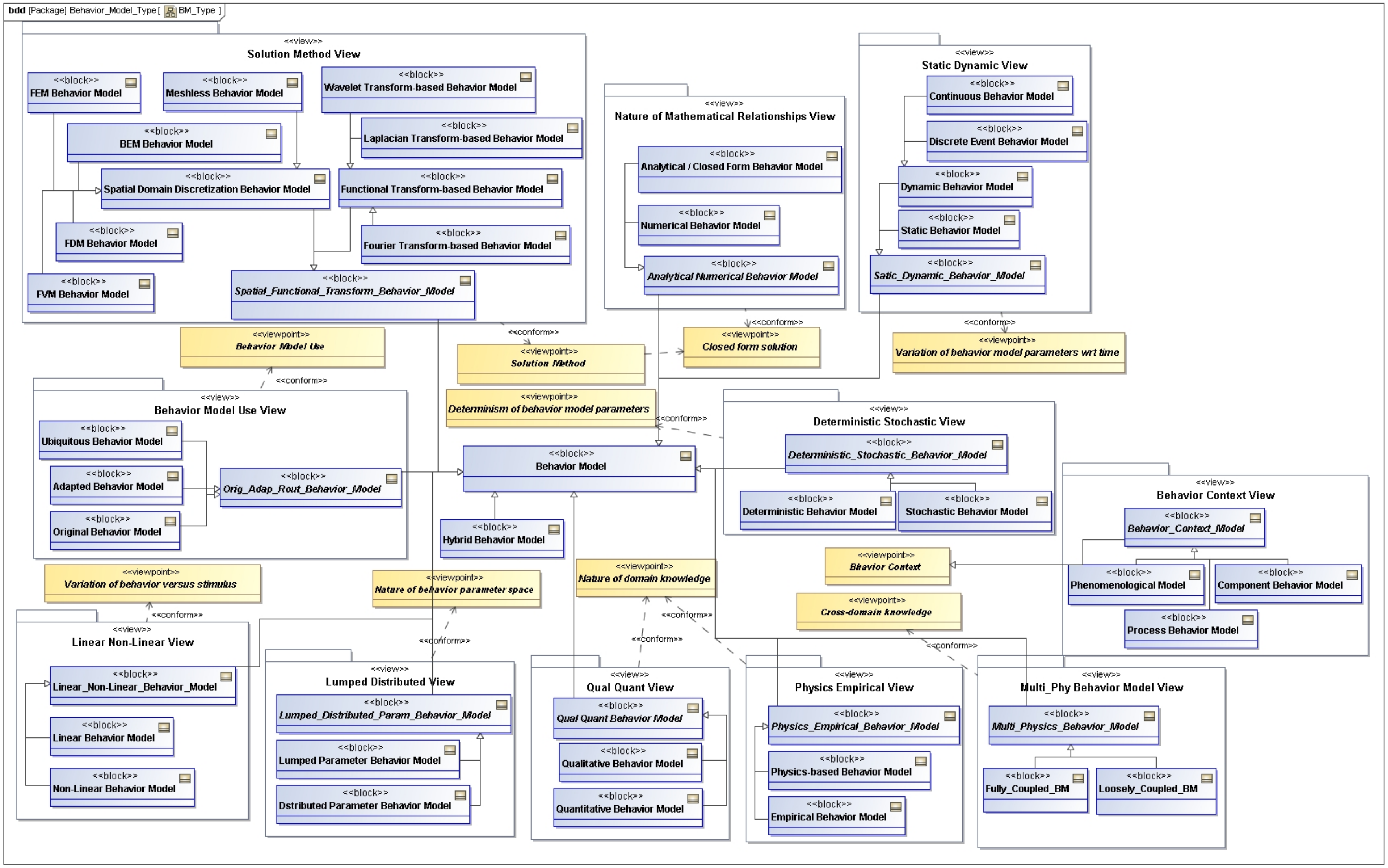


Figure 3.2: Types of behavior models from different viewpoints

- *Viewpoint: Nature of domain knowledge*

This viewpoint is concerned with all specializations of behavior models from a standpoint of nature of domain knowledge used for formulating and solving behavior models (structures and instances). A confirming view (Qual Quant View) consists of specializations of behavior model based on qualitative or quantitative nature of domain knowledge. In this view, there are two broad classes of behavior models—Qualitative Behavior Model and Quantitative Behavior Model. As the names suggest, a quantitative behavior model is used to compute the behavior of a system quantitatively in contrast to a qualitative behavior model which is used to predict the behavior of a system in qualitative terms. (de Kleer and Brown 1984; de Kleer 1992) have presented extensive work on qualitative physics and its use to create qualitative behavior models. An analytical behavior model or a numerical behavior model (such as a FEA model) is an example of a Quantitative Behavior Model.

Another view confirming to this viewpoint is the Physics Empirical View. This view consists of specializations of a behavior model based on whether the behavior model is founded on physics-based concepts and theories, or empirical information. A finite element model to predict the warpage behavior of PCBs is an example of a quantitative Physics-based Behavior Model (Bajaj, Peak et al. 2006), while an analytical model to predict warpage behavior based on the expertise of a PCB fabricator is an example of an Empirical Behavior Model.

The focus of this research is to develop methods for efficient formulation of quantitative physics-based behavior model structures. However, the intent is to not to underestimate the valuable insights that may be obtained from formulating and solving qualitative behavior models. Beyond verifying design alternatives, qualitative results may guide analysts formulate higher fidelity quantitative behavior models. Though this research focuses on physics-based behavior models, the formulation methods may be extended to use quantitative empirical building blocks.

- *Viewpoint: Variation of behavior versus stimulus*

This viewpoint is concerned with all specializations of a behavior model from a standpoint of the variation of the behavior represented by a behavior model and the stimulus for that behavior. A confirming view (Linear Non-Linear View) consists of specializations of behavior model based on the whether the behavior represented by a behavior varies linearly or non-linearly with respect to the stimulus. In the context of structural behavior analysis, this would imply the behavior of the deformation of the structure with respect to the applied loads. There may be several causes of non-linear behavior, such as non-linear material behavior, and large deformations.

- *Viewpoint: Nature of behavior parameter space*

  This viewpoint is concerned with all specializations of behavior models from a standpoint of the nature of behavior parameter space. A confirming view (Lumped Distributed View) consists of specializations of behavior model based on the lumped behavior parameters or distributed behavior parameters. A Lumped Parameter Behavior Model is one in which the spatial distribution of behavior parameters is idealized as a single value, in contrast to a Distributed Parameter Behavior Model in which the behavior parameters are spatially distributed. For example, if the temperature distribution along a heated bar is idealized as an average temperature value in a thermal behavior model for the bar, the thermal model would be a Lumped Parameter Behavior Model. However, if the spatial distribution of temperature in the bar is accounted in the thermal behavior model for the bar, the thermal model would be a Distributed Parameter Behavior Model.

- *Viewpoint: Behavior model use*

  This viewpoint is concerned with all specializations of behavior models from a usage standpoint. A confirming view (Behavior Model Use View) consists of specializations of behavior model based on if a behavior model is formulated for the first time (Original Behavior Model), is adapted from an existing behavior model (Adapted Behavior Model), or is being reused as-is (Ubiquitous Behavior Model). These behavior models correspond to the idea of original, adaptive, or ubiquitous analysis presented in section 2.2.2.1.

- *Viewpoint: Closed form solution*

This viewpoint is concerned with all specializations of behavior models from a standpoint of solvability of mathematical relationships in a behavior model. A confirming view (Nature of Mathematical Relationships View) consists of specializations of behavior model based on whether they have a closed form solution or need to be solved numerically. If all such relationships have a closed form solution, then such a behavior model is a Closed Form Behavior Model. If these relationships do not have a closed form solution, such a behavior model needs to be solved numerically and is known as a Numerical Behavior Model. It is possible that some relationships in a behavior model have a closed-form solution while others do not. All such cases in different views are specializations of a Hybrid Behavior Model (described at the end of this section).

- *Viewpoint: Solution method*

This viewpoint is concerned with all specializations of behavior models from a standpoint of solution methods for solving the mathematical relationships in a behavior model. The solution methods depend on the nature of mathematical relations (e.g. closed form). Hence, this viewpoint depends on the Closed form solution viewpoint as indicated in Figure 3.2. A confirming view, Solution Method View, consists of specializations of behavior model based on solution methods. It consists of two main classes of solution method-based behavior models—Spatial Domain Discretization Behavior Model and Functional Transform-based Behavior Model. The former represents those behavior models in which the spatial domain is discretized to solve the mathematical relationships in each discretization, such as finite element method-, finite difference method-, finite volume method, and boundary element method-based behavior models. These are denoted as Meshless, FEA, FDM, FVM, and BEM Behavior Model blocks in the figure. The block Functional Transform-based Behavior Model represents those behavior models in which analytical relationships are derived from behavior experimental data, or an analytical relationship is decomposed into a series of analytical relationships or transformed from one analytical form to another to aid mathematical operations (such as integrals). This class of behavior models is represented by the Function Transform-based Behavior Model block that has specialization such as Fourier Transform-based Behavior

Model, Laplacian Transform-based Behavior Model, and Wavelet Transform-based Behavior Model.

- *Viewpoint: Variation of behavior model parameters with respect to time*

  This viewpoint is concerned with all specializations of behavior models from a standpoint of variation of behavior parameters with respect to time. Static (or steady state) behavior models are those wherein behavior model parameters are idealized to be constant with respect to time, and dynamic (or transient) behavior models are those wherein behavior model parameters vary with time. In Figure 3.2, the Static Behavior Model block represents the former class of behavior models, and the Dynamic Behavior Model block represents the latter class of behavior models. Dynamic behavior models can be further specialized into continuous time behavior models and discrete event behavior models depending on whether behavior model parameters are provided or computed as continuous functions of time, or at discrete points in time. These are represented by Continuous Behavior Model and Discrete-Event Behavior Model blocks respectively in the figure.

- *Viewpoint: Determinism of behavior model parameters*

  This viewpoint is concerned with all specializations of behavior models from a standpoint of determinism of behavior model parameters. Deterministic behavior models are those wherein all behavior model parameters are deterministic in nature, while Stochastic behavior models are those wherein one or more behavior model parameters are stochastic in nature. In Figure 3.2, the Deterministic Behavior Model block represents the former class of behavior models, and the Stochastic Behavior Model block represents the latter class of behavior models.

- *Viewpoint: Behavior Context*

  This viewpoint is concerned with all specializations of a behavior model from a standpoint of the context of the behavior model. Here, "context" implies the specific "thing" whose behavior is being represented by a behavior model. A confirming view, Behavior Context View, consists of specializations of a behavior model from this viewpoint. These specializations include: (a) Phenomenological model—represent the behavior of a phenomena, such as an Euler Beam bending model, (b) Component behavior model—represents the behavior of a component (physical artifact), and (c) Process behavior model—represents the behavior of a process.

In general, a behavior model may be hybrid of the specializations in a given view. All such hybrid behavior models are represented by the Hybrid Behavior Model block in the figure. A Hybrid Behavior Model specializes one or more behavior model blocks in a view since all specializations within a view may not be mutually disjoint.

3.2.2 Formulating behavior models

In this section, research related to the formulation of behavior models is presented with special focus on the following aspects: (a) Formulating structure vs. instance of behavior models, and (b) Formulating solution method-, and solver-independent behavior models.

3.2.2.1 CAD-FEA integration

A major research thrust in formulating distributed parameter behavior models has been in the area of CAD-FEA integration. Methods developed in this area are aimed at efficient and intelligible idealization of CAD geometry to make it more amenable to FEA. Gordon (Gordon 2001) has identified three primary geometry idealization categories: (1) design and analysis geometry are same and no idealizations are required (seamless case); (2) design geometry is too complex and has wrong intent, so it has to be extensively modified to create a geometric model amenable to analyses; and (3) engineering analysis is performed first on an idealized form to create specifications for the actual design form. These three use cases affirm the necessity of non-causal associativity between design and behavior models to enable the creation of one from the other.

Armstrong et al. (Armstrong 1995; Donaghy 1996) have proposed geometric operations for dimensional reduction and addition / suppression of features based on medial-axis transforms and Saint Venant’s principle for creating idealized geometry for simpler FEA meshes and faster analyses. Arabshahi et al. (Arabshahi, Barton et al. 1991; Arabshahi, Barton et al. 1993) have proposed CAD-FEA transformation methods for analysis to respond to changes in design, and Belaziz et al. (Belaziz, Bouras et al. 2000) have developed a feature-based tool based on morphological analysis of solid models for integrating the design model and its idealized form. This analysis views the detailed solid model available from CAD tools as one created from a “gross model” with addition/modification of features. Given a solid model, this analysis creates a form feature model (detecting the gross model and the process of feature addition / removal), followed by simplification of features to create an idealized model, and iterative FEA based on the idealized model. The updated idealized model is then mapped back to update the native CAD model.

The contribution of these and other research efforts in this area that have developed intelligent methods for creating idealized geometric models from details design geometry is valuable but not sufficient for formulating behavior models. Turkiyyah and Fenves (Turkiyyah and Fenves 1996) aptly state that the functional description of the system is a key for creating behavior models. Spatial information by itself provides little information about desired behavior and hence, insufficient for behavioral evaluation. In addition to the idealized form, the formulation process requires idealization of the material behavior of the artifact, and associated behavior conditions and stimulus (such as loads)—stated in details in the definition of behavior model formulation in section 2.1.

The workflow for formulating behavior models in most current-day CAE tools (such as finite element tools) typically starts at creating the idealized form, or importing it from a COTS CAD tool via their native interfaces or standard STEP- or IGES-based interfaces. More often than not, CAE tools have limited support for importing design form from multiple CAD tools and minimal[12] support for open standards-based interfaces. In effect, an analyst has to re-create the idealized form or refine the imported design form. Even in the case of seamless import, there is no explicit associativity between the design form and idealized form that will be used for analysis (such as FEA). An additional limiting factor is the inability of most CAE tools to recognize the imported shapes as parts, and their usages as components in an assembly, and to interpret that the interactions between geometric shapes is the interaction between assembly components,

[12] Here, minimal implies confirming to (or importing/exporting) limited aspects of standards-based description of design form.

thus compelling users to work with basic geometry entities such as vertices, edges, areas, and volumes. However, once an idealized form is available in their analysis modeling environment, most CAE tools provide a broad set of capabilities to formulate solution method- and solver-specific behavior models. Newer capabilities in some FEA tools provide for a one-way associativity between a CAD model and corresponding FEA model (Simulia ABAQUS 2008). It enables an automated update of the FEA model when the design form is changed. However, this associativity is static and one-way. For changes in assembly system topology of design models, the idealization process leading to creation of the FEA model has to be repeated.

The limitations in formulating behavior models directly in CAE tools (and hence specific to a solver for a given solution method) can be summarized as follows:

- Inability to capture analysis intent, such as attribution of material behavior and loads to specific parts in the design form as opposed to volumes in the idealized form
- Lack of explicit associativity between design form and idealized form
- Lack of support for VTMB analysis problems
- Need to reformulate behavior models from scratch for using capabilities of other CAE solvers, and other analysis methods (such as FEA (Reddy 1993) and meshless analysis (Chen, Lee et al. 2006))

Hence, this research focuses on formulating behavior models independent of solution method and solver, and to establish explicit associativity relationships between the design form and idealized form so as to preserve the analysis intent. In the context of the MRA-based simulation pattern presented in section 2.2.2, this implies formulating the CBAM. Behavior models formulated in this manner may then be solved in whole or parts using different methods and solvers.

(Shephard, Beall et al. 2004) corroborate the approach for having an abstract design-component model to capture analysis intent and to interface between CAD and FEA tools. The Simulation Application Suite (Simmetrix Inc. 2006)) is one such FEA mesh generation tool that is founded on this abstract component model. In the MRA-based simulation pattern, the ABB system consists of an assembly of analysis bodies and their associativities to individual parts and components in the design form. This satisfies the requirement for having an abstract design-component model for supporting multi-fidelity analyses and can additionally be used for other solution methods apart from FEA.

3.2.2.2 Heuristic frameworks

In the past heuristic methods have been proposed to formulate problem-specific equations from general domain equations, such as the framework developed by Yip et al. (Yip 1993) for simplifying the Navier Stokes equations and by Ling et al. (Ling, Steinberg et al. 1993) for generating governing equations for analysis of thermal systems. A challenge with these approaches is to develop and assemble equations for different fidelities for a multi-body design alternative. Most modern CAE tools possess the capability of assembling and solving a set of relevant equations for a multi-body problem, given a consistent set of analysis specifications (idealizations). Even then, the issue lies in the lack of explicit associativity between the behavior model and the design model (both at the structure and instance level) thus making the behavior model formulation process inefficient for handling VTMB problems for adaptive and original analyses.

The heuristics-based approaches may not be sufficient but are can play an important role on the overall solution towards model-based communication between designers and analysts. Heuristics may help guide analysts in selecting appropriate idealizations based on the given artifact, behavior conditions, and desired analysis accuracy. Additionally, it may used for refining behavior models such as in adaptive control tools for FEA pre-processors (Shephard, Beall et al. 2004).

3.2.2.3 Simulation templates

In this sub-section, behavior model formulation approaches that laid special emphasis on integration with design models and modularity of the formulation method are presented.

The Composable Simulation research (Diaz-Calderon, Paredis et al. 2000; Sinha, Paredis et al. 2000; Paredis, Diaz-Calderon et al. 2001) is aimed at performing system level behavior simulations by composing behavior models of the system components. Each physical component is represented by means of port-based models that formally describe its form and behavior with explicit mapping between the form and behavior ports. These port-based models can be composed to create the system level behavior model. The ports and the internal behavioral implementations are separate, thus providing the capability to easily reconfigure the system for different fidelities of behavioral simulations. Although specific to mechatronics systems that are typically modeled using lumped parameters, the composability ideas may be leveraged for creating behavior models of a system from behavior models of components. However, much of this research depends on the availability of behavior models of the components and this research does not prescribe efficient ways to formulate them. Also, the methods in this research are specifically developed for behavior models that are described using differential algebraic equations.

The Multi-Representation Architecture (Peak 1993; Peak, Fulton et al. 1998; Wilson, Peak et al. 2001; Peak, Paredis et al. 2005) research prescribes a modular and reusable approach for creating behavior models from design models by stepping through four intermediate models, as described in details in section 2.2.2. As described in that section, the MRA can be viewed as a *simulation template pattern*—analogous to design patterns (Gamma, Johnson et al. 1995) in software engineering. The reusability of this approach is due to (a) use of analysis building blocks (such as linear elastic material) and systems of ABBs (such as Euler beam system), and (b) non-causal description of ABBs, ABB systems, and their associativity to design models, thus providing a model structure for solving analysis problems and inverse problems. The process of composing the ABB system structure from ABBs and establishing associativities to the design model structure is manual, thus making the process inefficient for adaptive and original analyses wherein designers and analysts perform trade studies on idealizations. Additionally, the model structure needs to be “rewired” for assembly system topology changes inherent to VTMB analysis problems. However, once the structure is available, it can be used to formulate behavior model instances automatically for a family of design model instances (XaiTools (Georgia Tech) 1999)

In the MOSAIC project-related research (Sellgren 2003), a product is divided into sub-systems, and their mating features (what is connected) and interface features (how it is connected) are identified. It proposes a three-layered architecture for organizing the information in design and analysis models – the *design layer* for design-specific information such as geometry and material; the *generic behavior layer* for information specific to behaviors of the design model, mating and interface features; and the *application layer* for representing this information in a software tool (such as FEA tools). The modularization rationale is similar to the composable simulation work—separating the interface definition and its behavior implementations. However, this research does not deal with organization of analysis knowledge or formulation of behavior models.

The open standards-based information exchange methods are focused on use of open standards to represent analysis models and their relationships to the design model. STEP AP209 (ISO 10303-209 2001) is an ontology for representing analysis models and the associativity between the shape representations of the design form and the idealized form for analyses, and the idealized form for analysis to a solution method-specific form (such as FE meshed model). Here, a relationship ("basis") is used to link the idealized and the nominal design shape (Hunten 2001). With the modularized STEP architecture, the generic design model concepts in AP209 are shared with other application domain APs, such other AP210 (ISO 10303-210 2001) for electromechanical products and AP203 (ISO 10303-203 2000) for generic mechanical products. Further, Part 104 (ISO 10303-104 2000) provides an ontology for representing finite-element based models. Overall, these open standards are useful for representing some types of idealization relations (esp. geometry-related) between the design model and the analysis model, but they do not prescribe a standards-based ontology for representing ABBs (such as material behavior models, load models, and behavior condition models) that may be used for creating analysis models. In the research presented in this dissertation, relevant aspects of STEP-based ontologies are leveraged in principle and a behavior meta-model is developed. Additionally, algorithms for automated composition of ABBs—typically outside the scope of the subject open standards-based ontologies—are also developed.

Several methods have been proposed in the past for organizing behavior models. Some notable methods are described here. Hoffman et al. (Hoffman and Joan-Arinyo 1998) propose a *product master model mechanism* so that the different behavior models of the artifact may be linked and synchronized with a master model that contains all the information about the artifact. Addanki et al. (Addanki, Cremonini et al. 1991) have proposed the *graph of models* approach for automated selection of analysis models organized in a graph, on the basis of assumption-checking. This method is implemented for systems characterized by ODEs and is founded on model-based reasoning techniques. Falkenhainer and Forbus (Falkenhainer and Forbus 1991) have proposed a compositional modeling approach using which appropriate analysis models may be searched from a repository of analysis knowledge, based on the specific query and the structure of the subject system. This repository is founded on the relevant domain theories (such as thermodynamic analysis of steam plants). Since the approach is targeted for searching models and not formulating behavior models, all possible combinations of idealizations are explicitly modeled in this approach, which is typical known only when analysis knowledge is mature. For adaptive and original analyses, analysts need to dynamically compose, verify and reconfigure behavior model structures using different combinations of idealizations and perform trade studies to select the appropriate set of idealizations. However, the use case of efficiently organizing behavior models is a valuable one. If behavior model structures can be characterized along some key dimensions, then algorithms can be created to compute the "differential" between any two behavior model structures and thus determine their degree and dimension of separation in a repository of behavior model structures. For a given behavior model structure, one may also create a repository of behavior model instances.

Tools such as Model Center (Engineous Software 2007) and iSight (Phoenix Integration 2007) provide a modeling and computation framework for linking design parameters in native CAD models and behavior parameters computed in different solver tools (such as FEA tools). These linkages are specific to the assembly system topology of artifacts and have to be manually updated for families of VTMB design alternatives. In addition, mathematical relationships embodied in these linkages need to be manually updated both for topology variations in design alternatives and idealization decisions taken by analysts.

3.2.3 Analysis knowledge and reuse

The term "analysis knowledge" has been used in different flavors in related research efforts. Different research efforts model different aspects of analysis knowledge that are essential to realize their specific use cases. In essence, analysis knowledge is the union of all such aspects. Some well-known aspects of analysis knowledge are listed below.

- Domain theoretic knowledge—including first principles such as conservation of energy and equilibrium principles and derived behavior theories like Euler-Beam and Timoshenko beam theories (Timoshenko and Goodier 1970)—used for computing behaviors of artifacts
- Consistent combinations (and limitations) of different aspects of domain theoretic knowledge, such as assumptions under which the Newton's laws of inertia are valid
- An answer to the following question: "What domain theoretic knowledge concepts have to be used for a specific behavior computation problem?", i.e. when and how to apply existing knowledge to compute behavior. Heuristic-based approaches presented in section 3.2.2.2 specifically address this question. Other research efforts in this direction involve automated selection of assumptions given the analysis objectives (Finn 1993; Turkiyyah and Fenves 1996).
- Analysis intent—description of idealization decisions that help formulate a behavior model structure and its relationships to a design model structure
- Analysis rationale—justification of why certain pieces of knowledge (and hence certain idealizations) are used for computing behavior. The justification typically relates to experiential knowledge of the analysts.
- Objectives of the analysis problem and limitations of analysis models

In this research, analysis knowledge specifically implies domain theoretic knowledge, modeled as computer-interpretable analysis building blocks (ABBs), and the consistent combinations of these ABBs that reflect valid combinations of domain theoretic concepts. In particular, this research does not focus—without limiting such extensions—on developing a knowledge base relating domain theoretic concepts to family of analysis problems for which they may be used or are most useful. The methodology developed in this research is targeted to be used by analysts in formulating behavior model structures. Designers use a simulation template pattern that embodies the behavior model structure to perform trade studies on instances. This assumes that analysts are aware of the analysis rationale and hence the reasons behind the assumptions—embodied as ABBs. However, this research does aim at representing analysis intent by (a) explicitly relating a design model structure to behavior model structures (as shown in the definition of simulation templates in section 2.2.2), and (b) representing idealization decisions taken by analysts as computer-interpretable specifications for automated formulation of a behavior model structure.

For efficient formulation of behavior model structures—the primary objective of this research—it is necessary that ABBs be reused for formulating different behavior model structures, and behavior model structures of components be reused for formulating behavior model structures of systems. In the context of the simulation template for plane stress analysis of Flap Link part example illustrated in Figure 2.3, this would imply using the plane stress ABB model for plane stress analysis of different mechanical parts, and reusing the entire Flap Link plane stress CBAM for developing a plane stress CBAM of a system with multiple Flap Link parts.

Peak et al. (Peak 1993; Peak, Fulton et al. 1998; Peak, Fulton et al. 1999; Zeng 2004; Bajaj, Peak et al. 2006) have demonstrated the advantages of abstracting domain theories as ABBs and using ABBs to create behavior models. Here ABBs embody specific assumptions that are used for creating a behavior model. They have shown special types of primitive ABBs for mechanical and thermo-mechanical analyses, such as material behavior ABBs, load ABBs, geometry ABBs, and boundary condition ABBs. These ABBs can then be used to create phenomenological models, such as Linear Extensional Rod model, Euler Beam model, Linear Torsion model, and Plane Stress behavior model. A phenomenological model is a type of a complex ABB. Phenomenological models can then be used to create component behavior models, such as the Plane Stress ABB is used to create plane stress behavior model for the Flap Link part (Figure 2.3). Turkiyyah and Fenves (Turkiyyah and Fenves 1996) propose that analysis assumptions should be modeled explicitly using declarative aspects that define the scope, content, and the validity of assumptions, and procedural aspects that define the transformations to the behavior model when the subject assumption is applied. It is to be noted that ABBs are representative of types of assumption choices available to analysts. The above effort only models the declarative aspects of ABBs. In the research presented in this dissertation, this will be augmented with the procedural aspects, thus aiding automated composition of behavior models (structures) from ABBs. In addition, this research shall also investigate the characteristic dimension of ABBs and develop a meta-model for building a library of ABBs.

Robinson et al. (Robinson, Nance et al. 2004) aptly state that the validity of simulation models when being used in a context different from the original use is factor that limits reuse. In this light, ABBs themselves embody domain theoretic knowledge, and the creation of each ABB should be followed by a formal verification method to check if it repreents the domain knowledge correctly. On the other hand, the use of ABBs for creating component and assembly-level behavior model structures deserves rigorous validation for the following reasons: (a) not all ABBs (assumption choices) are mutually consistent, and (b) ABBs used (assumptions) for creating behavior model structures may not be valid when analysis specifications are changed—the linear extensional model of the Flap Link part will not be a valid behavior model if the end loads on the part were torsional in nature.

The research presented in this dissertation leverages the work of (Finn 1993) that states the different types of approximations to physical system and phenomena for developing a behavior model. These include approximation of: (a) geometry of physical system, (b) physical phenomena being modeled, (c) boundary conditions, (d) material properties, and (d) approximation of control volume (esp. for thermal convection problems).

Grosse et al. (Grosse, Milton-Benoit et al. 2005) have proposed an ontology for supporting reuse, adaptation, and interoperability of engineering analysis models. This ontology provides an extensive listing of generic properties of analysis models that can be used to archive, identify and reuse them. In comparison, this is akin to the secondary use case of this research. The primary use case is to create behavior models. In the work presented by Grosse et al., an analyst (or a knowledge engineer per their terminology) has to explicitly categorize and document the decisions taken while creating an analysis model in terms of these generic properties. Further, most of the key properties, such as model idealization and model limitation, are represented as text strings. This limits the ability to search analysis models based on these properties since typically there are no commonly well-accepted standard string values for these properties. Also, the idealizations and limitations identified by an analyst may be coupled (or even contradict) each other. It is difficult for algorithms to identify these couplings and contradictions if the instances are text strings with no bounds on values. The ontology proposed by Grosse et al. agrees well with our perspective on model formulation versus solution methods – it identifies continuum, lumped parameter, and empirical-based idealizations for physics-based models, and several numerical solution techniques for solving these problems. In their ontology, the related physical system (or the design model) is a property of an analysis model. This is a coarse-grained associativity between an analysis model and a design model as opposed to fine-grained associativity that the automated methods developed in this research aim to establish. The research presented in this dissertation develops an extensible behavior meta-model based on an ABB meta-model for representing behavior model structures, which are then used to represent behavior model instances. It is strongly believed that behavior model structures confirming to this meta-model will provide an inherent description of the idealizations (performed to create them) by the virtue of the ABBs that compose them.

3.3 Model Definition and Transformation

In this section, declarative model definition and transformations approaches are described in the context of the modeling requirements for this research.

3.3.1 Model Definition

This section focuses on modeling paradigms and languages necessary for representing the types of models relevant to this research—artifact design models, behavior models, and analysis building block models (all three at both the structure and instance levels).

Some well-known representations for modeling knowledge are: productions (rules), semantics nets, schemata, frames, scripts and logic (Giarratano and Riley 1998). Productions formalize the knowledge by identifying preconditions, which when satisfied will result in actions. Semantic nets are used to model propositional information and formalize knowledge by identifying relationships (such as is-a, has-a) between nodes. Though they provide ease-of-expression, semantic nets have a non-definite (lack of representation for cardinality of relationships, aggregates of nodes) and shallow knowledge structure (attributes of a concept are represented as nodes, like the concept itself). A Schemata or a Schema is a deep knowledge structure, unlike semantic nets.

Using this, we can represent knowledge related to the properties of artifacts. Frames and Scripts (time-ordered sequence of frames) are different types of schema. Frames are used to describe knowledge typical to a given situation (snapshot in time). They may be: (a) situational frames – knowledge as to what to expect in a given situation, (b) action frames - knowledge about what to do in a given situation and (c) causal knowledge frames - combining situational and action frames to represent causal knowledge. The attributes of a frame are known as slots and their values are known as fillers. For example, a frame "car A" has an attribute "color" with value "black". Frames can be grouped together into new frames (such as "car"). This is similar to the *class* and *object* terminology in object-oriented programming and *schema* and *instance* terminology in databases.

Most declarative formalisms for information and knowledge modeling in engineering are frame-based, such as EXPRESS (ISO 10303-11 2001) which is used by the STEP family of standards and SysML (SysML 2006) which specializes the UML formalism for systems engineering. Essentially, they provide entities to represent concepts in a given universe-of-discourse, attributes to represent the properties of this concept, constraints to bound the values of the attributes (such as *where-rules* in EXPRESS, *constraint blocks* in SysML, *constraints* in COBs (Wilson 2000), OCL (UML 2 OCL 2004)), and relations to represent the relationships between the attributes and entities (such as association, aggregation in (UML 2 2004)). In the recent past, the term *ontology* is used to define a set of representational primitives to model a universe of discourse. These representational primitives are classes (or sets), attributes (or properties), and relationships (relations between classes) (Gruber 1995; Gruber 2007). An ontology provides semantics to communicate about a domain. As an example, STEP AP210 (ISO 10303-210 2001) is an ontology for describing the design of electro-mechanical products. It provides concepts, their inter-relationships, and validity for describing design-related information for electromechanical artifacts.

Logic is the study of the rules of exact reasoning. Formal logic focuses on the structure or the form of logic and not the semantics. Just as algebra can be used for uniquely formulating problems with different semantics, formal logic can be used for reasoning about objects without concerning itself with semantics of the objects. Predicate Logic was developed to analyze the internal structure of statements, and propositional logic (subset of predicate logic) deals with IF- THEN structure only. The simplest form of predicate logic is first order predicate logic that consists of universal and existential quantifiers.

Description Logics (a.k.a. DL) (Calvanese, Lenzerini et al. 1998) is a family of knowledge representation languages that provides the capabilities of “description”—describing a domain, and “logic”—rules to reason about the domain. The purpose of DL languages is to model domains in a manner that formal reasoning can be performed on these domains. With reference to object-oriented modeling, in DL a class is modeled as an atomic or complex *concept* representing a set of objects, and a relationship is modeled as atomic (or complex) *role* representing sets of pairs of objects. Complex concepts and relationships are modeled as expressions consisting of atomic concepts, roles, and logical operations. Examples of these operations are: $\neg$ negation (complement), $\cup$ disjunction (or union), and $\cap$ conjunction (or intersection). In addition restrictions can be placed on sets by using the value restriction quantifier $\forall$ and the existential quantifier $\exists$. Representing a set of concepts using DL constructs allows one to use DL reasoners such as (RacerPro 1997) to verify the non-redundancy of concepts, non-empty concepts, and check subsumption relationships (subset) between concepts. DL languages and reasoning engines can be helpful in developing a knowledge base of concepts. In the context of this research, this technology can be helpful in extending and validating a library of ABBs provided ABB models can be formalized as DL expressions. The primary objective of this dissertation is to identify key characteristics of ABBs and to develop model transformation methods to formulate behavior model structures. Developing formal methods to validate a library of ABBs will be a valuable future extension. It is also to be noted that several object-oriented languages (such as EXPRESS) themselves are founded on set theory-based concepts. A reasoning engine could possible be built to validate the semantic consistency and non-redundancy of models expressed in these languages. It is also worth noting that object-oriented languages ((ISO 10303-11 2001; UML 2 2004; SysML 2007)) provide enhanced ease of expressiveness for modeling real world concepts. DL languages provide constructs to enable formal reasoning based on these concepts. It is best to combine the easy of expressiveness with formal reasoning capabilities in developing model repositories and ontologies. In this dissertation, SysML is used extensively to represent design and behavior meta-models and models for the following three reasons in particular: (a) ease of expressiveness in defining the models, (b) representation of fine-grained relationships in a non-causal manner, as modeled using SysML parametric diagrams, and (c) applicability to systems design and analysis problems in general—representation of different types of systems and behaviors.

3.3.2 Model Transformations

Existing foundations of model transformations hold key to the research presented in this dissertation. The formulation of behavior model structures from an artifact design model structure given a set of analysis specifications is a type of model transformation. The intent of this aspect of the technical survey is to understand existing approaches to model transformation and to select one that is more suitable for the primary use case of this research.

Analogous to traditional data computing wherein the operands are numbers and operators transform numbers (such as add, subtract, divide, and multiply numbers), model transformation can be viewed as a form of computing where the operand is a model and the operators are transformation rules. Over time, the term model transformation has tended to imply transformations of object-oriented models as opposed to program transformation that deals with transformations of computation statements (such as those in imperative programming and functional programming) and is a relatively mature field in computer science. In contrast to program transformation systems that are based on mathematically-oriented concepts such as term rewriting, functional programming, and attribute grammars, model transformation systems tend to be based on object-oriented principles (Czarnecki and Helsen 2006).

Figure 3.3 illustrates the basic idea of model transformation. A model transformation process transforms a source model that confirms to a source meta-model (or schema) to a target model that confirms to a target meta-model (or schema). The two enablers for this transformation are: (a) transformation definition—describes how the transformation is to be achieved, and (b) a transformation engine—executes the transformations described in the definition. It is to be noted that the definition of a transformation is based on the source and target meta-models while the transformation engine executes this definition on source models (instances of source meta-model).

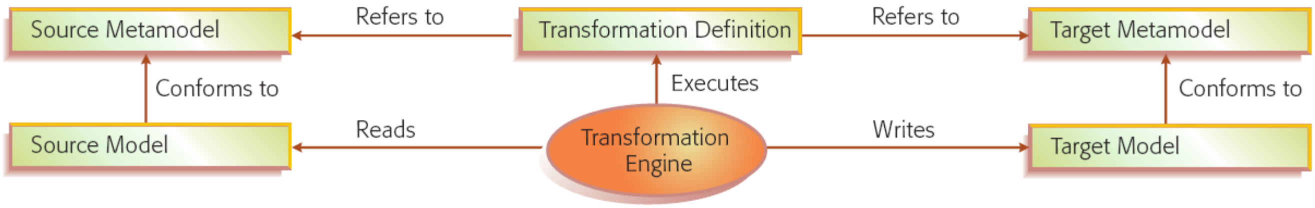


Figure 3.3: Basic concepts of model transformation (Czarnecki and Helsen 2006)

(Czarnecki and Helsen 2006) present a classification scheme to characterize different model transformation approaches. Figure 3.4 illustrates this classification scheme. Here, different aspects of this classification scheme are summarized and their relevance to this research is described.

- *Specification* implies the definition of the transformation itself. There are two main methods to specify a transformation. In one method, the source model (operand) and the transformation function (operator) are given and the target model (result) is computed. In the second method, the source model (operand) and the target model (result) are given and the transformation engine automatically figures a way to achieve the target model from the source model. In method 1, the operator may be encoded as a procedural code. In contrast, method 2 is more declarative in the sense that one describes the source and target models and not the specific computation process. In the context of this research, method 2 is adopted versus method 1.

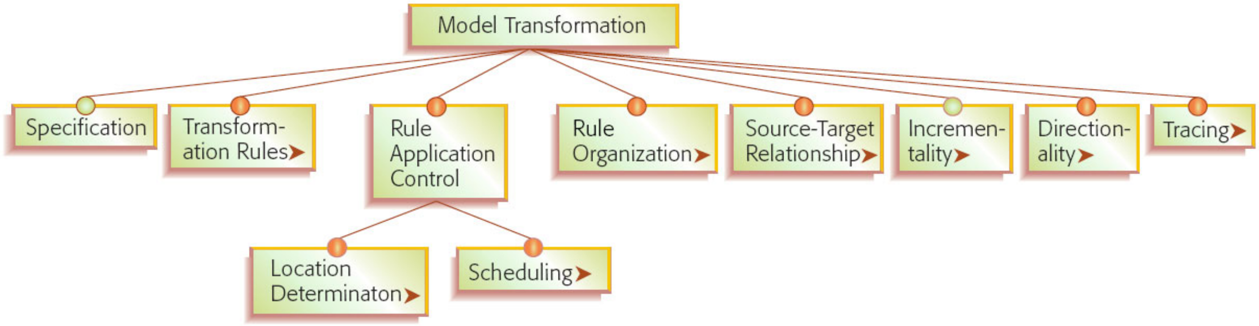


Figure 3.4: Classification scheme for model transformation approaches (Czarnecki and Helsen 2006)

- *Transformation rule* is the atomic unit of the model transformation process. Typically, transformation rules are declaratively represented with a LHS pattern and a RHS pattern. However, they may also be imperatively represented as a procedure or a function. Transformation rules consist of three main building blocks: (a) Variables that bind to model elements such as entities and relationships, (b) Patterns that consist of variables and bind to model fragments, and (c) Logic that defines the computation process. Variables, patterns, and logic could be syntactically typed or semantically typed. For declarative transformation rules, the transformation engine binds the LHS pattern to all matching fragments of the source model and replaces them with the RHS pattern to create the target model. For an imperative transformation rule, the transformation engine executes the procedure or function in the transformation rule. In the context of this research, the transformation rules are described declaratively as this will allow analysts to express the intent of behavior model structure formulation process without having to describe a procedure to formulate it. For example, for the plane stress CBAM for the Flap Link part in Figure 2.3, the analysis intent is to idealize the Flap Link part as a plane stress body. Declaratively, this is achieved by specifying the source Flap Link model and the target model—CBAM fragment that shows the Flap Link part wired with a Plane Stress body. Imperatively, this would have to be realized by writing a procedure to create the target model from the source model. Some other notable aspects of transformation rules are:

  - *Multi-directionality* describes if a transformation rule can be executed in multiple directions and causalities. In this research, transformation rules are being used to create the structure of a behavior model and are uni-directional. However, the structure itself may have relationships that may be solved in multiple directions (for inherently non-causal relations) to compute instance values.
  - *Application conditions* describe necessary conditions that must be satisfied before a rule can be executed.
  - *Parameterization* allows for passing parameter value (flags), data types, or even other rules to influence the behavior of a given rule.

- *Rule application control* primarily deals with the scope (local determination) of the model fragment to which a given rule is applied, and scheduling strategy to determine which rules are executed before others. There may be multiple matches of the LHS pattern of a rule in the source model. Transformation engines implement different application strategies—deterministic, non-deterministic, and interactive. In the context of this research, the deterministic strategy is required as it is desired that the transformations be applied to all possible matches in the source model. This is one of the key requirements for selecting a transformation engine that can be used for VTMB problems. Another notable aspect of rule application control is scheduling. In the context of this research, a transformation engine that allows for explicit scheduling is preferred since there is a sequence to the process of formulating a behavior model structure. In contrast, transformation engines with implicit scheduling do not allow users to control the execution order of rules.

- *Rule organization* deals with how rules may be packaged in a repository for reuse.

- *Source-Target Relationship* deals with the following transformation options: (a) creating a new target model that is different from the source, or (b) updating the source model to be the target model. In the context of this research, the latter approach is preferred as it allows for not duplicating the source model (artifact design model) and establishing associativities from the design model structure to the behavior model structure.

- *Incrementality* deals with the capability to efficiently synchronize the source and the target models when either one is changed.

- *Directionality* deals with the ability to execute transformations in one versus multiple directions. For model synchronization, it is desired that transformations be executable in multiple directions. This distinction holds importance when the source and the target models are not related. However, in the context of this research, the target model includes the source model and associativities to the source model itself. This is similar to the triple graph grammar approach (Konigs 2005) wherein the transformation rule not only creates the target model but also the associativities between the source and the target model.

- *Tracing* deals with recording the runtime process of transformation execution.

(Czarnecki and Helsen 2006) discusses several model to model transformation approaches. Of particular interest to this research is the *graph transformation-based approach to model transformations*. This particular approach is founded on the strong mathematical theory behind graphs and graph transformation—summarized by (Andries, Engels et al. 1999; Engels and Heckel 2000). This approach typically applies to models that may be abstracted as typed, attribute, labeled graphs which as (Czarnecki and Helsen 2006) point out is a formal representation of simplified class models. Two of the outstanding features of this approach are: (a) the ability to specify transformation rules in a declarative manner, leading to ease of modeling and model maintenance, and (b) the ability to apply transformations simultaneously to all fragments of the source model that match with the LHS pattern of a transformation rule—in contrast to sequential application for imperative transformations. A pitfall with this approach—in its native form—is the lack of explicit rule scheduling, thus leading to issues such as non-termination of transformations. However, newer graph transformation tools such as VIATRA (VIATRA 2007) fill this gap by providing a state machine-based controller to schedule the order of application of rules. It is worth pointing out that the definition of the transformation rule itself is declarative but the application of transformation rules is specified as a procedure. This approach is also intuitive to the realm of object oriented modeling as such models can be viewed as graphs in an abstract sense. Another potential weakness of the graph transformation-based approach is the treatment of ordered graphs, such as when representing methods as graphs where the ordering of statements is important (Czarnecki and Helsen 2006).

The objective of this research is to select a graph transformation system that satisfies the specific requirements for the primary use case. The research contribution lies in demonstrating the impact of such a graph transformation approach and system on the problem that this research addresses versus making improvements in the fundamental paradigms and algorithms that graph transformation approaches and systems are founded on.

A graph transformation rule *r = (L, R, K, glue, emb, appl)* consists of a left hand side graph *L* and a right hand side graph *R*, an *interface graph K* which is a subgraph of *L,* an occurrence *glue* of *K* in *R*, an embedding relation *emb* that relates the nodes in L to the nodes in R, and a set of application conditions *appl* that need to be satisfied for a subject graph for this rule to execute on it (Andries, Engels et al. 1999). The application of the rule *r* to a given graph G yields a graph H, denoted as G $=>_r$ H, in the following five steps (also illustrated in Figure 3.4).

Step 1: CHOOSE an occurrence of the left hand side graph *L* in graph G.

Step 2: CHECK if the application conditions, *appl* are satisfied

Step 3: REMOVE the occurrence of L upto the occurrence of K in G. Also remove any dangling edges—edges incident on deleted nodes.

Step 4: GLUE the resulting graph D in Step 3 with the right hand side graph R of rule r. This results in a disjoint union of graph D and R.

Step 5: EMDED graph R in D, i.e. establish all relationships between R and D per the embedding relations in *emb*.

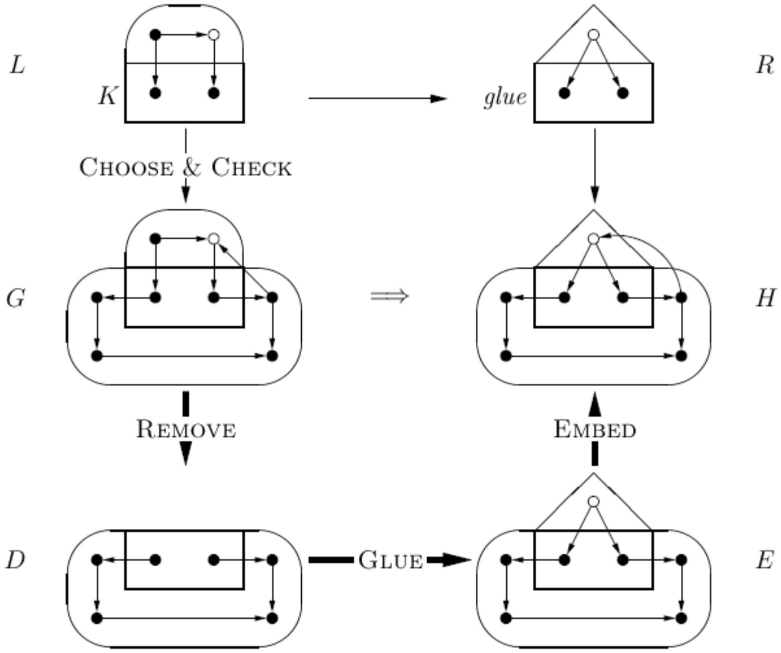


Figure 3.5: Illustrative definition of a graph transformation rule (Andries, Engels et al. 1999)

Different graph transformation approaches realize these basics steps in different ways. In general, the CHOOSE step requires the Injectivity condition—the occurrence of

L in G be isomorphic to L. Less restrictive conditions are the Contact condition—no dangling edges will arise in the REMOVE step, and Identification condition—occurrence of L in G should only compare nodes and edges in the interface graph K. A *single pushout rule* has empty application conditions and empty embedding relations. A *double pushout rule* has a contact and identification condition but empty application condition. A rule with a single node in the left hand side graph L and empty interface graph is a *node replacement rule*.

It is worth noting that there is a fundamental difference in the use case of graph grammars versus model transformation using graph transformations. Graph grammars consist of a set of formal production rules to generate a language (or expressions) based on a set of terminal symbols. The terminal symbols have semantics in their own right, and the syntactic arrangement of terminal symbols in an expression obeys the grammar. The semantics of an expression is determined by the semantics of the terminal symbols and the relative arrangement of terminal symbols in the expression. This is similar to the English language wherein the semantics of a sentence is determined by the semantics of the individual words and the relative arrangement of words. The primary use case for graph grammars is to generate a language of graphs based on terminal graphs and productions specified in the grammar. This would be useful when one intends to generate all possible models that could be created using a given set of transformation rules, as in generating a family of all possible design alternatives (Mullins and Rinderle 1991). However, the primary use case for this research is to create a specific behavior model structure that embodies the idealizations specified by an analyst. Graph transformations with explicit scheduling serve the needs of this specific use case.

3.4 Summary

A summary of the technical survey presented in this chapter is shown in Table 3.2. The table shows only the most relevant research efforts in the columns. The rows correspond to qualitative metrics for evaluating and comparing these research efforts. The coloring grades these research efforts based on the qualitative metrics.

Table 3.2: Summary of technical survey (shows most relevant references only)

	Related Research (right) **Evaluation Metrics (below)**	CPM2 (Fenves 2004)	STEP	ECAD / MCAD / PLM Tools	Armstrong 1995; Donaghy 1996	Belaziz, Bouras et al. 2000	Simulia ABAQUS 2008	Shephard, Beall et al. 2004	Yip 1993	Ling, Steinberg et al. 1993	Paredis, Diaz-Calderon et al. 2001	Peak 1993; Peak, Fulton et al. 1998	Sellgren 2003	STEP AP209, Part 104	Turkiyyah and Fenves 1996	Finn 1993	Grosse, Milton-Benoit et al. 2005	EXPRESS	SysML	XML	Description Logics	STEP	CPM2	Model Xform by Graph Xform (w/Explicit Scheduling)
1	**Design information and knowledge modeling (design meta-model)**																							
a	Represent conceptual and detailed design models	🟩	▨	▨																				
b	Domain-specific detailed design ontologies		🟩																					
c	Open-standard and non-proprietary ontologies	🟩	🟩																					
d	Extensibility	🟩	🟩																					
e	Associated behavior models			▨																				
f	Export model structure from design tools (such as ECAD, MCAD tools)		🟩																					
g	Export model instances from design tools		🟩					▨																
2	**Behavior modeling**																							
a	Formulating behavior models (solution method and solver-independent)										🟩	🟩												
	Qualitative								▨	▨														
	Quantitative										🟩	🟩												
b	Relationship between design models and behavior models						▨				▨	🟩	▨				▨							
c	Solution method-, and solver-specific behavior models				▨	▨	▨	▨			🟩	🟩			🟩									
d	Behavior model building blocks (and library) & reuse										▨	🟩	▨		🟩	🟩	▨							
e	Auto-generate behavior model structure from building blocks										▨													
3	**Simulation templates**																							
a	Template patterns and templates for trade studies										▨	🟩	▨											
b	Auto-generate simulation templates and their components				🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨							
d	Multi-directional solution of simulation templates (and inverse problems)											🟩												
e	Automatically adapting simulation templates to changes in idealization decisions				🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨							
f	Ability to address VTMB problems				🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨							
4	**Model definition and transformation**																							
a	Declarative representation of models (and their associativities)																	🟩	🟩	🟩	🟩			
	OO Modeling languages																	🟩	🟩	🟩				
	Formal languages for model v&v																				🟩			
	Domain-specific ontologies																					🟩	🟩	
b	Declarative representation of model transformations																							🟩

🟩 Strong support

▨ Partial support

🟨 No support from existing research - Research Gap

Chapter 4 : RESEARCH GAPS, QUESTIONS & HYPOTHESES

The objective of this chapter is to transition from the statement and descriptions of the problem and gaps that this research addresses to developing hypotheses for a possible solution approach. The primary question that motivated this research is as follows: **How can we improve the effectiveness of the analysis problem formulation process for VTMB problems?** In this light, two key research gaps identified in Chapter 2 were:

- lack of formalization of the knowledge used by analysts in formulating simulation templates
- inability to leverage this knowledge to define model composition methods for formulating simulation templates

Based on the related research presented in Chapter 3, it can be concluded that existing methods and approaches are *ineffective* in formulating and adapting simulation templates for VTMB problems and changes in idealization decisions taken by analysts. Based on the factors contributing to the effectiveness of analysis problem formulation presented in Chapter 2 and survey of existing methods in Chapter 3, the primary research hypothesis is presented in this chapter. Based on the primary hypothesis, two secondary research questions are posed for this research. Hypotheses for the secondary research questions are also stated.

4.1 Primary Research Question (PRQ) and Hypothesis (PRH)

PRQ: How can we improve the effectiveness of the analysis problem formulation process for VTMB problems?

PRH: We can improve the effectiveness of the analysis problem formulation process for VTMB problems by:

- abstracting the analysis building blocks (ABBs) that may be reused for composing simulation templates
- abstracting the intent of the idealization decisions taken by analysts, and using it to drive the process of formulating simulation templates

- systematically and automatically composing simulation templates using ABBs and the idealization decisions taken by analysts

4.2 Secondary Research Questions and Hypotheses (SRQ/Hs)

SRQ1: How can we formalize an ABB such that it can be reused for composing simulation templates?

SRH1: We can formalize an ABB such that it can be reused for composing simulation templates by:

- using a non-causal, declarative formalism to describe the concept and the knowledge represented by an ABB
- using a model transformation-based formalism to describe the method for using an ABB when composing simulation templates

SRQ2: How can we systematically and automatically compose simulation templates from ABBs?

SRH2: We can systematically and automatically compose simulation templates from ABBs by:

- representing idealization decisions in terms of specific ABBs to be used in composing simulation templates and the conditions for using these ABBs
- formalizing the process of composing simulation templates as a model transformation process that automatically creates simulation templates for VTMB design alternatives and idealization decisions

PART 2: KNOWLEDGE COMPOSITION METHODOLOGY (KCM)

Chapter 5 : KCM OVERVIEW

The Knowledge Composition Methodology (KCM) is the contribution of the research presented in this dissertation. KCM is a collection of models and methods that enable effective formulation of analysis problems. KCM models and methods are based on the research hypotheses stated in the previous chapter. The KCM Framework is a computational embodiment of the KCM. The purpose of the KCM Framework is to (a) provide a testbed for KCM implementations, and (b) test research hypotheses. The chapters presented in Part 2 of this dissertation describe different aspects of the KCM Framework. Figure 5.1 illustrates the components of the KCM Framework as a SysML package diagram.

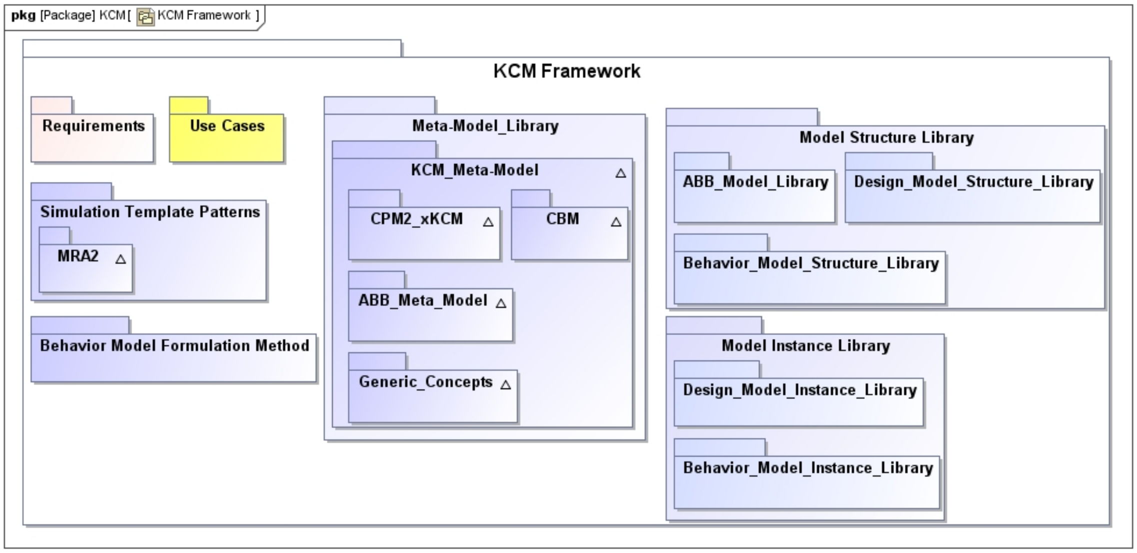

pkg [Package] KCM [KCM Framework]
KCM Framework
Requirements
Use Cases
Simulation Template Patterns
MRA2
Behavior Model Formulation Method
Meta-Model_Library
KCM_Meta-Model
CPM2_xKCM
CBM
ABB_Meta_Model
Generic_Concepts
Model Structure Library
ABB_Model_Library
Design_Model_Structure_Library
Behavior_Model_Structure_Library
Model Instance Library
Design_Model_Instance_Library
Behavior_Model_Instance_Library

Figure 5.1: KCM Framework components

The components of the KCM Framework are as follows:

- Requirements – functional and design requirements of KCM based on research hypotheses and research gaps presented in the previous chapter. KCM requirements are presented in this chapter.
- Use Cases – use cases of KCM based on research hypotheses presented in the previous chapter. KCM use cases are presented in this chapter.
- Simulation Template Patterns – patterns that define the structure of simulation templates for analysis problems. In this dissertation, the MRA pattern is used for formulating simulation templates for computing physics-based behavior. Similarly,

other simulation template patterns may be included in this package. See section 2.2.2 for the definition of simulation templates and the MRA pattern in the context of this research.

- Behavior Model Formulation Method – method to formulate behavior model structures and hence simulation templates (presented in Chapter 8).
- *Meta-Model Library* – library of meta-models relevant to KCM. This consists primarily of the KCM_Meta-Model which is a collective meta-model for representing design and analysis models for VTMB problems, and consists of:
  - CPM2_xKCM – extension of the Core Product Model (Fenves 2004) meta-model for representing abstractions of an artifact, such as designed artifact, manufacturable artifact, and analyzable artifact (presented in Chapter 6)
  - CBM – a meta-model for representing artifact behavior models—both structures and instances—for VTMB problems (presented in Chapter 7)
  - ABB Meta-Model – a meta-model for representing ABBs for composing behavior models for VTMB problems (presented in Chapter 7)
  - Generic_Properties – a meta-model for representing generic properties such as geometry and material that are used for all meta-models in the KCM_Meta-Model (referred in Part 2 and defined in Appendix 3).

  Other meta-models in this library may include for example STEP (ISO 10303)-based modules that provide concepts for representing detailed design aspects of domain-specific VTMB alternatives. For instance,
- Model Structure Library – a library of model structures for test cases in the KCM Framework
- Model Instance Library – a library of model instances for test cases in the KCM Framework

The components of the KCM Framework are designed to be extensible for different design domains and different types of analyses.

5.1 Requirements

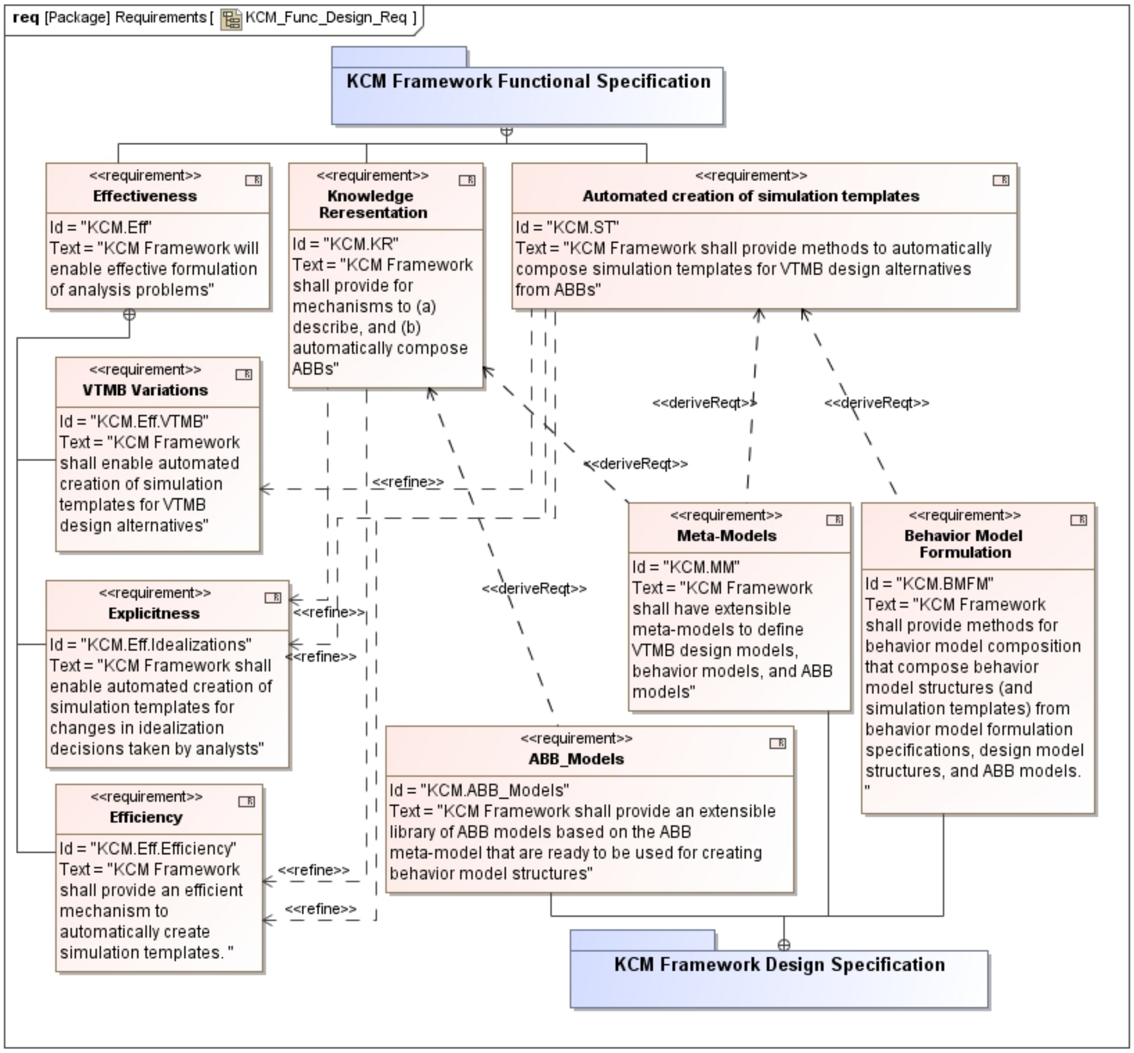
The Requirements component of the KCM Framework consists of requirements for KCM distilled from the research gaps and hypotheses. These requirements are formalized in two sets:

- KCM Framework Functional Specification consists of KCM requirements distilled from research hypotheses and research gaps.
- KCM Framework Design Specification consists of KCM requirements from the standpoint of a methodology developer and which when satisfied will also satisfy the functional specification above.

Figure 5.2 illustrates the functional and design specifications of KCM using a SysML requirements diagram. The KCM Framework Functional Specification consists of the following three requirements:

- Effectiveness – requirement related to the effectiveness of formulating analysis problems. This consists of three sub-requirements, namely VTMB Variations, Idealization Variations, and Efficiency. Collectively, these requirements state that the KCM will enable effective formulation of analysis problems by providing effective methods to handle VTMB variations and variations in idealization decisions taken by analysts. The Effectiveness is based on the definition of effectiveness in the context of this research (section 2.4).
- Knowledge Representation – requirement related to representing ABBs that embody the knowledge used by analysts in formulating behavior model structures.
- Automated creation of simulation templates – requirement related to providing methods to automatically compose simulation templates for VTMB problems from ABBs.

The formal statements of these requirements are presented in Figure 5.2 as requirements text property. Note that these three functional requirements are not mutually independent. The Effectivenss requirement and its sub-requirements are refined by the other two requirements, as shown by the <<refine>> relationship between these requirements. The Knowledge Representation requirement and Automated creation of simulation templates requirement are based on a specific approach to enhance the effectiveness of formulating analysis problems for VTMB design alternatives. This approach is founded on the research hypotheses presented in the previous chapter.



req [Package] Requirements [KCM_Func_Design_Req]
KCM Framework Functional Specification
KCM Framework Design Specification
<<refine>>
<<refine>>
<<refine>>
<<refine>>
<<refine>>
<<deriveReqt>>
<<deriveReqt>>
<<deriveReqt>>
<<deriveReqt>>

<<requirement>>
Effectiveness
Id = "KCM.Eff"
Text = "KCM Framework will enable effective formulation of analysis problems"

<<requirement>>
Knowledge Reresentation
Id = "KCM.KR"
Text = "KCM Framework shall provide for mechanisms to (a) describe, and (b) automatically compose ABBs"

<<requirement>>
Automated creation of simulation templates
Id = "KCM.ST"
Text = "KCM Framework shall provide methods to automatically compose simulation templates for VTMB design alternatives from ABBs"

<<requirement>>
VTMB Variations
Id = "KCM.Eff.VTMB"
Text = "KCM Framework shall enable automated creation of simulation templates for VTMB design alternatives"

<<requirement>>
Explicitness
Id = "KCM.Eff.Idealizations"
Text = "KCM Framework shall enable automated creation of simulation templates for changes in idealization decisions taken by analysts"

<<requirement>>
Efficiency
Id = "KCM.Eff.Efficiency"
Text = "KCM Framework shall provide an efficient mechanism to automatically create simulation templates. "

<<requirement>>
Meta-Models
Id = "KCM.MM"
Text = "KCM Framework shall have extensible meta-models to define VTMB design models, behavior models, and ABB models"

<<requirement>>
Behavior Model Formulation
Id = "KCM.BMFM"
Text = "KCM Framework shall provide methods for behavior model composition that compose behavior model structures (and simulation templates) from behavior model formulation specifications, design model structures, and ABB models.
"

<<requirement>>
ABB_Models
Id = "KCM.ABB_Models"
Text = "KCM Framework shall provide an extensible library of ABB models based on the ABB meta-model that are ready to be used for creating behavior model structures"

Figure 5.2: KCM Framework requirements – functional and design specifications

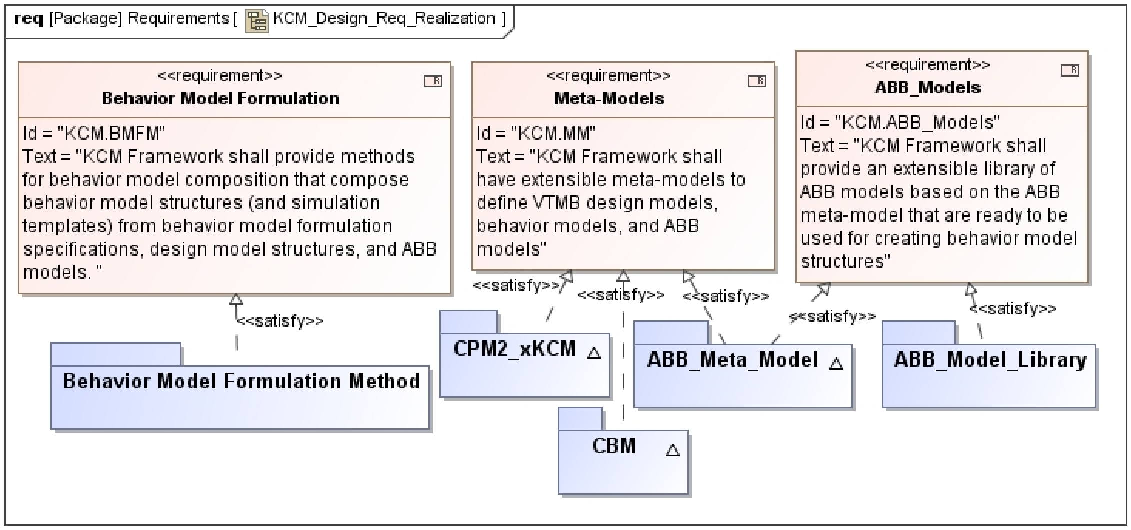
The KCM Framework Design Specification consists of the following three requirements:

- Meta-Models – requirement related to providing meta-models for representing all VTMB design models, behavior models, and ABB models.
- ABB Models – requirement related to providing an extensible library of ABBs that are building blocks of behavior model structures.
- Behavior Model Formulation – requirement related to providing methods to compose behavior model structures (and hence simulation templates) from design model

structures, ABBs, and behavior model formulation specifications (embody the idealization decisions taken by analysts).

The KCM Framework design requirements are derived from the functional requirements. Hence, each design requirement is related to the corresponding functional requirement with a <<deriveReqt>> relationship.

Figure 5.3 illustrates the KCM Framework components that satisfy the KCM design requirements (KCM Framework Design Specification). The Behavior Model Formulation Method component of the KCM Framework shall satisfy the Behavior Model Formulation requirement; the KCM_Meta-Model components (CPM2_xKCM, CBM, ABB_Meta_Model) shall satisfy the Meta-Models requirement; and ABB_Model_Library component of the KCM Framework shall satisfy the ABB_Models requirement.



Id = "KCM.BMFM"
Text = "KCM Framework shall provide methods for behavior model composition that compose behavior model structures (and simulation templates) from behavior model formulation specifications, design model structures, and ABB models. "

Id = "KCM.MM"
Text = "KCM Framework shall have extensible meta-models to define VTMB design models, behavior models, and ABB models"

Id = "KCM.ABB_Models"
Text = "KCM Framework shall provide an extensible library of ABB models based on the ABB meta-model that are ready to be used for creating behavior model structures"

Figure 5.3: KCM Framework design requirements satisfied by other components

5.2 Use Cases

A Use Case is the specification of actions performed by the system which yields an observable result that is of value for one or more actors or other stakeholders of the system (UML 2 2007). KCM use cases—represented by the Use Cases component of the KCM Framework—are a collection of use cases relevant to the KCM Framework. A use case diagram identifies a system, the use cases for that system, and the actors who are related to these use cases. In the context of the KCM Framework, the subject system is the framework itself as illustrated as a SysML Use Case diagram in Figure 5.4. The figure also shows the use cases and the actors who are the key stakeholders in the framework. A line connecting an actor to use case(s) represents the communication that occurs between the actor and the framework in realizing the actions specified by the use case(s). The primary use case of the KCM Framework is to automatically create simulation templates. This is represented by the Generate Simulation Template use case in Figure 5.4. The primary end-users of the KCM Framework are designers and analysts. However, the use cases are presented from the context of the KCM Framework as a whole, including actors such as framework developers and modelers who define and extend the KCM Framework. Hence, it shows use cases that are relevant to the methodology developer (author of this dissertation).

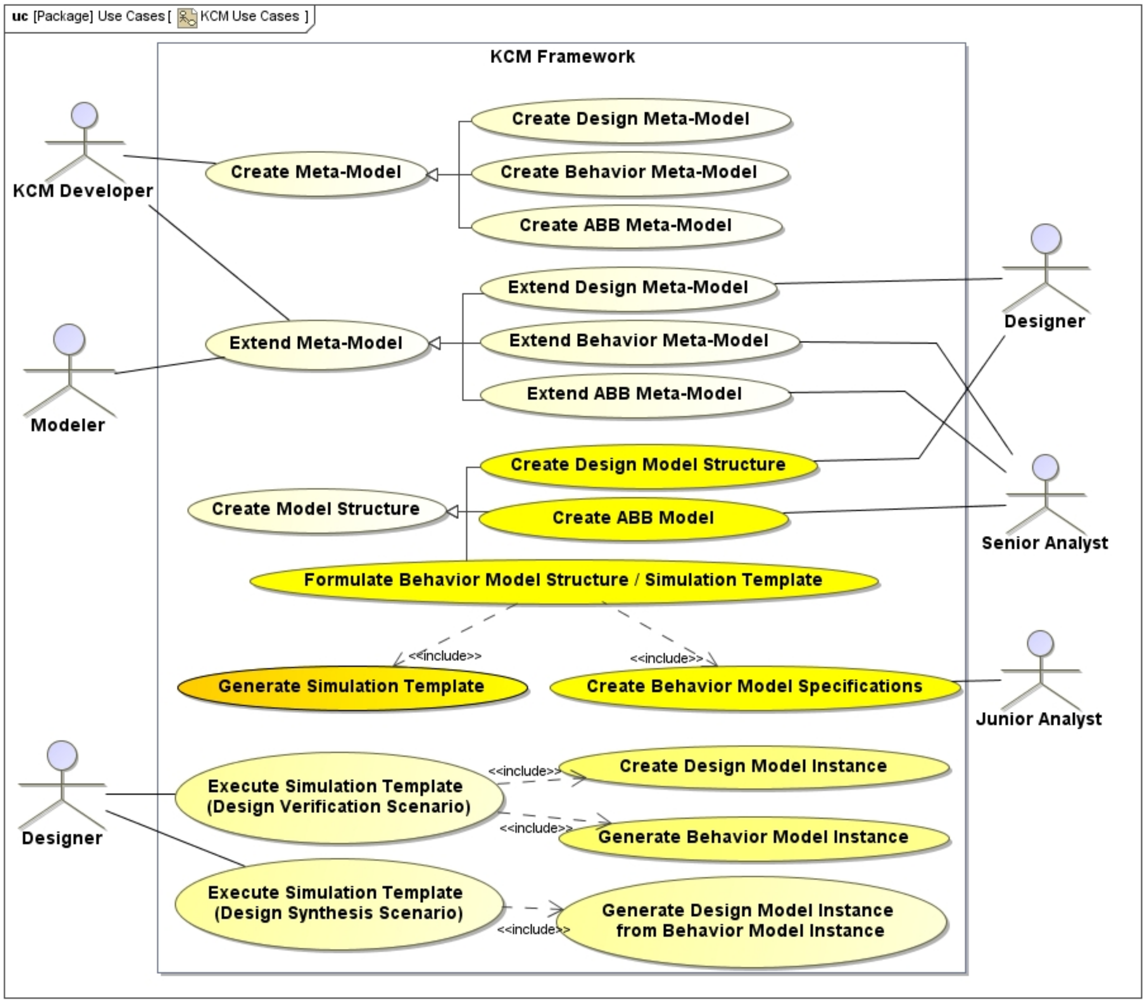

uc [Package] Use Cases [KCM Use Cases]
KCM Framework
KCM Developer
Create Meta-Model
Create Design Meta-Model
Create Behavior Meta-Model
Create ABB Meta-Model
Extend Design Meta-Model
Extend Meta-Model
Extend Behavior Meta-Model
Extend ABB Meta-Model
Designer
Modeler
Create Design Model Structure
Create Model Structure
Create ABB Model
Senior Analyst
Formulate Behavior Model Structure / Simulation Template
<<include>>
<<include>>
Generate Simulation Template
Create Behavior Model Specifications
Junior Analyst
Designer
Execute Simulation Template (Design Verification Scenario)
<<include>>
Create Design Model Instance
<<include>>
Generate Behavior Model Instance
Execute Simulation Template (Design Synthesis Scenario)
<<include>>
Generate Design Model Instance from Behavior Model Instance

Figure 5.4: KCM Use Cases

In these use cases, the term *"Generate"* implies automated creation by a computer-based method in these use cases. The use cases of the KCM Framework are summarized as below:

- Create Meta-Model use case concerns creation of meta-models for KCM Framework. The KCM Developer actor (author of this dissertation) shall realize these use cases in the form of KCM_Meta-Model component (see Figure 5.1) of the KCM Framework.
- Extend Meta-Model use cases concerns extending meta-models of the KCM Framework by KCM Developer actors and Modeler actors—users who are well-versed in the object-oriented concepts of the KCM. Designers and senior analysts provide specific scenarios (modeling needs) that aid in extending design and behavior meta-models respectively.
- Create Model Structure use case concerns the creation of model structures by designers (represented by the Designer actor), junior analysts (represented by the Junior Analyst actor), and senior analysts (represented by the Senior Analyst actor). It consists of the following three specialized use cases:
  - Create Design Model Structure use case concerns creating VTMB design model and analyzable design model structures by designers.
  - Create ABB Model use case concerns creating the structure of the ABB models by senior analysts.
  - Formulate Behavior Model Structure / Simulation Template use case concerns automated generation of the behavior model structure and simulation templates. It consists of following two sub-use cases (as also illustrated by the <<include>> relationship in Figure 5.4):
    - Create Behavior Model Specifications use case concerns formulating analyst idealization decisions as specifications for formulating behavior model structure.
    - Generate Simulation Template use cases concerns automatically creating the behavior model structure (and hence simulation template).
- Execute Simulation Template (Design Verification Scenario) use case concerns execution of simulation templates for design instances, thereby automatically generating behavior model instances.

- Execute Simulation Template (Design Synthesis Scenario) use case concerns execution of simulation templates for behavior model instances, thereby automatically generating design model instances.

5.3 Organization of KCM Components

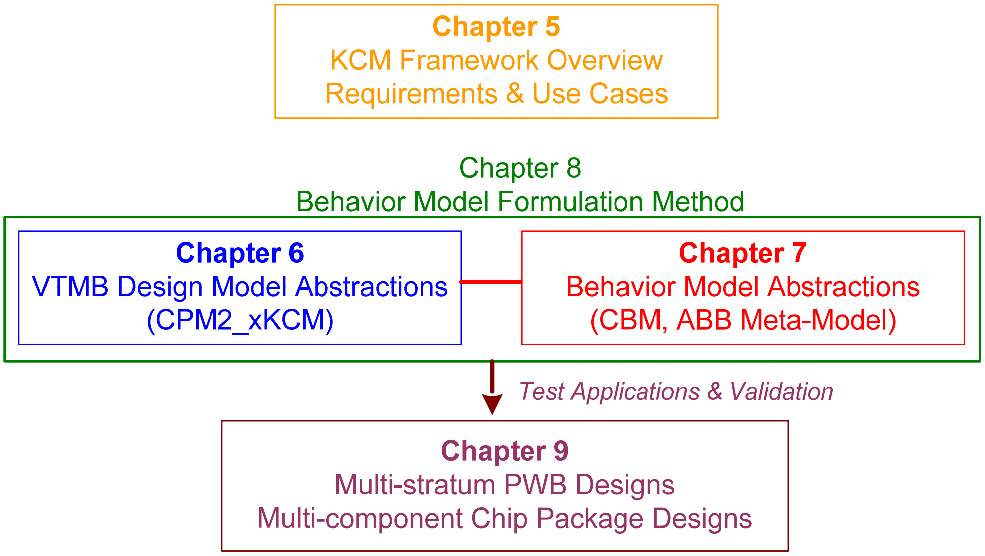

Chapter 5
KCM Framework Overview
Requirements & Use Cases
Chapter 8
Behavior Model Formulation Method
Chapter 6
VTMB Design Model Abstractions
(CPM2_xKCM)
Chapter 7
Behavior Model Abstractions
(CBM, ABB Meta-Model)
Test Applications & Validation
Chapter 9
Multi-stratum PWB Designs
Multi-component Chip Package Designs

Figure 5.5: Organization of KCM Components

Figure 5.5 illustrates the organization of KCM components in this dissertation. In this chapter (Chapter 5), an overview of the KCM Framework components was presented followed by requirements and use cases of the KCM Framework. In Chapter 6, different abstractions of VTMB design models are presented. This includes definition and detailed description of the CPM2_xKCM meta-model and its abstractions. In Chapter 7, the different abstractions of behavior models are presented. This includes definition and detailed description of the Core Behavior Model (CBM) and the ABB Meta-Model. In Chapter 8, the Behavior Model Formulation Method (BMFM) and the underlying model transformation approach is presented. The BMFM is used for formulating simulation templates—composing behavior model structures given VTMB design model structures, ABB models, and analyst idealization decisions. In Chapter 9, two test applications of the Behavior Model Formulation Method are presented in details. These test applications concern the formulation of simulation templates for thermo-mechanical analyses of multi-stratum printed wiring boards (PWB) and multi-component chip package designs respectively.

Chapter 6 : CPM2_xKCM - *An Artifact Meta-Model*

The focus of this chapter is to present the different abstractions of design models for representing variable topology multi-body alternatives. Representation of design models is a central component in formulating simulation templates using the Behavior Model Formulation Method. The different abstractions of design models for VTMB design alternatives are founded on CPM2_xKCM—a meta-model for representing VTMB design alternatives for all families of artifacts. In this chapter, the CPM2_xKCM meta-model is presented first, followed by the other abstraction levels and examples in section 6.2.

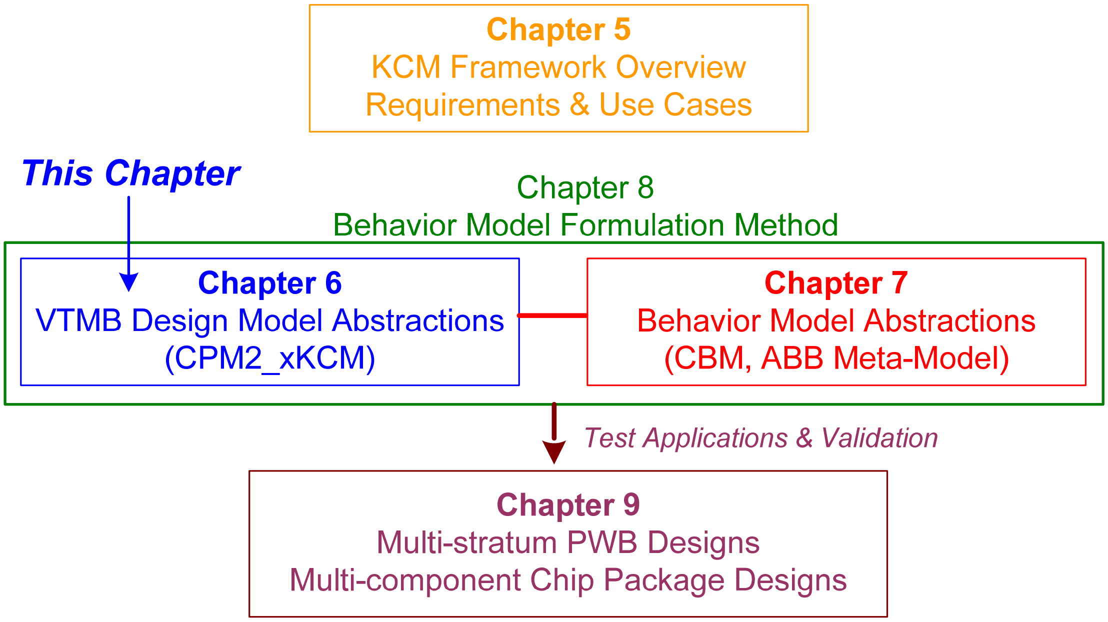

Chapter 5
KCM Framework Overview
Requirements & Use Cases
This Chapter
Chapter 8
Behavior Model Formulation Method
Chapter 6
VTMB Design Model Abstractions
(CPM2_xKCM)
Chapter 7
Behavior Model Abstractions
(CBM, ABB Meta-Model)
Test Applications & Validation
Chapter 9
Multi-stratum PWB Designs
Multi-component Chip Package Designs

Figure 6.1: VTMB Design Model Abstractions based on CPM2_xKCM – focus of this chapter

CPM2_xKCM is an extension of the Core Product Model, CPM2 (Fenves 2004), for the Knowledge Composition Methodology (KCM), and it is used to represent abstractions of an artifact for design, analysis, and manufacturing lifecycle phases, and the relationships between these abstractions. In the context of simulation templates these abstractions are necessary to define an artifact for the purposes of formulating behavior models of that artifact. In some scenarios behaviors of an artifact may be computed from its design description, while in other scenarios they may be computed from an artifact’s manufacturing description. Depending upon the product realization process, additional artifacts and features may be added to an artifact assembly when transforming design descriptions to manufacturing descriptions. Hence, it becomes necessary to evaluate the behavior of artifacts from both design and manufacturing descriptions. In both of these scenarios, analysts perform idealizations or add additional details to an artifact's description for analysis purposes. In doing so they create a description of an artifact that is ready for a family of analyses. In the KCM, this description is known as the Analyzable Artifact Model (or Analyzable Product Model) as shown in Figure 6.2. While the Core Product Model provides a basic foundation for representing artifacts across their lifecycle, CPM2_xKCM extends it by adding these abstractions.

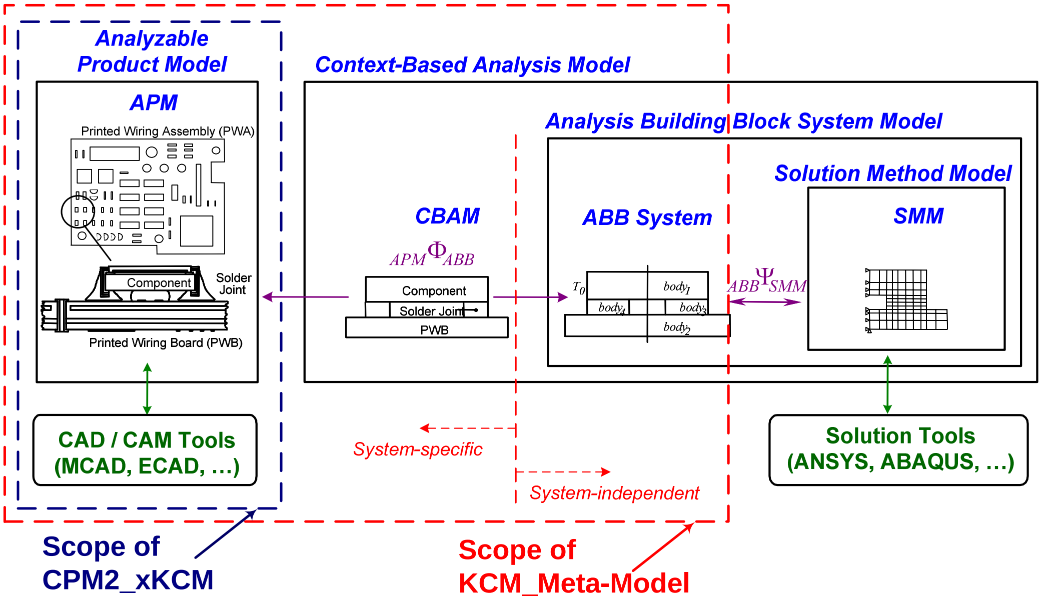


Figure 6.2: Scope of CPM2_xKCM in MRA simulate template pattern

In essence, CPM2_xKCM is a meta-model for defining an artifact as originating from CAD/CAM tools and its idealizations (AAM / APM) for analysis purposes. CPM2_xKCM is a component of the KCM_Meta-Model as shown in Figure 5.1, and can be specialized for different product domains. Detailed analyses in each product domain shall require a detailed domain-specific meta-model. The Knowledge Composition Methodology presented in this dissertation relies on the STEP (ISO 10303) application protocols (APs) and modules for detailed product definition. The STEP APs provide domain-specific meta-models that can be viewed as specializations of CPM2 and CPM2_xKCM.

Hence, behavior model formulation methods developed using CPM2_xKCM are applicable both for low fidelity analyses performed during conceptual design phases and high fidelity analyses performed during detailed design phases. In this chapter, CPM2_xKCM is described in section 6.1, and is illustrated with examples in section 6.2.

6.1 Description of CPM2_xKCM

In this section, CPM2_xKCM is described. There are two types of extensions to CPM2 that were done to formalize CPM2_xKCM. These are: (a) minor modifications to the existing concepts (esp. relationships) in CPM2, and (b) addition of new concepts—entities and relationships—to CPM2. The basic concepts in CPM2 with minor modifications are described in section 6.1.1 and new concepts are described in section 6.1.2.

6.1.1 CPM2_xKCM View 1: CPM2 with minor modifications for the Knowledge Composition Methodology

Figure 6.3 illustrates all the key classes in CPM2. While the Core Product Model was originally presented using UML, it is presented using SysML in this dissertation. Refer to Appendix 2 for a summary of SysML constructs used in this dissertation. All models and meta-models in KCM are described using SysML as it provides a common formalism to define and relate models at different levels of abstractions and to establish fine-grained associativities between them. In the SysML-based version of CPM2, a UML class maps to a SysML block, and a UML association class maps to a block with reference properties (names prefixed with "related" and "relating") to the blocks being associated.

The Core Product Model schema consists of two main abstract blocks:

- CommonCoreObject is the base abstract block for all objects.
- CommonCoreRelationship is the base abstract block for all relationships between objects.

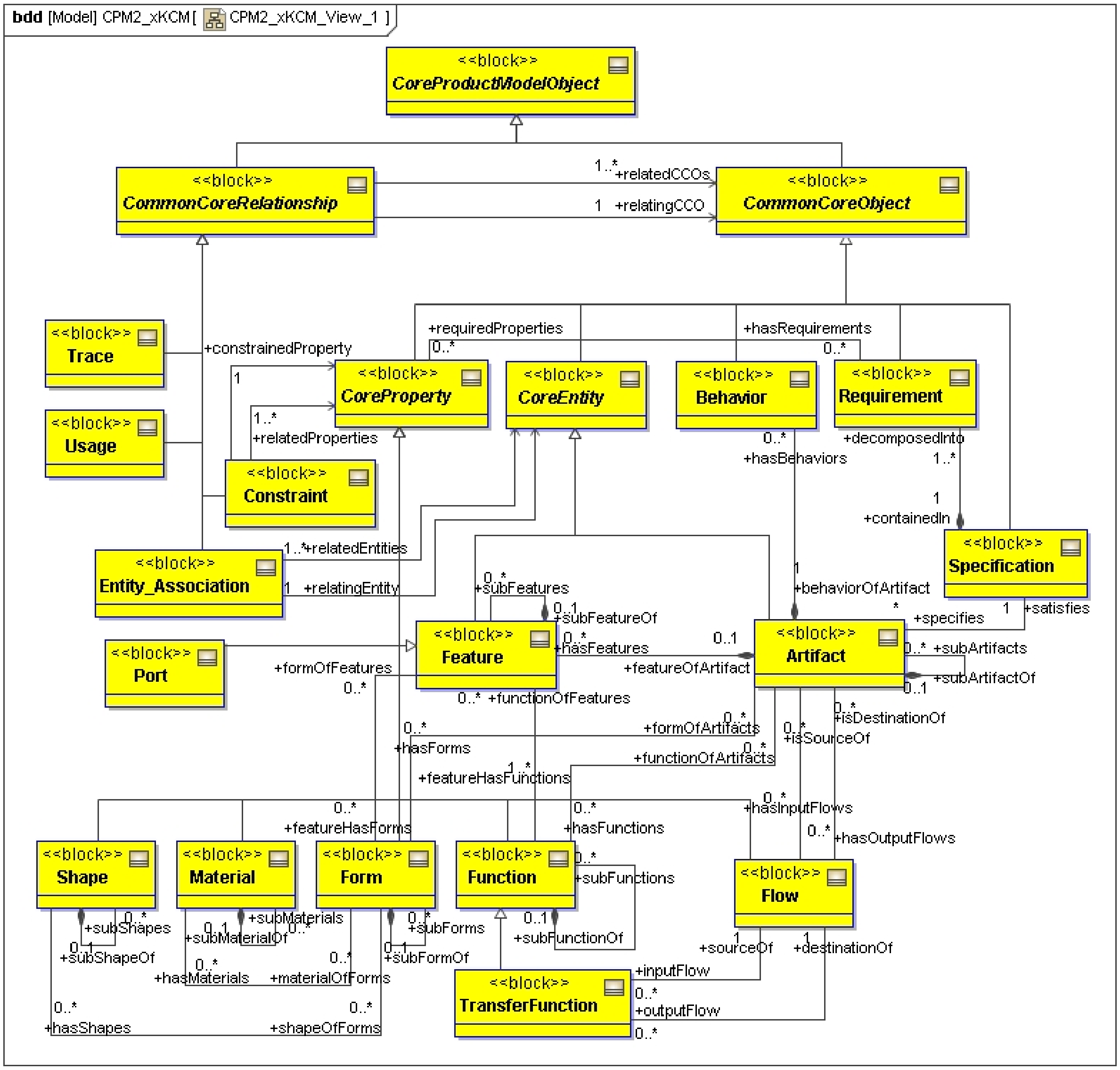


Figure 6.3: CPM2_xKCM View 1 – shows minor modifications to CPM2

The CommonCoreObject block is specialized into the following blocks.

- CoreEntity is the base abstract block for representing artifacts and their features.
- CoreProperty is the base abstract block for representing properties of artifacts such as form, function, material, shape (geometry), and flow.
- Behavior is the base block for representing behaviors of artifacts. Behavior is the response of an artifact to external stimuli such as applied forces and temperature. While function describes what an artifact is supposed to do, behavior describes what an artifact does. During analysis, specific behaviors of an artifact are computed and compared against the functional requirements. An instance of Behavior has no existence on its own,

and must be associated with the artifact whose behavior is being computed. This is reflected by the Behavior block property Behavior.behaviorOfArtifact. In CPM2, the Behavior block has the following four properties:

- behaviorOfArtifact for referencing the artifact whose behavior is to be computed and evaluated
- behaviorModels for referencing behavior models that are used to compute the subject behavior of the artifact
- observedBehavior for describing the results of computing the behavior
- evaluatedBehavior for evaluating the computed behavior against requirements

One of the key contributions of this dissertation is the Core Behavior Model (CBM)—a meta-model for describing behavior models—which is described in Chapter 7.

- Requirement is the base block for representing requirements for artifacts. A requirement applies to one or more properties of an artifact—form, function, flow, material, or geometry. Requirements are contained in a specification.
- Specification is the base block for representing a collection of requirements based on end user needs or engineering specification derived from it. A specification may or may not be satisfied by existing artifacts. Typically during early design stages, an artifact that satisfies a specification does not exist.

The CoreEntity is block is further specialized into the following blocks.

- Artifact is the base block for representing artifacts. An artifact is a distinct entity of a product, such as component, sub-assembly, or an assembly. An artifact may have multiple features as represented by the block property Artifact.hasFeatures, and a feature must be owned by an artifact, as represented by the block property Feature.featureOfArtifact. An artifact may have sub-artifacts as represented by the recursive composition relationship with roles Artifact.subArtifactOf and Artifact.subArtifacts. This is used to represent the part-assembly structure of artifacts.
- Feature is the base block for representing features of artifacts. A feature is a specific part of an artifact's form that implements one or more functions. A design or analysis or manufacturing feature implements one or more functions for the purposes of design or analysis or manufacturing process respectively. A feature may have sub-features as

represented by the recursive composition relationship with roles Feature.subFeatureOf and Feature.subFeatures.

The CoreProperty block is further specialized into the following blocks.

- Form is the base block for representing forms of artifacts and features. The form of an artifact is described using its geometric shape and constituent material. Further specializations of form depend on specializations of the artifact to which it is associated. Form may represent a proposed design form, a specific idealization of a proposed design form for analyses, or the final design form that may be used to create a bill of materials for manufacturing. Artifact.hasForm associates a form to an artifact. An artifact may have multiple forms associated with it, each representing a specific view of the artifact's form for a specific purpose (such as generating a bill of materials) or as a version of the form in-development. Form.formOfArtifacts associates artifacts to a given form. A given form may be used by multiple artifacts. A form may have sub-forms as represented by the recursive composition relationship with roles Form.subFormOf and Form.subForms.
- Function is the base block for representing functions of artifacts. Functions of an artifact describes what an artifact is supposed to do, and is derived from end user and engineering specifications. A transfer function—represented by the block TransferFunction—is a specific type of function that involves the transfer (or conversion) of an input flow to an output flow. For example, a generator is an artifact that implements a transfer function that converts mechanical energy to electrical energy. Artifact.hasFunctions associates functions to an artifact and Function.functionOfArtifacts associates artifacts to a given function. A function may be realized by multiple artifacts. A function may have sub-functions, as represented by the recursive composition relationship with roles Function.subFunctionOf and Function.subFunctions.
- Material is the base block for representing the constituent material(s) of artifacts. A material may be associated with one or more forms. Form.hasMaterials associates a material to a given form, and Material.ofForms associates a form to a given material. A given form may be associated with multiple materials, each representing a version of the material in-development for the subject form, or a view of the material used in the subject form. A material may have sub-materials as represented by the recursive composition

relationship with roles Material.subMaterials and Material.subMaterialOf. This may be used to represent alloys that are materials composed of other alloys or basic materials.

- Shape is the base block for representing shapes of artifacts and features. A given shape may be associated with one or more forms. Form.hasShapes associates shapes to a given form and Shape.ofForms associates forms to a given shape. A given form may be associated with multiple shapes, each representing a version of the shape in-development for the subject form, or a view of the shape used in the subject form. A given shape may have sub-shapes as represented by a recursive composition relationship with roles Shape.subShapeOf and Shape.hasShapes. The Knowledge Composition Methodology relies on shape representation concepts in STEP Part 42 (ISO 10303-42 2000). Those concepts are specializations of the Shape block in CPM2_xKCM.
- Flow is the base block for representing flows. A flow is the medium, such as fluid, energy, or messages that is used to realize transfer function(s). A flow can be realized by one or more artifacts, and an artifact may have multiple flow inputs and outputs.

The CommonCoreRelationship is an abstract block that associates a “relating” CommonCoreObject block to one or more “related” CommonCoreObject blocks. The CommonCoreRelationship block is specialized into the four main blocks.

- EntityAssociation block is used for representing set membership relation between CoreEntity blocks.
- Constraint is the base block for defining constraints (and more generically relations) between the properties of artifacts and features.
- Usage block is used to specify the relationship between the definition of a CommonCoreObject and its usages (possibly in different contexts). For example, if a part defined in a database occurs as a component in an assembly, the occurrence of the part and the definition are related by the Usage block.
- Trace block is similar to the Usage block. It is used to specify relationships between one CommonCoreObject and another when one depends on the other in the following manner: (a) alternative of, (b) version of, (c) derived from, (d) based on, (e) same as. So, typically relationships defined using a Trace block have a directionality. For example, if a part is an alternate / derived from / version of another source part, then the Trace block is used to associate the subject part to its source part.

The list of minor modification done to CPM2 to create CPM2_xKCM is as follows.

- Use of SysML language constructs instead of UML. The use of a SysML block with reference properties to represent relationships between concepts instead of a UML association class resulted in minor modifications to name and cardinality of reference properties. For example, CommonCoreRelationship block has reference properties relatingCCO and relatedCCOs instead of associatedCCO and ccRelationship attributes in the UML association class. The cardinality change reflects that a relationship instance must have the relating and related properties populated, i.e. an instance of a relationship is not valid unless it is associated with object instances that are being related.
- Use of SysML Composition Relationship between two blocks—denoted by a line connecting the blocks with a filled black diamond on the end of the composed block—to represent that the composed block has a part property of type of the composing block. When the composed block instance is deleted, the composing block instances shall also be deleted. This is used to represent the composition relationships between blocks representing the following pairs of concepts: Artifact-Artifact, Feature-Feature, Form-Form, Function-Function, Shape-Shape, Material-Material, Artifact-Feature, and Specification-Requirement.
- Use of SysML Association Relationship between two blocks—denoted by a line connecting the blocks—to represent that each block has a reference property of type of the referenced block. This implies that when an instance of one of the blocks is deleted, the reference relationship will be deleted but the referring block instance will not be deleted. This is used to represent the association relationships between blocks representing the following pairs of concepts: Artifact-Form, Feature-Form, Artifact-Function, Feature-Function, Form-Material, and Form-Shape.
- When a composition or an association relationship has roles with cardinality 0 or more (0..*), the name of the roles is pluralized. For example, featureHasFunction is changed to featureHasFunctions. Similar changes were done for the association relationships between blocks representing the following pairs of concepts: Artifact-Form, Feature-Form, Artifact-Form, and Artifact-Function.

- Changed the name of class (block) Geometry to Shape to better align with the term "shape" as used in STEP-based product models to represent both geometry and topology of products. Changes were made to all relationship and role names that used the term Geometry.
- Changed the cardinality of association relationship Form–Shape. An instance of a Form may exist without an instance of a Shape. This represents the use case during conceptual design or work-in-progress designs when a shape has not been defined for a form. Also, a shape may be used in multiple forms.
- Changed the cardinality of association relationship Form–Material with the same rationale as above.
- Changed the cardinality of some association relationships to allow for reuse of instances. For example, the cardinality of reference property Form.formOfArtifacts changed from 0 or 1 (0..1) to 0 or more (0..*) to allow reusing the same instance of a form for multiple instances of an artifact. Similar changes were done for the following association relationships:
    - Function.functionOfArtifacts
    - Function.functionOfFeatures
    - Form.formOfFeatures
    - Flow.isSourceOf
    - Flow.isDestinationOf
    - Shape.shapeOfForms
    - Material.materialOfForms
- Changed the name of the root object to CoreProductModelObject instead of CoreProductModel. The package containing the entities and relationships is named as CPM2_xKCM.

In this dissertation only those aspects of the Core Product Model are described that are relevant to the technical contributions of this research. It is suggested that readers refer to (Fenves 2004) for a more complete description.

6.1.2 CPM2_xKCM View 2: New concepts added to CPM2 for the Knowledge Composition Methodology

In this section, the new concepts—entities and relationships—that are added to CPM2 to create CPM2_xKCM are described. The new entities are formalized as blocks in the SysML-based representation of CPM2_xKCM, and highlighted in blue color in Figure 6.4. The figure is a SysML Block Definition Diagram (BDD) of CPM2_xKCM and only shows concepts that are added with respect to CPM2. These new concepts are described a below.

The Artifact block is specialized into Designed_Artifact, Manufactured_Artifact, and Analyzable_Artifact blocks that are described below.

- The block Designed_Artifact represents a designed artifact—the definition of an artifact in the design process. It is the central entity used for representing design-oriented information of an artifact. The design-oriented information of an artifact includes the designed artifact and sub-artifacts, designed features and sub-features.
- The block Manufacturable_Artifact represents a manufacturable artifact—the definition of an artifact for the purposes of manufacturing. It is the central entity used for representing manufacturing-oriented information of an artifact. This includes a manufacturable artifact and sub-artifacts, manufacturable features and sub-features. The manufacturing-oriented definition of an artifact is typically derived from the design-oriented definition for a particular manufacturing technology.
- The block Analyzable_Artifact is used for representing an analyzable artifact—the definition of an artifact for analyses purposes. It is the central entity used in representing analysis-oriented information of an artifact. This typically includes an analyzable artifact and sub-artifacts, analyzable features and sub-features. The analysis-oriented information of an artifact is derived from its design-, or manufacturing-, or existing analysis-oriented information of the artifact for a family of analyses.

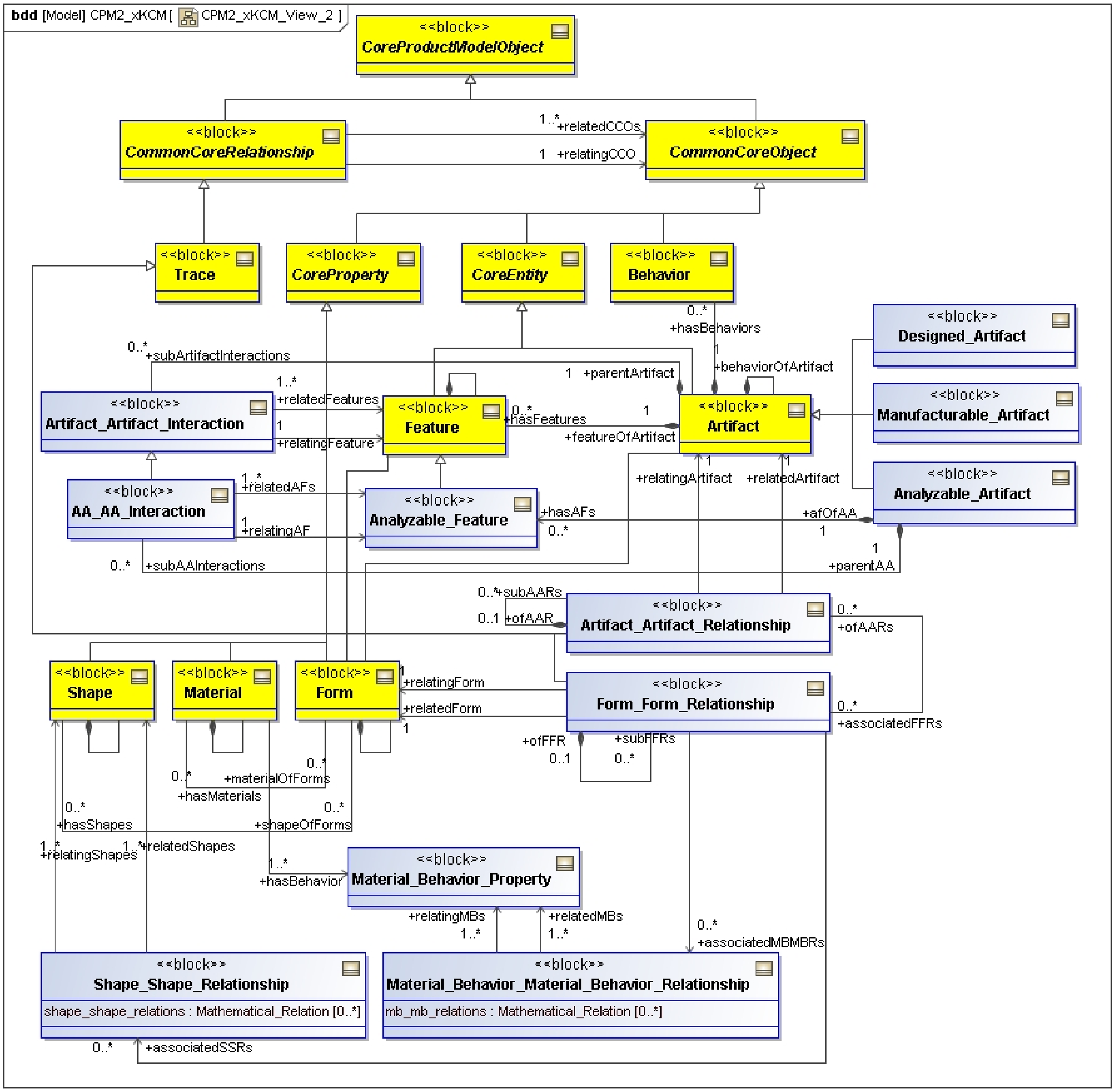


Figure 6.4: CPM2_xKCM View 2 – shows addition of new concepts to CPM2 for KCM

The Artifact_Artifact_Relationship (AAR) block is a specialization of the Trace block and represents relationships between two artifacts (or their specializations), such as those between two designed artifacts, or a designed artifact and a manufacturable (or analyzable) artifact, or between two analyzable artifacts. A designed artifact may be derived from another designed artifact. This relationship is useful for relating these abstractions of an artifact when one is derived from others in a particular context. A manufacturable artifact may be derived from another manufacturable artifact or a designed artifact, and an analyzable artifact may be derived from a designed artifact, or a manufacturable artifact, or another analyzable artifact. The AAR block has two reference properties relatingArtifact and relatedArtifacts that refer to the subject artifact (or its specialization) and all other related artifacts respectively. For example, if an analyzable artifact instance is derived from a designed artifact instance, then the analyzable artifact instance will be referred as the relatingArtifact and the designed artifact instances will be referred as the relatedArtifact in the Artifact_Artifact_Relationship instance. The AAR block also has a recursive composition relationship with roles subAARs and ofAAR. When two artifact assemblies are related using an AAR block instance, then their parts also related using AAR block instances. The composition relationship is used to contain all part AAR instances in the assembly AAR instance.

The Form_Form_Relationship (FFR) block is a specialization of the Trace block and represents relationships between two forms (or their specializations). The intent of the FFR block is similar to the Artifact_Artifact_Relationship block. It may be used for example to relate forms of two designed artifacts, or a form of a designed artifact and a form of an analyzable artifact. An Artifact_Artifact_Relationship block instance may be associated with zero or more (0..*) FFR block instances, and a FFR block instance may be associated with zero or more (0..*) Artifact_Artifact_Relationship block instances as represented by the association end roles associatedFFRs and ofAARs respectively. The cardinality of these roles is derived from the cardinality of the associated between the Artifact and Form blocks. The FFR block also has a recursive composition relationship with roles subFFRs and ofFFR. This is similar in intent to the recursive composition relationship of the AAR block. The FFR composition relationship is used for collecting FFR block instances relating child forms into a FFR block instance that relates the parent forms.

The form of an artifact refers to definitions of the constituent material and shape of that artifact. Hence, the relationship between two forms will also results in a relationship between the referred shapes, and a relationship between the referred materials. Instead of relating two materials, a relationship between two forms relates two material behaviors that characterize these materials. The FFR block has two reference properties associatedSSRs and associatedMBMBRs of type Shape_Shape_Relationship and

Material_Behavior_Material_Behavior_Relationship respectively. The block Shape_Shape_Relationship (SSR) is used to describe relationships between two or more shapes. For example, an instance of SSR block may be used to relate two shapes such that one is the result of an affine transformation on the other. A relationship between a master relating shape and set of related shapes is described using mathematical relations, and is represented by the property shape_shape_relations of type Mathematical_Relation (defined using SysML Constraint Block and explained in Chapter 7). The block Material_Behavior_Material_Behavior_Relationship (MBMBR) is used to describe relationships between two or material behaviors. For example, an instance of MBMBR may be used to relate source materials behaviors and a target material behavior such that the target is the effective material behavior computed from the source material behaviors (say by Rule of Mixtures). A relationship between source and target material behaviors is described using mathematical relations, and is represented by the property mb_mb_relations of type Mathematical_Relation (defined using SysML Constraint Block and explained in Chapter 7). MBMBR relates two or more material behaviors, each represented by the block Material_Behavior_Property. The block Material (originally in CPM2) has a reference property hasBehavior of type Material_Behavior_Property in CPM2_xKCM. This represents the relationship between the definition of a material and the definition of its behaviors.

Note that in some cases, material behavior idealization relationships are also dependent on the shape idealization relationship, such as when relating a homogenous material distribution to a heterogeneous material distribution. In such case, a new block shape_and_material_behavior_relationship may be defined as a specialization of MBMBR and SSR blocks.

The block Analyzable_Feature represents an analyzable feature. An analyzable feature is a feature defined for the purposes of analyses. Analyzable features are typically defined to specify (a) geometric features where behavior parameters are to be computed, and (b) geometric features that participate in component interactions in an analyzable artifact assembly. An analyzable feature could be same as (or derived from) a design feature or defined new for specifying analysis conditions. For the purposes of analyses, some design features may be neglected. For example, if an analyst wants to compute the shear stress at the interface between two components of an assembly, then the interface will be defined as an analyzable feature. The block Analyzable_Feature is a specialization of the block Feature. An analyzable artifact may have multiple analyzable features, and analyzable feature must be owned by an analyzable artifact. These relationships are represented by Analyzable_Artifact.hasAFs and Analyzable_Feature.afOfAA properties.

The block Artifact_Artifact_Interaction (AAI) was added to CPM2_xKCM to represent interactions between components and the features participating in these interactions when defining an assembly. The composition relationship Artifact.subArtifacts represents the component artifacts in an assembly artifact, and the composition relationship Artifact.hasFeatures represents the features of an artifact. The composition relationship Artifact.subArtifactInteractions was added in CPM2_xKCM to more explicitly represent the interactions between components in the context of defining an assembly of these components. An interaction must be defined in the context of an artifact and cannot exist on its own. This is realized by the cardinality (1) of the property Artifact_Artifact_Interaction.parentArtifact. An interaction between any two components of an assembly is realized by the features of the components participating in the interaction. An interaction is realized between a relating feature and one or more related features. The relating and related features are represented by the reference properties relatingFeature and relatedFeatures of the block Artifact_Artifact_Interaction. The block AA_AA_Interaction is a specialization of Artifact_Artifact_Interaction and is used to represent interactions between components of an analyzable artifact assembly. An interaction between two components represented by the Analyzable_Artifact block is realized by analyzable features of these two components. An interaction between any two analyzable artifacts must exist in the context of their analyzable artifact assembly. This is realized by the cardinality of the property AA_AA_Interaction.parentArtifact.

6.2 VTMB Artifact Design Models – Abstractions and Examples

In this section, the different abstractions of artifact models in the Knowledge Composition Methodology are presented. Examples of each abstraction are also presented. Figure 6.5 is a SysML block definition diagram that conceptually illustrates these five levels of abstractions (Levels 1-5, a.k.a D1-D5) of artifact models.

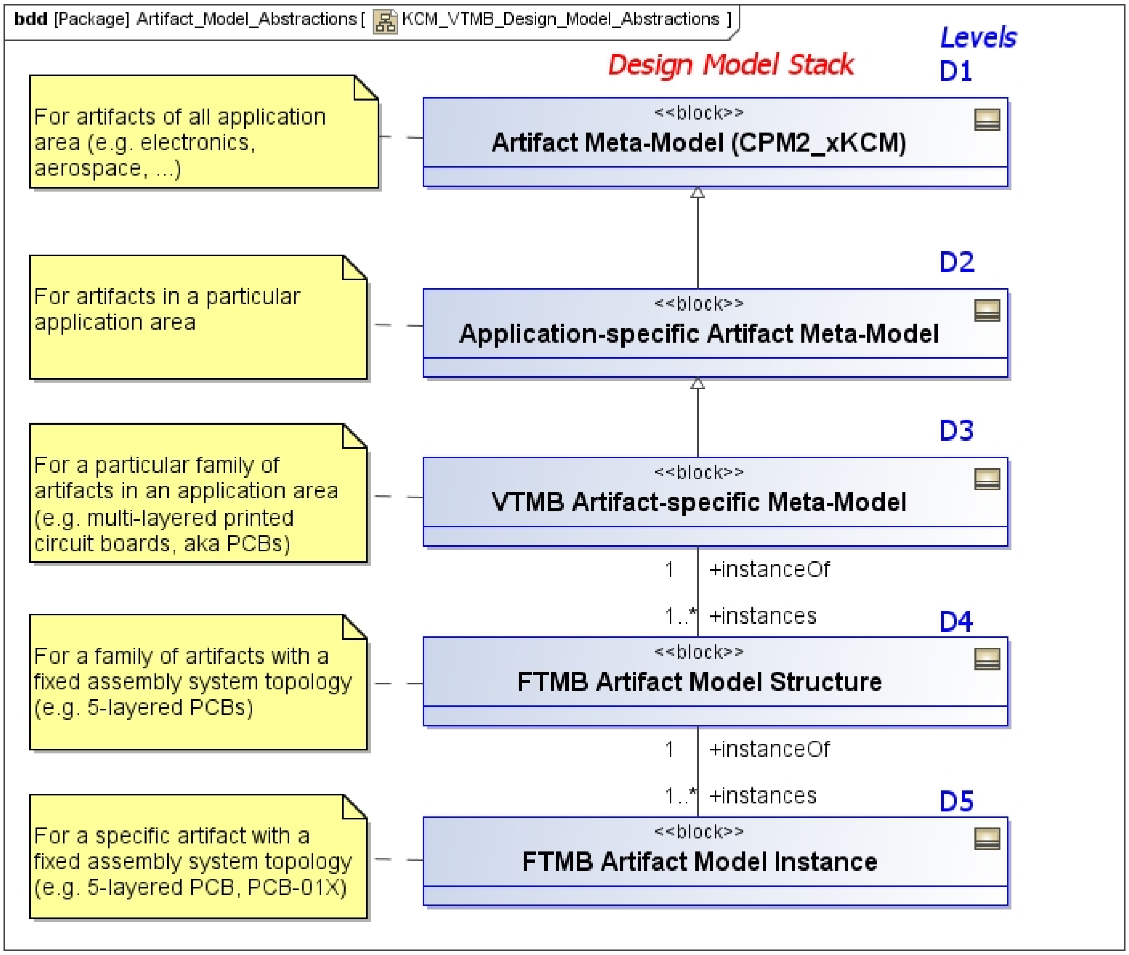

bdd [Package] Artifact_Model_Abstractions [KCM_VTMB_Design_Model_Abstractions]
Design Model Stack
Levels
D1
For artifacts of all application area (e.g. electronics, aerospace, ...)
<<block>>
Artifact Meta-Model (CPM2_xKCM)
D2
For artifacts in a particular application area
<<block>>
Application-specific Artifact Meta-Model
D3
For a particular family of artifacts in an application area (e.g. multi-layered printed circuit boards, aka PCBs)
<<block>>
VTMB Artifact-specific Meta-Model
1
+instanceOf
1..*
+instances
D4
For a family of artifacts with a fixed assembly system topology (e.g. 5-layered PCBs)
<<block>>
FTMB Artifact Model Structure
1
+instanceOf
1..*
+instances
D5
For a specific artifact with a fixed assembly system topology (e.g. 5-layered PCB, PCB-01X)
<<block>>
FTMB Artifact Model Instance

Figure 6.5: Abstractions of artifact design models in KCM – Design Model Stack

The rationale for developing these abstractions of artifact models are: (a) defining design meta-models that represent variable topology design alternatives of a particular product, and (b) identifying desing models that are associated with behavior models in simulation templates. For efficient formulation of analysis problems (and hence behavior models), it is necessary that behavior model formulation methods be applied to artifact models that represent a set of artifacts and not necessarily a specific artifact. In this manner, the resulting behavior models can be used to compute the behavior parameters for a set of artifacts. The five levels of abstractions of artifact models in KCM are described below.

- **Level 1 (D1): Artifact Meta-Model -** An Artifact Meta-Model is a meta-model that defines constructs and relationships to represent artifacts in all application areas, such as Automotive, Electronics, and Aerospace. The Core Product Model extended by the Knowledge Composition Methodology (CPM2_xKCM) is an example of such a meta-model.

- **Level 2 (D2): Application-specific Artifact Meta-Model** - An Application-specific Artifact Meta-Model defines the constructs and relationships for representing artifact in a specific application area, such as electronics or automotive. An Application-specific Artifact Meta-Model specializes an Artifact Meta-Model to represent application area-specific concepts. STEP AP210 is an example of an Application-specific Artifact Meta-Model for electromechanical artifacts, such as printed circuit boards, assemblies, and chip packages. Similarly, STEP AP214 is an example of an application-specific artifact meta-model for representing automotive artifacts.

- **Level 3 (D3): VTMB Artifact-specific Meta-Model –** A VTMB Artifact-specific Meta-Model defines the constructs and relationships for representing a specific family of artifacts, such as printed circuit boards. A VTMB Artifact-specific Meta-Model is created as a specialization of or abstracted from an Application-specific Artifact Meta-Model. In the context of KCM, a VTMB Artifact-specific Meta-Model is used for representing design and analyzable design-related information for multi-body artifacts with different assembly system topologies. Typically, D3 models are represented by artifact design templates created and maintained by designers, using system design tools such as CAD tools.

- **Level 4 (D4): FTMB Artifact Model Structure –** A FTMB Artifact Model Structure is an instance of a VTMB Artifact-specific Meta-Model, and it represents a family of multi-body artifacts with equivalent assembly system topologies, such as family of 5-

layered printed circuit boards. Here, FTMB stands for Fixed Topology Multi-Body. Typically, D4 models are represented as design models—conforming to a specific design template—where topology-specific decisions have been taken.

- **Level 5 (D5): FTMB Artifact Model Instance –** A FTMB Artifact Model Instance is an instance of an FTMB Artifact Model Structure and it represents a specific artifact in the family of FTMB artifacts, such a specific 5-layer printed circuit board. Typically, D5 models are represented in system design tools as a specific instance of a D4 model.

Note that the design model stack shown in Figure 6.5 is a conceptual model shown in SysML. In implementation, SysML does not allow instantiation of instance models—D5 is an instance of D4, and D4 is an instance of D3. In implementation, D4 is modeled as a partially-specified instance of D3, and D5 is modeled as a fully-specified instance of D3. Multiple levels of meta-modeling (not supported by UML and SysML) is a much desired feature of modeling languages (Atkinson and Kuhne 2001), especially when model transformations may be applied at different levels of model abstractions, and models at a given abstraction may serve as meta-models for transformations of models at lower (instance) levels of abstraction.

An FTMB Artifact Model Structure can be viewed as partially-specified instance of a VTMB Artifact Meta-Model where only topology-specific decisions have been taken. In contrast, a FTMB Artifact Model Instance can be viewed as a fully-specified instance of a VTMB Artifact Meta-Model.

Having defined the five different levels of abstractions of artifact models in KCM, specific examples of these abstractions are now presented. CPM2_xKCM as described in the previous section is an example of Level 1 abstraction—an Artifact Meta-Model for representing artifacts in all application domains. In this section, a Printed Circuit Board (PCB) artifact is used for illustrating the other four abstractions of artifact models. Figure 6.6 illustrates the 2D layout and through-thickness stackup of a typical PCB. A PCB consists of a stackup of materials as shown in the through-thickness view. Each layer of material is known as a stratum. A stackup is made of alternatively electrically conductive and non-conductive stratums. Conductive stratums have conductive features such as lands and traces as shown in the planar layout view. Vias and through-holes are openings in the stackup from one conductive layer to another—primarily meant to provide electrical connections across stratums.

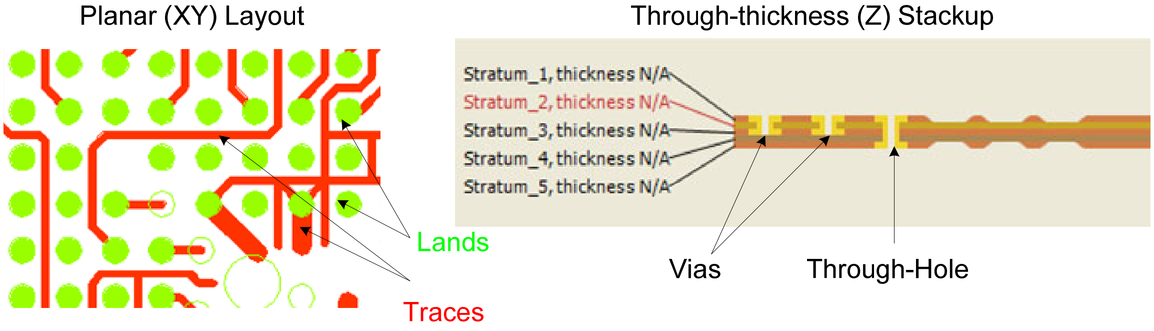


Figure 6.6: A typical Printed Circuit Board design (shown here with 5 stratums)

STEP AP210 is an example of a Level 2 abstraction for electromechanical products. Figure 6.7 and Figure 6.8 together illustrate a VTMB Artifact-specific Meta-Model for representing design and analyzable design aspects of multi-stratum printed circuit boards. Figure 6.7 illustrates PDMM—a meta-model for representing mechanical design aspects of printed circuit boards, and Figure 6.8 illustrates PAMM—a meta-model for representing analyzable design aspects of printed circuit boards (for thermo-mechanical analyses in particular). Together PDMM and PAMM constitute a Level 3 artifact model for representing printed circuit boards with different assembly system topologies. PDMM and PAMM are represented as specializations of CPM2_xKCM and contain PCB product concepts abstracted from the STEP AP210 meta-model.

Figure 6.7 illustrates the PDMM. The blocks highlighted in yellow and blue belong to CPM2_xKCM meta-model and the blocks highlighted in pink belong to PDMM. The entities and relationships represented in the PDMM are abstracted from STEP AP210. The block Electronics_Designed_Artifact is the central entity for representing design-oriented information of an artifact in the electronics domain, and is a specialization of the block Designed_Artifact. Similarly, the block Electronics_Design_Feature is the central entity used for representing design-oriented information of a feature (of an artifact) in the electronics domain, and is a specialization of the block Feature. The block PCB represents design-oriented information for printed circuit boards, and the block Stratum is used to represent design-oriented information for stratums that are stacked together to define a PCB. A PCB is composed of multiple stratums. Each stratum has a form (represented by the block Stratum_Form) that refers to the shape and material of a stratum (represented by blocks Stratum_Shape and Material). The block Adjacent_Stratum_Surface_Interaction is a specialization of Artifact_Artifact_Interaction and is used for representing the interactions between any two adjacent stratums in a stackup. Each interaction is realized by the mating of the secondary surface of the preceding stratum and the primary surface of the succeeding stratum.

This is represented by the two reference properties precedingStratumSurface and succeedingStratumSurface of the block Adjacent_Stratum_Surface_Interaction. Each stratum also has design-oriented features (represented by the block Stratum_Feature). A stratum feature may lie within a stratum (intra-stratum feature) such as in the case of lands and traces, or extend across stratums (inter-stratum feature) such as in the case of vias and plated through holes. A plated through hole is a stratum feature that extends across the entire depth of the stackup of a PCB. Intra-stratum features are represented by the block Intra_Stratum_Feature, and inter-stratum features are represented by the block Inter_Stratum_Feature. A PCB is composed of stratums, their interactions, and inter-stratum features. A stratum is composed of intra-stratum features. The PDMM can be used to represent 2-, 3-, or n-stratum PCBs and hence is a VTMB meta-model.

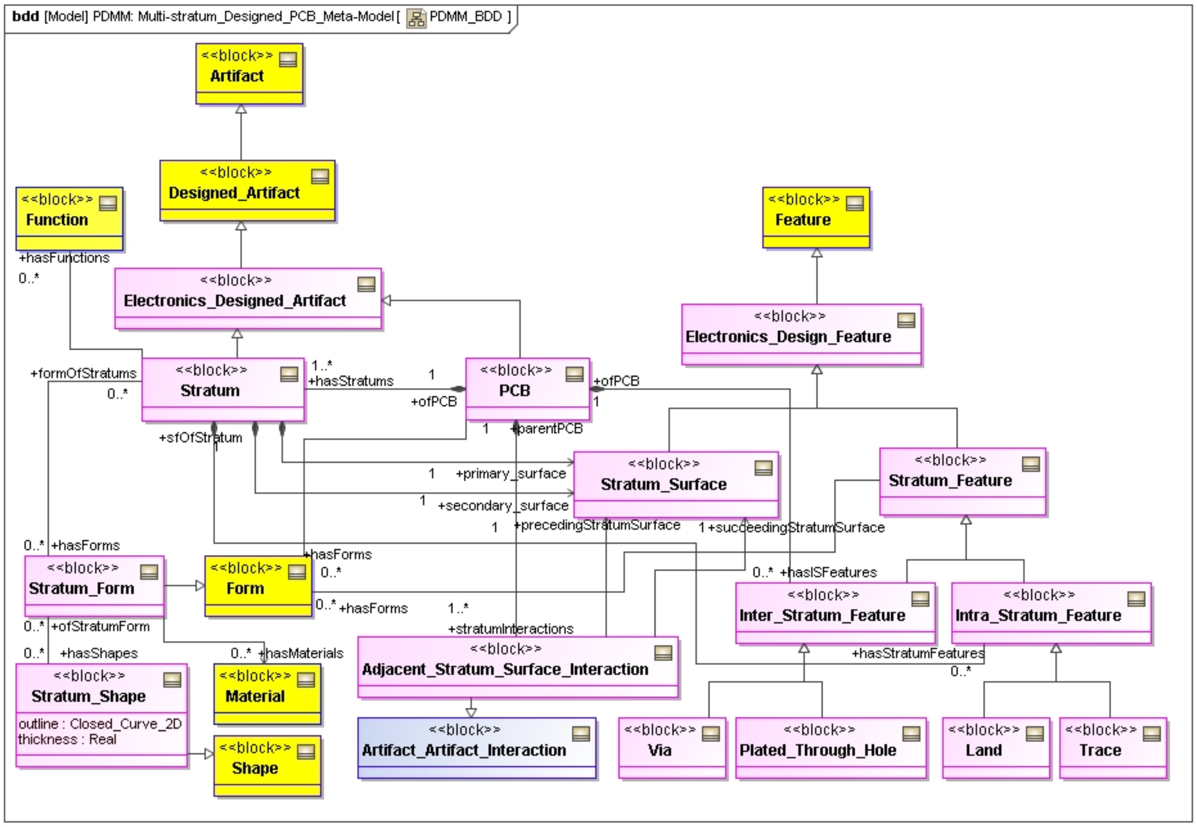


Figure 6.7: PDMM (D3) for representing mechanical design aspects of (VTMB) multi-stratum PCBs

Figure 6.8 illustrates PAMM—a meta-model for representing analyzable design aspects of printed circuit boards (for thermo-mechanical analyses in particular). This meta-model represents a specific idealization of the multi-stratum PCB designed artifact meta-model (PDMM). In this idealization—as illustrated by Figure 6.9 for a 5-stratum PCB—the intra- and inter-stratum features have been ignored for analyses purposes. Each stratum is idealized as a homogenous layer of material. In the PAMM shown in Figure 6.8, the blocks highlighted in yellow and blue belong to CPM2_xKCM meta-model and the blocks highlighted in pink belong to PAMM.

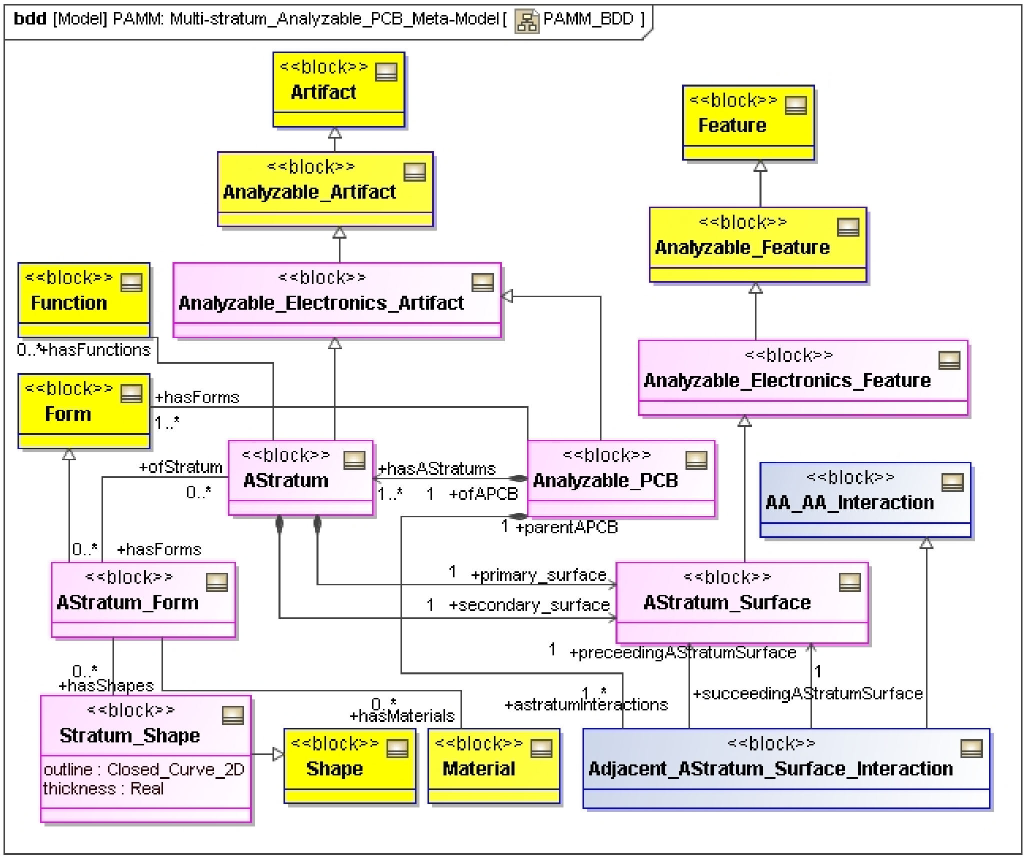


Figure 6.8: PAMM (D3): An analyzable artifact meta-model for (VTMB) multi-stratum PCBs

The blocks Analyzable_Electronics_Artifact and Analyzable_Electronics_Feature are used for representing artifacts and their features for analyses purposes. These blocks are specializations of Analyzable_Artifact and Analyzable_Feature blocks respectively. The block Analyzable_PCB and AStratum are used to represent analyzable PCBs and analyzable stratums respectively. An analyzable PCB is composed of analyzable stratums and the interactions between them (represented by the block Adjacent_AStratum_Surface_Interaction).

In the idealization represented by the PAMM here, an analyzable stratum is a homogenous layer of material and hence does not contain inter-stratum features. Similarly, an analyzable PCB does not contain intra-stratum features. The PAMM can be used to represent 2-, 3-, or n-stratum analyzable PCBs and hence is a VTMB meta-model. In a similar manner, other PAMMs can be idealized for the PDMM shown in Figure 6.7. For example, one may define an analyzable PCB that contains all the intra- and inter-stratum features (or only specific types of features)—as in the designed PCB—if such details are relevant for the specific types of analyses.

Figure 6.10 illustrates PDM_5Sx—a FTMB artifact model structure for representing mechanical design-related information for 5-stratum PCBs. This model structure is at Level 4 abstraction, and is an instance of PDMM (Level 3 abstraction). PDM_5Sx represents design-oriented information for a family of PCBs with 5 stratums. The number of stratums, interactions, and their types are fixed. Hence, the members of the family of 5-stratum PCBs have equivalent assembly system topologies.

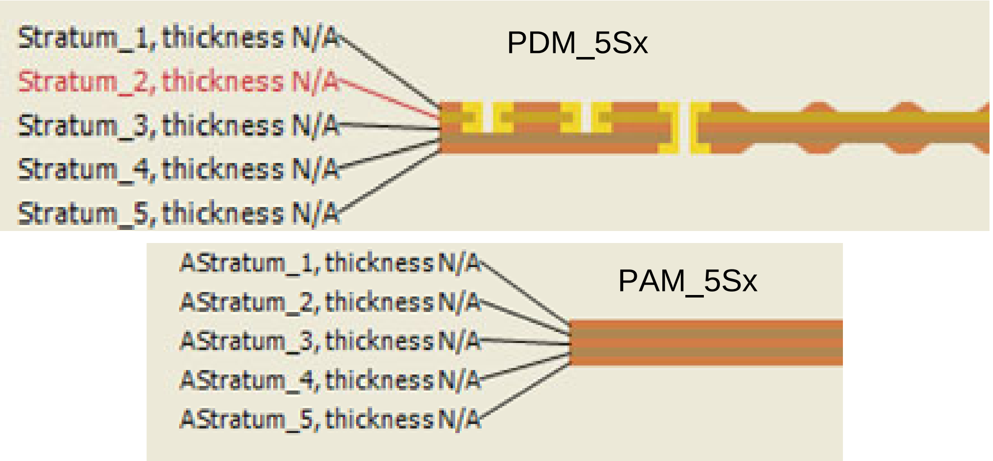


Figure 6.9: Pictoral view of PDM_5Sx and PAM_5Sx (D4 models)

The instance block PCB_5Sx represents a family of 5-stratum PCBs, and is an instance of the block PCB. PCB_5Sx has 5 stratums as represented by instances (Stratum_1 to Stratum_5) of the Stratum block, and also consists of 4 stratum interactions instances (stratum_12_interaction and so on) of the Adjacent_Stratum_Surface_Interaction block.

The preceding and the succeeding stratum surfaces in each interaction are also instantiated. The stratum interaction instances for other stratums are not shown in the figure. In the figure, not all instance information is show for each stratum but the type of instance information that exists for stratums is illustrated. For example, inter-stratum feature instances are shown only for Stratum_1 instance while the form (shape and material) instance is shown only for Stratum_5 instance. Stratum_1, Stratum_3, and Stratum_5 are conductive stratums, while Stratum_2 and Stratum_4 are non-conductive stratums—represented by the relationship between these stratum instance blocks and Conductive and Non-conductive instance blocks (of type Function block). The conductive stratums also have intra-stratum features such as lands and traces. The intra-stratum features are shown only for stratum 1. Stratum_1 instance block has 1000 lands—represented by instances Land_1_1 to Land_1_1000 of the Land block, and 400 traces—represented by instances Trace_1_1 to Trace_1_400 of the Trace block. Via_13_1 to Via_13_40 instance blocks are instances of the Via block and represent vias between conductive stratums Stratum_1 and Stratum_3. PTH_15_1 instance block is an instance of the Plated_Through_Hole block and represents a plated through hole between stratums Stratum_1 to Stratum_5. Vias and plated through-holes are examples of intra-stratum features.

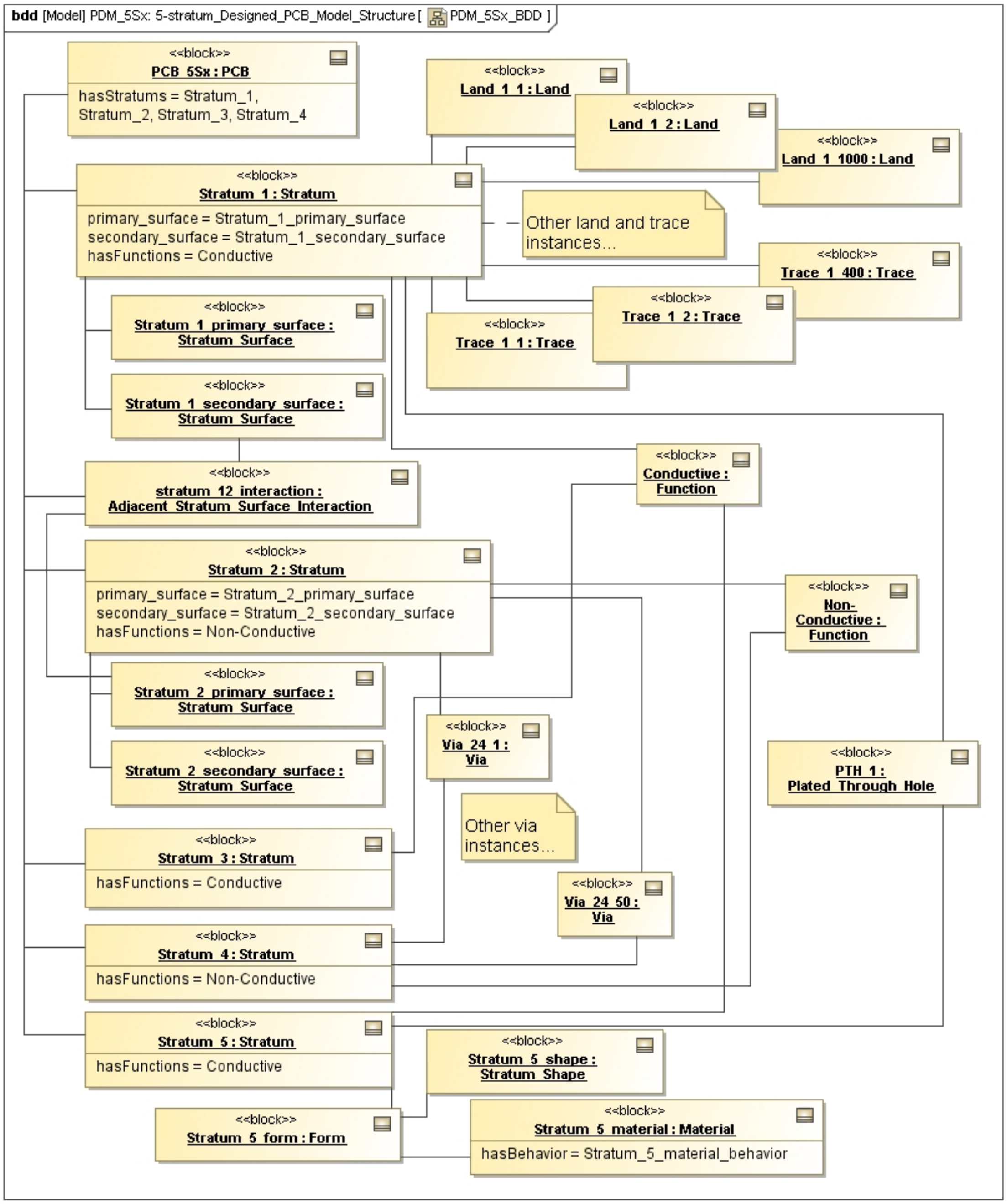

bdd [Model] PDM_5Sx: 5-stratum_Designed_PCB_Model_Structure [PDM_5Sx_BDD]
<<block>>
PCB_5Sx : PCB
hasStratums = Stratum_1, Stratum_2, Stratum_3, Stratum_4
<<block>>
Land_1_1 : Land
<<block>>
Land_1_2 : Land
<<block>>
Land_1_1000 : Land
<<block>>
Stratum_1 : Stratum
primary_surface = Stratum_1_primary_surface
secondary_surface = Stratum_1_secondary_surface
hasFunctions = Conductive
Other land and trace instances...
<<block>>
Trace_1_400 : Trace
<<block>>
Trace_1_2 : Trace
<<block>>
Trace_1_1 : Trace
<<block>>
Stratum_1_primary_surface : Stratum_Surface
<<block>>
Stratum_1_secondary_surface : Stratum_Surface
<<block>>
Conductive : Function
<<block>>
stratum_12_interaction : Adjacent_Stratum_Surface_Interaction
<<block>>
Stratum_2 : Stratum
primary_surface = Stratum_2_primary_surface
secondary_surface = Stratum_2_secondary_surface
hasFunctions = Non-Conductive
<<block>>
Non-Conductive : Function
<<block>>
Stratum_2_primary_surface : Stratum_Surface
<<block>>
Stratum_2_secondary_surface : Stratum_Surface
<<block>>
Via_24_1 : Via
<<block>>
PTH_1 : Plated_Through_Hole
Other via instances...
<<block>>
Stratum_3 : Stratum
hasFunctions = Conductive
<<block>>
Via_24_50 : Via
<<block>>
Stratum_4 : Stratum
hasFunctions = Non-Conductive
<<block>>
Stratum_5 : Stratum
hasFunctions = Conductive
<<block>>
Stratum_5_shape : Stratum_Shape
<<block>>
Stratum_5_material : Material
hasBehavior = Stratum_5_material_behavior
<<block>>
Stratum_5_form : Form

Figure 6.10: PDM_5Sx (D4): A designed artifact model structure for (FTMB) 5-Stratum PCBs

PDM_5Sx (Level 4) is a partially-specified instance of the PDMM (Level 3) because although the decisions related to the assembly system topology of the designed artifact have been taken, decisions related to specific numeric instance values (such as the exact size and shape of the PCB and stratums) have not been taken. Hence, PDM_5Sx represents a family of 5-stratum PCBs and not a specific 5-stratum PCB.

Figure 6.11 illustrates PAM_5Sx—an analyzable artifact meta-model structure for representing (FTMB) 5-stratum analyzable PCBs. PAM_5Sx is a Level 4 model and is an instance of PAMM. The instance block APCB_5Sx represents a family of analyzable PCBs with 5-stratums, and is an instance of Analyzable_PCB block. APCB_5Sx has 5 analyzable stratum instances (AStratum_1 to AStratum_5) of type AStratum block. Each analyzable stratum instance is composed of two stratum surfaces (in roles of primary and secondary surface). The interactions between adjacent stratums are realized by instances of Adjancent_AStratum_Surface_Interaction block. Each analyzable stratum also has a function—represented by instance blocks Conductive and Non-Conductive for conductive and non-conductive functions respectively.

PAM_5Sx is an idealized artifact model for the purposes of analyses, and is derived from PDM_5Sx. However, these model structures are not stand-alone. They are related. The relationships between these model structures represent the idealizations. Figure 6.12 illustrates PM_5Sx—an artifact model structure that represents the designed and analyzable model structures, and their inter-relationships for 5-stratum PCBs. PM_5Sx is at Level 4 abstraction and consists of PDM_5Sx (Level 4), PAM_5Sx (Level 4), and their inter-relationships. The figure does not illustrate all instances in these model structures and their inter-relationships. Only instances relating to designed stratum Stratum_5 and corresponding analyzable stratum AStratum_5 are shown. The designed and the analyzable artifact instances are related by instances of Artifact_Artifact_Relationship (AAR) block, and the designed and analyzable forms are related by instances of Form_Form_Relationship (FFR) block.

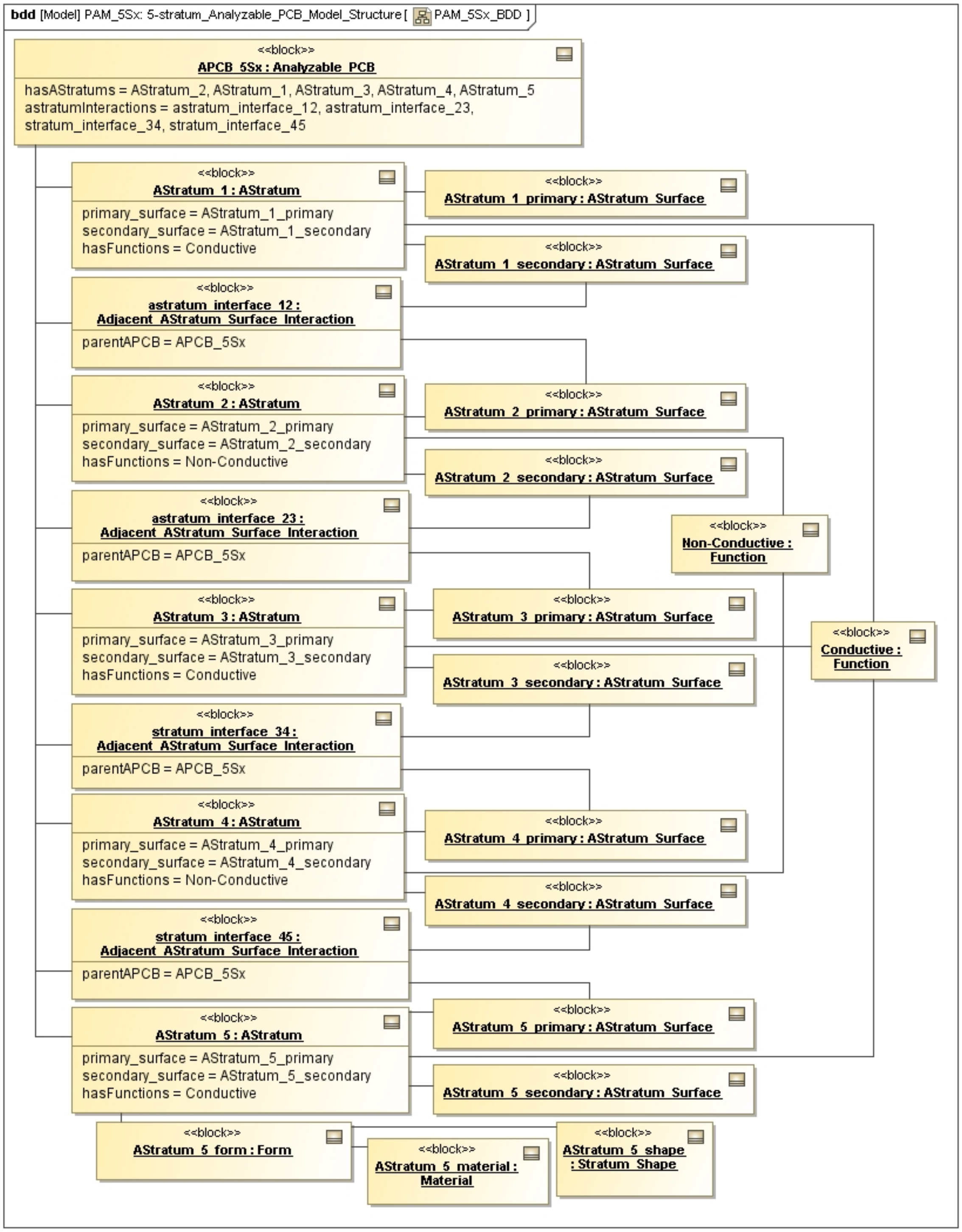

bdd [Model] PAM_5Sx: 5-stratum_Analyzable_PCB_Model_Structure [PAM_5Sx_BDD]
<<block>>
APCB 5Sx : Analyzable PCB
hasAStratums = AStratum_2, AStratum_1, AStratum_3, AStratum_4, AStratum_5
astratumInteractions = astratum_interface_12, astratum_interface_23, stratum_interface_34, stratum_interface_45
<<block>>
AStratum 1 : AStratum
primary_surface = AStratum_1_primary
secondary_surface = AStratum_1_secondary
hasFunctions = Conductive
<<block>>
AStratum 1 primary : AStratum Surface
<<block>>
AStratum 1 secondary : AStratum Surface
<<block>>
astratum interface 12 : Adjacent AStratum Surface Interaction
parentAPCB = APCB_5Sx
<<block>>
AStratum 2 : AStratum
primary_surface = AStratum_2_primary
secondary_surface = AStratum_2_secondary
hasFunctions = Non-Conductive
<<block>>
AStratum 2 primary : AStratum Surface
<<block>>
AStratum 2 secondary : AStratum Surface
<<block>>
astratum interface 23 : Adjacent AStratum Surface Interaction
parentAPCB = APCB_5Sx
<<block>>
Non-Conductive : Function
<<block>>
AStratum 3 : AStratum
primary_surface = AStratum_3_primary
secondary_surface = AStratum_3_secondary
hasFunctions = Conductive
<<block>>
AStratum 3 primary : AStratum Surface
<<block>>
AStratum 3 secondary : AStratum Surface
<<block>>
Conductive : Function
<<block>>
stratum interface 34 : Adjacent AStratum Surface Interaction
parentAPCB = APCB_5Sx
<<block>>
AStratum 4 : AStratum
primary_surface = AStratum_4_primary
secondary_surface = AStratum_4_secondary
hasFunctions = Non-Conductive
<<block>>
AStratum 4 primary : AStratum Surface
<<block>>
AStratum 4 secondary : AStratum Surface
<<block>>
stratum interface 45 : Adjacent AStratum Surface Interaction
parentAPCB = APCB_5Sx
<<block>>
AStratum 5 : AStratum
primary_surface = AStratum_5_primary
secondary_surface = AStratum_5_secondary
hasFunctions = Conductive
<<block>>
AStratum 5 primary : AStratum Surface
<<block>>
AStratum 5 secondary : AStratum Surface
<<block>>
AStratum 5 form : Form
<<block>>
AStratum 5 material : Material
<<block>>
AStratum 5 shape : Stratum Shape

Figure 6.11: PAM_5Sx (D4): An analyzable artifact model structure for (FTMB) 5-Stratum PCBs

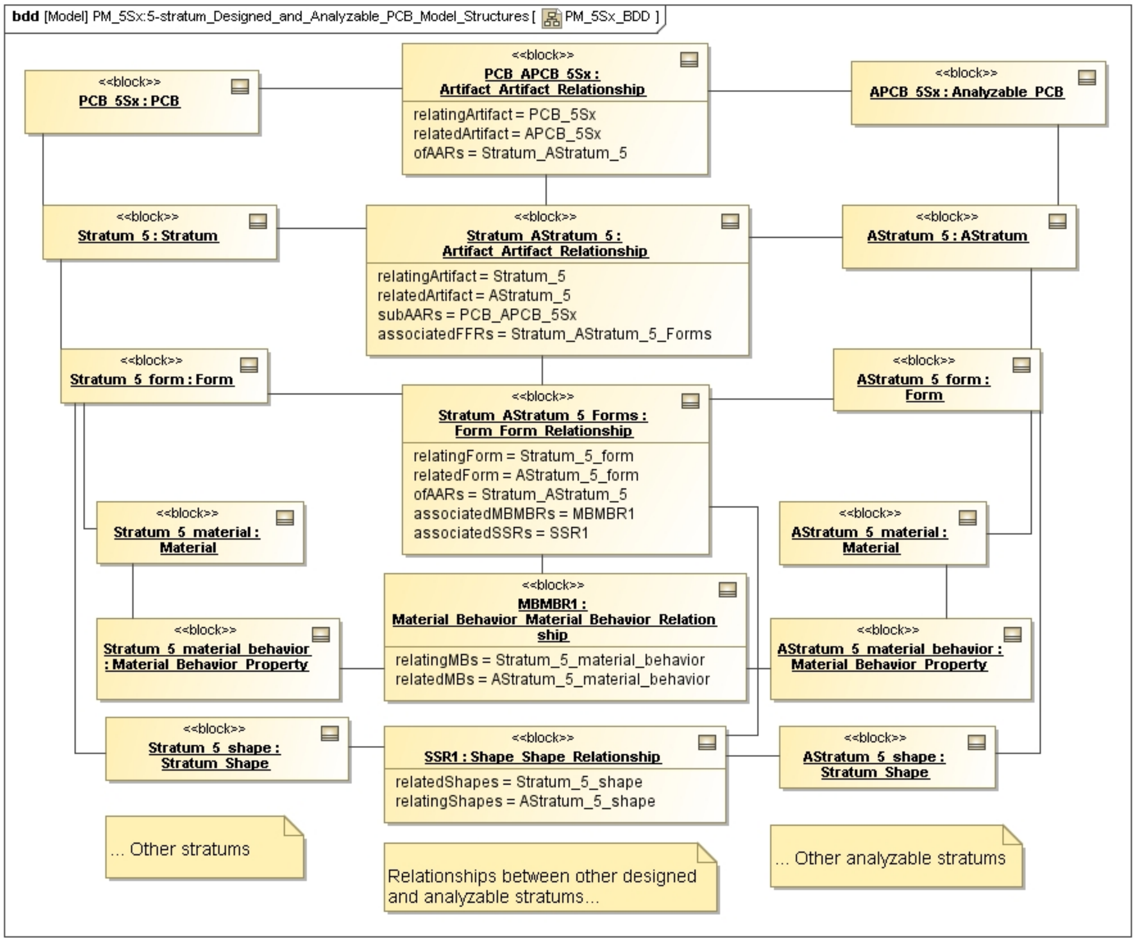

bdd [Model] PM_5Sx:5-stratum_Designed_and_Analyzable_PCB_Model_Structures [PM_5Sx_BDD]
<<block>>
PCB_5Sx : PCB
<<block>>
PCB_APCB_5Sx : Artifact_Artifact_Relationship
relatingArtifact = PCB_5Sx
relatedArtifact = APCB_5Sx
ofAARs = Stratum_AStratum_5
<<block>>
APCB_5Sx : Analyzable_PCB
<<block>>
Stratum_5 : Stratum
<<block>>
Stratum_AStratum_5 : Artifact_Artifact_Relationship
relatingArtifact = Stratum_5
relatedArtifact = AStratum_5
subAARs = PCB_APCB_5Sx
associatedFFRs = Stratum_AStratum_5_Forms
<<block>>
AStratum_5 : AStratum
<<block>>
Stratum_5_form : Form
<<block>>
Stratum_AStratum_5_Forms : Form_Form_Relationship
relatingForm = Stratum_5_form
relatedForm = AStratum_5_form
ofAARs = Stratum_AStratum_5
associatedMBMBRs = MBMBR1
associatedSSRs = SSR1
<<block>>
AStratum_5_form : Form
<<block>>
Stratum_5_material : Material
<<block>>
AStratum_5_material : Material
<<block>>
MBMBR1 : Material_Behavior_Material_Behavior_Relationship
relatingMBs = Stratum_5_material_behavior
relatedMBs = AStratum_5_material_behavior
<<block>>
Stratum_5_material_behavior : Material_Behavior_Property
<<block>>
AStratum_5_material_behavior : Material_Behavior_Property
<<block>>
Stratum_5_shape : Stratum_Shape
<<block>>
SSR1 : Shape_Shape_Relationship
relatedShapes = Stratum_5_shape
relatingShapes = AStratum_5_shape
<<block>>
AStratum_5_shape : Stratum_Shape
... Other stratums
Relationships between other designed and analyzable stratums...
... Other analyzable stratums

Figure 6.12: PM_5Sx (D4): An artifact model structure for representing designed and analyzable (FTMB) 5-Stratum PCBs

The AAR instance between the designed and analyzable PCB is composed of the AAR instances between the designed stratums and the corresponding analyzable stratums. For example, the AAR instance block PCB_APCB_5Sx consists of the AAR instance block Stratum_AStratum_5. Each instance of AAR refers to an instance of FFR that relates the forms of the artifact instances. For example, Stratum_AStratum_5 instance block refers to Stratum_AStratum_5_Forms instance block of type FFR. Each instance of FFR refers to an instance of the Material_Behavior_Material_Behavior_Relationship block (MBMBR) and Shape_Shape_Relationship block (SSR) that relates the material behaviors and the shapes of the forms being related by the subject FFR instance. For example, Stratum_AStratum_5_Forms instance block refers to MBMBR1 and SSR1 instance blocks.

Figure 6.13 illustrates PAMI_5S1—an analyzable artifact model instance that represents a specific 5-stratum analyzable PCB. PAMI_5S1 is a fully-specified instance of PAMM, and also an instance of PAM_5Sx. It is at Level 5 abstraction. PDMI_5S1 which is at Level 5 and represents design-related information of a specific PCB is not shown here. The specific analyzable PCB represented by PAMI_5S1 model is APCB_5S1. Note that PAMI_5S1 is a fully specified instance of PAMM as opposed to PAM_5Sx—a partially specified instance of PAMM—because not only is the assembly system topology decision has been taken (as in PAM_5Sx) but also specific shapes, sizes, and materials of the artifact and features have been decided. For example, analyzable stratum 5 (represented by the instance block AStratum_5) has a rectangular outline of width 5 inches and length 10 inches, and is 0.1 inches thick.

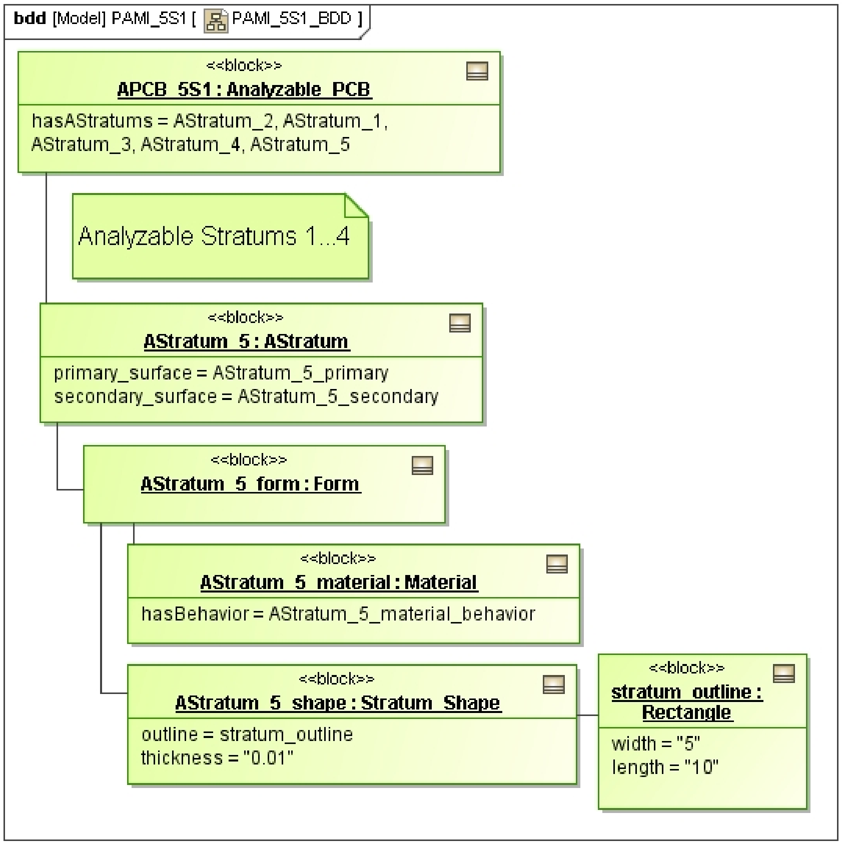

bdd [Model] PAMI_5S1 [PAMI_5S1_BDD]
<<block>>
APCB_5S1 : Analyzable_PCB
hasAStratums = AStratum_2, AStratum_1, AStratum_3, AStratum_4, AStratum_5
Analyzable Stratums 1...4
<<block>>
AStratum_5 : AStratum
primary_surface = AStratum_5_primary
secondary_surface = AStratum_5_secondary
<<block>>
AStratum_5_form : Form
<<block>>
AStratum_5_material : Material
hasBehavior = AStratum_5_material_behavior
<<block>>
AStratum_5_shape : Stratum_Shape
outline = stratum_outline
thickness = "0.01"
<<block>>
stratum_outline : Rectangle
width = "5"
length = "10"

Figure 6.13: Example D5 model - A analyzable artifact model instance for a 5-stratum analyzable PCB

The focus of this dissertation is to define transformations for formulating behavior model structures from analyzable design model structures (for VTMB problems in particular) and to provide a method for executing these transformations. Idealizations used for transforming design models to analyzable design models such as geometry-specific idealizations (Finn 1993) are well-developed. In particular (Tamburini 1999) presents the Analyzable Product Model (APM) representation—in the context of MRA simulation template pattern—for formally representing analyzable design-related information. The graph transformation-based approach for formulating behavior models for variable topology problems, as presented in this dissertation, also provides a fundamental approach for formulating analyzable models from design (or manufacturing models).

6.3 Summary

In this chapter, CPM2_xKCM has been presented as a meta-model for representing design, manufacturing, and analysis-oriented information of artifacts. Five different abstractions of artifact models are presented in the context of KCM, and illustrated for (VTMB) multi-stratum printed circuit boards. These abstractions of the designed and analyzable artifact models are central to the Behavior Model Formulation Method in the KCM. For effective formulation of behavior models, it is required that the formulation methods may be applied to analyzable artifact models that represent a set of analyzable artifacts, and not necessarily represent a single analyzable artifact. As a result of this approach, the formulated behavior model structure will represent a family of behavior models—one for each member in the family of analyzable artifacts to which the formulation methods were applied. Applying the formulation method to design alternatives with different assembly system topologies will result in corresponding behavior model structures (and simulation templates) for VTMB analysis problems.

Chapter 7 : CORE BEHAVIOR MODEL (CBM) – *AN ARTIFACT BEHAVIOR META-MODEL*

The focus of this chapter is to present the different abstractions of behavior models of variable topology multi-body design alternatives. All abstractions of behavior models are founded on the Core Behavior Model (CBM)—a meta-model that defines the constructs and relationships for representing behavior models. In this chapter, the CBM is presented first. This is followed by a presentation of Analysis Building Block (ABB) meta-models and ABB models in sections 7.2 and 7.3 respectively. ABBs define the units of knowledge that are composed for creating behavior models; and the ABB Meta-Model defines the constructs and relationships for representing different types of ABBs. In section 7.4, different abstractions of behavior models based on the CBM are presented. The analysis knowledge embodied in ABBs, and the structure of the Core Behavior Model is founded on the Analysis Knowledge Dimensions presented in section 7.5. The analysis knowledge dimensions define the types of decisions taken by analysts in formulating behavior models (and hence simulation templates) and the choices available for each type of decision. In the KCM, behavior models also include relationships to VTMB design models. Hence, the formulation of behavior models implies the formulation of simulation templates.

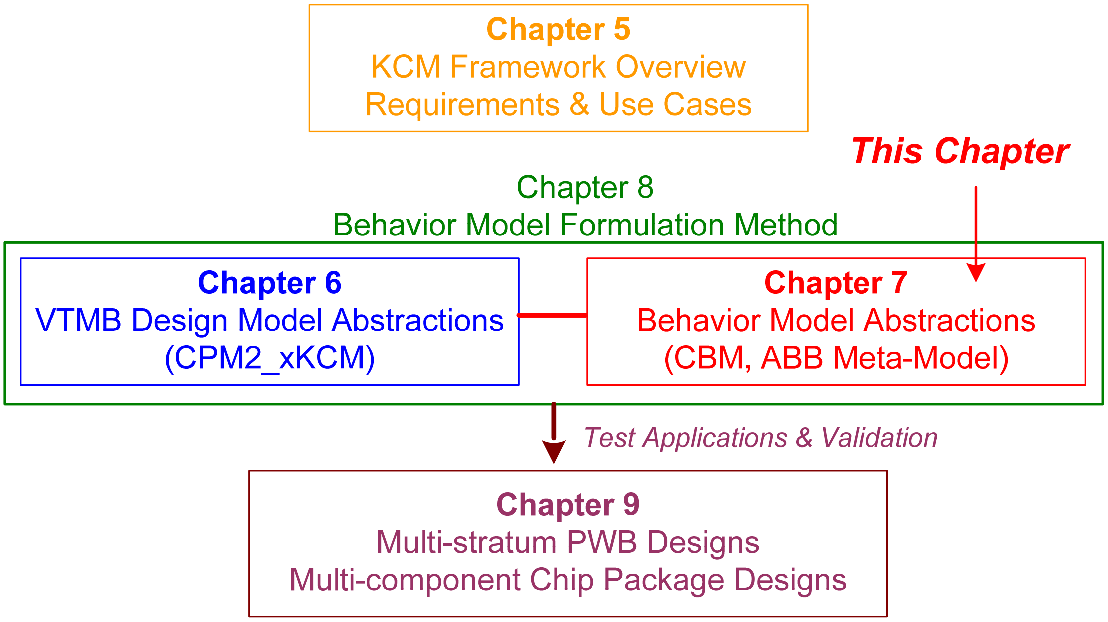

Chapter 5
KCM Framework Overview
Requirements & Use Cases
This Chapter
Chapter 8
Behavior Model Formulation Method
Chapter 6
VTMB Design Model Abstractions
(CPM2_xKCM)
Chapter 7
Behavior Model Abstractions
(CBM, ABB Meta-Model)
Test Applications & Validation
Chapter 9
Multi-stratum PWB Designs
Multi-component Chip Package Designs

Figure 7.1: Behavior Model Abstractions based on Core Behavior Model (CBM)

7.1 Core Behavior Model

The Core Behavior Model (CBM) is a behavior meta-model. It defines the constructs and relationships for representing behavior models of artifacts. A behavior model represents a set of idealized behaviors of an artifact in a behavior environment. The behavior environment is the set of external conditions under which the behavior is being computed. For example, a linear deformation model of a mechanical spring is a behavior model of the mechanical spring that can be used to compute the axial deformation behavior of a spring when axial forces are applied to the ends of the spring. The linear deformation model idealizes the behavior of the spring to be linear—deformation is directly proportional to the end forces.

7.1.1 Overview

In the KCM, an artifact behavior model is represented as an instance of the Core Behavior Model. The Core Behavior Model embodies the concept of context-based analysis models defined in the MRA simulation template pattern. In this pattern, a context-based analysis model consists of (a) an ABB system model, and (b) behavior idealization relationships (APMΦABB) between an analyzable artifact (product) model and the ABB system model. An analyzable artifact model represents an idealized artifact for a class of behavior analyses (Chapter 6). An ABB system model is a system of analysis building block models (ABB models) such as those representing analysis bodies, loads, and boundary conditions, and it represents the behavior of a system of analysis bodies. The behavior idealization relationships between an analyzable artifact and an ABB system model idealize the analyzable artifact as a system of analysis bodies. Hence, a set of behaviors of the idealized artifact are approximated as behaviors of the system of analysis bodies. For example, the deformation of a printed circuit board during the manufacturing process can be idealized as the deformation of a laminated shell subjected to thermal loading during the manufacturing process. Here, the printed circuit board is the artifact whose behavior is to be computed. The laminated shell, the thermal loading and the boundary conditions are defined in an ABB system model. Thus, an artifact behavior model that is represented as an instance of the Core Behavior Model is composed of (a) an ABB system model, and (b) idealization relationships that approximate the idealized artifact as a system of analysis bodies represented by the ABB system model. In addition, the ABB system model also represents the behavior environment in which the behaviors are computed.

The central idea in KCM—and the MRA pattern that it embodies—is that an ABB system model is the core ingredient of an artifact behavior model, and an ABB system model can be composed from reusable ABB models. Thus, the efficiency of formulating behavior models can be significantly improved if there were methods to automatically compose a behavior model from reusable ABB models. The behavior model formulate methods in KCM address this need, and are described in Chapter 8.

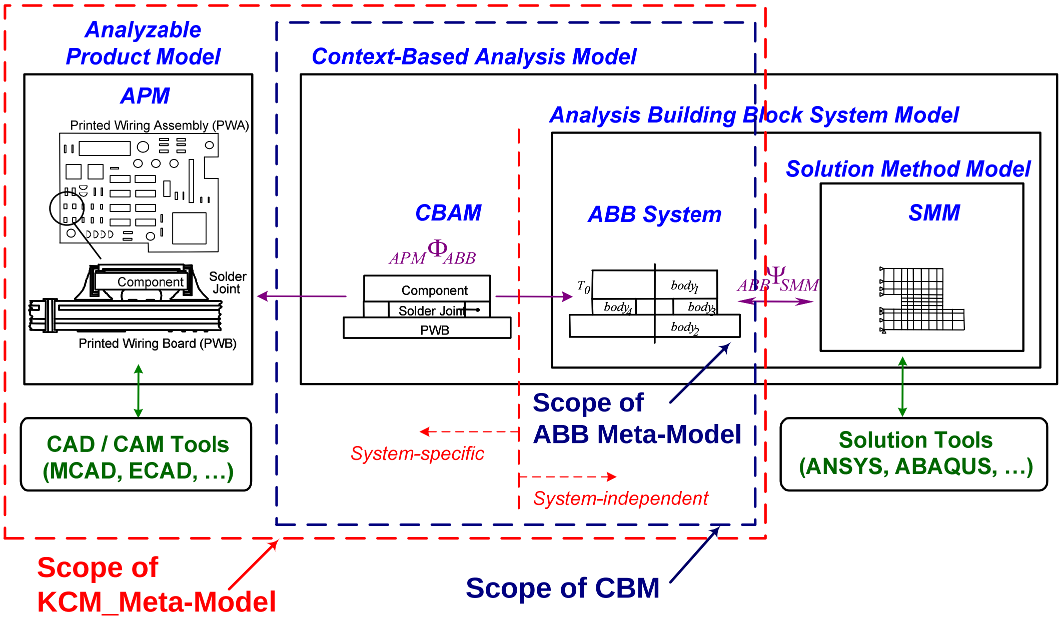


Figure 7.2: Scope of CBM and ABB Meta-Model in MRA simulate template pattern

Another meta-model presented in this chapter and closely related to the Core Behavior Model is the ABB Meta-Model. The ABB Meta-Model is a meta-model for representing ABB models and ABB system models. Figure 7.2 illustrates the scope of the Core Behavior Model and the ABB Meta-Model in the context of the MRA simulation template pattern. While the Core Behavior Model is used to represent artifact behavior models, the ABB Meta-Model is used for representing ABB models and ABB system models. The ABB Meta-Model is defined separately from the Core Behavior Model since ABB models may exist in a library of ABBs independent of their usage in an ABB system model. Additionally, an ABB system model may exist independently of its usage in an artifact behavior model.

The constructs and relationships in both the meta-models—Core Behavior Model and ABB Meta-Model—are founded on analysis knowledge dimensions described in section 7.5. Analysis knowledge dimensions represent the types of decisions taken by analysts when creating a behavior model, and provide the rationale for defining ABB models. Each ABB model is a choice for specific type(s) of decision(s) taken by analysts.

In this chapter, the Core Behavior Model is described in section 7.1. The ABB Meta-Model is described in section 7.2. An initial library of ABB models (each represented as an instance of the ABB Meta-Model) is presented in section 7.3, and in section 7.4 different abstractions of a behavior model relevant in the context of Variable Topology Multi-Body problems are presented. The analysis knowledge dimensions are described in section 7.5. Note that in this chapter, the CBM and ABB Meta-Model are described using examples. The transformations that compose ABB models to create a behavior model are presented as part of the behavior model formulation methods in Chapter 8.

7.1.2 Description

The Core Behavior Model is presented in this section. Figure 7.3 illustrates a SysML block definition diagram of the Core Behavior Model.

The Behavior_Model block is main construct for representing artifact behavior models. The Behavior block (section 6.1.1) is used for representing behaviors of an artifact. A given behavior of an artifact may be computed using several behavior models—each of a different fidelity. For example, the planar deformation of a printed circuit board is a specific behavior that may be computed using any of the following behavior models that idealize the printed circuit board as a: (a) homogenous solid, (b) homogenous shell, (c) laminated solid, or (d) laminated shell. The reference property Behavior.behaviorModels is used for representing this use case. The lower bound on cardinality of this property (0..*) represents the use case that a behavior may be instantiated without a behavior model to compute it. A given behavior model must be associated with atleast one behavior. A behavior may be used as the computation model for several behaviors. For example, a behavior model in which a printed circuit board is idealized as a laminated shell can be used for computing planar deformation behavior

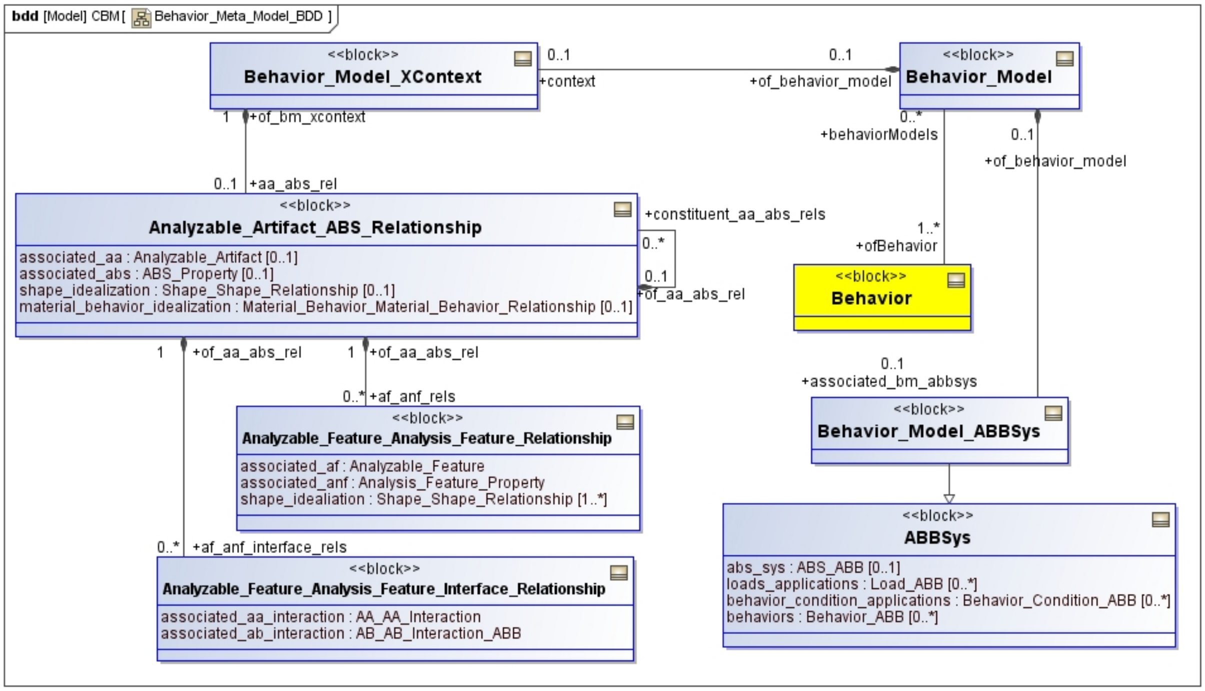

bdd [Model] CBM [Behavior_Meta_Model_BDD]
<<block>>
Behavior_Model_XContext
0..1
+context
0..1
+of_behavior_model
<<block>>
Behavior_Model
1
+of_bm_xcontext
0..*
+behaviorModels
0..1
+of_behavior_model
0..1
+aa_abs_rel
<<block>>
Analyzable_Artifact_ABS_Relationship
associated_aa : Analyzable_Artifact [0..1]
associated_abs : ABS_Property [0..1]
shape_idealization : Shape_Shape_Relationship [0..1]
material_behavior_idealization : Material_Behavior_Material_Behavior_Relationship [0..1]
+constituent_aa_abs_rels
0..*
0..1
+of_aa_abs_rel
1..*
+ofBehavior
<<block>>
Behavior
1
+of_aa_abs_rel
1
+of_aa_abs_rel
0..1
+associated_bm_abbsys
0..*
+af_anf_rels
<<block>>
Analyzable_Feature_Analysis_Feature_Relationship
associated_af : Analyzable_Feature
associated_anf : Analysis_Feature_Property
shape_idealiation : Shape_Shape_Relationship [1..*]
<<block>>
Behavior_Model_ABBSys
<<block>>
ABBSys
abs_sys : ABS_ABB [0..1]
loads_applications : Load_ABB [0..*]
behavior_condition_applications : Behavior_Condition_ABB [0..*]
behaviors : Behavior_ABB [0..*]
0..*
+af_anf_interface_rels
<<block>>
Analyzable_Feature_Analysis_Feature_Interface_Relationship
associated_aa_interaction : AA_AA_Interaction
associated_ab_interaction : AB_AB_Interaction_ABB

Figure 7.3: SysML block definition diagram of the Core Behavior Model (CBM)

and out-of-plane deformation behavior. The reference property Behavior_Model.ofBehavior is used for representing this use case.

Per the MRA simulation template pattern illustrated in Figure 7.2, a behavior model is composed of (a) an ABB system model, and (b) behavior idealization relationships (APMΦABB) between ABB system model and an analyzable artifact model. The blocks Behavior_Model_ABBSys and Behavior_Model_XContext are used for representing ABB system model and behavior idealization relationships that constitute the behavior model. The part properties Behavior_Model.context and Behavior_Model.associated_bm_abbsys realize the composition relationships. The cardinality of these part properties indicate that a behavior model instance may exist without an instance of an ABB system or an instance of Behavior_Model_XContext block that encapsulates the behavior idealization relationships, such as during the behavior model development process.

For brevity, an ABB system used in a behavior model is referred as a behavior model ABB system and it is represented by the Behavior_Model_ABBSys block. *A behavior model ABB system in itself is the behavior model of a system of analysis bodies*. Behavior_Model_ABBSys block is a specialization of the ABBSys block and has the following part properties that represent the types of ABB models that are composed to define an ABB system model:

- abs_sys part property is used for composing an analysis body assembly in an ABB system model. An analysis body represents the physical continuum whose behavior is to be computed. Note that the behavior of an analysis body assembly is an idealization of the behavior of the analyzable artifact assembly. The property type ABS_ABB is a generalization of blocks representing an analysis body or an analysis body assembly, and is described in section 7.2. Analysis body and analysis body assembly are special types of ABBs.
- load_applications part property is used for composing loads—applied to the analysis body assembly—in an ABB system model. A load is an external stimulus to which the behavior of an analysis body assembly is to be computed. The property type Load_ABB represents loads (a special type of ABB) and is described in section 7.2.
- behavior_condition_applications part property is used for composing behavior conditions—applied to an analysis body assembly—in an ABB system model. A behavior condition represents a constraint imposed on the analysis body assembly. The property type Behavior_Condition_ABB represents behavior conditions (a special type of ABB) and is described in section 7.2.
- behaviors reference property is used for representing the set of behavior parameters that may be computed for the subject ABB system model. The property type Behavior_ABB represents behaviors (characterized by behavior parameters) and is a special type of ABB described in section 7.2.

The lower bound on the cardinality of these part properties denote that during model development, an ABB system model instance may exist without the ABB model instances that compose it. The upper bound on the cardinality indicates the maximum number of ABB instances of each type that may compose an ABB system model. Note that the ABB system—as defined here—is targeted specially towards physics-based behavior models. However, specializations of the ABB system can be defined for representing different types of behaviors, such as physics-based behaviors (as in this case) and state-based behaviors.

The Behavior_Model_ABBSys block is a specialization of the ABBSys block to distinguish an ABB system model used in a behavior model from any other ABB system model. An ABB system model may be composed of two or more ABBs and may not necessarily represent the behavior of a physical continuum (analysis body system). In contrast, a Behavior_Model_ABBSys is designed to represent an ABB system model that may be solved using a solution method to compute behavior parameters of a physical continuum. Hence, a Behavior_Model_ABBSys instance must be composed of: (a) one instance of ABS_ABB that represents a physical continuum, (b) atleast one instance of Load_ABB that represents the external stimulus under which the behavior is to be computed, (c) atleast one instance of Behavior_Condition_ABB that represents the conditions under which the behavior is being computed, and (d) atleast one instance of Behavior_ABB that represents the behavior parameters that may be computed for the subject analysis body system. The first two requirements are necessary for computing behavior parameters in a solver. In addition, the third requirement may be necessary for certain class of problems. The fourth requirement is necessary a more complete definition of the model. Note that the cardinality of the part properties may have been constrained to represent these requirements but they are relaxed to represent in-development Behavior_Model_ABBSys instances.

Behavior_Model_XContext block represents the context of the behavior model—the specific analyzable artifact model whose behavior shall be computed using the behavior model. It is the main construct for representing behavior idealization relationships between an analyzable artifact model and an ABB system model. These idealization relationships associate an analyzable artifact assembly to an analysis body assembly. Specifically, idealization relationships between the following pairs of entities realize this association: (a) between components of analyzable artifact assembly and components of analysis body assembly, (b) between analyzable features and analysis features, and (c) between interactions among analyzable artifact components and interactions among analysis body components. Analyzable features are features defined in an analyzable artifact assembly (section 6.1.2) while analysis features are features defined in an analysis body assembly. Like an analysis body, analysis feature is a special type of ABB and described in section 7.2. The three types of idealization relationships above are represented by Analyzable_Artifact_ABS_Relationship, Analyzable_Feature_Analysis_Feature_Relationship, and Analyzable_Feature_Analysis_Feature_Interface_Relationship blocks respectively. The part property aa_abs_rel is of type Analyzable_Artifact_ABS_Relationship and is used for composing the behavior idealization relationship between the analyzable artifact assembly and analysis body assembly in the behavior model. The cardinality of the part property indicates that a Behavior_Model_XContext instance may exist independent of the idealization relationship but the reverse is not permitted. An idealization relationship must always be defined in the context of a behavior model.

Analyzable_Artifact_ABS_Relationship block is used for representing behavior idealization relationships between an analyzable artifact assembly and an analysis body assembly. In essence, these relationships idealize the behavior of an analyzable artifact assembly as the behavior of an analysis body assembly continuum. The Analyzable_Artifact_ABS_Relationship block has following four reference properties:

- associated_aa reference property is used for referring to the analyzable artifact assembly that is participating in the idealization relationship
- associated_abs reference property is used for referring to the analysis body (or analysis body assembly) participating in the idealization relationship
- shape_idealization reference property is used for representing the relationship between the geometric shapes of the analyzable artifact assembly and analysis body (or analysis body assembly).
- material_behavior_idealization is used for representing the relationship between material behaviors of the analyzable artifact assembly and analysis body (or analysis body assembly).

The Analyzable_Artifact_ABS_Relationship block has the following three part properties in addition to the reference properties above:

- constituent_aa_abs_rels part property is a recursive relationship used for composing idealization relationships between analyzable artifact and analysis body sub-assemblies (children) in the idealization relationship between parent assemblies.

- af_anf_rels part property is of type Analyzable_Feature_Analysis_Feature_Relationship block which is used for representing relationships between analyzable features and analysis features.
- af_anf_interface_rels part property is of type Analyzable_Feature_Analysis_Feature_Relationship which is used for representing relationships between component interfaces in the analyzable artifact assembly and component interfaces in the analysis body assembly. Specifically, it maps component interfaces in the analyzable artifact assembly to analysis body interfaces and behaviors in an analysis body assembly.

Table 7.1: Guidelines for modeling idealization relationships between analyzable artifacts and analysis bodies

Idealization case	***Modeled as***
Single analyzable artifact corresponds to a single analysis body	▪ One AA_ABS_Rel instance that relates the analyzable artifact to the analysis body
Single analyzable artifact decomposed to create an analysis body assembly	▪ One AA_ABS_Rel instance that relates the analyzable artifact to the analysis body assembly; the instance is composed of multiple AA_ABS_Rel instances of the following type. ▪ For each analysis body, an AA_ABS_Rel instance that relates the analyzable artifact to the analysis body.
Assembly of analyzable artifacts composed (or lumped) to create a single analysis body	▪ One AA_ABS_Rel instance that relates the analyzable artifact assembly to the analysis body; this instance is composed of multiple AA_ABS_Rel instances of the following type. ▪ For each analyzable artifact, an AA_ABS_Rel instance that relates each analyzable artifact to the analysis body.
Combination of decomposition and composition	▪ Combination of above

Table 7.1 above shows guidelines to model different idealization cases using Analyzable_Artifact_ABS_Relationship (AA_ABS_Rel) block instances.

Analyzable_Feature_Analysis_Feature_Relationship block is used for representing idealization relationships between analyzable features and analysis features. Analyzable features are geometric features of an analyzable artifact assembly that are defined for analysis purposes (section 6.1.2). Analysis features are geometric features of an analysis body assembly that are also defined for analysis purposes. They are a special type of ABB and are described in section 7.2.

The Analyzable_Feature_Analysis_Feature_Relationship block has the following reference properties:

- associated_af is used for referring to the analyzable feature participating in the idealization relationship
- associated_anf is used for referring to the analysis feature participating in the idealization relationship
- shape_idealization is used for representing the geometric relationship between the analyzable feature and the analysis feature.

Table 7.2 below shows guidelines to model different idealization cases using Analyzable_Feature_Analysis_Feature_Relationship (AF_ANF_Rel) block instances.

Table 7.2: Guidelines for modeling idealization relationships between analyzable features and analysis features

Idealization case	***Modeled as***
Single analyzable feature corresponds to a single analysis feature	▪ One AF_ANF_Rel instances relates the analyzable feature to the analysis feature
Single analyzable feature is decomposed to create several analysis features	▪ For each analysis feature, an AF_ANF_Rel instance that relates the analyzable features to the analysis feature ▪ One may also create an AF_ANF_Rel instance that relates the analyzable feature to the parent analysis feature—composed of the individual analysis features.

Several analyzable features composed (or lumped) to create a single analysis feature	▪ For each analyzable feature, an AF_ANF_Rel instance that relates the analyzable feature to the analysis features. ▪ One may also create an AF_ANF_Rel instance that relates the analysis feature to the parent analyzable feature—composed of the individual analyzable features.
Combination of decomposition and composition	▪ Combination of above

Analyzable_Feature_Analysis_Feature_Interface_Relationship is used for representing relationships between component interfaces in the analyzable artifact assembly and component interfaces in the analysis body assembly. It has the following three reference properties:

- associated_aa_interaction reference property is of type AA_AA_Interaction block which is used for representing component interfaces in the analyzable artifact assembly (section 6.1.2).
- associated_ab_interaction reference property is of type AB_AB_Interaction_ABB block which is used for representing analysis body interactions in an analysis body assembly. Analysis body interaction is a special type of ABB and described in section 7.2. The interaction is described by specifying analysis features participating in the interaction and the interaction behavior in terms of mathematical relations between behavior parameters of the participating analysis bodies.

The Core Behavior Model accounts for multi-physics analyses in two possible ways: (a) defining behavior models that have specialized analysis bodies representing coupled behavior, such as analysis bodies that represent both structural and thermal behaviors, and (b) defining separate behavior models—one corresponding to each analysis discipline—and relating the behavior parameters in one model to the load (or behavior condition) parameters in another behavior model, such as when thermal loads result in temperature changes in an analysis body system, causing structural deformation.

The Core Behavior Model is illustrated using examples in section 7.4.

7.2 ABB Meta-Model

In this section, the ABB Meta-Model is presented. The ABB Meta-Model is a meta-model for representing analysis building blocks models (ABB models) and analysis building block system models. The ABB Meta-Model is described here by specifically focusing on the following key questions.

1. What is an ABB model?
2. What are the different types of ABB models?
3. What is the type of knowledge embodied in ABB models?
4. What is an ABB system model?
5. What is the type of knowledge embodied in an ABB system model?

Aspects of the ABB Meta-Model that address questions 1, 2, and 3 are presented in section 7.2.1 and those that address questions 4 and 5 are presented in section 7.2.2.

7.2.1 Analysis building block (ABB) model

The ABB Meta-Model defines the constructs and relationships for representing analysis building block models. In the Knowledge Composition Methodology, an ABB model is defined as follows.

An **Analysis building block (ABB) model** represents a specific aspect of domain theoretic knowledge (section 3.2.3) that is necessary for defining behavior models of artifacts. An ABB model is the atomic unit for representing this knowledge.

ABB models (referred as ABBs for brevity) represent choices available to analysts when taking decisions for creating behavior models. There are several types of ABBs. All ABBs of a given type correspond to choices available to analysts for taking a specific type of decision. Examples of types of ABBs (and choices for each type) are as follows: Analysis Body ABB (plane stress analysis body, shell analysis body); Load ABB (point force load, temperature load); and Analysis Body Interaction ABB (shell-shell interaction, solid-shell interaction). In the KCM, ABBs are derived and organized based on analysis knowledge dimensions (section 0). The dimensions are a conceptual organization of types of decisions (and available choices) and consistency and completeness of a set of decisions for creating behavior models. The SysML block definition diagram shown in Figure 7.4 below illustrates the types of ABBs that are represented using the ABB Meta-Model.

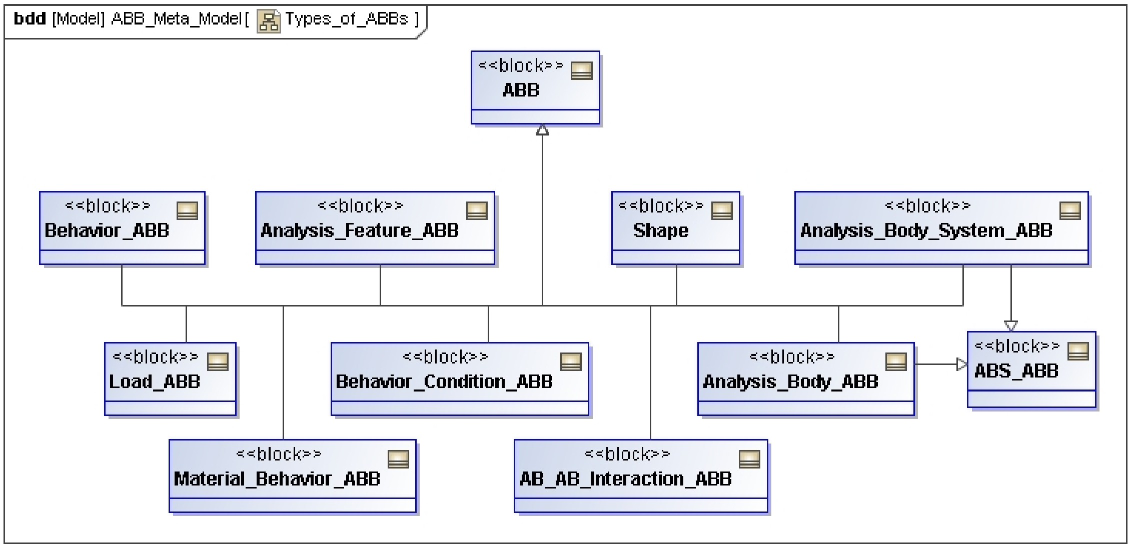


Figure 7.4: Types of ABBs represented using the ABB Meta-Model

All ABBs are modeled as specializations of the ABB block. The type of decision represented by each specialization of the ABB block is as follows:

- Analysis_Body_ABB block is used for representing analysis bodies. It represents the form and idealized behavior of a family of analysis bodies.

An **analysis body** is an idealization of an artifact such that it exhibits an idealized sub-set of behaviors of the artifact. These behaviors are formalized as mathematical expressions relating the behavior parameters, the form parameters of the analysis body, and the behavior environment in which the behaviors are computed (such as load and behavior conditions). Some examples of analysis bodies are plates, shells, membranes, linear springs, and linear resistor. For instance, when an artifact is idealized as a linear spring, its axial extension/compression behavior is abstracted from other behaviors that an artifact may exhibit and this extension/compression behavior is idealized to be linear and elastic like a spring (i.e. linear deformation is directly proportional to the applied extensional forces). The intent of defining an analysis body is to idealize the behavior of an analyzable artifact as the behavior of an analysis body (or an analysis body assembly). The behavior models of an analysis body are well-established from existing knowledge—analytical models derived from domain theories to response surface models derived from physical experiments.

- Analysis_Body_System_ABB block is used for representing analysis body assemblies (or systems). An analysis body assembly is represented using a set of analysis bodies that are components of the assembly and the interactions among these analysis bodies. The analysis body components in an analysis body assembly are usages of pre-defined analysis bodies (represented as analysis body models), and the interactions among these components are usages of pre-defined interactions (represented as analysis body interaction behavior models). Example of analysis body assemblies are solid-shell assembly, or beam-shell assembly.

- ABS_ABB block is generalization of Analysis_Body_ABB block and Analysis_Body_System_ABB block.

- Analysis_Feature_ABB block is used for representing analysis features defined on analysis bodies or an analysis body assemblies.

An **analysis feature** is a specific aspect of the shape of analysis body or analysis body assembly that is defined for analysis purposes such as to define geometric regions where behavior parameters are to be computed, or regions where loads and behavior conditions are to be applied.

- AB_AB_Interaction_ABB block is used for representing interaction behaviors between analysis bodies. The interaction behavior among analysis bodies in an assembly can be defined using math models relating behavior parameters of the analysis bodies at their interaction regions (represented as analysis features).

- Shape block is used for representing geometric shapes. Since this construct is used for other meta-models in the KCM, its name does not have the ABB suffix as the case with other types of ABB models described here.

- Material_Behavior_ABB block is used for representing constitutive material behavior of analysis bodies. Examples of material behavior models are: linear elastic isotropic temperature-independent material behavior and linear viscoelastic isotropic temperature-independent mater behavior.

- Load_ABB block is used for representing loads applied to an analysis body or an analysis body assembly.

A **load** is the stimulus to which the response of an analysis body (or analysis body system) is to be computed. Loads are applied to analysis features defined on analysis bodies or analysis body assemblies. Some examples of loads are: force, moment, temperature, and heat generation rate.

- Behavior_Condition_ABB block is used for representing behavior conditions. Behavior conditions are additional conditions applied to analysis body or analysis body assemblies under which their response to loads is to be computed. Examples of behavior conditions include initial value conditions or boundary conditions. Behavior conditions are typically described using math constraints involving behavior parameters.

- Behavior_ABB block is used for representing the set of behavior parameters that may be computed for a given analysis body or an analysis body assembly.

The ABB Meta-Model also defines the specific aspects of domain theoretic knowledge represented for each ABB type. It defines four foundational aspects of this knowledge. These are:

- Context—to identify the domain theoretic concept being represented by an ABB. The context for each ABB type is defined in section 7.2.1.1. The context attribute of an ABB is static—not instantiated with an ABB instance. This is because the context attribute of an ABB defines the characteristics of the specific ABB class and not its instances.
- Property—to model the domain theoretic concept as parameters and relations. The properties for each ABB type are defined in section 7.2.1.2.

- Application Conditions—to describe the conditions that must be satisfied for using an ABB when composing ABB systems or sub-systems. The application conditions for each ABB type are defined in section 7.2.1.3. The application conditions attribute of an ABB is also static since it defines the characteristics of the specific ABB class and not its instances.
- Application Transforms—to define the behavior model composition transformations when an ABB is used to compose ABB systems (and hence behavior models). The application transforms for each ABB type are defined in section 7.2.1.4. The application transforms attribute of an ABB is also static since it defines the characteristics of the specific ABB class and not its instances.

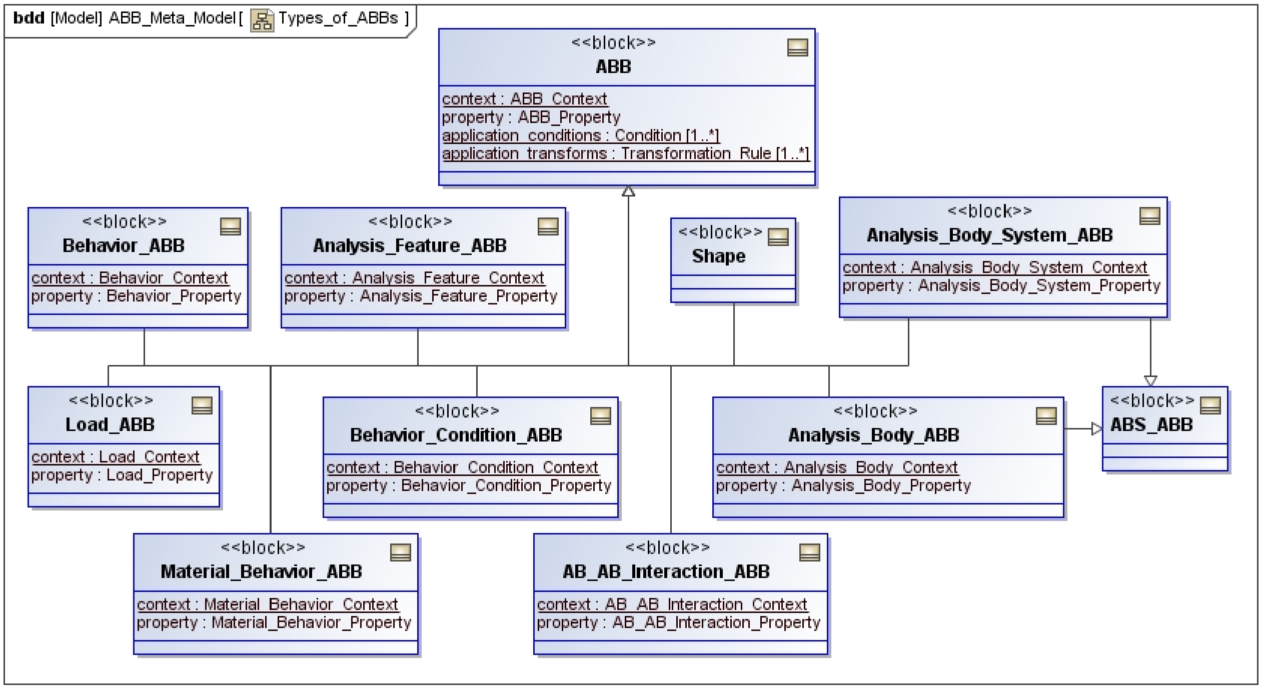

bdd [Model] ABB_Meta_Model [Types_of_ABBs]
<<block>>
ABB
context : ABB_Context
property : ABB_Property
application_conditions : Condition [1..*]
application_transforms : Transformation_Rule [1..*]
<<block>>
Behavior_ABB
context : Behavior_Context
property : Behavior_Property
<<block>>
Analysis_Feature_ABB
context : Analysis_Feature_Context
property : Analysis_Feature_Property
<<block>>
Shape
<<block>>
Analysis_Body_System_ABB
context : Analysis_Body_System_Context
property : Analysis_Body_System_Property
<<block>>
Load_ABB
context : Load_Context
property : Load_Property
<<block>>
Behavior_Condition_ABB
context : Behavior_Condition_Context
property : Behavior_Condition_Property
<<block>>
Analysis_Body_ABB
context : Analysis_Body_Context
property : Analysis_Body_Property
<<block>>
ABS_ABB
<<block>>
Material_Behavior_ABB
context : Material_Behavior_Context
property : Material_Behavior_Property
<<block>>
AB_AB_Interaction_ABB
context : AB_AB_Interaction_Context
property : AB_AB_Interaction_Property

Figure 7.5: Aspects of domain theoretic knowledge represented in each ABB type

Figure 7.5 above illustrates how these four foundational aspects are represented for each ABB in the ABB Meta-Model. Note that the static attributes of each ABB are underlined.

Details of the four foundational aspects of ABBs are as described below.

7.2.1.1 ABB Context - what concept is being represented?

This aspect of an ABB is used to represent contextual knowledge about the domain theoretic concept represented by the ABB. This contextual knowledge can be used by analysts to query ABBs in a library and to test the mutual compatibility of candidate ABBs selected for composing ABB system models. The contextual knowledge is modeled by populating the contextual attributes of each ABB with pre-defined keywords. The type of the each contextual attribute is a list of allowable keywords for that attribute. The allowable keywords for each attribute are governed by the blocks (classes) in the Analysis Body Dimension model defined in section 7.5.2 and KCM's Generic Properties Meta-Model defined in section Appendix 3. In effect, the keywords tag an ABB thus making it easier to search it in a large library of ABBs. The set of keyword tags for each ABB is unique and unambiguous. For example, the contextual attributes for material behavior ABB are the following: (i) material behavior parameters—set of parameters used for characterizing material behavior, such as Young's Modulus and Poisson's Ratio, (ii) material behavior discipline, such as structural behavior and thermal behavior, (iii) material behavior distribution, such as isotropic and orthotropic, (iv) material behavior variation, such as linear, bi-linear, non-linear. Material behavior variation is further characterized as (a) variation of stress with strain, (b) variation of material behavior parameters with temperature, and (c) variation of stress and material behavior parameters with strain rate. In this manner, the context attribute of each ABB, when populated, allows analysts to query ABBs from a library of ABBs.

Figure 7.6 illustrates the ABB Context Meta-Model (subset of the ABB Meta-Model) for representing the contextual knowledge in ABBs. The ABB Context Model defines the contextual attributes for each ABB type. The ABB Context Meta-Model is founded on the analysis knowledge dimensions (section 7.5) and is extensible to defining the contextual attributes of other types of ABBs..

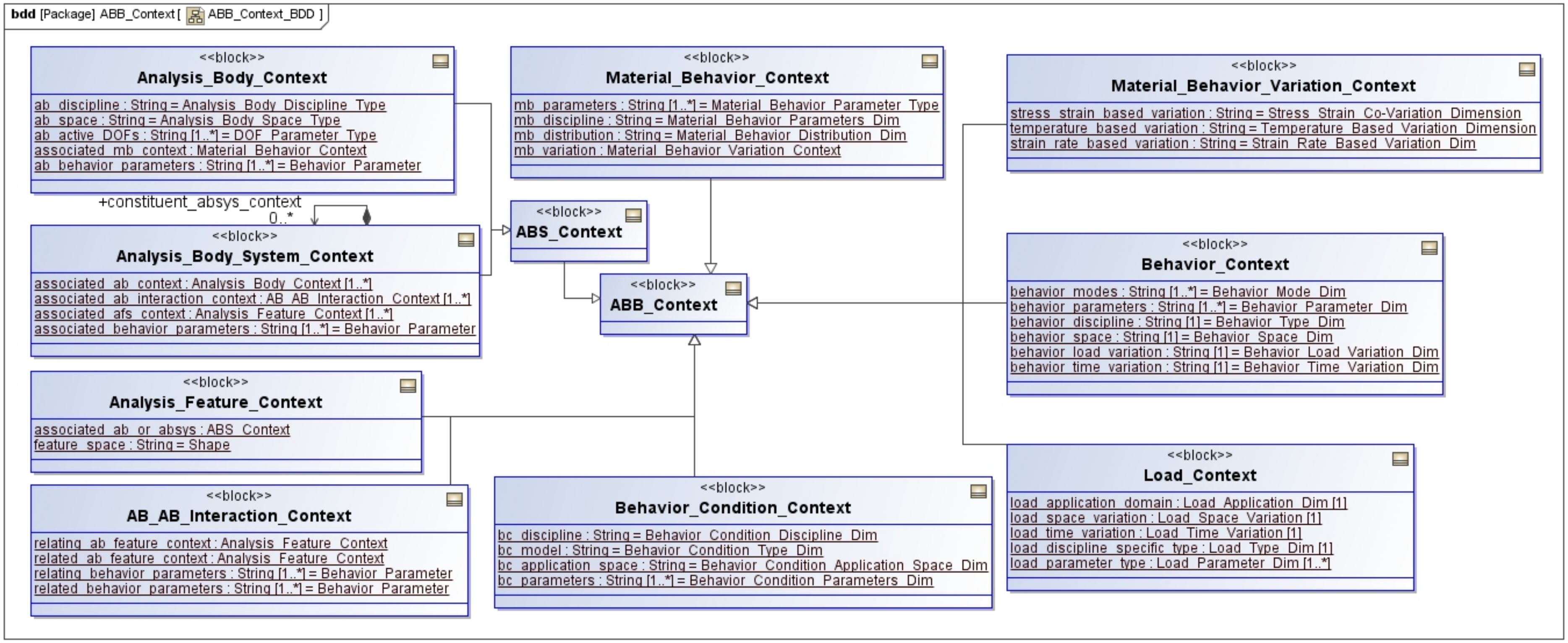

bdd [Package] ABB_Context [ABB_Context_BDD]
<<block>>
Analysis_Body_Context
ab_discipline : String = Analysis_Body_Discipline_Type
ab_space : String = Analysis_Body_Space_Type
ab_active_DOFs : String [1..*] = DOF_Parameter_Type
associated_mb_context : Material_Behavior_Context
ab_behavior_parameters : String [1..*] = Behavior_Parameter
<<block>>
Material_Behavior_Context
mb_parameters : String [1..*] = Material_Behavior_Parameter_Type
mb_discipline : String = Material_Behavior_Parameters_Dim
mb_distribution : String = Material_Behavior_Distribution_Dim
mb_variation : Material_Behavior_Variation_Context
<<block>>
Material_Behavior_Variation_Context
stress_strain_based_variation : String = Stress_Strain_Co-Variation_Dimension
temperature_based_variation : String = Temperature_Based_Variation_Dimension
strain_rate_based_variation : String = Strain_Rate_Based_Variation_Dim
+constituent_absys_context
0..*
<<block>>
ABS_Context
<<block>>
Analysis_Body_System_Context
associated_ab_context : Analysis_Body_Context [1..*]
associated_ab_interaction_context : AB_AB_Interaction_Context [1..*]
associated_afs_context : Analysis_Feature_Context [1..*]
associated_behavior_parameters : String [1..*] = Behavior_Parameter
<<block>>
ABB_Context
<<block>>
Behavior_Context
behavior_modes : String [1..*] = Behavior_Mode_Dim
behavior_parameters : String [1..*] = Behavior_Parameter_Dim
behavior_discipline : String [1] = Behavior_Type_Dim
behavior_space : String [1] = Behavior_Space_Dim
behavior_load_variation : String [1] = Behavior_Load_Variation_Dim
behavior_time_variation : String [1] = Behavior_Time_Variation_Dim
<<block>>
Analysis_Feature_Context
associated_ab_or_absys : ABS_Context
feature_space : String = Shape
<<block>>
AB_AB_Interaction_Context
relating_ab_feature_context : Analysis_Feature_Context
related_ab_feature_context : Analysis_Feature_Context
relating_behavior_parameters : String [1..*] = Behavior_Parameter
related_behavior_parameters : String [1..*] = Behavior_Parameter
<<block>>
Behavior_Condition_Context
bc_discipline : String = Behavior_Condition_Discipline_Dim
bc_model : String = Behavior_Condition_Type_Dim
bc_application_space : String = Behavior_Condition_Application_Space_Dim
bc_parameters : String [1..*] = Behavior_Condition_Parameters_Dim
<<block>>
Load_Context
load_application_domain : Load_Application_Dim [1]
load_space_variation : Load_Space_Variation [1]
load_time_variation : Load_Time_Variation [1]
load_discipline_specific_type : Load_Type_Dim [1]
load_parameter_type : Load_Parameter_Dim [1..*]

Figure 7.6: ABB Context Meta-Model for representing contextual knowledge in ABBs

The central construct in the ABB Context Meta-Model is the ABB_Context block. All other blocks are specializations of the ABB_Context block and are used for representing the context of the corresponding ABBs. For example, the Analysis_Body_Context block is used for representing the context of analysis body ABB and so on. Note that context of an ABB is a static attribute. Thus, the attributes of context blocks (used for populating the context of each ABB) are also static and shown as underlined in Figure 7.6. The constructs in the ABB Context Meta-Model and their properties are described below

- Analysis_Body_Context block is used for representing contextual knowledge for analysis body ABB. The following five reference properties are used for characterizing this contextual knowledge:
  - ab_discipline refers to the analysis discipline (such as structural or thermal) associated which the idealized behaviors represented by an analysis body.
  - ab_space refers to geometric space used for defining the shape of an analysis body.
  - ab_active_DOFs refers to the number and type of degrees-of-freedom used for defining the behavior of an analysis body
  - associated_mb_context refers to the context of the material behavior associated with an analysis body
  - ab_behavior_parameters refers to the behavior parameters that can be computed for an analysis body
- Analysis_Feature_Context block is used for representing contextual knowledge for analysis feature ABB. The following two reference properties are used for characterizing this contextual knowledge:
  - associated_ab refers to the context of the analysis body that owns the analysis feature.
  - feature_space refers to the geometric space of an analysis feature (e.g. 1D feature—point; 2D features—line and plane; and 3D features—surface and volume).
- Material_Behavior_Context block is used for representing contextual knowledge for material behavior ABB. The following four reference properties are used for characterizing this contextual knowledge:

  - mb_parameters refers to parameters used for describing the material behavior (such as Young's Modulus, Poisson's Ratio, etc.)
  - mb_discipline refers to the analysis disciplines for which the material behavior is being described (such as structural discipline and thermal discipline). The type and number of material behavior parameters depend on the discipline.
  - mb_distribution refers to the idealized distribution of material in the analysis body such as isotropic, transversely isotropic, and orthotropic. The material distribution governs the number of material behavior parameters.
  - mb_variation refers to the variation of material behavior parameters, such as linear, bi-linear, and non-linear. Material_Behavior_Variation_Context block is used for characterizing the variation.
- Material_Behavior_Variation_Context block is used for characterizing the types of variation of material behavior. The following three reference properties are used for characterizing this contextual knowledge:
  - stress_strain_based_variation represents variation characterized as the variation of stress-strain response of a material.
  - temperature_based_variation represents variation characterized as the variation of material behavior parameter values with respect to temperature.
  - strain_rate_based_variation represents variation characterized as the variation of stress with respect to strain rate (or deformation rate).
- AB_AB_Interaction_Context block is used for representing contextual knowledge for analysis body interaction ABB. The following three reference properties are used for characterizing this contextual knowledge:
  - relating_ab_feature_context and related_ab_feature_context refer to the context two analysis features participating in an analysis body interaction.
  - relating_behavior_parameters and related_behavior_parameters refer to behavior parameters (at each analysis feature) used for defining the interaction. For example, if two solid bodies are glued together, then the displacement parameters (translation and rotation) at the glued surfaces are used for populating the relating_behavior_parameters and related_behavior_parameters contextual properties.

- Analysis_Body_System_Context block is used for representing contextual knowledge for analysis body system ABB. The following two reference properties are used for characterizing this contextual knowledge:
  - associated_ab_context refers to the context of each analysis body used for creating an analysis body assembly.
  - associated_ab_interaction_context refers to the context of each interaction (between analysis bodies) in an analysis body assembly.
  - constituent_absys_context refers to the context of sub-assemblies in the top level analysis body assembly. An analysis body assembly may be composed of analysis bodies, or analysis body sub-assemblies, or combinations of both.
  - associated_behavior_parameters refers to the behavior parameters used for representing the behavior of the analysis body system.
- Behavior_Condition_Context block is used for representing contextual knowledge for behavior condition ABB. The following four reference properties are used for characterizing this contextual knowledge:
  - bc_discipline refers to the analysis discipline for which the behavior condition is described. For example, structural boundary conditions and thermal boundary conditions are described for structural and thermal analysis disciplines respectively. The value of this property is governed by the discipline associated with the behavior parameters that are used for describing the behavior condition.
  - bc_model refers to the type of behavior condition. Boundary conditions (for boundary value problems) and initial conditions (for initial value problems) are examples of different types of behavior conditions.
  - bc_application_space refers to the geometric space (such as point, line or surface) over which a behavior condition is applied.
  - bc_parameters refers to the behavior parameters used for describing behavior conditions. For example, displacement parameters (u_x, u_y, u_z, θ_x, θ_y, θ_z) are used for describing a displacement boundary condition.
- Load_Context block is used for representing contextual knowledge for load ABB. The following five reference properties are used for characterizing this contextual knowledge:

  - load_application_domain refers to the geometric space over which the load is applied. For example, a concentrated point load is applied at a point, and a distributed load may be applied along a line/curve, or over a surface.
  - load_space_variation refers to the variation of load over the geometric space over which it is applied. For example, a load may be distributed uniformly or non-uniformly over the application domain.
  - load_time_variation refers to the variation of load over the time domain. For example, a point force may be constant or vary with time.
  - load_discipline_specific_type refers to the analysis discipline for which the load is described. The types of loads are different for each analysis discipline. For example, force, moment, and temperature are loads in the structural analysis discipline, and heat flux and heat generation rate are loads in the thermal analysis discipline.
  - load_parameter_type refers to the parameters used for representing loads, such as force parameter, moment parameter, and temperature parameter.
- Behavior_Context block is used for representing contextual knowledge for behavior ABB. The following six reference properties are used for characterizing this contextual knowledge:
  - behavior_modes refers to the different modes of behavior of the analysis body system. For example, in the structural analysis discipline, small deformation and large deformation are examples of different behavior modes. The governing concepts and the analytical formulations for these modes are different. Stress stiffening, fatigue, and fracture modes are specific types of large deformation mode.
  - behavior_parameters refers to the behavior parameters used for quantifying the behavior of an analysis body system. Displacement, stresses, and strains are examples of behavior parameters in the structural discipline while temperature is an example of a behavior parameter in the thermal discipline.
  - behavior_discipline refers to the analysis disciplines such as structural, thermal, electromagnetics
  - behavior_space refers to the behavior space of the analysis body system. Behavior space is characterized by: (a) geometric space defined to describe the form of an analysis body system, such as 1D or 2D, and (b) number of independent behavior

parameters used for characterizing the behavior of an analysis body system. For example, the geometric space used for defining a beam-rod—an analysis body that exhibits axial deformation and bending behavior—with constant cross-section is 1-dimensional, while it has 2 independent behavior parameters (axial deformation and transverse deflection). The purpose of this reference property is to better characterize the meaning of the commonly used terms such as “1D analysis problem” and “2D analysis problem”.

- behavior_load_variation refers to the variation of behavior parameters with respect to the applied loads. For example in the case of an idealized linear spring, the deformation varies linearly with the applied forces. This is an example of linear behavior.
- behavior_time_variation refers to the variation of behavior parameters with respect to time. For transient behavior, behavior parameters vary with temperature while for steady-state or static behavior, behavior parameters are idealized to be constant with respect to time.

7.2.1.2 ABB Property - how is this concept represented?

The knowledge represented by the context attribute of an ABB can enable analysts to search ABBs in a library, and identify semantic conflicts between the different ABBs used for composing behavior models. In contrast, the property attribute of an ABB is used for representing parameters and relations that mathematically define the domain-theoretic concept represented by the ABB. When ABBs are composed to create an ABB system, the property attributes of different ABBs are associated with each other via mathematical relations. As an example, the contextual attribute of the Hook’s Law Material Behavior ABB has keywords that collectively state that the stress-strain co-variation is linear but the property attribute of this ABB represents the behavior parameters (Stress σ, Strain ε, and Young’s Modulus Y), and the linear mathematical equation between them ($\sigma/\varepsilon = Y$).

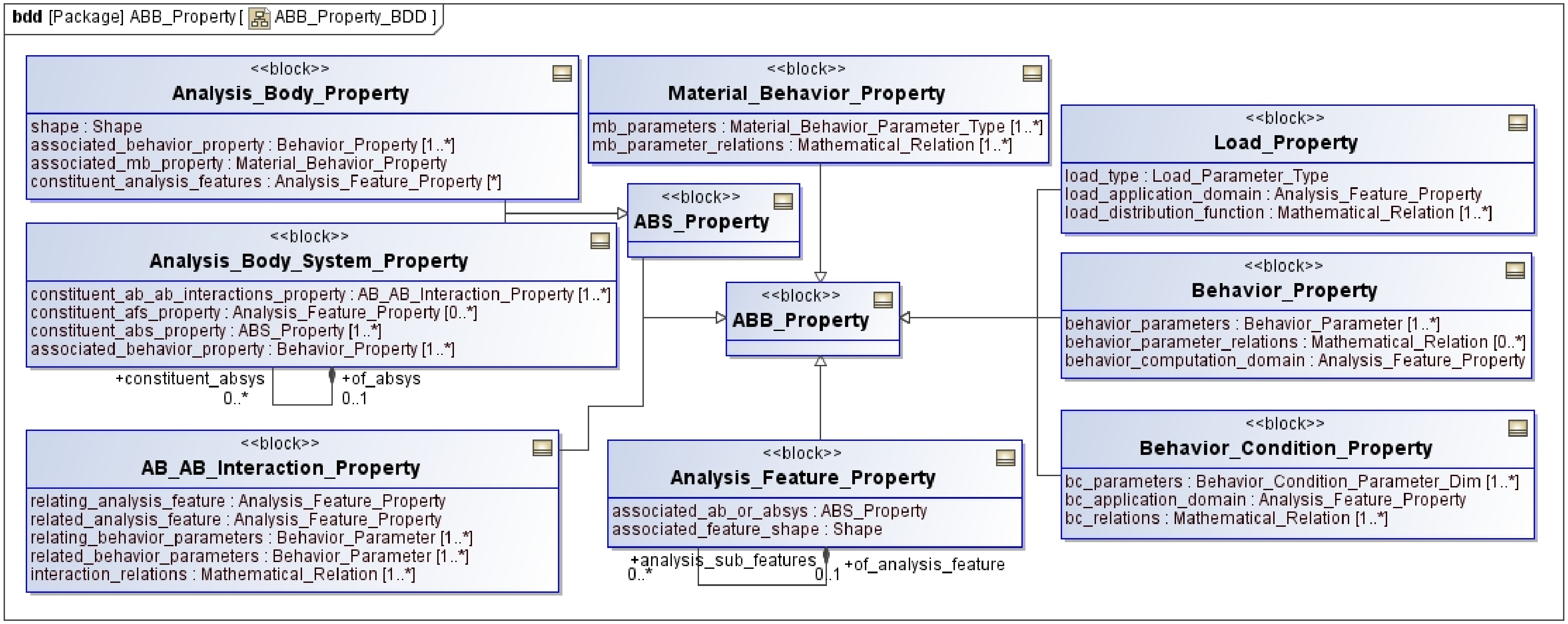

bdd [Package] ABB_Property [ABB_Property_BDD]
<<block>>
Analysis_Body_Property
shape : Shape
associated_behavior_property : Behavior_Property [1..*]
associated_mb_property : Material_Behavior_Property
constituent_analysis_features : Analysis_Feature_Property [*]
<<block>>
Material_Behavior_Property
mb_parameters : Material_Behavior_Parameter_Type [1..*]
mb_parameter_relations : Mathematical_Relation [1..*]
<<block>>
Load_Property
load_type : Load_Parameter_Type
load_application_domain : Analysis_Feature_Property
load_distribution_function : Mathematical_Relation [1..*]
<<block>>
ABS_Property
<<block>>
Analysis_Body_System_Property
constituent_ab_ab_interactions_property : AB_AB_Interaction_Property [1..*]
constituent_afs_property : Analysis_Feature_Property [0..*]
constituent_abs_property : ABS_Property [1..*]
associated_behavior_property : Behavior_Property [1..*]
<<block>>
ABB_Property
<<block>>
Behavior_Property
behavior_parameters : Behavior_Parameter [1..*]
behavior_parameter_relations : Mathematical_Relation [0..*]
behavior_computation_domain : Analysis_Feature_Property
+constituent_absys
0..*
+of_absys
0..1
<<block>>
AB_AB_Interaction_Property
relating_analysis_feature : Analysis_Feature_Property
related_analysis_feature : Analysis_Feature_Property
relating_behavior_parameters : Behavior_Parameter [1..*]
related_behavior_parameters : Behavior_Parameter [1..*]
interaction_relations : Mathematical_Relation [1..*]
<<block>>
Analysis_Feature_Property
associated_ab_or_absys : ABS_Property
associated_feature_shape : Shape
<<block>>
Behavior_Condition_Property
bc_parameters : Behavior_Condition_Parameter_Dim [1..*]
bc_application_domain : Analysis_Feature_Property
bc_relations : Mathematical_Relation [1..*]
+analysis_sub_features
0..*
0..1
+of_analysis_feature

Figure 7.7: ABB Property Meta-Model for representing properties of ABBs

Figure 7.7 illustrates the ABB Property Meta-Model (subset of the ABB Meta-Model) for representing the property attribute of ABBs. The ABB Property Meta-Model defines the constructs for representing the property attribute of different types of ABBs. It is founded on the analysis knowledge dimensions (section 7.5) and is extensible to defining the properties of other types of ABBs.

The central construct in the ABB Property Meta-Model is the ABB_Property block. All other blocks are specializations of the ABB_Property block and are used for representing the properties of corresponding ABBs. For example, the Analysis_Body_Property block is used for representing the property of analysis body ABB and so on. Note that the values populating the property attribute of an ABB do not explicitly convey the semantics of the physical concept being represented by the ABB. For example, the property attribute of Hooke's Law Material Behavior ABB represents the stress, strain, and Young's Modulus parameters and the linear equation relating the three (stress = strain * Young's Modulus) but it does not explicitly describe the nature of the equation (linear) or the material distribution assumed (isotropic versus orthotropic).

The constructs in the ABB Property Meta-Model are described below.

- Analysis_Body_Property block is used for representing the property attributes of analysis body ABB. It has the following four reference properties:
  - shape refers to the geometric shape of the analysis body. The reference property type is the Shape block that is reused across all meta-models in the KCM—CPM2_xKCM, CBM, and ABB Meta-Model. KCM leverages STEP Part 42 (ISO 10303-42 2000) standard for representing geometric shapes. Thus, the Shape block is an abstraction for geometry representation entities in Part 42.
  - associated_behavior_property refers to the behavior parameters that may be computed for the analysis body, and the relations among these behavior parameters in the context of the analysis body. The reference property type is the Behavior_Property block. For example, for a linear mechanical spring (an analysis body ABB), the only behavior parameter that may be computed is the deformation of the spring along its axis (Ux).

- associated_mb_property refers to the material behavior ABB that represents the constitutive material behavior of the analysis body. The reference property type is the Material_Behavior_Property block.
- constituent_analysis_features_property refers to the analysis features defined on the analysis body. The reference property type is the Analysis_Featuare_Property block. Analysis features are defined to identify geometric regions defined on an analysis body (or assembly) where behavior parameters are to be computed, interactions need to be defined among analysis bodies, and/or load and behavior conditions need to be applied.

- Analysis_Feature_Property block is used for representing property attributes of analysis feature ABB. It has the following two reference properties:
  - associated_ab_or_absys refers to the analysis body or analysis body assembly on which the analysis feature is defined. The reference property type is the Analysis_Body_Property block.
  - associated_feature_shape refers to the shape of the analysis feature defined on the analysis body or analysis body assembly. The reference property type is the Shape block.
  - analysis_sub_features refers to analysis features that are sub-features of the given analysis feature. This reference property represents the composition of analysis features from analysis features. For example, if a surface is identified as an analysis feature and a point on the surface is identified as another analysis feature, then the two analysis features are related by this reference property.
- AB_AB_Interaction_Property block is used for representing property attributes of analysis body interaction ABB (AB_AB_Interaction_ABB block). The analysis body interaction ABB represents the interaction behavior among two analysis bodies in an analysis body assembly. The interaction behavior is defined between analysis features of the analysis bodies participating in the interaction. The AB_AB_Interaction_Property block has the following two reference properties:
  - relating_analysis_feature and related_analysis_feature refer to the two analysis features (each defined on an analysis body) participating in the interaction.

- relating_behavior_parameters and related_behavior_parameters refer to two sets of behavior parameters that are used for defining the interaction behavior.
- interaction_relations refer to mathematical relations defined using the relating and related behavior parameters.

For example, the No-slip interaction ABB (type of analysis body interaction ABB) can be used to create a tie constraint between two analysis features—at which the corresponding analysis bodies contact each other—such that there is no relative displacement between the analysis features. In the No-slip interaction ABB, the relating and related analysis features would refer to the two analysis features participating in the contact respectively; the relating and related behavior parameters refer to the displacement parameters (u_x, $u_{y,}$ $u_{z,}$ $\theta_{x,}$ $\theta_{y,}$ θ_z) defined at each analysis feature; and the interaction relations would refer to the mathematical equations that bind the displacement parameters at the analysis features ($u_x^1=u_x^2$, $u_y^1=u_y^2$, …:where u_x^1 and u_x^2 are the displacement parameters at analysis features 1 and 2 respectively, and so on).

- Analysis_Body_System_Property block is used for representing property attributes of analysis body system ABB. It has the following four reference properties.
  - constituent_ab_ab_interactions_property refer to the interactions defined between analysis bodies in the context of the analysis body system. The reference property type is AB_AB_Interaction_Property block.
  - constituent_af_property refer to the analysis features defined on the analysis body system. The reference property type is Analysis_Feature_Property block.
  - constituent_abs_property refer to the analysis body components of the analysis body system. The reference property type is Analysis_Body_Property block.
  - constituent_absys and of_absys refer to the children sub-systems and parent sub-system of an analysis body system respectively. The reference property type is Analysis_Body_System_Property block.
  - asociated_behavior_property is used representing the behavior parameters computed for the analysis body system and the analysis features at which they are computed.
- ABS_Property block is used for representing property attributes of an analysis body ABB or analysis body system ABB. It is the generalization of Analysis_Body_Property block and Analysis_Body_System property block.

- Material_Behavior_Property block is used for representing property attributes of material behavior ABB. It has the following two reference properties.
  - mb_parameters refers to the material behavior parameters used for defining the material behavior.
  - mb_parameter_relations refer to the mathematical relations established among material behavior parameters to define the material behavior. These mathematical relations may have an analytical form (such as equations) or a tabulated form (such as material property-value tables generated in physical experiments). In general, KCM has pre-defined specializations of the Mathematical_Relation block for representing analytical, logical, tabulated relations, and is extensible to developing other specializations.
- Load_Property block is used for representing property attributes of the load ABB. It has the following three reference properties.
  - load_type refers to the type of load and the load parameter used for defining the load. For example, force is a structural load defined using the force parameter (denoted as F) and heat generation rate is a thermal load defined using the heat generation rate parameter (denoted as q_{gen}).
  - load_application_domain refers to the analysis features of an analysis body or analysis body system to which the load is applied. Depending on the load type, loads may be applied to a point, surface, or volume features.
  - load_distribution_function refers to the mathematical relations that describe the variation of the load over the application domain. For example, a constant force load would have a distribution function as F=constant while a force load that varies linearly over a straight edge analysis feature would have the following distribution function: $F_x=(x/L)*F_L$ where: F_x is the force magnitude at a distance x from the origin of the edge feature, L is the length of the edge feature, and F_L is the force magnitude at the end of the edge feature.
- Behavior_Condition_Property block is used for representing property attributes of the behavior condition ABB. It has the following three reference properties.
  - bc_parameters refers to parameters used for defining behavior conditions.

  - bc_application_domain refers to the analysis features of an analysis body or analysis body system on which the behavior conditions are defined.
  - bc_relations refers to mathematical relations—established among behavior condition parameters—that are used for defining behavior conditions.

  For example, if a boundary condition that constrains all degrees of freedom at a point on an analysis body is to be defined, the behavior condition parameters are the displacement parameters (u_x, $u_{y,}$ $u_{z,}$ $\theta_{x,}$ $\theta_{y,}$ θ_z); the application domain is the point analysis feature; and the behavior condition relations are the mathematical equations that bind displacement parameters at the point analysis feature to 0, such as $u_x=0$, $u_y=0$ and so on.

- Behavior_Property block is used for representing property attributes of the behavior ABB. Note that the behavior ABB is defined as an ABB to characterize and reuse the definition of different types of idealized behaviors. It has the following two reference properties.
  - behavior_parameters refers to the set of behavior parameters (such as displacement, temperature, stress, and strain) that are used for characterizing the behavior.
  - behavior_computation_domain refers to the analysis features where the subject behavior parameters are being computed.
  - behavior_parameter_relations refers to a set of mathematical relations defined using behavior parameters. Together, the behavior parameters and the mathematical relations are used for characterizing a behavior.

While the context and property attributes of an ABB define the concept represented by the ABB, the application condition and the transformation rules attributes define how an ABB may be used in composing an ABB system and hence a behavior model.

7.2.1.3 ABB Application Conditions – what are the conditions for using this concept?

The *application condition* attribute of an ABB defines the pre-conditions for using / applying the concept embodied in an ABB when composing ABB system model. ABB application conditions are represented using mathematical relation such as analytical, logical, or tabular that must be satisfied for an ABB to be used. For example, when an analyzable artifact is idealized as a shell, it is assumed that the in-plane deformation (stretching) and bending effects dominate the deformation of the shell and the out-of-plane tensile or compressive deformations are negligible. As the thickness of the shell decreases, the stretching behavior dominates and all other deformations behaviors are neglected. Thus, when an analyzable artifact is idealized as a shell, it is assumed that the ratio of the thickness of the shell to the radius of curvature is significantly less than unity. The application condition attribute of the shell analysis body ABB is used to represent the mathematical relation (`h/R <<1`, where `h` is the thickness of the shell and `R` is the radius of curvature). Hence, the application condition attribute of an ABB represents an aspect of the domain theoretic concept represented by the ABB.

7.2.1.4 ABB Transformation Rules – how does one use this concept?

ABB *transformation rules* attribute of an ABB represents the model transformations that are executed when the ABB is composed in an ABB system model and when the ABB system model is composed in a behavior model. The following two types of transformation rules are defined for an ABB: (i) transformation rules that establish composition relationship between an ABB and the ABB system where is it to be used, and (ii) transformation rules that establish idealization relationships between an ABB and the corresponding analyzable artifact. While the former is defined for all ABBs, the latter is defined only for those ABBs that are idealizations of some specific aspect of the analyzable artifact model. These are analysis body ABB, analysis body system ABB,

In the KCM, the graph transformations and patterns are used for mathematically defined these transformation rules. The transformation rules for each ABB type are defined as part of the behavior model formulation method presented in Chapter 8.

7.2.2 Analysis building block (ABB) system model

An analysis body system model (referred to as ABB system for brevity) is a model composed of ABB models. If ABBs are choices available to analysts for a certain type of decision, then an ABB system is a grouping of selected choices for a certain set of decisions. Similar to an ABB, an ABB system can be reused to create other ABB systems. Figure 7.8 illustrates the ABB System Meta-Model (a sub-set of the ABB Meta-

Model). An ABB system is represented by the ABBSys block and has the following four part properties:

- abs_sys refers to an analysis body or analysis body system ABBs in the ABB system.
- load_applications refers to load ABBs in the ABB system.
- behavior_condition_applications refers to the behavior condition ABBs in the ABB system
- behaviors refers to behavior ABBs in the ABB system

While there are a significantly large number of ABB systems that may be composed from nine different types of ABBs defined in the previous section, two key types of ABB systems defined using the ABBSys block in the KCM are as follows:

- Behavior Model ABB Systems - These types of ABB systems are represented by the Behavior_Model_ABBSys block—described in details in section 7.1.
- ABB systems that are logical groupings of ABBs and are used relatively frequently when creating behavior models. For example, an ABB system composed of a solid analysis body ABB with linear elastic isotropic material behavior ABB.

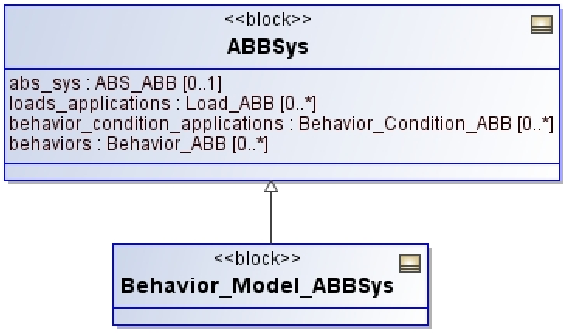

<<block>>
ABBSys
abs_sys : ABS_ABB [0..1]
loads_applications : Load_ABB [0..*]
behavior_condition_applications : Behavior_Condition_ABB [0..*]
behaviors : Behavior_ABB [0..*]
<<block>>
Behavior_Model_ABBSys

Figure 7.8: ABB System Meta-Model

The ABB System Meta-Model illustrated in Figure 7.8 is also designed to allow composition of multiple ABB systems to create a higher-level ABB system. This allows for greater reuse of ABB systems across different behavior models. For example, if an electronics designer/analyst was interested to compute the warpage behavior of printed circuit assemblies and printed circuit boards, they could create a warpage behavior model for printed circuit boards and reuse that behavior model to create a warpage behavior model for printed circuit assemblies. Note that the ABBSys block does not have an explicit composition relationship to itself for realizing this use case. Instead, this composition is realized by defining a new analysis body system that is composed of the analysis body systems from the ABB systems that were to be composed. Similarly, the load application, behavior conditions applications, and behavior attributes from the ABB systems are ‘merged’ to define the higher-level ABB system.

The composition of an ABB system from ABBs along with the composition rules defined in each type of ABB (as transformation rule attribute) are described in details in Chapter 8 as part of the Behavior Model Formulation Method of the Knowledge Composition Methodology.

7.3 ABB Model Library

In this section, ABB models of each type are presented. Figure 7.9 illustrates the three levels of ABB model abstractions. The ABB Meta-Model presented in section 7.2 defines the constructs for defining eight different types of ABBs. For each ABB type, multiple ABB models may be defined as specializations of the corresponding type in the ABB Meta-Model. For example, the analysis body ABB defined in the ABB Meta-Model may be specialized to define Rod analysis body ABB (or Rod ABB for brevity), Beam ABB, Shell ABB, and so on. ABB models are used for composing behavior meta-models and behavior model structures—specifically the VTMB Artifact Behavior Meta-Model (Level 3) and FTMB Artifact Behavior Model Structure (Level 4) as described later in section 7.4. ABB models are instantiated to define behavior model instances—specifically, the FTMB Artifact Behavior Model instance as described in section 7.4.

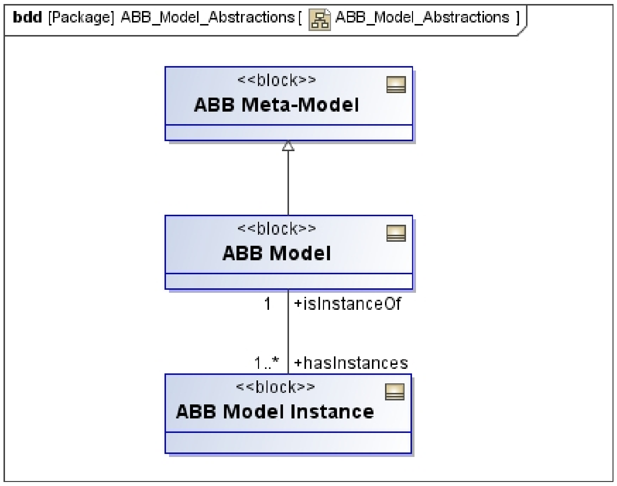


Figure 7.9: ABB model abstractions in KCM

As an example, the block Analysis_Body_ABB defined in the ABB Meta-Model represents analysis body ABBs. This block is specialized to define different types of analysis body ABBs (such as Rod ABB, Beam ABB, and Shell ABB). Each of these ABBs may be then instantiated such that its property attributes are populated—the Rod ABB may be instantiated with specific values of the rod's length, cross-sectional shape, and its material behavior properties.

Examples of each of the eight ABB types are described below. For each ABB type, one example is presented in details to describe how the ABB meta-model may be specialized. Note that only the context and property attributes of ABBs are presented here. The application conditions and transformation attributes are described in Chapter 8 as part of the behavior model formulation methods.

7.3.1 Analysis Body ABBs

An analysis body is an idealization of an artifact such that it exhibits an idealized sub-set of behaviors of the artifact. An analysis body ABB represents the form and idealized behavior of a family of analysis bodies. In the ABB Meta-Model (section 7.2), the Analysis_Body_ABB block is used for representing analysis body ABBs. This block may be specialized to represent several types of analysis body ABB models as shown in Figure 7.10. The blocks representing different analysis bodies are stated below:

- Structural_Body – represents analysis bodies with idealized structural behavior

- Thermal_Body – represents analysis bodies with idealized thermal behavior
- Electric_Body – represents analysis bodies with idealized electric behavior
- Magnetic_Body - represents analysis bodies with idealized magnetic behavior
- Fluid_Body - represents analysis bodies with idealized fluid flow behavior

The structural body ABB (represented by Structural_Body block) may be further specialized into different types of structural analysis bodies such as Rod, Shaft, Beam, Column, Plate, Shell, and Membrane as shown in Figure 7.10

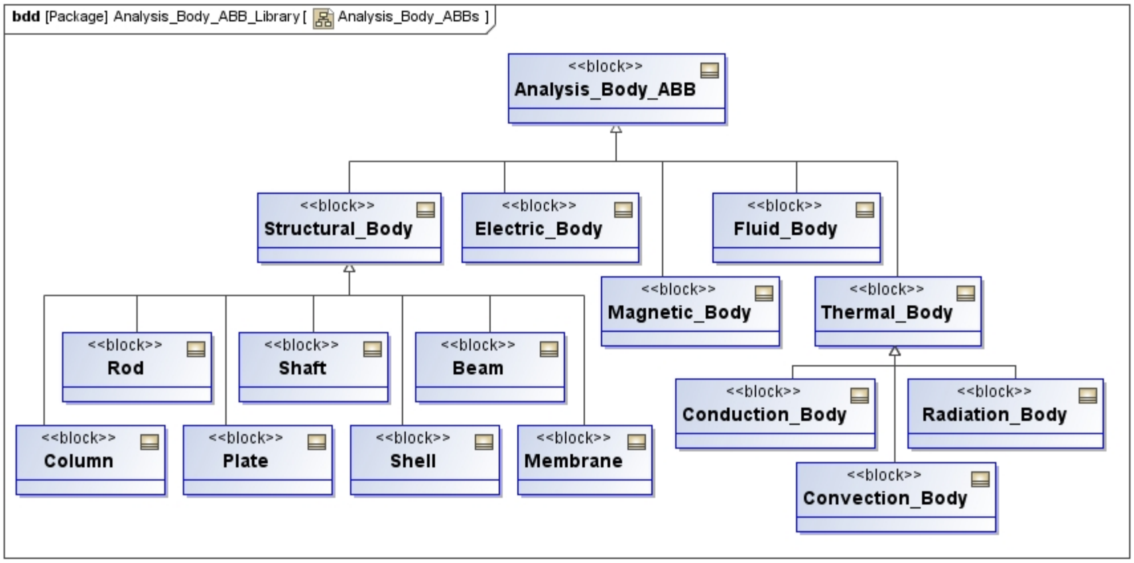


Figure 7.10: Analysis body ABBs

In addition, analysis bodies may be defined such that inherit the characteristics of one more analysis bodies within the same discipline or across disciplines. For example, beam-rod is a special type of analysis body that exhibits both transverse and axial deformation behavior.

Note that ABBs are characterized using their context attribute, and the context of each type of ABB is defined in the ABB Meta-Model. For example, the block Analysis_Body_Context represents the context of analysis body ABBs in general. The properties of this block characterize the context of analysis body ABBs. Specifically, the context of analysis body ABBs can be represented in terms of the analysis discipline, geometric space, active degrees-of-freedom, and material behavior. Any of these characteristics may be used for organizing different types of analysis body ABBs in a hierarchy. In Figure 7.10, analysis body ABBs are organized based on the analysis discipline.

For each of the analysis body ABBs described above, the context and the property attribute types may be defined. For example, for the shell analysis body ABB, Shell_Context and Shell_Property blocks represent the context and property type respectively, defined as specializations of Analysis_Body_Context and Analysis_Body_Property block respectively. The shell analysis body ABB defined here is based on Kirchhoff-Love assumptions for thin elastic shells (Ventsel and Krauthammer 2001). Thus, the attribute values of the Shell_Context and Shell_Property block properties indicate the shell analysis body ABB has an elastic material behavior and the stress and strain behavior parameters normal to the mid-surface are neglected.

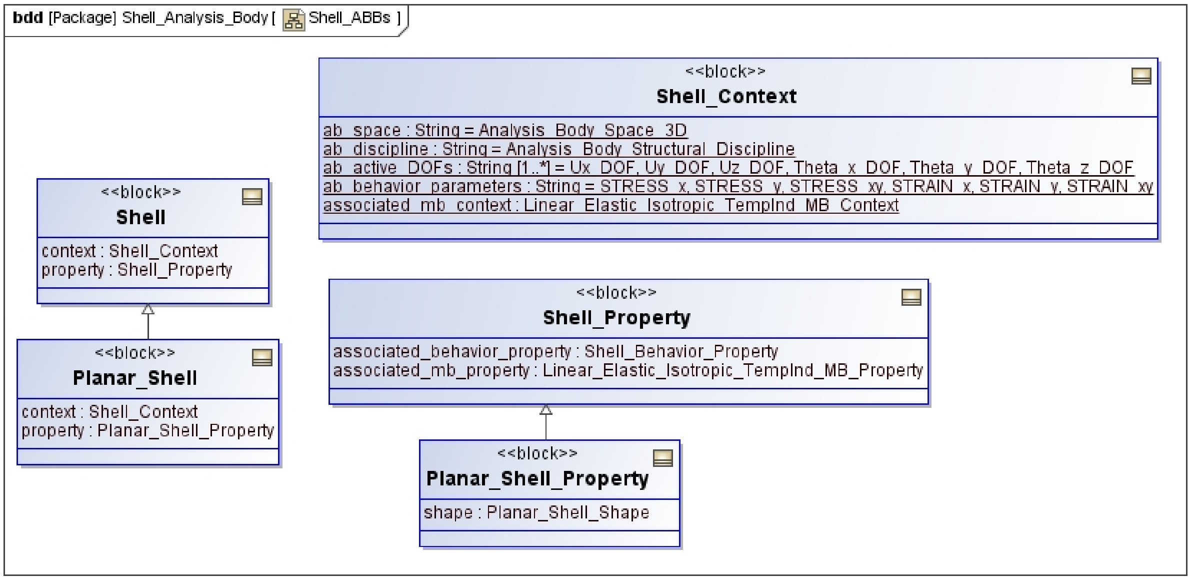

bdd [Package] Shell_Analysis_Body [Shell_ABBs]
<<block>>
Shell_Context
ab_space : String = Analysis_Body_Space_3D
ab_discipline : String = Analysis_Body_Structural_Discipline
ab_active_DOFs : String [1..*] = Ux_DOF, Uy_DOF, Uz_DOF, Theta_x_DOF, Theta_y_DOF, Theta_z_DOF
ab_behavior_parameters : String = STRESS_x, STRESS_y, STRESS_xy, STRAIN_x, STRAIN_y, STRAIN_xy
associated_mb_context : Linear_Elastic_Isotropic_TempInd_MB_Context
<<block>>
Shell
context : Shell_Context
property : Shell_Property
<<block>>
Planar_Shell
context : Shell_Context
property : Planar_Shell_Property
<<block>>
Shell_Property
associated_behavior_property : Shell_Behavior_Property
associated_mb_property : Linear_Elastic_Isotropic_TempInd_MB_Property
<<block>>
Planar_Shell_Property
shape : Planar_Shell_Shape

Figure 7.11: Shell analysis body ABB

Each of the analysis body ABBs described above may be further specialized. For example, the shell analysis body ABB is specialized to represent planar shell analysis body ABB. A planar shell is a shell whose mid-surface has a planar shape as shown in Figure 7.11. Only the context and property attributes of the ABBs are shown here.

Note that the attributes of Analysis_Body_Context and its specializations are static—values do not change when an analysis body ABB is instantiated. However, the attributes of Analysis_Body_Property block are populated with specific values when an analysis body ABB is instantiated. The context attribute of the planar shell analysis body

ABB is of type Planar_Shell_Context block. The attribute values of this block characterize a planar shell. In a similar manner, the Analysis_Body_Property block is specialized to represent shell property and planar shell property in particular. The shape attribute of the Planar_Shell_Property block is of type Planar_Shell_Shape. Also note that during specialization the parent attributes are re-defined to better characterize the specializations.

7.3.2 Material Behavior ABBs

Material behavior ABBs are used for representing the constitutive material behavior of analysis body ABBs. The Material_Behavior_ABB block is the parent block for represent material behavior ABBs. The context attribute of material behavior ABBs defines the key dimensions for characterizing material behavior ABBs, namely analysis discipline, behavior parameters, distribution, and variation (described in details in section 7.2.1. Any of these dimensions may be used for creating a hierarchy of material behavior ABBs. Figure 7.12 below illustrates blocks corresponding to different types of material behavior ABBs organized based on analysis discipline, and defined as specializations of the Material_Behavior_ABB block.

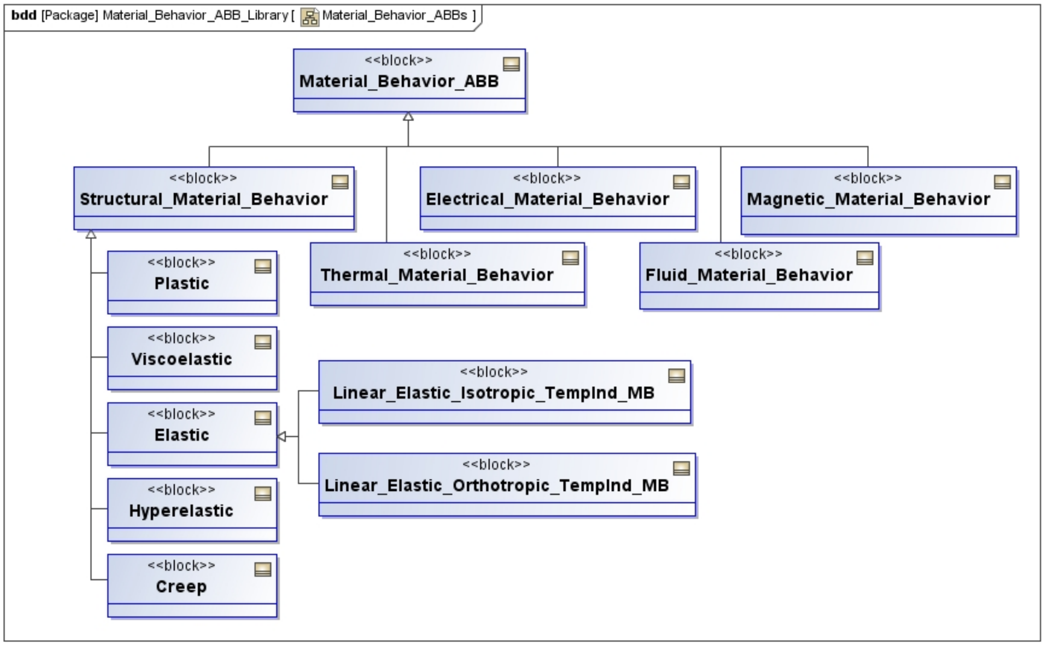

bdd [Package] Material_Behavior_ABB_Library [Material_Behavior_ABBs]
<<block>>
Material_Behavior_ABB
<<block>>
Structural_Material_Behavior
<<block>>
Electrical_Material_Behavior
<<block>>
Magnetic_Material_Behavior
<<block>>
Thermal_Material_Behavior
<<block>>
Fluid_Material_Behavior
<<block>>
Plastic
<<block>>
Viscoelastic
<<block>>
Elastic
<<block>>
Hyperelastic
<<block>>
Creep
<<block>>
Linear_Elastic_Isotropic_TempInd_MB
<<block>>
Linear_Elastic_Orthotropic_TempInd_MB

Figure 7.12: Material behavior ABBs

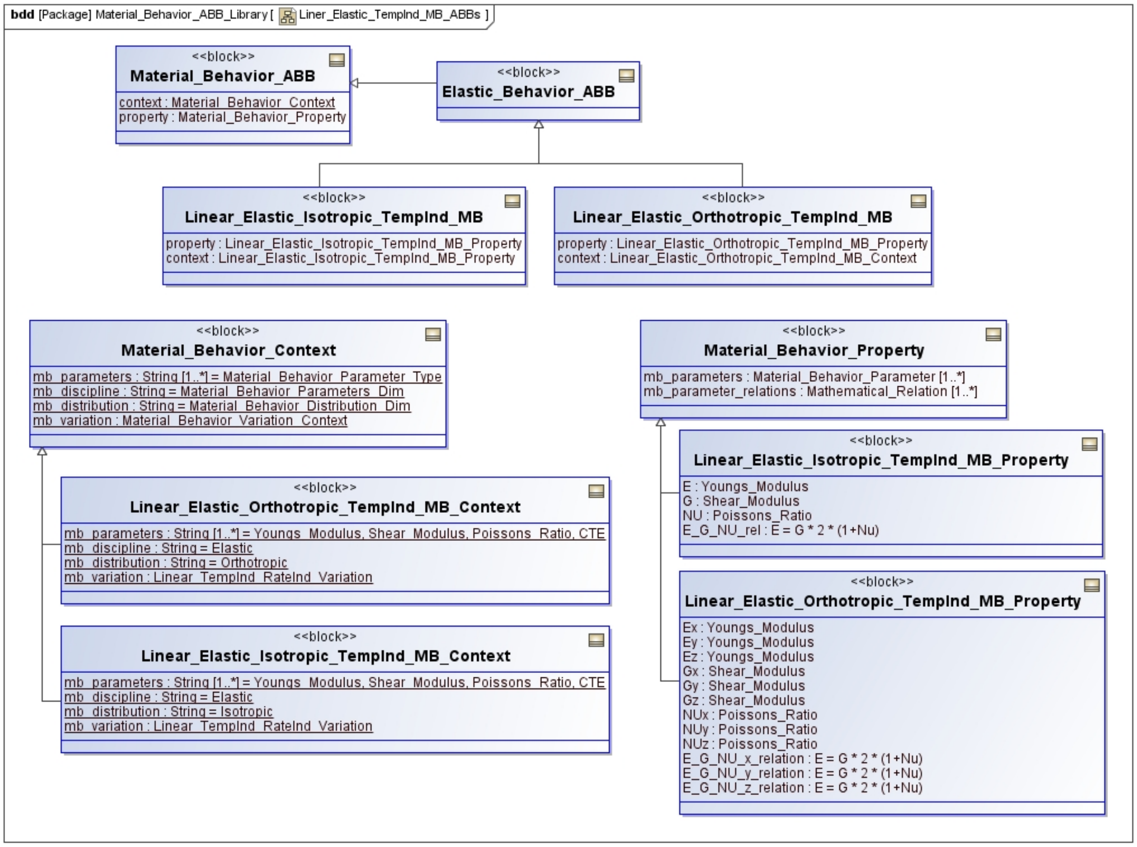

bdd [Package] Material_Behavior_ABB_Library [Liner_Elastic_TempInd_MB_ABBs]
<<block>>
Material_Behavior_ABB
context : Material_Behavior_Context
property : Material_Behavior_Property
<<block>>
Elastic_Behavior_ABB
<<block>>
Linear_Elastic_Isotropic_TempInd_MB
property : Linear_Elastic_Isotropic_TempInd_MB_Property
context : Linear_Elastic_Isotropic_TempInd_MB_Property
<<block>>
Linear_Elastic_Orthotropic_TempInd_MB
property : Linear_Elastic_Orthotropic_TempInd_MB_Property
context : Linear_Elastic_Orthotropic_TempInd_MB_Context
<<block>>
Material_Behavior_Context
mb_parameters : String [1..*] = Material_Behavior_Parameter_Type
mb_discipline : String = Material_Behavior_Parameters_Dim
mb_distribution : String = Material_Behavior_Distribution_Dim
mb_variation : Material_Behavior_Variation_Context
<<block>>
Material_Behavior_Property
mb_parameters : Material_Behavior_Parameter [1..*]
mb_parameter_relations : Mathematical_Relation [1..*]
<<block>>
Linear_Elastic_Isotropic_TempInd_MB_Property
E : Youngs_Modulus
G : Shear_Modulus
NU : Poissons_Ratio
E_G_NU_rel : E = G * 2 * (1+Nu)
<<block>>
Linear_Elastic_Orthotropic_TempInd_MB_Context
mb_parameters : String [1..*] = Youngs_Modulus, Shear_Modulus, Poissons_Ratio, CTE
mb_discipline : String = Elastic
mb_distribution : String = Orthotropic
mb_variation : Linear_TempInd_RateInd_Variation
<<block>>
Linear_Elastic_Orthotropic_TempInd_MB_Property
Ex : Youngs_Modulus
Ey : Youngs_Modulus
Ez : Youngs_Modulus
Gx : Shear_Modulus
Gy : Shear_Modulus
Gz : Shear_Modulus
NUx : Poissons_Ratio
NUy : Poissons_Ratio
NUz : Poissons_Ratio
E_G_NU_x_relation : E = G * 2 * (1+Nu)
E_G_NU_y_relation : E = G * 2 * (1+Nu)
E_G_NU_z_relation : E = G * 2 * (1+Nu)
<<block>>
Linear_Elastic_Isotropic_TempInd_MB_Context
mb_parameters : String [1..*] = Youngs_Modulus, Shear_Modulus, Poissons_Ratio, CTE
mb_discipline : String = Elastic
mb_distribution : String = Isotropic
mb_variation : Linear_TempInd_RateInd_Variation

Figure 7.13: Linear elastic isotropic and orthotropic material behavior ABBs

Two specializations of the elastic material behavior ABB —linear elastic isotropic temperature-independent material behavior ABB and linear elastic orthotropic temperature independent material behavior ABB—are illustrated in details in Figure 7.13. The context and property attribute types of these ABBs are also shown in the figure—Linear_Elastic_Isotropic_TempInd_MB_Context and Linear_Elastic_Isotropic_TempInd_MB_Property blocks for the linear elastic isotropic temperature-independent material behavior ABB, and Linear_Elastic_Orthotropic_TempInd_MB_Context and Linear_Elastic_Orthtropic_TempInd_MB_Property blocks for the linear elastic orthotropic temperature-independent material behavior ABB. The attribute values of the context blocks are populated with keywords that indicate the isotropic versus orthotropic material distribution as the only difference between the two material behavior ABBs. The property blocks for the two ABBs are defined as specializations of the Material_Behavior_Property block, and the Material_Behavior_Property .mb_parameters and

Material_Behavior_Property.mb_relations attributes are specialized to 3 parameters and 1 constraint relation for the linear elastic isotropic temperature-independent material behavior ABB, and 9 parameters and 3 constraint relations for the linear elastic orthotropic temperature-independent material behavior ABB. For the linear elastic isotropic temperature-independent material behavior ABB, the three parameters are the Young's Modulus, Poisson's Ratio, and Shear Modulus; and the constraint relation relates the three parameters as shown in the parametric diagram in Figure 7.14. For the linear elastic orthotropic temperature-independent material behavior ABB, the nine parameters are the Young's Modulus, Poisson's Ratio, and Shear Modulus in three principal directions; and the constraint relation relates the three parameters in each principal direction as shown in the parametric diagram in Figure 7.15.

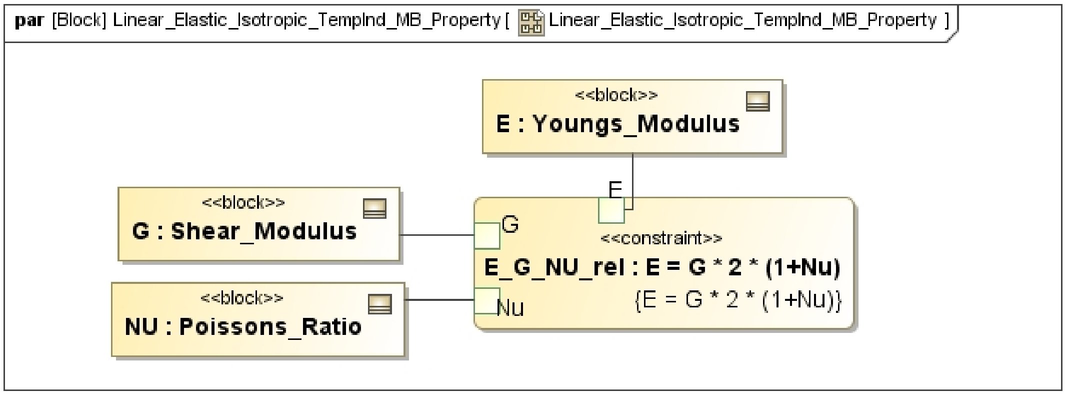

par [Block] Linear_Elastic_Isotropic_TempInd_MB_Property [Linear_Elastic_Isotropic_TempInd_MB_Property]
<<block>>
E : Youngs_Modulus
<<block>>
G : Shear_Modulus
<<block>>
NU : Poissons_Ratio
E
G
Nu
<<constraint>>
E_G_NU_rel : E = G * 2 * (1+Nu)
{E = G * 2 * (1+Nu)}

Figure 7.14: Constraint relations between E, G, and Nu parameters for linear elastic isotropic temperature-independent material behavior ABB

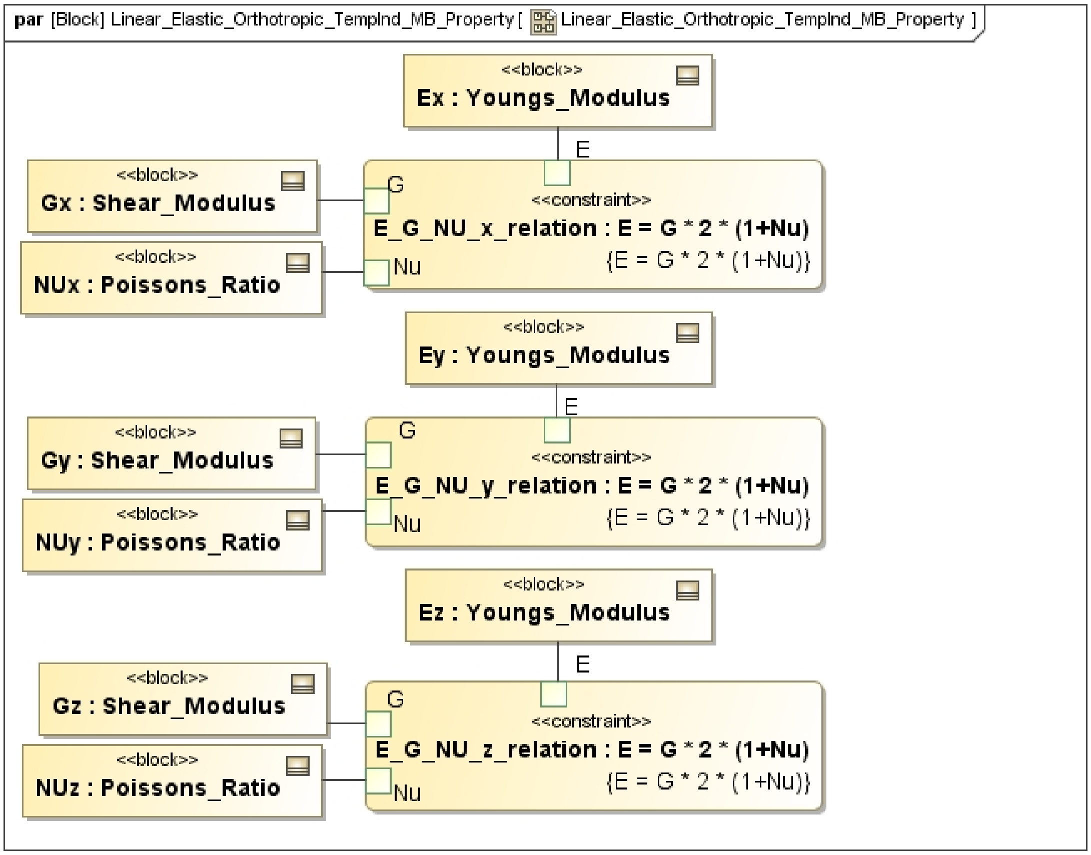


Figure 7.15: Constraint relations between E, G, and Nu parameters in each principal direction for linear elastic orthotropic temperature-independent material behavior ABB

In a similar manner, the property types of other material behavior ABBs may be defined with their corresponding parameters and relations. The constraint relations between material behavior parameters may be available as a tabulated data between the parameters.

7.3.3 Behavior ABBs

A Behavior ABB represents a set of idealized behaviors. When an analysis body ABB is associated with a behavior ABB, it implies that the subject analysis body exhibits the specific set of behaviors. In the KCM, behavior is characterized by the context and property attribute of the behavior ABB, represented by the Behavior_Context and Behavior_Property blocks respectively. The Behavior_Context block properties provide several dimensions for characterizing behavior, as described in section 7.2. Figure 7.16 shows a set of behavior ABBs organized in a hierarchy based on the behavior discipline dimension. The structural behavior ABBs represent different types of primitive structural behaviors, such as tension, compression, bending, torsion. A composite behavior may be defined by multiply inheriting two or more types of behavior ABBs.

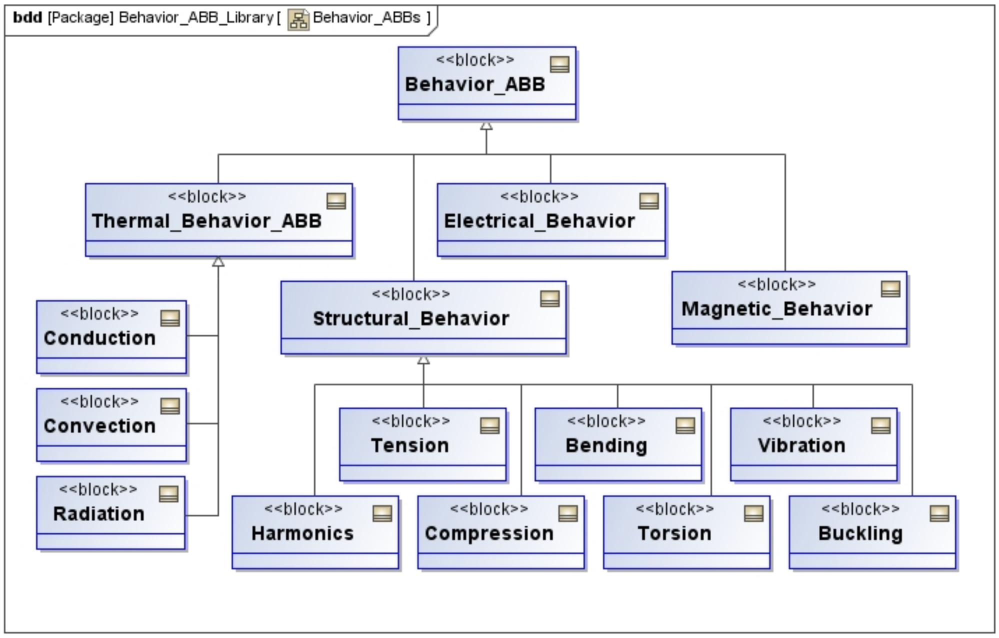


Figure 7.16: Behavior ABBs

Figure 7.17 shows Structural_Behavior ABB block—representing structural behavior ABB—defined as a specialization of the Behavior_ABB block. The property values of Structural_Behavior_Context block characterize the structural behavior ABB, and the Structural_Behavior_Property.behavior_parameters is specialized to represent structural behavior parameters only.

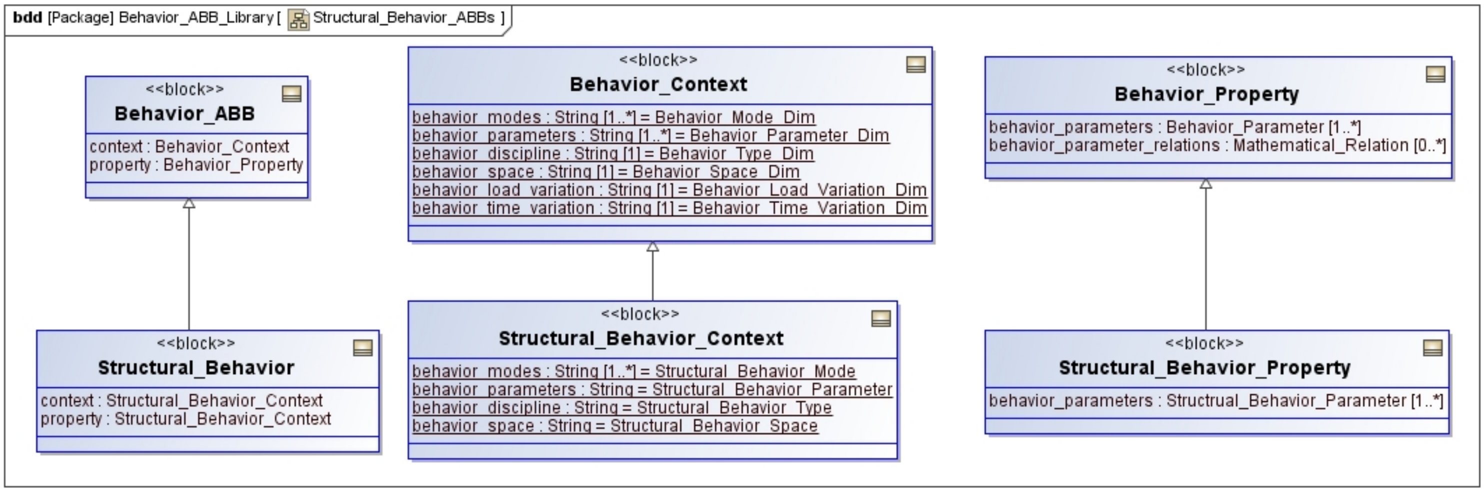

bdd [Package] Behavior_ABB_Library [Structural_Behavior_ABBs]
<<block>> Behavior_ABB
context : Behavior_Context
property : Behavior_Property
<<block>> Behavior_Context
behavior_modes : String [1..*] = Behavior_Mode_Dim
behavior_parameters : String [1..*] = Behavior_Parameter_Dim
behavior_discipline : String [1] = Behavior_Type_Dim
behavior_space : String [1] = Behavior_Space_Dim
behavior_load_variation : String [1] = Behavior_Load_Variation_Dim
behavior_time_variation : String [1] = Behavior_Time_Variation_Dim
<<block>> Behavior_Property
behavior_parameters : Behavior_Parameter [1..*]
behavior_parameter_relations : Mathematical_Relation [0..*]
<<block>> Structural_Behavior
context : Structural_Behavior_Context
property : Structural_Behavior_Context
<<block>> Structural_Behavior_Context
behavior_modes : String [1..*] = Structural_Behavior_Mode
behavior_parameters : String = Structural_Behavior_Parameter
behavior_discipline : String = Structural_Behavior_Type
behavior_space : String = Structural_Behavior_Space
<<block>> Structural_Behavior_Property
behavior_parameters : Structrual_Behavior_Parameter [1..*]

Figure 7.17: Structural behavior ABB

7.3.4 Analysis Feature ABBs

Analysis features ABBs are associated with analysis body (or analysis body system) ABBs since an analysis feature is specific aspect of the shape of analysis body or analysis body systems. Figure 7.18 illustrates different types of analysis features defined for a shell. The blocks representing these analysis features are defined as specializations of the Analysis_Feature_ABB block.

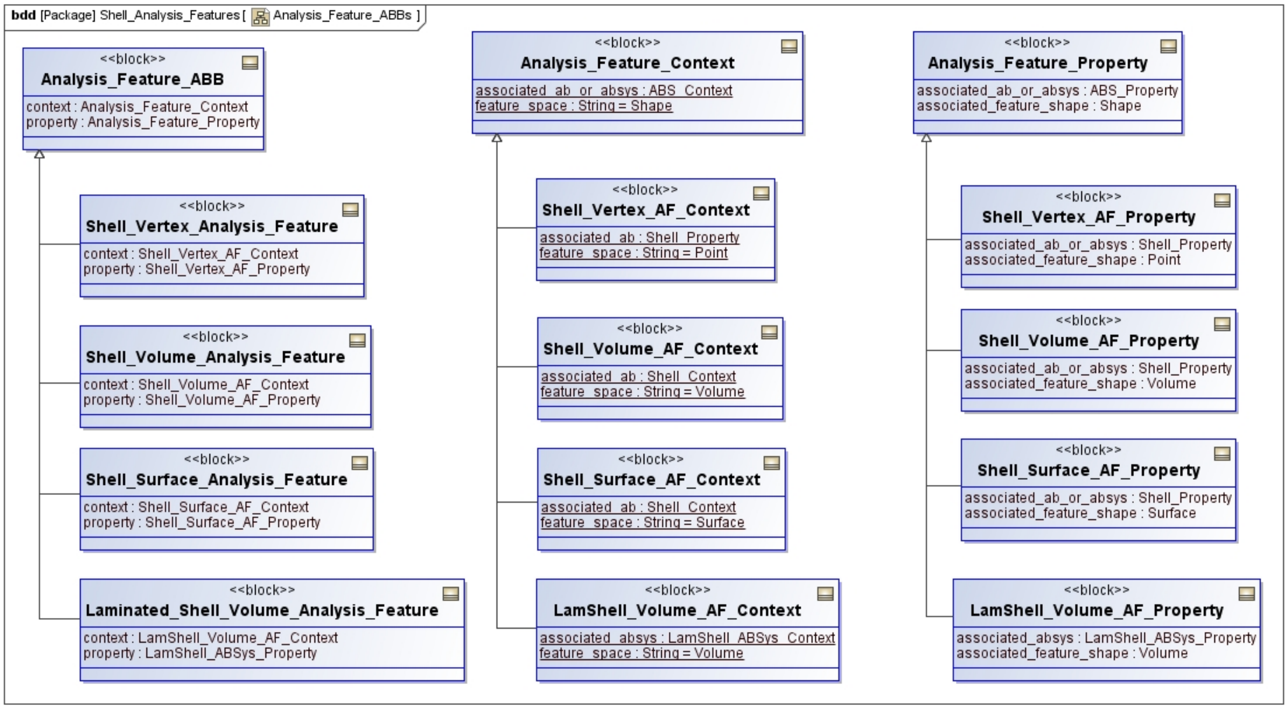

bdd [Package] Shell_Analysis_Features [Analysis_Feature_ABBs]
<<block>>
Analysis_Feature_ABB
context : Analysis_Feature_Context
property : Analysis_Feature_Property
<<block>>
Analysis_Feature_Context
associated_ab_or_absys : ABS_Context
feature_space : String = Shape
<<block>>
Analysis_Feature_Property
associated_ab_or_absys : ABS_Property
associated_feature_shape : Shape
<<block>>
Shell_Vertex_Analysis_Feature
context : Shell_Vertex_AF_Context
property : Shell_Vertex_AF_Property
<<block>>
Shell_Vertex_AF_Context
associated_ab : Shell_Property
feature_space : String = Point
<<block>>
Shell_Vertex_AF_Property
associated_ab_or_absys : Shell_Property
associated_feature_shape : Point
<<block>>
Shell_Volume_Analysis_Feature
context : Shell_Volume_AF_Context
property : Shell_Volume_AF_Property
<<block>>
Shell_Volume_AF_Context
associated_ab : Shell_Context
feature_space : String = Volume
<<block>>
Shell_Volume_AF_Property
associated_ab_or_absys : Shell_Property
associated_feature_shape : Volume
<<block>>
Shell_Surface_Analysis_Feature
context : Shell_Surface_AF_Context
property : Shell_Surface_AF_Property
<<block>>
Shell_Surface_AF_Context
associated_ab : Shell_Context
feature_space : String = Surface
<<block>>
Shell_Surface_AF_Property
associated_ab_or_absys : Shell_Property
associated_feature_shape : Surface
<<block>>
Laminated_Shell_Volume_Analysis_Feature
context : LamShell_Volume_AF_Context
property : LamShell_ABSys_Property
<<block>>
LamShell_Volume_AF_Context
associated_absys : LamShell_ABSys_Context
feature_space : String = Volume
<<block>>
LamShell_Volume_AF_Property
associated_absys : LamShell_ABSys_Property
associated_feature_shape : Volume

Figure 7.18: Analysis feature ABBs

The shell analysis features represented by these blocks are as follows:

- Shell_Vertex_Analysis_Feature block represents a point or vertex analysis feature defined on a shell analysis body.
- Shell_Surface_Analysis_Feature block represents a surface analysis feature defined on a shell analysis body.
- Shell_Volume_Analysis_Feature block represents a volume analysis feature defined on a shell analysis body. The volume feature could be the entire volume of the analysis body or a sub-volume.

- Laminated_Shell_Volume_Analysis_Feature block represents a volume analysis feature defined on a laminated shell analysis body system. A laminated shell analysis body system is a stackup of planar shell analysis bodies.

7.3.5 Analysis Body Interaction ABBs

Analysis body interaction ABB represents the behavior of the interaction between analysis bodies in an analysis body system. Since it is the behavior of the interaction that is core to representing the interaction, analysis body interaction ABBs may be organized based on the analysis disciplines. Figure 7.19 illustrates different types of analysis body interaction ABBs organized based on analysis discipline, and each represented by a block that is a specialization of the AB_AB_Interaction_ABB block. For analysis body interaction ABBs corresponding to a specific analysis discipline, the interaction is defined in terms of the corresponding behavior parameters. For example, for structural interaction ABBs, the interaction is defined in terms of structural behavior parameters. Note that the context attribute of analysis body interaction ABB is of type AB_AB_Interaction_Context block, and the properties of this block define characteristics on the basis of which analysis body interaction ABBs may be organized. Figure 7.19 shows one such hierarchy based on a analysis disciplines.

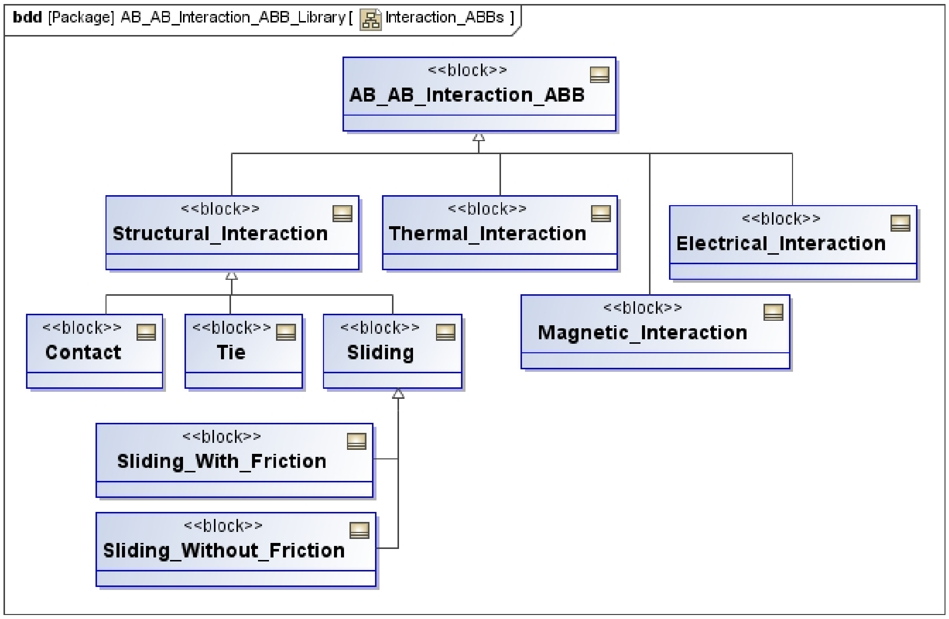


Figure 7.19: Analysis Body Interaction ABBs

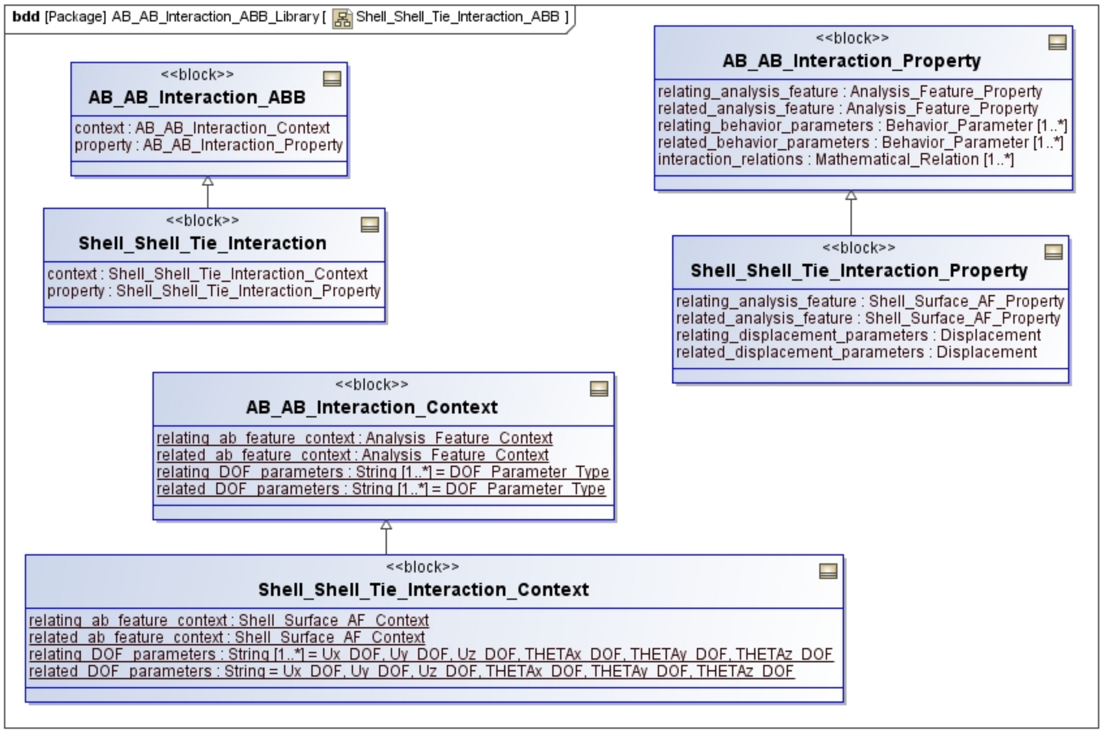

bdd [Package] AB_AB_Interaction_ABB_Library [Shell_Shell_Tie_Interaction_ABB]
<<block>>
AB_AB_Interaction_ABB
context : AB_AB_Interaction_Context
property : AB_AB_Interaction_Property
<<block>>
Shell_Shell_Tie_Interaction
context : Shell_Shell_Tie_Interaction_Context
property : Shell_Shell_Tie_Interaction_Property
<<block>>
AB_AB_Interaction_Property
relating_analysis_feature : Analysis_Feature_Property
related_analysis_feature : Analysis_Feature_Property
relating_behavior_parameters : Behavior_Parameter [1..*]
related_behavior_parameters : Behavior_Parameter [1..*]
interaction_relations : Mathematical_Relation [1..*]
<<block>>
Shell_Shell_Tie_Interaction_Property
relating_analysis_feature : Shell_Surface_AF_Property
related_analysis_feature : Shell_Surface_AF_Property
relating_displacement_parameters : Displacement
related_displacement_parameters : Displacement
<<block>>
AB_AB_Interaction_Context
relating_ab_feature_context : Analysis_Feature_Context
related_ab_feature_context : Analysis_Feature_Context
relating_DOF_parameters : String [1..*] = DOF_Parameter_Type
related_DOF_parameters : String [1..*] = DOF_Parameter_Type
<<block>>
Shell_Shell_Tie_Interaction_Context
relating_ab_feature_context : Shell_Surface_AF_Context
related_ab_feature_context : Shell_Surface_AF_Context
relating_DOF_parameters : String [1..*] = Ux_DOF, Uy_DOF, Uz_DOF, THETAx_DOF, THETAy_DOF, THETAz_DOF
related_DOF_parameters : String = Ux_DOF, Uy_DOF, Uz_DOF, THETAx_DOF, THETAy_DOF, THETAz_DOF

Figure 7.20: Shell-shell tie interaction ABB

Figure 7.20 illustrates shell-shell tie interaction ABB that represents perfectly bonded (or glued) interaction between two shell analysis bodies, and is represented by Shell_Shell_Tie_Interaction block. As the context and property attribute types of this interaction ABB illustrate, the shell-shell tie interaction ABB associates the displacement parameters defined at the surfaces of two shell analysis bodies. The mathematical relations between the displacement parameters are illustrated by the parametric diagram in Figure 7.21. As shown in the diagram, the displacement parameters (both translation and rotation) at the surfaces of two shell analysis bodies participating in the interaction are equated[13] to each other.

[13] The lines connecting the displacement parameters are called binding connectors—used for binding values of connected objects.

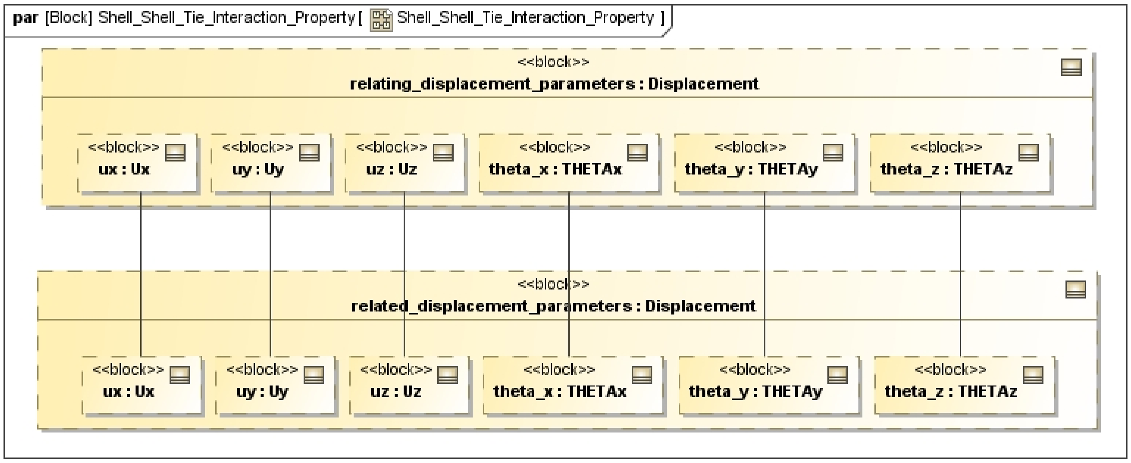


Figure 7.21: Interaction relations for shell-shell tie interaction ABB

7.3.6 Analysis Body System ABBs

An analysis body system is an idealization of an artifact[14] such that it exhibits an idealized sub-set of behaviors of the artifact. An analysis body system ABB represents an analysis body system. Figure 7.22 illustrates a laminated shell analysis body system ABB. A laminated shell analysis body system ABB is a stackup of shells such that surfaces of adjacent shells are glued together. As illustrated in the figure, this composition is reflected in the context and property attributes of this ABB, represented by LamShell_ABSys_Context and LamShell_ABSys_Property blocks respectively. In addition, a volume analysis feature that represents the volume of the laminated shell analysis body system is also defined.

[14] Typically a multi-body artifact unless a single artifact is chopped to define multiple analysis bodies

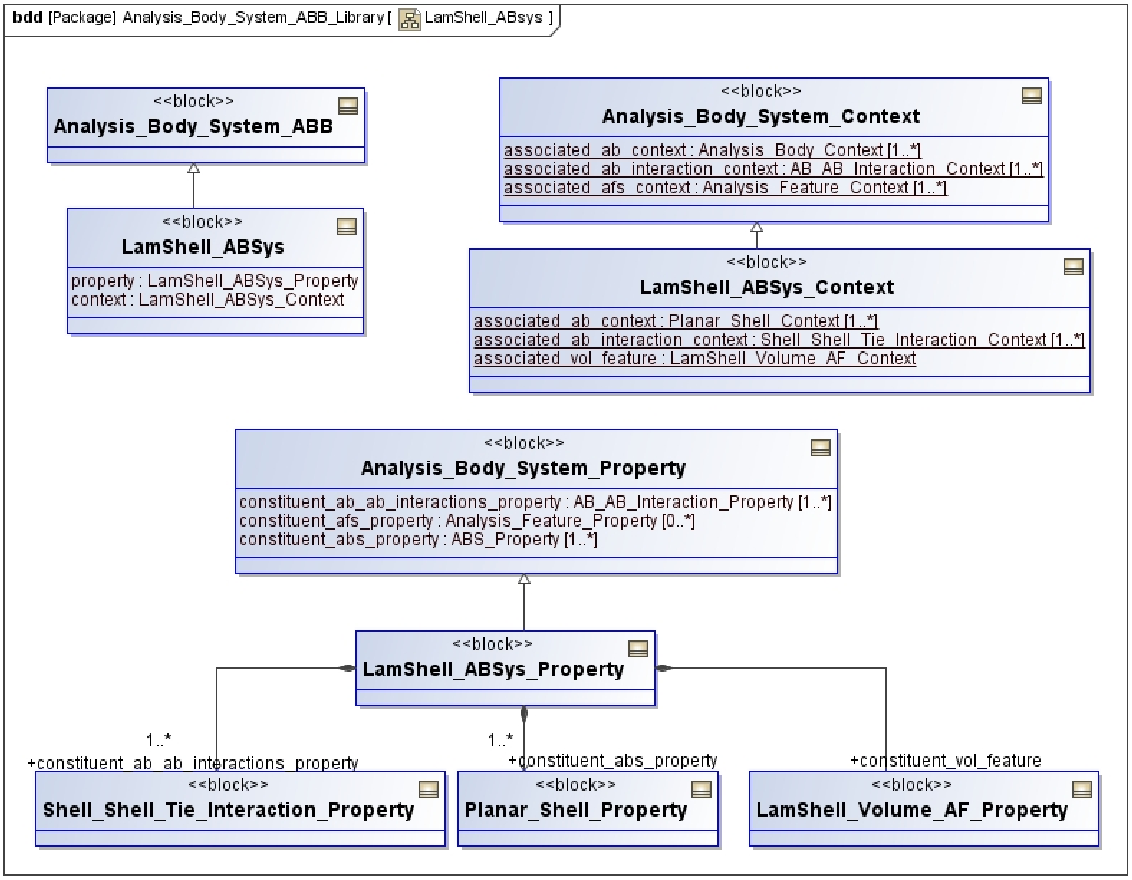

bdd [Package] Analysis_Body_System_ABB_Library [LamShell_ABsys]
<<block>>
Analysis_Body_System_ABB
<<block>>
LamShell_ABSys
property : LamShell_ABSys_Property
context : LamShell_ABSys_Context
<<block>>
Analysis_Body_System_Context
associated_ab_context : Analysis_Body_Context [1..*]
associated_ab_interaction_context : AB_AB_Interaction_Context [1..*]
associated_afs_context : Analysis_Feature_Context [1..*]
<<block>>
LamShell_ABSys_Context
associated_ab_context : Planar_Shell_Context [1..*]
associated_ab_interaction_context : Shell_Shell_Tie_Interaction_Context [1..*]
associated_vol_feature : LamShell_Volume_AF_Context
<<block>>
Analysis_Body_System_Property
constituent_ab_ab_interactions_property : AB_AB_Interaction_Property [1..*]
constituent_afs_property : Analysis_Feature_Property [0..*]
constituent_abs_property : ABS_Property [1..*]
<<block>>
LamShell_ABSys_Property
1..*
+constituent_ab_ab_interactions_property
1..*
+constituent_abs_property
+constituent_vol_feature
<<block>>
Shell_Shell_Tie_Interaction_Property
<<block>>
Planar_Shell_Property
<<block>>
LamShell_Volume_AF_Property

Figure 7.22: Analysis Body System ABBs

7.3.7 Load ABBs

A load is the stimulus to which the response of an analysis body (or analysis body system) is to be computed. The context and property attributes of load ABB are represented by Load_Context and Load_Property blocks respectively. The attributes of the Load_Context block define several dimensions on which load ABBs may be organized. Figure 7.23 illustrates a set of load ABBs organized based on the analysis discipline, and Figure 7.24 illustrates uniform temperature load ABB defined as a special types of load ABB. The uniform temperature load ABB represents a temperature load (increase or decrease in ambient temperature) to which an analysis body (or analysis body system) may be subjected to. Since temperature change affects the entire volume of the analysis body or analysis body system, this load is applied to the volume analysis feature. The load distribution function in the Uniform_Temperature_Load_Property block shows that the final temperature is a constant (as also illustrated in the parametric diagram in Figure 7.24. The change from reference temperature to final temperature is a straight ramp.

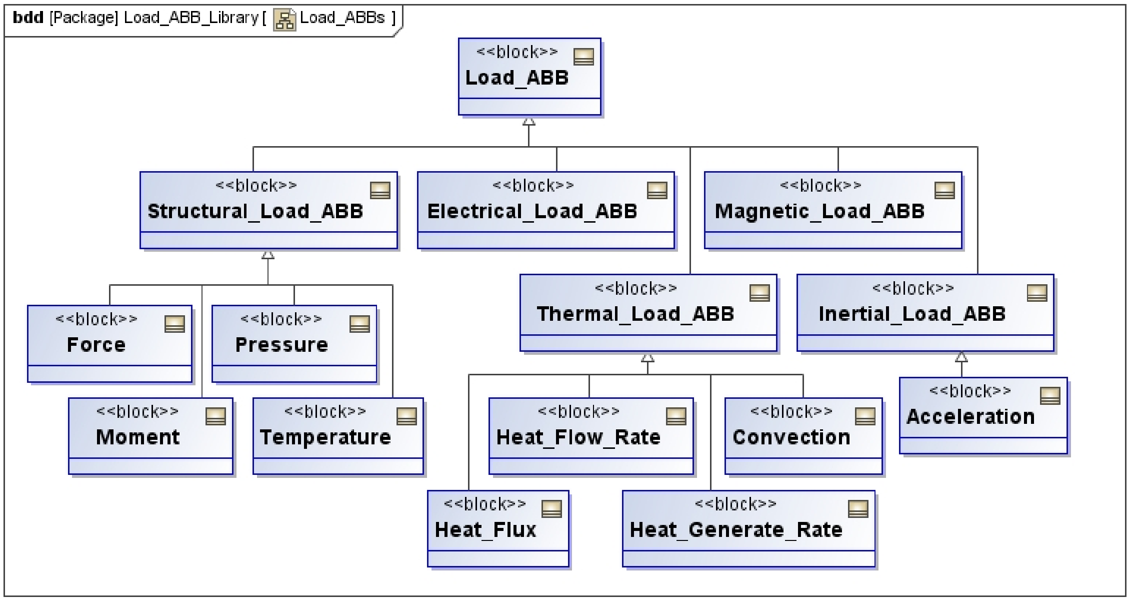


Figure 7.23: Load ABBs

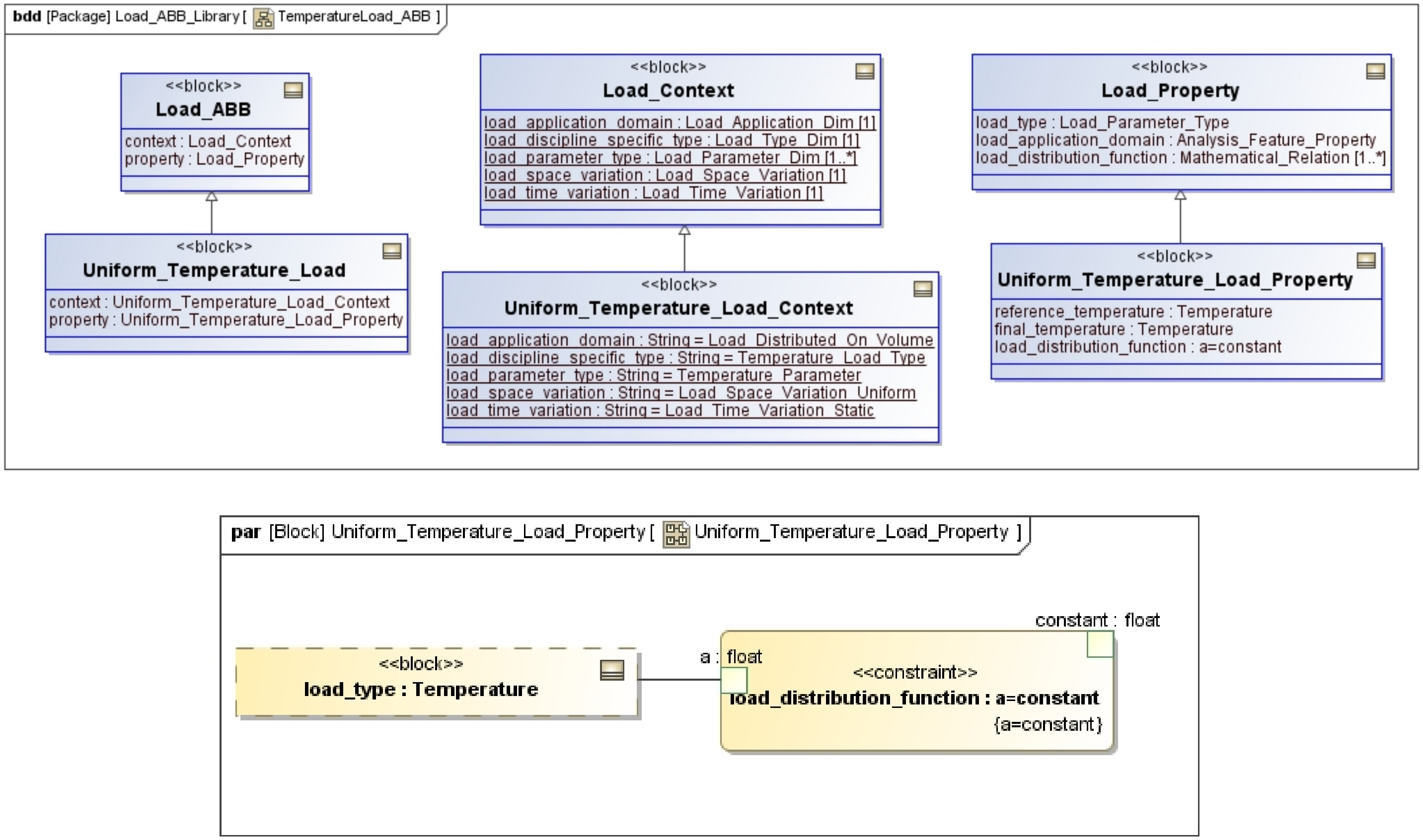

bdd [Package] Load_ABB_Library [TemperatureLoad_ABB]
<<block>>
Load_ABB
context : Load_Context
property : Load_Property
<<block>>
Uniform_Temperature_Load
context : Uniform_Temperature_Load_Context
property : Uniform_Temperature_Load_Property
<<block>>
Load_Context
load_application_domain : Load_Application_Dim [1]
load_discipline_specific_type : Load_Type_Dim [1]
load_parameter_type : Load_Parameter_Dim [1..*]
load_space_variation : Load_Space_Variation [1]
load_time_variation : Load_Time_Variation [1]
<<block>>
Uniform_Temperature_Load_Context
load_application_domain : String = Load_Distributed_On_Volume
load_discipline_specific_type : String = Temperature_Load_Type
load_parameter_type : String = Temperature_Parameter
load_space_variation : String = Load_Space_Variation_Uniform
load_time_variation : String = Load_Time_Variation_Static
<<block>>
Load_Property
load_type : Load_Parameter_Type
load_application_domain : Analysis_Feature_Property
load_distribution_function : Mathematical_Relation [1..*]
<<block>>
Uniform_Temperature_Load_Property
reference_temperature : Temperature
final_temperature : Temperature
load_distribution_function : a=constant
par [Block] Uniform_Temperature_Load_Property [Uniform_Temperature_Load_Property]
<<block>>
load_type : Temperature
a : float
constant : float
<<constraint>>
load_distribution_function : a=constant
{a=constant}

Figure 7.24: Uniform temperature load ABBs

7.3.8 Behavior Condition ABBs

Behavior_Condition_ABB block is used for representing behavior conditions. Behavior conditions are additional conditions applied to analysis body or analysis body system under which their response to loads is to be computed. Figure 7.25 illustrates behavior condition ABBs organized based on analysis disciplines. The PointDisplacementFixed_Condition block represents a behavior condition in which a point analysis feature is held static, i.e. the displacement parameters are set to zero.

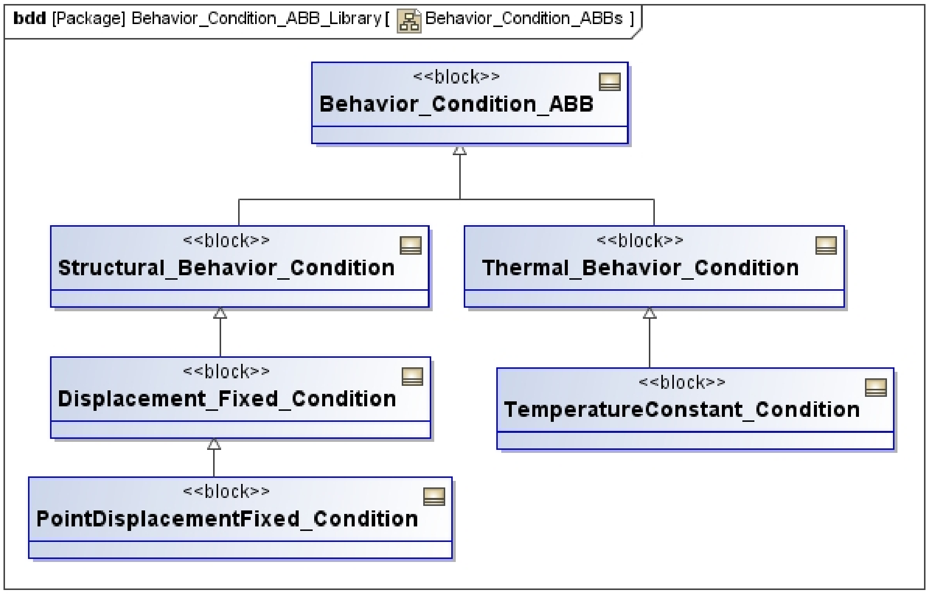

bdd [Package] Behavior_Condition_ABB_Library [Behavior_Condition_ABBs]
<<block>>
Behavior_Condition_ABB
<<block>>
Structural_Behavior_Condition
<<block>>
Thermal_Behavior_Condition
<<block>>
Displacement_Fixed_Condition
<<block>>
TemperatureConstant_Condition
<<block>>
PointDisplacementFixed_Condition

Figure 7.25: Behavior condition ABBs

Figure 7.26 illustrates the context and property attributes of this behavior condition ABB and Figure 7.27 illustrates the behavior condition relations for this ABB.

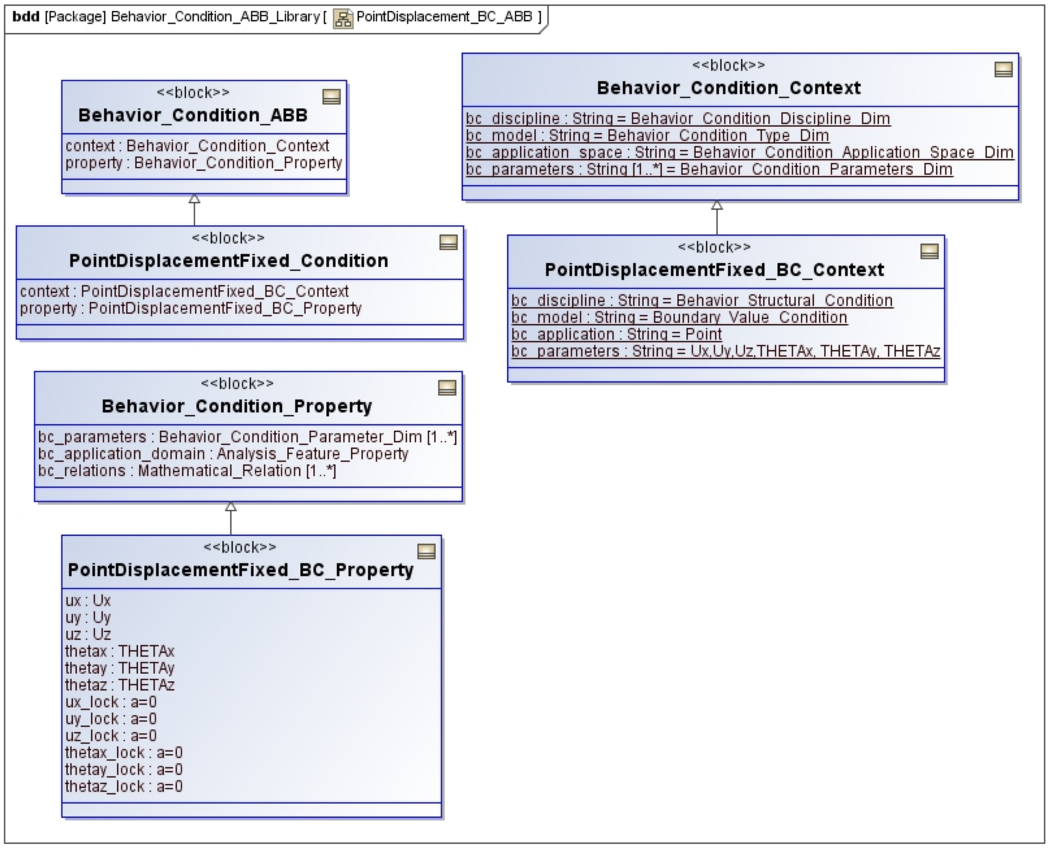

bdd [Package] Behavior_Condition_ABB_Library [PointDisplacement_BC_ABB]
<<block>>
Behavior_Condition_ABB
context : Behavior_Condition_Context
property : Behavior_Condition_Property
<<block>>
Behavior_Condition_Context
bc_discipline : String = Behavior_Condition_Discipline_Dim
bc_model : String = Behavior_Condition_Type_Dim
bc_application_space : String = Behavior_Condition_Application_Space_Dim
bc_parameters : String [1..*] = Behavior_Condition_Parameters_Dim
<<block>>
PointDisplacementFixed_Condition
context : PointDisplacementFixed_BC_Context
property : PointDisplacementFixed_BC_Property
<<block>>
PointDisplacementFixed_BC_Context
bc_discipline : String = Behavior_Structural_Condition
bc_model : String = Boundary_Value_Condition
bc_application : String = Point
bc_parameters : String = Ux,Uy,Uz,THETAx, THETAy, THETAz
<<block>>
Behavior_Condition_Property
bc_parameters : Behavior_Condition_Parameter_Dim [1..*]
bc_application_domain : Analysis_Feature_Property
bc_relations : Mathematical_Relation [1..*]
<<block>>
PointDisplacementFixed_BC_Property
ux : Ux
uy : Uy
uz : Uz
thetax : THETAx
thetay : THETAy
thetaz : THETAz
ux_lock : a=0
uy_lock : a=0
uz_lock : a=0
thetax_lock : a=0
thetay_lock : a=0
thetaz_lock : a=0

Figure 7.26: Point displacement fixed behavior condition ABB

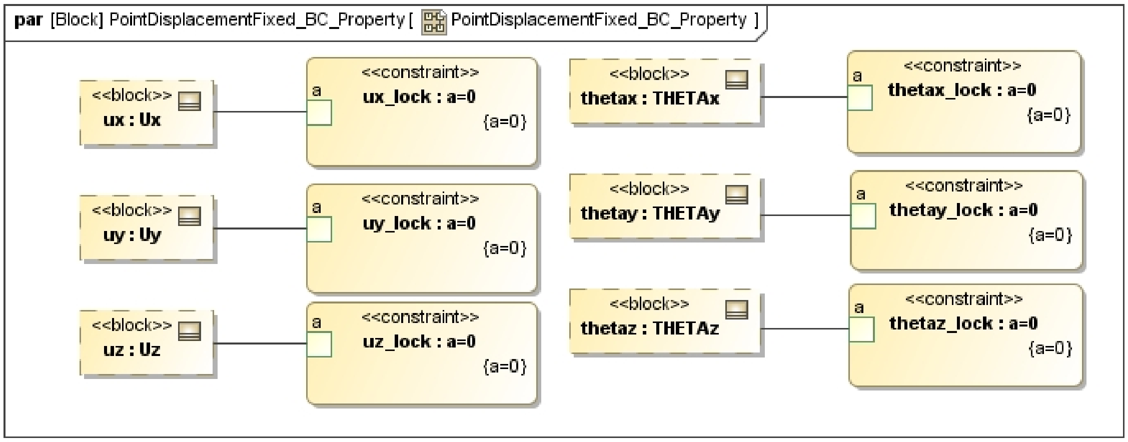


Figure 7.27: Behavior condition relations for point displacement fixed behavior condition ABB

7.4 Behavior Models

7.4.1 Abstractions

In this section, the five levels of abstractions of behavior models relevant in the KCM are described. Figure 7.28 shows a conceptual hierarchy of models in the KCM as a SysML block definition diagram. The five levels of abstractions of behavior models are grouped as the behavior model stack in the diagram and are described in this section. The behavior model abstraction at each level in the stack represents a set of behavior models. As one moves down the stack (increase in levels), the models become more specialized and represent a sub-set of behavior models represented by the preceding abstraction level.

The five different abstractions of behavior models in the KCM are useful for the following reasons:

- Since the primary use case of KCM is the automated composition of behavior models, it is necessary to distinguish the abstraction levels where the model composition transformations are specified versus the levels at which they are executed versus the level at which behavior models are solved to compute behavior parameters. *This approach allows one to define transformations to create a set of behavior models and not just a specific behavior model*.
- Since KCM is targeted to address VTMB problems, the abstractions allow one to distinguish between behavior models that represent the behavior of artifacts with different assembly system topologies versus those that represent the behavior of artifacts with a fixed topology. The former type of behavior models are those for which assembly system topology-specific decisions have not been taken by analysts while the latter type of behavior models at those where these decisions have been taken.
- The abstractions also allow one to study variations of behavior models. Each level in the abstraction corresponds to a specific type of variation of behavior models and hence represents a set of behavior models. For example, at Level 3 in the behavior model stack the assembly system topology of artifact may vary; at Level 4 the topology is fixed but size and properties of artifacts may vary.

.

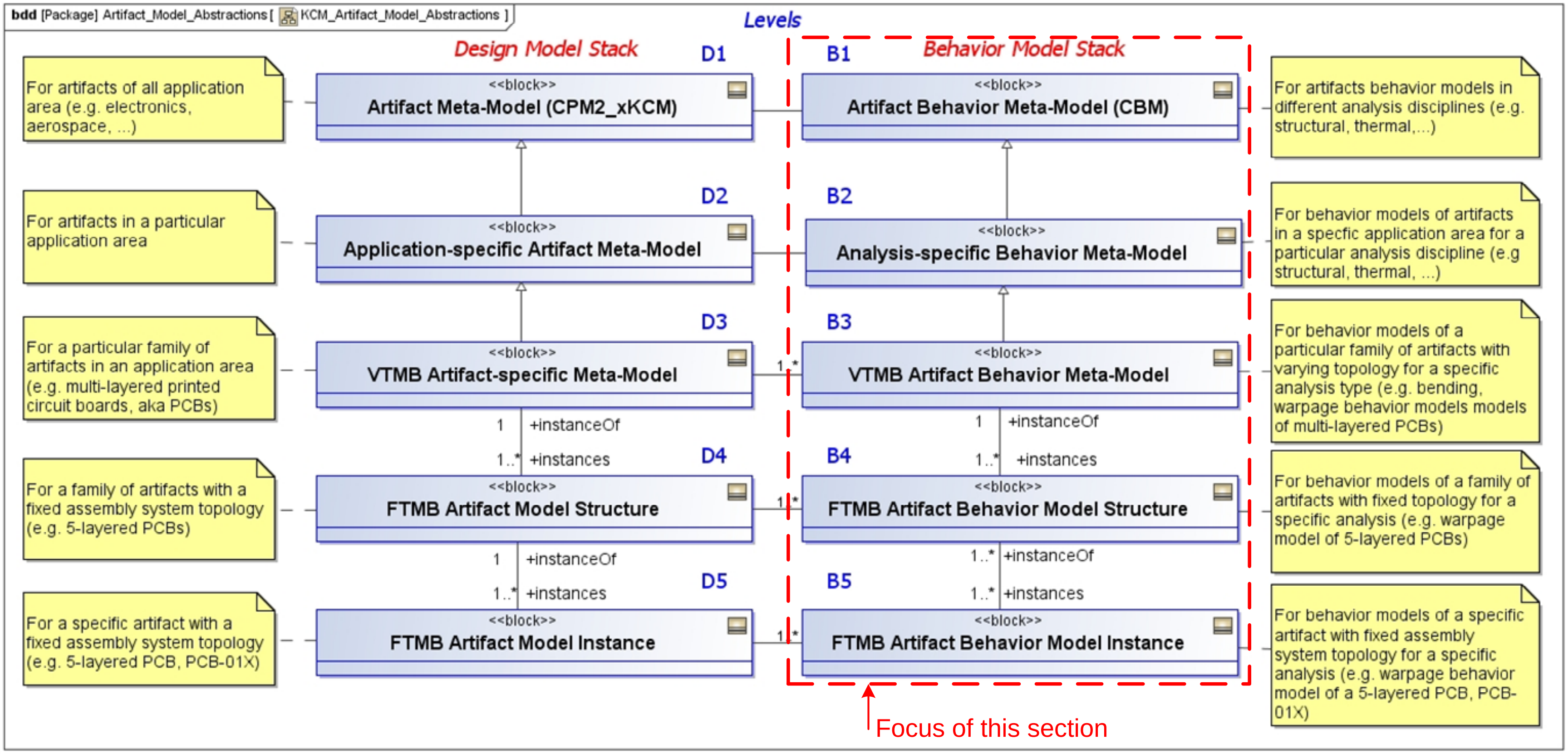

bdd [Package] Artifact_Model_Abstractions [KCM_Artifact_Model_Abstractions]
Levels
Design Model Stack
Behavior Model Stack
D1
D2
D3
D4
D5
B1
B2
B3
B4
B5
For artifacts of all application area (e.g. electronics, aerospace, ...)
<<block>> Artifact Meta-Model (CPM2_xKCM)
<<block>> Artifact Behavior Meta-Model (CBM)
For artifacts behavior models in different analysis disciplines (e.g. structural, thermal,...)
For artifacts in a particular application area
<<block>> Application-specific Artifact Meta-Model
<<block>> Analysis-specific Behavior Meta-Model
For behavior models of artifacts in a specfic application area for a particular analysis discipline (e.g structural, thermal, ...)
For a particular family of artifacts in an application area (e.g. multi-layered printed circuit boards, aka PCBs)
<<block>> VTMB Artifact-specific Meta-Model
<<block>> VTMB Artifact Behavior Meta-Model
For behavior models of a particular family of artifacts with varying topology for a specific analysis type (e.g. bending, warpage behavior models models of multi-layered PCBs)
1
+instanceOf
1..*
+instances
For a family of artifacts with a fixed assembly system topology (e.g. 5-layered PCBs)
<<block>> FTMB Artifact Model Structure
<<block>> FTMB Artifact Behavior Model Structure
For behavior models of a family of artifacts with fixed topology for a specific analysis (e.g. warpage model of 5-layered PCBs)
1..*
+instanceOf
1..*
+instances
For a specific artifact with a fixed assembly system topology (e.g. 5-layered PCB, PCB-01X)
<<block>> FTMB Artifact Model Instance
<<block>> FTMB Artifact Behavior Model Instance
For behavior models of a specific artifact with fixed assembly system topology for a specific analysis (e.g. warpage behavior model of a 5-layered PCB, PCB-01X)
1..*
Focus of this section

Figure 7.28: Behavior Model Abstractions in KCM

The five different levels of abstractions in the behavior model stack are described below with examples

- **Level 1 (B1): Artifact Behavior Meta-Model (Core Behavior Model)**

The Level 1 abstraction in the behavior model stack is known as Artifact Behavior Meta-Model. This meta-model defines the constructs and relationships for representing behavior models of artifacts in different application areas (such as electronics, automotive, and aerospace) for different types of analyses (such as structural analyses, thermal analyses, and electromagnetic analyses). *The Core Behavior Model presented in this chapter is a specific example of an Artifact Behavior Meta-Model* (with special focus on VTMB artifacts). The Artifact Behavior Meta-Model is related to the Artifact Meta-Model (Level 1 in design model stack) and this relation represents the relation between an artifact (specifically its analyzable abstraction) and its behavior models. In KCM, CPM2_xKCM (specific example of Artifact Meta-Model) is related to CBM through the Behavior block as described in section 7.1 and illustrated in Figure 7.3.

- **Level 2 (B2): Analysis-specific Behavior Meta-Model**

The Level 2 abstraction in the behavior model stack is known as Analysis-specific Behavior Meta-Model. This meta-model is a specialization of the Artifact Behavior Meta-Model (B1) and it defines the constructs and relationships for representing behavior models of artifacts in a specific application area for a specific analysis domain. For example, the CBM may be specialized to create a behavior meta-model for thermo-mechanical analyses of electronics artifacts.

- **Level 3 (B3) : VTMB Artifact Behavior Meta-Model**

The Level 3 abstraction in the behavior model stack is known as VTMB Artifact Behavior Meta-Model, where VTMB stands for Variable Topology Multi-Body. This meta-model may be defined as a specialization of Analysis-specific Behavior Meta-Model (B2) or directly as a specialization of Artifact Behavior Meta-Model (B1). This is so because it may not be practical to develop a behavior meta-model for each analysis domain for some artifact application areas. The VTMB Artifact Behavior Meta-Model defines the constructs and relationships for representing behavior models of a family of artifacts with different assembly system topologies and for a specific type of analysis. All members of this family are a specific type of artifact in a given application area and with different (non-equivalent) assembly system topologies. An example of the VTMB Artifact Behavior Model is a behavior meta-model defined for thermo-mechanical analyses of multi-layered printed circuit boards. Here, the behavior meta-mode is for a family of artifacts of a specific type (printed circuit boards) but with different assembly system topologies (such as 5-layered, 10-layered, or 13-layered PCBs).

The PCB_nSx_ThermoMech_Behavior_Meta-Model illustrated in section 7.4.2 is an example of the VTMB Artifact Behavior Meta-Model. It is a meta-model for representing thermo-mechanical behavior models of n-stratum printed circuit boards.

- **Level 4 (B4): FTMB Artifact Behavior Model Structure**

The Level 4 abstraction in the behavior model stack is known as FTMB Artifact Behavior Model Structure, where FTMB stands for Fixed Topology Multi-Body. This model is defined as an instance of the VTMB Artifact Behavior Model Structure (B3). In instantiating the B3 model, only decisions pertaining to the assembly system topology are populated. While the B3 abstraction is a meta-model for representing behavior models of a family of artifacts with varying assembly system topology, the B4 abstraction represents behavior models of artifacts with a fixed topology. For example, one may create a B4 model for representing 5-layered PCBs, or 10-layered PCBs, or 15-layered PCBs. Since only topology-specific decisions have been populated in an FTMB Artifact Behavior Model Structure, it is a partially-specified instance model and it provides a structure for creating several fully-specified instances. Hence, B4 abstraction is a behavior model structure and not a specific behavior model. It represents a set of behavior models for artifacts with the equivalent assembly system topologies.

The PCB_5Sx_ThermoMech_Behavior_Model_Structure illustrated in section 7.4.2 is an example of the FTMB Artifact Behavior Model Structure. It represents thermo-mechanical behavior models of a set of 5-layered printed circuit boards.

- **Level 5 (B5): FTMB Artifact Behavior Model Instance**

The B5 abstraction in the behavior model stack is known as FTMB Artifact Behavior Model Instance. This model is a fully-specified instance of B3 model (VTMB Artifact Behavior Meta-Model). In contrast to a B4 model, a B5 model is a specific behavior model and is intended to be solvable. It incorporates all decisions that have been taken to completely define a behavior model (i.e. a solvable behavior model). For a given B4 behavior model (structure), several B5 models (instances) may be created. A B5 model is a behavior model of a specific artifact for a specific analysis.

The PCB_5S1_ThermoMech_Behavior_Model_Instance model illustrated in section 7.4.2 is an example of the FTMB Artifact Behavior Model Instance. It represents a thermo-mechanical behavior model of a specific 5-layered printed circuit board.

7.4.2 Examples

In this section, specific examples of the model abstractions in the behavior model stack are presented. The Level 1 abstraction is the Core Behavior Model which was presented in section 7.1. In this section, B3, B4, and B5 models are presented to illustrate how the CBM is used for representing specialized meta-models and models. The examples presented here show different levels of abstractions of thermo-mechanical behavior models for multi-layered printed circuit boards. The relation of these models to the corresponding models in the design model stack is also illustrated.

- **Level 3 (B3) example:** PCB_nSx_ThermoMech_Behavior_Meta-Model

Figure 7.29 below illustrates PCB_nSx_ThermoMech_Behavior_Meta-Model —a thermo-mechanical behavior meta-model for n-layered printed circuit boards. This meta-model is created as a specialization of the Core Behavior Model, and it defines the constructs and relationships for representing thermo-mechanical behavior models of multi-layered PCBs. The central entity in this meta-model is PCB-LamShell_ThermoMech_BM block (specialization of Behavior_Model block) and it represents a thermo-mechanical behavior meta-model where an n-layered PCB is idealized as an n-layered laminated shell system.

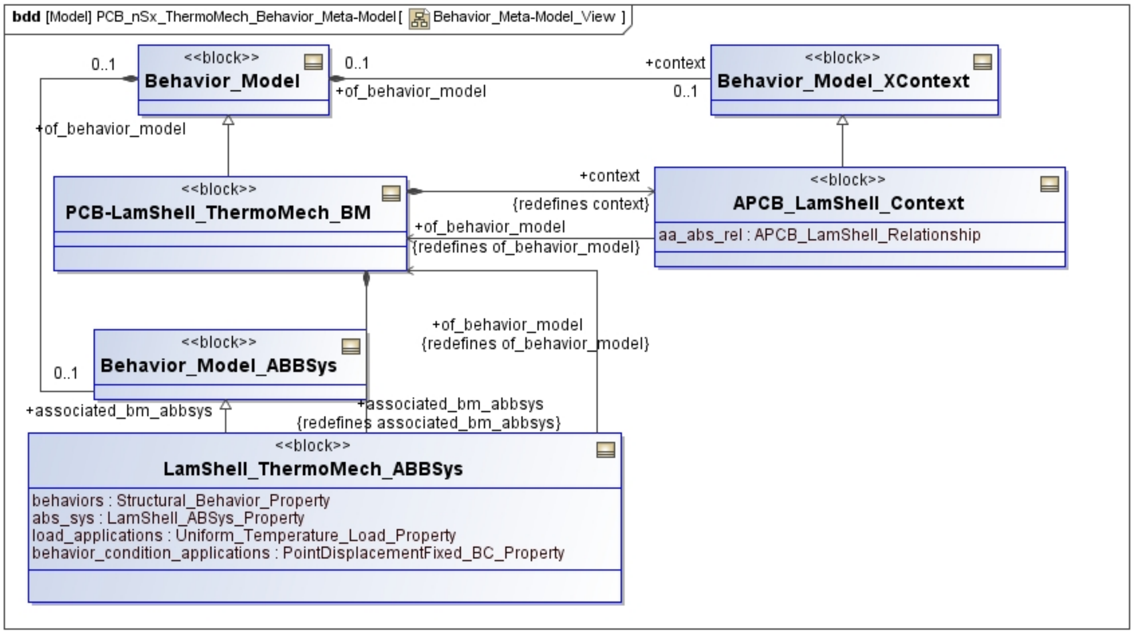

bdd [Model] PCB_nSx_ThermoMech_Behavior_Meta-Model [Behavior_Meta-Model_View]
<<block>>
Behavior_Model
0..1
0..1
+of_behavior_model
+context
0..1
<<block>>
Behavior_Model_XContext
+of_behavior_model
<<block>>
PCB-LamShell_ThermoMech_BM
+context
{redefines context}
<<block>>
APCB_LamShell_Context
aa_abs_rel : APCB_LamShell_Relationship
+of_behavior_model
{redefines of_behavior_model}
+of_behavior_model
{redefines of_behavior_model}
<<block>>
Behavior_Model_ABBSys
0..1
+associated_bm_abbsys
+associated_bm_abbsys
{redefines associated_bm_abbsys}
<<block>>
LamShell_ThermoMech_ABBSys
behaviors : Structural_Behavior_Property
abs_sys : LamShell_ABSys_Property
load_applications : Uniform_Temperature_Load_Property
behavior_condition_applications : PointDisplacementFixed_BC_Property

Figure 7.29: PCB_nSx_ThermoMech_Behavior_Meta-Model (B3): A thermo-mechanical behavior meta-model for multi-layered PCBs (View 1)

This behavior meta-model is composed of an ABB system meta-model, and a context meta-model that relates the ABB system meta-model to the analyzable artifact meta-model (D3 abstraction in the design model stack). LamShell_ThermoMech_ABBSys block represents the specialized ABB system meta-model and PCB_LamShell_Context represents the specialized context meta-model. Note that the specializations also redefine the block properties. For example, Behavior_Model.context is of type Behavior_Model_XContext but PCB-LamShell_ThermoMech_BM.context is of type APCB_LamShell_Context. The ABB system meta-model (LamShell_ThermoMech_ABBSys block) is composed of:
(i) n-layered laminated shell system (represented by LamShell_ABSys_Property block illustrated in Figure 7.30), (ii) uniform temperature load applied to the laminated shell system (represented by Uniform_Temperature_Load_Property block), (iii) point displacement boundary condition (represented by PointDisplacementFixed_BC_Property block), and (iv) set of structural behavior parameters (Structural_Behavior_Property block). These four types of ABBs are described are described in section 7.3. Note that in PCB_nSx_ThermoMech_Behavior_Meta-Model, specific types of loads and boundary conditions have also been specified as part of the meta-model. However, it is not

necessary to specify these and keep the meta-model more generic. The context meta-model (represented by the APCB_LamShell_Context block) is composed of idealization relationships between n-stratum analyzable PCB (Level 3 model in the design model stack) and n-layered laminated shell system (special type of analysis body system). These relationships are represented by the APCB_LamShell_Relationship block in Figure 7.30.

Figure 7.30 illustrates a more detailed view of the meta-model. In particular, it shows the n-layered laminated shell system and its relationship to the n-layered analyzable PCB model. The n-layered laminated shell system is composed of n individual planar shells (represented by Planar_Shell_Property block), the tie interactions between these planar shells that are stacked together (represented by Shell_Shell_Tie_Interaction_Property block), and the planar shell surfaces that participate in defining the tie interactions—the secondary surface of a preceding shell is tied to the primary surface of the succeeding shell—that are represented by Shell_Surface_AF_Property block. Planar shell, Shell-Shell tie interaction, and shell surface analysis features are special types of ABBs and blocks representing these ABBs are described in section 7.3. Corresponding to the n-layered laminated shell system in the behavior model stack is the n-layered analyzable PCB in the design model stack. An n-layered analyzable PCB is composed of individual stratums, the interactions between the stratums, and the stratum surface features that participate in the interactions—explained in details in section 6.2. The behavior idealization relationships that are defined as part of the context meta-model relate the stratums, interactions, and surface features to the planar shells, tie interactions, and shell surface analysis features respectively. The APCB_LamShell_Relationship block is the central entity in this context meta-model (part of the behavior meta-model) and represents the behavior idealization relationships between an n-layered analyzable PCB to an n-layered laminated shell system. The behavior idealization relationships are composed of: (i) idealization relationships between each analyzable stratum to the corresponding planar shell, (ii) idealization relationships between an analyzable stratum surface to the corresponding shell surface, and (iii) idealization relationships between the interactions between adjacent analyzable stratums and the tie interactions between adjacent planar shells. These three types of idealization relationships are represented by AStratum_PShell_Relationship, AStratSurf_PShellSurf_Relationship, and AdjStrat_PShellTie_Interaction_Relationship blocks respectively.

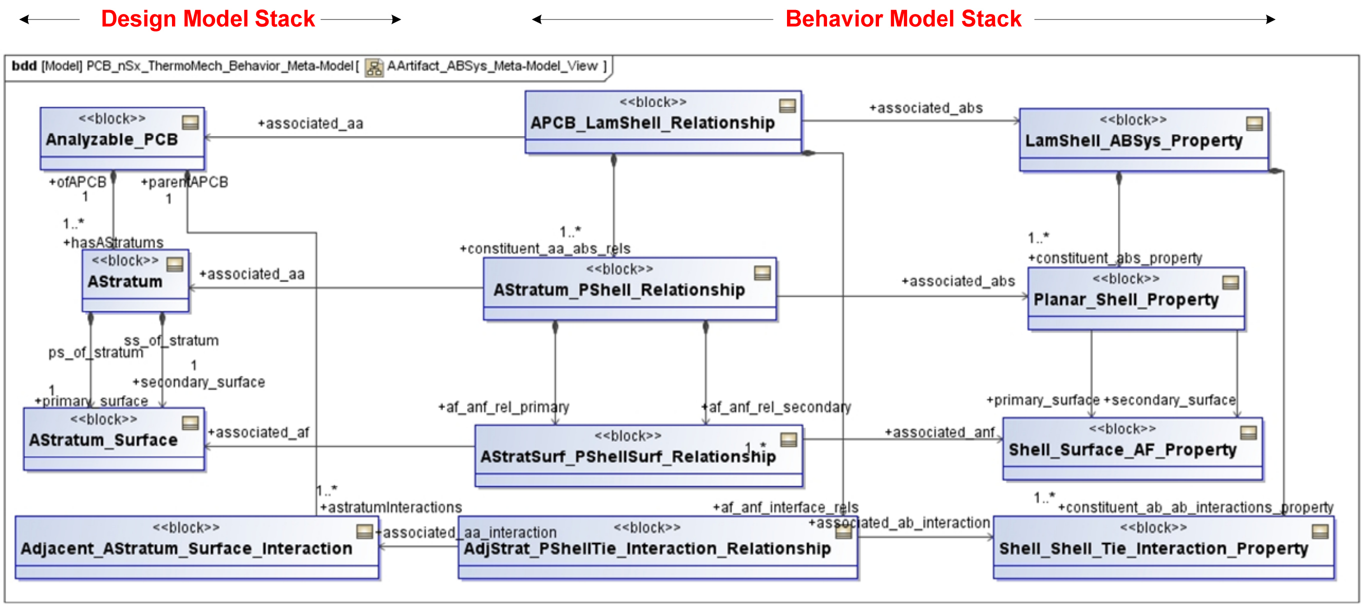


Figure 7.30: Example D3-B3 model showing relationships between n-stratum analyzable PCBs (D3) and corresponding n-layered laminated shell systems (B3)

- **Level 4 (B4) example:** PCB_5Sx_ThermoMech_Behavior_Model_Structure

Figure 7.31 below illustrates PCB_5Sx_ThermoMech_Behavior_Model_Structure—a thermo-mechanical behavior model structure for 5-stratum[15] PCBs. This behavior model structure is an instance of the PCB_nSx_ThermoMech_Behavior_Meta-Model and represents thermo-mechanical behavior models of printed circuit boards with 5 stratums. As shown in Figure 7.31, central entity in this B4 model is PCB-LamShell_5Sx_ThermoMech_BM (an instance of PCB-LamShell_ThermoMech_BM). Just like its parent meta-model and CBM, PCB_5Sx_ThermoMech_Behavior_Model_Structure is composed of an ABB system, and a context model that relates the ABB system to analyzable design model structure (D4 model in the design model stack). LamShell_5Sx_Thermo-Mech_ABB_System block

[15] The term 'layer' in printed circuit boards typically refers to design layers (electrically conductive). Hence the term 'stratum' is used to refer to all layers in general.

represents the ABB system, and APCB_LamShell_5Sx_Context block represents the context model for this behavior model structure.

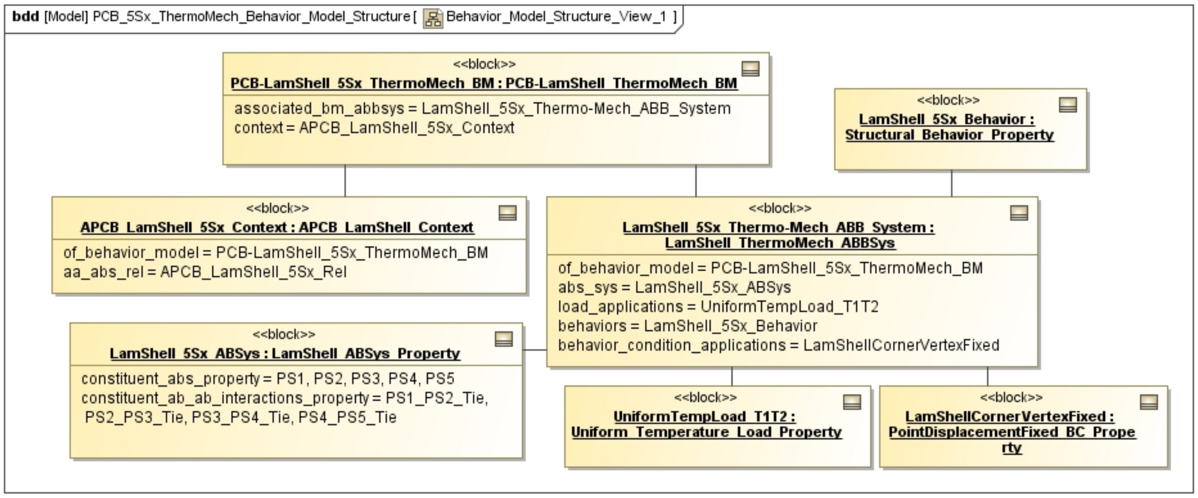

bdd [Model] PCB_5Sx_ThermoMech_Behavior_Model_Structure [Behavior_Model_Structure_View_1]
<<block>>
PCB-LamShell_5Sx_ThermoMech_BM : PCB-LamShell_ThermoMech_BM
associated_bm_abbsys = LamShell_5Sx_Thermo-Mech_ABB_System
context = APCB_LamShell_5Sx_Context
<<block>>
LamShell_5Sx_Behavior : Structural_Behavior_Property
<<block>>
APCB_LamShell_5Sx_Context : APCB_LamShell_Context
of_behavior_model = PCB-LamShell_5Sx_ThermoMech_BM
aa_abs_rel = APCB_LamShell_5Sx_Rel
<<block>>
LamShell_5Sx_Thermo-Mech_ABB_System : LamShell_ThermoMech_ABBSys
of_behavior_model = PCB-LamShell_5Sx_ThermoMech_BM
abs_sys = LamShell_5Sx_ABSys
load_applications = UniformTempLoad_T1T2
behaviors = LamShell_5Sx_Behavior
behavior_condition_applications = LamShellCornerVertexFixed
<<block>>
LamShell_5Sx_ABSys : LamShell_ABSys_Property
constituent_abs_property = PS1, PS2, PS3, PS4, PS5
constituent_ab_ab_interactions_property = PS1_PS2_Tie, PS2_PS3_Tie, PS3_PS4_Tie, PS4_PS5_Tie
<<block>>
UniformTempLoad_T1T2 : Uniform_Temperature_Load_Property
<<block>>
LamShellCornerVertexFixed : PointDisplacementFixed_BC_Property

Figure 7.31: PCB_5Sx_ThermoMech_Behavior_Model_Structure (B4): A thermo-mechanical behavior model structure for 5-layered PCBs (View 1)

The ABB system is composed of a 5-layered laminated shell system as illustrated in Figure 7.32. The figure shows 5 planar shells, their primary and secondary surfaces, and the tie interactions between planar shells in the laminate shell system. The planar shells, their surfaces, and shell-shell tie interactions are ABB instances. The properties of each ABB instance are also shown in the figure.

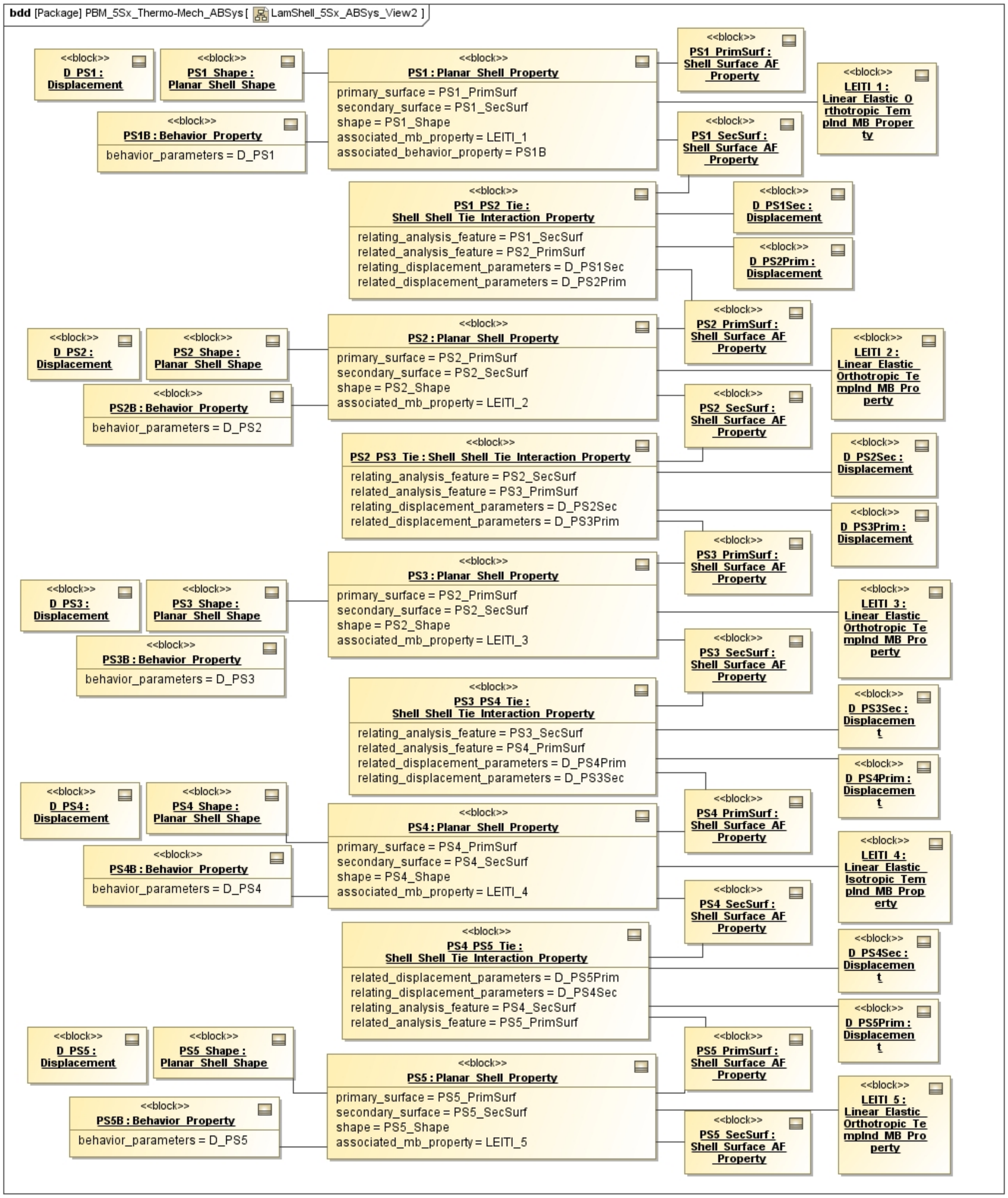

bdd [Package] PBM_5Sx_Thermo-Mech_ABSys [LamShell_5Sx_ABSys_View2]
<<block>> D_PS1 : Displacement
<<block>> PS1_Shape : Planar_Shell_Shape
<<block>> PS1 : Planar_Shell_Property
primary_surface = PS1_PrimSurf
secondary_surface = PS1_SecSurf
shape = PS1_Shape
associated_mb_property = LEITI_1
associated_behavior_property = PS1B
<<block>> PS1B : Behavior_Property
behavior_parameters = D_PS1
<<block>> PS1_PrimSurf : Shell_Surface_AF_Property
<<block>> PS1_SecSurf : Shell_Surface_AF_Property
<<block>> LEITI_1 : Linear_Elastic_Orthotropic_TempInd_MB_Property
<<block>> PS1_PS2_Tie : Shell_Shell_Tie_Interaction_Property
relating_analysis_feature = PS1_SecSurf
related_analysis_feature = PS2_PrimSurf
relating_displacement_parameters = D_PS1Sec
related_displacement_parameters = D_PS2Prim
<<block>> D_PS1Sec : Displacement
<<block>> D_PS2Prim : Displacement
<<block>> D_PS2 : Displacement
<<block>> PS2_Shape : Planar_Shell_Shape
<<block>> PS2 : Planar_Shell_Property
primary_surface = PS2_PrimSurf
secondary_surface = PS2_SecSurf
shape = PS2_Shape
associated_mb_property = LEITI_2
<<block>> PS2B : Behavior_Property
behavior_parameters = D_PS2
<<block>> PS2_PrimSurf : Shell_Surface_AF_Property
<<block>> PS2_SecSurf : Shell_Surface_AF_Property
<<block>> LEITI_2 : Linear_Elastic_Orthotropic_TempInd_MB_Property
<<block>> PS2_PS3_Tie : Shell_Shell_Tie_Interaction_Property
relating_analysis_feature = PS2_SecSurf
related_analysis_feature = PS3_PrimSurf
relating_displacement_parameters = D_PS2Sec
related_displacement_parameters = D_PS3Prim
<<block>> D_PS2Sec : Displacement
<<block>> D_PS3Prim : Displacement
<<block>> D_PS3 : Displacement
<<block>> PS3_Shape : Planar_Shell_Shape
<<block>> PS3 : Planar_Shell_Property
primary_surface = PS2_PrimSurf
secondary_surface = PS2_SecSurf
shape = PS2_Shape
associated_mb_property = LEITI_3
<<block>> PS3B : Behavior_Property
behavior_parameters = D_PS3
<<block>> PS3_PrimSurf : Shell_Surface_AF_Property
<<block>> PS3_SecSurf : Shell_Surface_AF_Property
<<block>> LEITI_3 : Linear_Elastic_Orthotropic_TempInd_MB_Property
<<block>> PS3_PS4_Tie : Shell_Shell_Tie_Interaction_Property
relating_analysis_feature = PS3_SecSurf
related_analysis_feature = PS4_PrimSurf
related_displacement_parameters = D_PS4Prim
relating_displacement_parameters = D_PS3Sec
<<block>> D_PS3Sec : Displacement
<<block>> D_PS4Prim : Displacement
<<block>> D_PS4 : Displacement
<<block>> PS4_Shape : Planar_Shell_Shape
<<block>> PS4 : Planar_Shell_Property
primary_surface = PS4_PrimSurf
secondary_surface = PS4_SecSurf
shape = PS4_Shape
associated_mb_property = LEITI_4
<<block>> PS4B : Behavior_Property
behavior_parameters = D_PS4
<<block>> PS4_PrimSurf : Shell_Surface_AF_Property
<<block>> PS4_SecSurf : Shell_Surface_AF_Property
<<block>> LEITI_4 : Linear_Elastic_Isotropic_TempInd_MB_Property
<<block>> PS4_PS5_Tie : Shell_Shell_Tie_Interaction_Property
related_displacement_parameters = D_PS5Prim
relating_displacement_parameters = D_PS4Sec
relating_analysis_feature = PS4_SecSurf
related_analysis_feature = PS5_PrimSurf
<<block>> D_PS4Sec : Displacement
<<block>> D_PS5Prim : Displacement
<<block>> D_PS5 : Displacement
<<block>> PS5_Shape : Planar_Shell_Shape
<<block>> PS5 : Planar_Shell_Property
primary_surface = PS5_PrimSurf
secondary_surface = PS5_SecSurf
shape = PS5_Shape
associated_mb_property = LEITI_5
<<block>> PS5B : Behavior_Property
behavior_parameters = D_PS5
<<block>> PS5_PrimSurf : Shell_Surface_AF_Property
<<block>> PS5_SecSurf : Shell_Surface_AF_Property
<<block>> LEITI_5 : Linear_Elastic_Orthotropic_TempInd_MB_Property

Figure 7.32: Analysis body system of PCB_5Sx_ThermoMech_Behavior_Model_Structure (B4)

Since B4 is a partially-specified instance model, the assembly system topology of the laminate shell system is fixed but the actually sizes and shapes of each shell, and their material behavior property values are not defined.

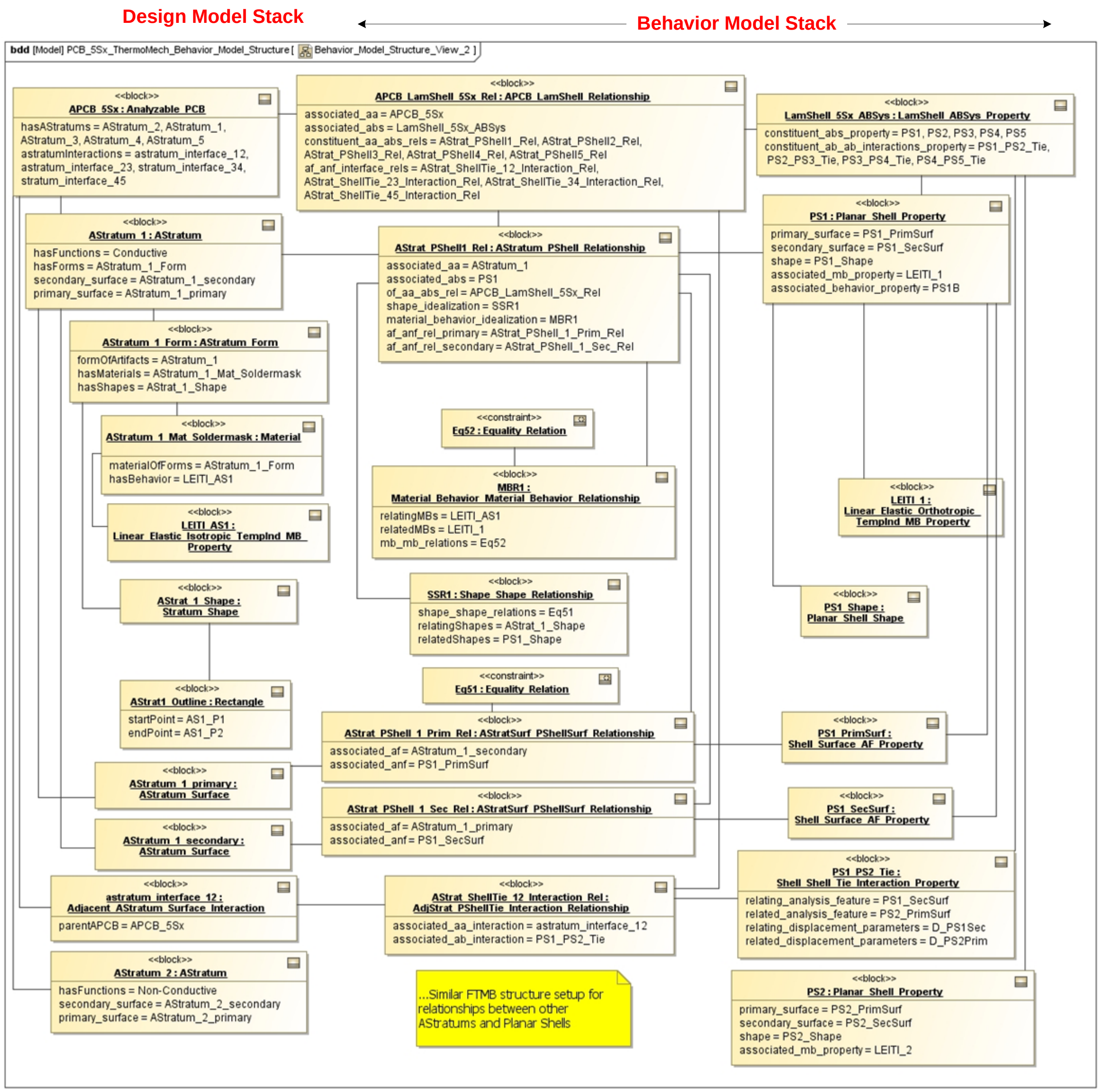

Design Model Stack
Behavior Model Stack
bdd [Model] PCB_5Sx_ThermoMech_Behavior_Model_Structure [Behavior_Model_Structure_View_2]
<<block>> APCB_5Sx : Analyzable_PCB
hasAStratums = AStratum_2, AStratum_1, AStratum_3, AStratum_4, AStratum_5
astratumInteractions = astratum_interface_12, astratum_interface_23, stratum_interface_34, stratum_interface_45
<<block>> APCB_LamShell_5Sx_Rel : APCB_LamShell_Relationship
associated_aa = APCB_5Sx
associated_abs = LamShell_5Sx_ABSys
constituent_aa_abs_rels = AStrat_PShell1_Rel, AStrat_PShell2_Rel, AStrat_PShell3_Rel, AStrat_PShell4_Rel, AStrat_PShell5_Rel
af_anf_interface_rels = AStrat_ShellTie_12_Interaction_Rel, AStrat_ShellTie_23_Interaction_Rel, AStrat_ShellTie_34_Interaction_Rel, AStrat_ShellTie_45_Interaction_Rel
<<block>> LamShell_5Sx_ABSys : LamShell_ABSys_Property
constituent_abs_property = PS1, PS2, PS3, PS4, PS5
constituent_ab_ab_interactions_property = PS1_PS2_Tie, PS2_PS3_Tie, PS3_PS4_Tie, PS4_PS5_Tie
<<block>> AStratum_1 : AStratum
hasFunctions = Conductive
hasForms = AStratum_1_Form
secondary_surface = AStratum_1_secondary
primary_surface = AStratum_1_primary
<<block>> AStrat_PShell1_Rel : AStratum_PShell_Relationship
associated_aa = AStratum_1
associated_abs = PS1
of_aa_abs_rel = APCB_LamShell_5Sx_Rel
shape_idealization = SSR1
material_behavior_idealization = MBR1
af_anf_rel_primary = AStrat_PShell_1_Prim_Rel
af_anf_rel_secondary = AStrat_PShell_1_Sec_Rel
<<block>> PS1 : Planar_Shell_Property
primary_surface = PS1_PrimSurf
secondary_surface = PS1_SecSurf
shape = PS1_Shape
associated_mb_property = LEITI_1
associated_behavior_property = PS1B
<<block>> AStratum_1_Form : AStratum_Form
formOfArtifacts = AStratum_1
hasMaterials = AStratum_1_Mat_Soldermask
hasShapes = AStrat_1_Shape
<<block>> AStratum_1_Mat_Soldermask : Material
materialOfForms = AStratum_1_Form
hasBehavior = LEITI_AS1
<<constraint>> Eq52 : Equality_Relation
<<block>> MBR1 : Material_Behavior_Material_Behavior_Relationship
relatingMBs = LEITI_AS1
relatedMBs = LEITI_1
mb_mb_relations = Eq52
<<block>> LEITI_1 : Linear_Elastic_Orthotropic_TempInd_MB_Property
<<block>> LEITI_AS1 : Linear_Elastic_Isotropic_TempInd_MB_Property
<<block>> AStrat_1_Shape : Stratum_Shape
<<block>> SSR1 : Shape_Shape_Relationship
shape_shape_relations = Eq51
relatingShapes = AStrat_1_Shape
relatedShapes = PS1_Shape
<<block>> PS1_Shape : Planar_Shell_Shape
<<block>> AStrat1_Outline : Rectangle
startPoint = AS1_P1
endPoint = AS1_P2
<<constraint>> Eq51 : Equality_Relation
<<block>> AStrat_PShell_1_Prim_Rel : AStratSurf_PShellSurf_Relationship
associated_af = AStratum_1_secondary
associated_anf = PS1_PrimSurf
<<block>> PS1_PrimSurf : Shell_Surface_AF_Property
<<block>> AStratum_1_primary : AStratum_Surface
<<block>> AStrat_PShell_1_Sec_Rel : AStratSurf_PShellSurf_Relationship
associated_af = AStratum_1_primary
associated_anf = PS1_SecSurf
<<block>> PS1_SecSurf : Shell_Surface_AF_Property
<<block>> AStratum_1_secondary : AStratum_Surface
<<block>> PS1_PS2_Tie : Shell_Shell_Tie_Interaction_Property
relating_analysis_feature = PS1_SecSurf
related_analysis_feature = PS2_PrimSurf
relating_displacement_parameters = D_PS1Sec
related_displacement_parameters = D_PS2Prim
<<block>> astratum_interface_12 : Adjacent_AStratum_Surface_Interaction
parentAPCB = APCB_5Sx
<<block>> AStrat_ShellTie_12_Interaction_Rel : AdjStrat_PShellTie_Interaction_Relationship
associated_aa_interaction = astratum_interface_12
associated_ab_interaction = PS1_PS2_Tie
<<block>> AStratum_2 : AStratum
hasFunctions = Non-Conductive
secondary_surface = AStratum_2_secondary
primary_surface = AStratum_2_primary
...Similar FTMB structure setup for relationships between other AStratums and Planar Shells
<<block>> PS2 : Planar_Shell_Property
primary_surface = PS2_PrimSurf
secondary_surface = PS2_SecSurf
shape = PS2_Shape
associated_mb_property = LEITI_2

Figure 7.33: Example D4-B4 model showing relationships between 5-stratum analyzable PCBs (D4) and corresponding 5-layered laminated shell systems (B4)

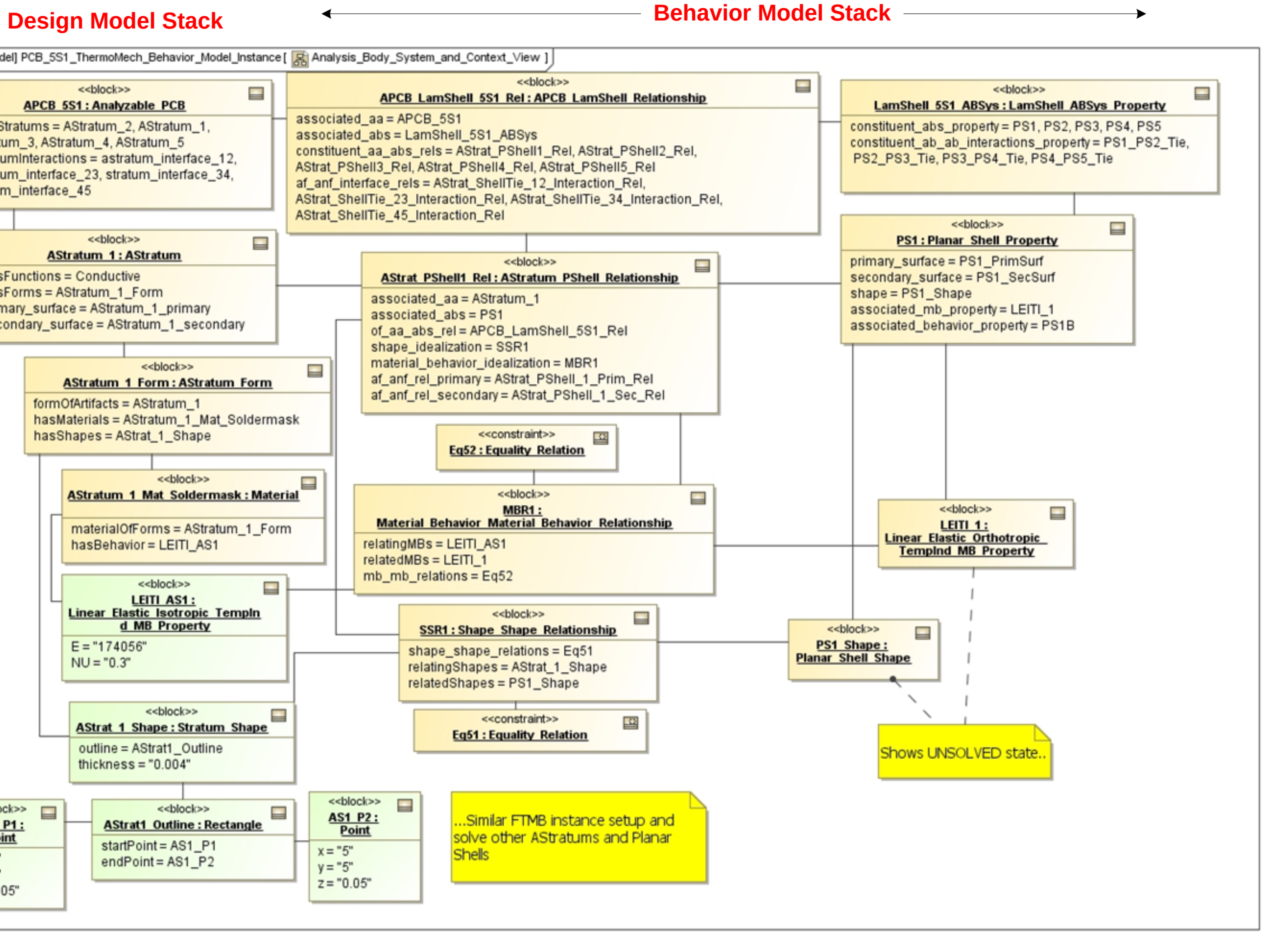
Figure 7.33 above shows the 5-layered laminate shell system and the context model that associates this laminate shell system to the 5-stratum analyzable PCB model structure. Behavior idealization relationship for shown only for stratum 1 as the structure repeats itself for other stratums. The central entity in the context model is the APCB_LamShell_5Sx_Context block (Figure 7.31). The context model refers to the behavior idealization relationship between the 5-stratum analyzable PCB and the 5-layered laminate shell system (APCB_LamShell_5Sx_Context.aa_abs_rel). As shown in Figure 7.33, this idealization represented is represented by the APCB_LamShell_5Sx_Rel block that relates the APCB_5Sx block (5-stratum analyzable PCB) and LamShell_5Sx_ABSys block (5-layered laminate shell system). The idealization relationship between the 5-stratum analyzable PCB and 5-layered laminate shell system is composed of: (i) idealization relationship between each stratum and shell (represented by blocks AStrat_PShell1_Rel, AStrat_PShell2_Rel,...), and (ii) the idealization relationships between the stratum interfaces and the shell interactions (represented by blocks AStrat_ShellTie_12_Interaction_Rel, AStrat_ShellTie_23_Interaction_Rel, …).

The idealization relationship between an analyzable PCB stratum and a planar shell in the laminate shell system refers to shape idealization relationships and material behavior idealization relationships. These relationships represent how the shapes and material behaviors respectively of the stratum and the shell are related—including the mathematical relationships between the shape parameters and the material behavior parameters. For the first stratum in the analyzable PCB and first planar shell in the laminate shell system, SSR1 and MBR1 are the shape idealization and material behavior idealization relationships respectively. Note that the mathematical relations associated with these idealization relationships indicate that the shape properties and material behavior properties of the stratum and shell are equal. The idealization relationship between an analyzable PCB stratum and a planar shell is also composed of the idealization relationships between their features. The primary and the secondary surfaces of the stratum are related to the primary and secondary surfaces of the corresponding shell by the blocks AStrat_PShell1_Prim_Rel and AStrat_PShell1_Sec_Rel.

- **Level 5 (B5):** PCB_5S1_ThermoMech_Behavior_Model_Instance

Figure 7.34 below illustrates PCB_5S1_ThermoMech_Behavior_Model_Instance—a thermo-mechanical behavior model for a specific 5-stratum PCB.. This behavior model is a fully-specified instance of PCB_nSx_ThermoMech_Behavior_Meta-Model (B3 model). In contrast to the B4 models all property values are fully-populated in B5 models, including the material behavior properties and the shapes of the analyzable PCB stratums.

The shape and material behavior idealization relationships between the stratums and the shells can then be solved using math solvers. Figure 7.34 shows the shape and material behavior properties for stratum 1 of the analyzable PCB_5S1 (in a D5 model). The values of the corresponding shape and material behavior properties for shell as shown in the figure are not computed.


Design Model Stack
Behavior Model Stack
bdd [Model] PCB_5S1_ThermoMech_Behavior_Model_Instance [Analysis_Body_System_and_Context_View]
Shows UNSOLVED state..
...Similar FTMB instance setup and solve other AStratums and Planar Shells

<<block>> APCB_5S1 : Analyzable_PCB
hasAStratums = AStratum_2, AStratum_1, AStratum_3, AStratum_4, AStratum_5
astratumInteractions = astratum_interface_12, astratum_interface_23, stratum_interface_34, stratum_interface_45

<<block>> APCB_LamShell_5S1_Rel : APCB_LamShell_Relationship
associated_aa = APCB_5S1
associated_abs = LamShell_5S1_ABSys
constituent_aa_abs_rels = AStrat_PShell1_Rel, AStrat_PShell2_Rel, AStrat_PShell3_Rel, AStrat_PShell4_Rel, AStrat_PShell5_Rel
af_anf_interface_rels = AStrat_ShellTie_12_Interaction_Rel, AStrat_ShellTie_23_Interaction_Rel, AStrat_ShellTie_34_Interaction_Rel, AStrat_ShellTie_45_Interaction_Rel

<<block>> LamShell_5S1_ABSys : LamShell_ABSys_Property
constituent_abs_property = PS1, PS2, PS3, PS4, PS5
constituent_ab_ab_interactions_property = PS1_PS2_Tie, PS2_PS3_Tie, PS3_PS4_Tie, PS4_PS5_Tie

<<block>> AStratum_1 : AStratum
hasFunctions = Conductive
hasForms = AStratum_1_Form
primary_surface = AStratum_1_primary
secondary_surface = AStratum_1_secondary

<<block>> AStrat_PShell1_Rel : AStratum_PShell_Relationship
associated_aa = AStratum_1
associated_abs = PS1
of_aa_abs_rel = APCB_LamShell_5S1_Rel
shape_idealization = SSR1
material_behavior_idealization = MBR1
af_anf_rel_primary = AStrat_PShell_1_Prim_Rel
af_anf_rel_secondary = AStrat_PShell_1_Sec_Rel

<<block>> PS1 : Planar_Shell_Property
primary_surface = PS1_PrimSurf
secondary_surface = PS1_SecSurf
shape = PS1_Shape
associated_mb_property = LEITI_1
associated_behavior_property = PS1B

<<block>> AStratum_1_Form : AStratum_Form
formOfArtifacts = AStratum_1
hasMaterials = AStratum_1_Mat_Soldermask
hasShapes = AStrat_1_Shape

<<constraint>> Eq52 : Equality_Relation

<<block>> AStratum_1_Mat_Soldermask : Material
materialOfForms = AStratum_1_Form
hasBehavior = LEITI_AS1

<<block>> MBR1 : Material_Behavior_Material_Behavior_Relationship
relatingMBs = LEITI_AS1
relatedMBs = LEITI_1
mb_mb_relations = Eq52

<<block>> LEITI_1 : Linear_Elastic_Orthotropic_TempInd_MB_Property

<<block>> LEITI_AS1 : Linear_Elastic_Isotropic_TempInd_MB_Property
E = "174056"
NU = "0.3"

<<block>> SSR1 : Shape_Shape_Relationship
shape_shape_relations = Eq51
relatingShapes = AStrat_1_Shape
relatedShapes = PS1_Shape

<<block>> PS1_Shape : Planar_Shell_Shape

<<block>> AStrat_1_Shape : Stratum_Shape
outline = AStrat1_Outline
thickness = "0.004"

<<constraint>> Eq51 : Equality_Relation

<<block>> AS1_P1 : Point
x = "0"
y = "0"
z = "0.05"

<<block>> AStrat1_Outline : Rectangle
startPoint = AS1_P1
endPoint = AS1_P2

<<block>> AS1_P2 : Point
x = "5"
y = "5"
z = "0.05"

Figure 7.34: Example D5-B5 model showing relationships between a specific 5-stratum analyzable PCB (D5) and corresponding 5-layered laminated shell system (B5 model in unsolved state)

7.5 Analysis Knowledge Dimensions

Analysis knowledge dimensions provide the basic foundation for defining the Core Behavior Model and the ABB Meta-Model. These constructs and the relationships defined in these meta-models were based on representing the types of decisions that analysts take in defining reusable building blocks of domain-theoretic concepts and composing behavior model structure based on these concepts. Analysis knowledge dimensions provide a conceptual organization of the types of these decisions and the choices available. The Analysis Knowledge Dimension Model presented in this dissertation is a conceptual model that is not instantiated itself but was used to define the Core Behavior Model and the ABB Meta-Model that may be specialized and instantiated.

In this section, the Analysis Knowledge Dimension model is presented using a set of SysML block definition diagrams. There are two types of constructs in this conceptual model: (a) constructs representing types of decision taken by analysts in creating behavior models, and (b) constructs representing choices available for these decisions. The former type of constructs is denoted as a 'dimension' or 'sub-dimension' (for sub-decisions). Both the constructs are presented as SysML blocks. There are two types of relationships defined among constructs:

- The composition relationship (line ending in a black diamond) denotes the composition of higher-level decisions into sub-decisions, and it is drawn between constructs representing dimensions.
- The generalization relationship (line ending in an arrow) denotes the choices available for a particular type of decision, and it is drawn between constructs representing choices to the construct representing the corresponding decision.

The central entity in Analysis Knowledge Dimension model is the Analysis_Knowledge_Dimensions block and it represents the collective decisions that need to be taken by analysts to create a behavior model structure. As shown in Figure 7.35, such a collective decision is decomposed into four dimensions (set of decisions), namely:

- Behavior dimension (represented by Behavior_Dimension block)
- Analysis body dimension (represented by Analysis_Body_Dimension block)
- Load dimension (represented by Load_Dimension block)
- Behavior condition dimension (represented by Behavior_Condition_Dimension block)

Note that the layout of the SysML diagrams presented in this section is circular and not hierarchical. The top-level concept is positioned in the middle of the diagram and related concepts are arranged around the top-level concept.

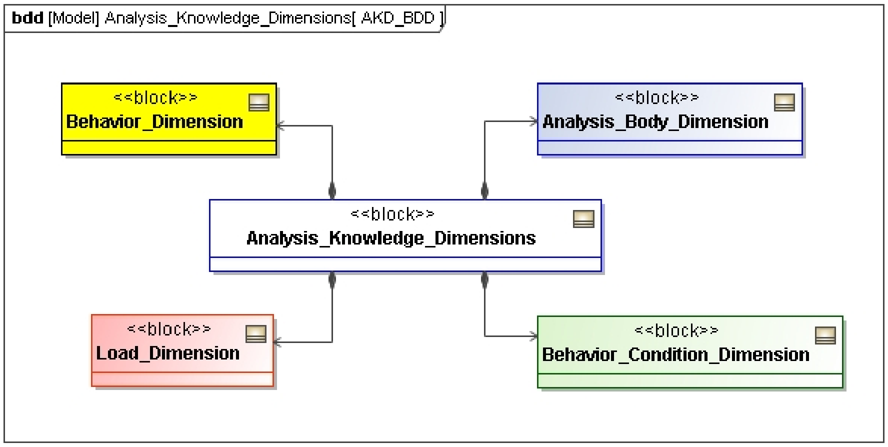

bdd [Model] Analysis_Knowledge_Dimensions[AKD_BDD]
<<block>>
Behavior_Dimension
<<block>>
Analysis_Body_Dimension
<<block>>
Analysis_Knowledge_Dimensions
<<block>>
Load_Dimension
<<block>>
Behavior_Condition_Dimension

Figure 7.35: Analysis Knowledge Dimension Model – top-level view

Blocks representing the four major dimensions have names ending in 'Dimension', and the blocks representing the sub-dimensions under each of the four major dimensions have names ending in 'Dim'. *The basis for abstracting these four dimensions lies in the assumption that the behavior of an artifact in a given environment is a function of the artifact's form, and the loads and behavior conditions to which an artifact is subjected in that environment*. Note that when formulating behavior models, the artifact is represented in its idealized form as an analysis body or analysis body system. Before describing the complete structure of each of these four dimensions, it is necessary to describe the semantic properties of the Analysis Knowledge Dimension Model. These are:

1. Decisions need to be taken on all dimensions and their sub-dimensions, either directly or indirectly, to create a complete behavior model structure. A complete behavior model structure is one which when instantiated is solvable.
2. Analysis knowledge dimensions and their sub-dimensions may not be mutually exclusive, and a decision taken on a particular sub-dimension may influence or constrain a decision on other sub-dimension(s).
3. There may not be a sequence in which decisions are taken along particular dimensions.
4. Decisions may be mutually consistent (or inconsistent), and/or redundant
5. The choices presented here for the dimensions are primarily primitive level choices. More choices may be defined by creating choices that are composed of one or more choices.

6. Choices available for a decision may be mutually exclusive. In the SysML block definition diagrams presented here, these choices are represented by blocks with italicized names.

These properties reflect the inherent nature of analysis problem formulation process in that there are several ways of creating a behavior model structure, and a valid and complete behavior model is one for which the set of idealization decisions were mutually complete, consistent and preferably non-redundant.

Note that the analysis knowledge dimensions presented here are extensible. The types of decisions and the choices for each decision type presented help to illustrate the conceptual model and in no way represent a fully exhaustive set of choices for all types of behavior models.

7.5.1 Behavior Dimension

The Behavior dimension is meant for categorizing decisions pertaining to the overall behavior of an artifact. It is represented by the Behavior_Dimension block and includes six sub-dimensions, as shown by the composition relationships in Figure 7.36. These sub-dimensions are as follows:

- Behavior_Type_Dim**:** This sub-dimension is used to categorize different types of idealized behaviors of an artifact As shown in Figure 7.36, the choices for this decision are categorized based on analysis disciplines, such as tension, torsion, vibration, and buckling for structural behavior; and conduction, convection, and radiation for thermal behavior. Choices that represent composite behaviors (such as bending and torsion) may be defined by creating blocks that specialize one or more blocks.

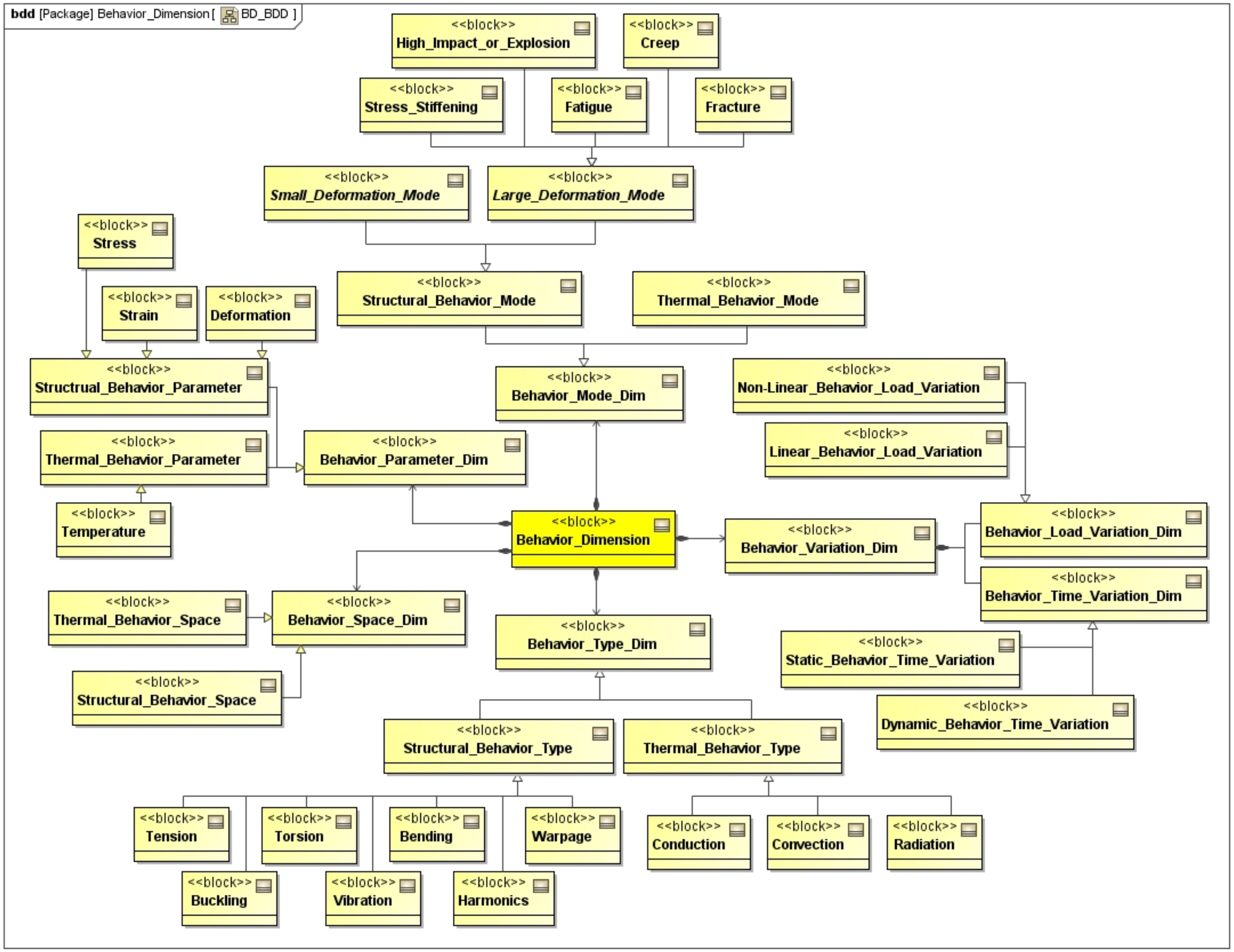

bdd [Package] Behavior_Dimension [BD_BDD]
<<block>> High_Impact_or_Explosion
<<block>> Creep
<<block>> Stress_Stiffening
<<block>> Fatigue
<<block>> Fracture
<<block>> Small_Deformation_Mode
<<block>> Large_Deformation_Mode
<<block>> Stress
<<block>> Strain
<<block>> Deformation
<<block>> Structural_Behavior_Mode
<<block>> Thermal_Behavior_Mode
<<block>> Structrual_Behavior_Parameter
<<block>> Behavior_Mode_Dim
<<block>> Non-Linear_Behavior_Load_Variation
<<block>> Linear_Behavior_Load_Variation
<<block>> Thermal_Behavior_Parameter
<<block>> Behavior_Parameter_Dim
<<block>> Temperature
<<block>> Behavior_Dimension
<<block>> Behavior_Variation_Dim
<<block>> Behavior_Load_Variation_Dim
<<block>> Behavior_Time_Variation_Dim
<<block>> Thermal_Behavior_Space
<<block>> Behavior_Space_Dim
<<block>> Behavior_Type_Dim
<<block>> Static_Behavior_Time_Variation
<<block>> Structural_Behavior_Space
<<block>> Dynamic_Behavior_Time_Variation
<<block>> Structural_Behavior_Type
<<block>> Thermal_Behavior_Type
<<block>> Tension
<<block>> Torsion
<<block>> Bending
<<block>> Warpage
<<block>> Conduction
<<block>> Convection
<<block>> Radiation
<<block>> Buckling
<<block>> Vibration
<<block>> Harmonics

Figure 7.36: Behavior Dimension

- Behavior_Mode_Dim: This sub-dimension is used to categorize different idealized behavior modes of an artifact for a given type of behavior. For example, depending on the magnitude of the load, an artifact's torsional behavior may be idealized as one resulting in small deformation or large deformation. Here, small deformation and large deformation are different modes of torsional behavior. Note that large deformation and small deformation are mutually exclusive choices (denoted with italicized block names)

- Behavior_Variation_Dim: This sub-dimension is used to categorize how the variation in the behavior of the artifact may be idealized, such as static or dynamic, linear or non-linear. In Figure 7.36, two main specializations of this sub-dimension are shown, namely (a) Linear_or_Non-Linear to specify if the response of an artifact to loads is idealized linear or non-linear, and (b) Static_or_Dynamic to specify if the response of an artifact to loads with respect to time is idealized as static or dynamic. Here, the response of an artifact is measured in terms of a behavior parameter, such as deformation or temperature. Other sub-dimensions may be added for relating the behavior of an artifact to other parameters apart from load and time.

- Behavior_Space_Dim: This sub-dimension is used to categorize the geometric space of an idealized artifact. There are 2 ways of measuring this: (a) the geometric space occupied by the idealized artifact, and (b) number of independent spatial variables in the analysis problem. As an example for a beam bending under transverse loads, the number of independent spatial parameters is one (distance measured along the axis of the beam). The transverse deflection is dependent on this distance. However, the geometric space occupied by a deformed beam is 2D. One needs the both the distance along the axis of the beam and the transverse deflection to describe the deformed shape of a beam. Traditionally, it is the former criterion that is used for characterizing behavior analysis problems as 1D, 2D, or 3D.

- Behavior_Parameter_Dim: This sub-dimension is used to categorize parameters used for measuring the idealized behaviors of an artifact, such as temperature, deformation, stress, and strain. The choices available for this dimension are organized based on analysis disciplines. So, there are choices available for structural behavior parameters, and thermal behavior parameters, and so on.

7.5.2 Analysis Body Dimension

The analysis body dimension is used for categorizing decisions pertaining to analysis bodies and analysis body systems. When creating behavior models, analyzable artifacts are idealized as analysis bodies (or analysis body systems) as described in section Figure 7.2. The analysis body dimension has 4 sub-dimensions, as illustrated in Figure 7.37 and described below.

- Analysis_Body_System_Composition_Dim: This sub-dimension is used to categorize decisions pertaining to the composition (part-assembly structure) of an analysis body system. The choices available for this sub-dimension are correspond to different types of single body and multi-body systems.
- Material_Behavior_Dim: This sub-dimension is used to categorize decisions pertaining to the constitutive material behavior of an analysis body. This sub-dimension is composed of three sub-sub-dimensions:
    - Material_Behavior_Type_Dim: This sub-dimension is used to categorize decisions pertaining to the response of the material to applied loads. The choices are categorized based on the type of load, such as such as elastic, plastic, and for structural loads.
    - Material_Behavior_Distribution_Dim: This sub-dimension is used for categorizing decisions pertaining to homogeneity or types of non-homogeneities of the material, such as isotropic, transversely isotropic, orthotropic, general anisotropic, etc.
    - Material_Behavior_Variation_Dim: This is used for categorizing decisions related to quantifying the idealized variation of material response to a load or deformation, loading rate or deformation rate, and temperature. These three criteria are represented by the following sub-dimensions as also shown in Figure 7.37: (a) Stress_Strain_Co-Variation_Dim for effects of loading / deformation, (b) Strain_Rate_Based_Variation_Dim for effects of strain rate, and (c) Temperature_Based_Variation_Dim for effects of temperature change. Other categories of material behavior variation may be added to this decision classification.

- Material_Behavior_Parameters_Dim is used to categorize parameters used for quantifying material behavior. The choices are described based on the analysis disciplines implying that different parameters are used for quantifying material behavior for different analysis discipline.

- Analysis_Body_Type_Dim: This sub-dimension is used for categorizing decisions pertaining to the type of an analysis body. The type is characterized based on the (a) the idealized behaviors that an analysis body exhibits, (b) geometric space used for representing the shape of an analysis body, and (c) the number and type of degrees-of-freedom associated with an analysis body. These criteria are represented by the following three sub-dimensions: (a) Analysis_Body_Discipline_Type, (b) Analysis_Body_Space_Type, and (c) Analysis_Body_DOF.
- Analysis_Body_Interaction_Behavior_Dim**:** This sub-dimension is used for categorizing decisions pertaining to the behavior of the interaction between analysis bodies in an analysis body system. Since the decision pertains to behavior, it is similar in nature to the types of decisions presented under the behavior dimension.

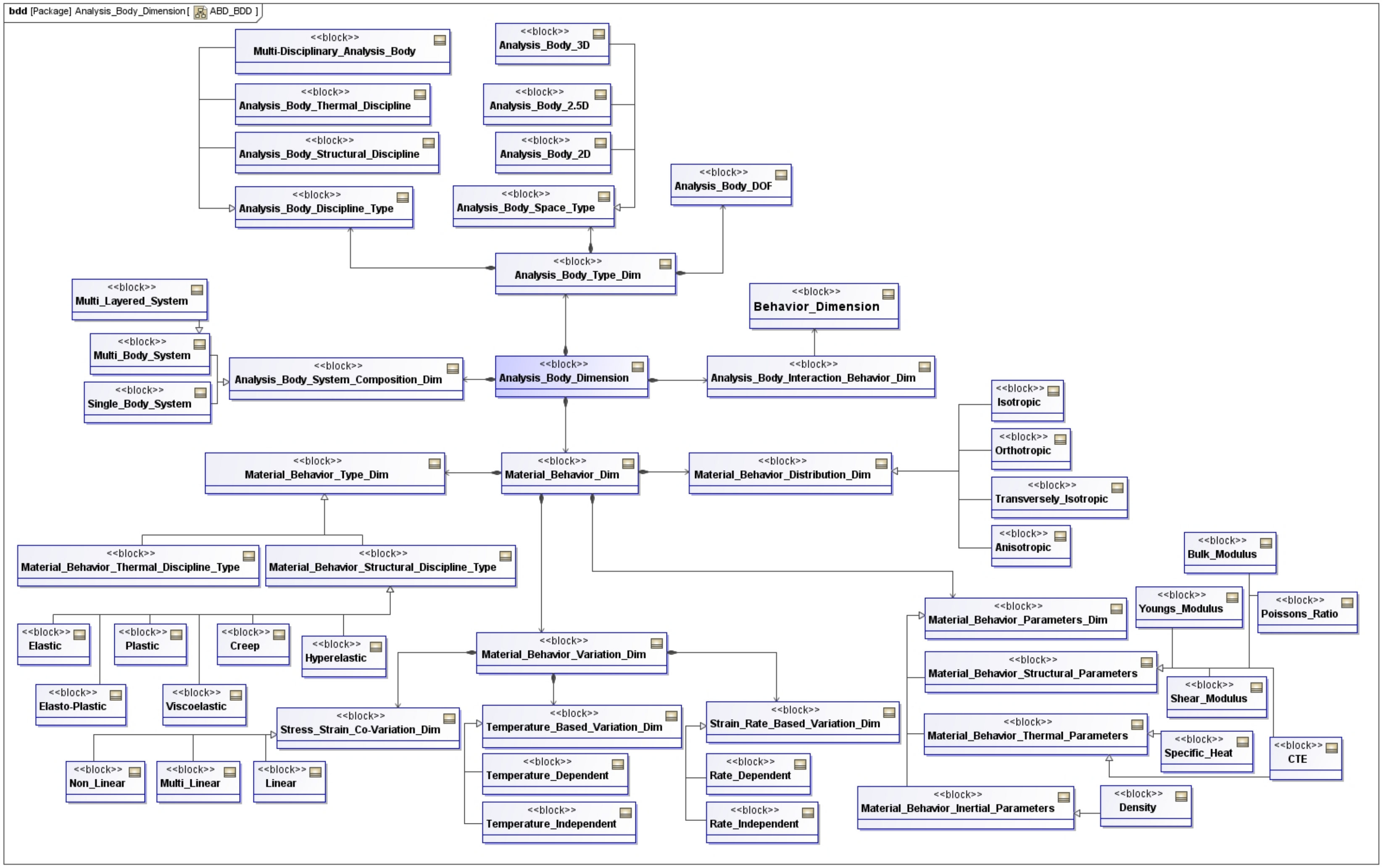

bdd [Package] Analysis_Body_Dimension [ABD_BDD]
<<block>> Multi-Disciplinary_Analysis_Body
<<block>> Analysis_Body_Thermal_Discipline
<<block>> Analysis_Body_Structural_Discipline
<<block>> Analysis_Body_Discipline_Type
<<block>> Analysis_Body_3D
<<block>> Analysis_Body_2.5D
<<block>> Analysis_Body_2D
<<block>> Analysis_Body_Space_Type
<<block>> Analysis_Body_DOF
<<block>> Analysis_Body_Type_Dim
<<block>> Multi_Layered_System
<<block>> Multi_Body_System
<<block>> Single_Body_System
<<block>> Analysis_Body_System_Composition_Dim
<<block>> Analysis_Body_Dimension
<<block>> Behavior_Dimension
<<block>> Analysis_Body_Interaction_Behavior_Dim
<<block>> Isotropic
<<block>> Orthotropic
<<block>> Transversely_Isotropic
<<block>> Anisotropic
<<block>> Material_Behavior_Type_Dim
<<block>> Material_Behavior_Dim
<<block>> Material_Behavior_Distribution_Dim
<<block>> Material_Behavior_Thermal_Discipline_Type
<<block>> Material_Behavior_Structural_Discipline_Type
<<block>> Elastic
<<block>> Plastic
<<block>> Creep
<<block>> Hyperelastic
<<block>> Elasto-Plastic
<<block>> Viscoelastic
<<block>> Material_Behavior_Variation_Dim
<<block>> Stress_Strain_Co-Variation_Dim
<<block>> Temperature_Based_Variation_Dim
<<block>> Strain_Rate_Based_Variation_Dim
<<block>> Non_Linear
<<block>> Multi_Linear
<<block>> Linear
<<block>> Temperature_Dependent
<<block>> Temperature_Independent
<<block>> Rate_Dependent
<<block>> Rate_Independent
<<block>> Bulk_Modulus
<<block>> Youngs_Modulus
<<block>> Poissons_Ratio
<<block>> Material_Behavior_Parameters_Dim
<<block>> Material_Behavior_Structural_Parameters
<<block>> Shear_Modulus
<<block>> Material_Behavior_Thermal_Parameters
<<block>> Specific_Heat
<<block>> CTE
<<block>> Material_Behavior_Inertial_Parameters
<<block>> Density

Figure 7.37: Analysis Body Dimension

7.5.3 Load Dimension

The load dimension is used for categorizing decisions pertaining to the applied load(s). The load dimension consists of four sub-dimensions, as illustrated in Figure 7.38 and described below:

- Load_Type_Dim: This sub-dimension is used to categorize decisions based on the type of loads. The choices are organized in terms of analysis disciplines. For example, pressure is a structural load while heat generation rate is a thermal load.
- Load_Application_Dim: This sub-dimension is used to categorize decisions concerning the application of load to an analysis body (or analysis body system). This decision is composed of the following two sub-decisions: (a) the application space of the load, such as whether the load is applied to an analysis feature (geometric space) or to an inertial mass, and (b) the direction of the load. These decisions are represented by the Load_Application_Domain_Dim and Load_Application_Direction_Dim respectively.
- Load_Variation_Dim: This sub-dimension is used to categorize decisions pertaining to the variation of loads over space and time, represented by the two sub-dimensions Load_Space_Variation and Load_Time_Variation respectively.
- Load_Parameter_Dim: This sub-dimension is used to categorize parameters for quantifying loads. The choices are organized on the basis of analysis disciplines.

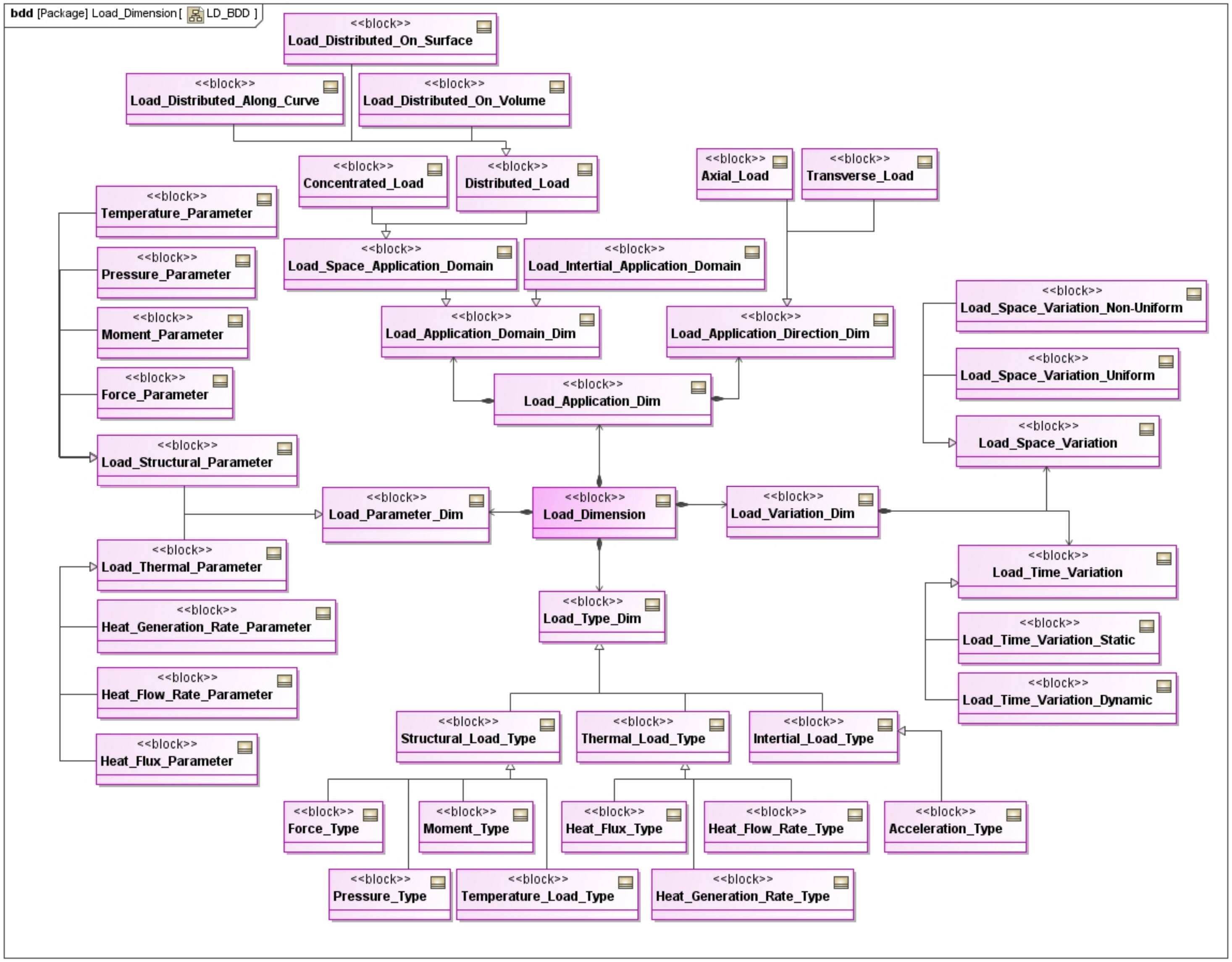

bdd [Package] Load_Dimension [LD_BDD]
<<block>> Load_Distributed_On_Surface
<<block>> Load_Distributed_Along_Curve
<<block>> Load_Distributed_On_Volume
<<block>> Concentrated_Load
<<block>> Distributed_Load
<<block>> Axial_Load
<<block>> Transverse_Load
<<block>> Temperature_Parameter
<<block>> Pressure_Parameter
<<block>> Moment_Parameter
<<block>> Force_Parameter
<<block>> Load_Structural_Parameter
<<block>> Load_Space_Application_Domain
<<block>> Load_Intertial_Application_Domain
<<block>> Load_Application_Domain_Dim
<<block>> Load_Application_Direction_Dim
<<block>> Load_Application_Dim
<<block>> Load_Space_Variation_Non-Uniform
<<block>> Load_Space_Variation_Uniform
<<block>> Load_Space_Variation
<<block>> Load_Parameter_Dim
<<block>> Load_Dimension
<<block>> Load_Variation_Dim
<<block>> Load_Thermal_Parameter
<<block>> Heat_Generation_Rate_Parameter
<<block>> Heat_Flow_Rate_Parameter
<<block>> Heat_Flux_Parameter
<<block>> Load_Type_Dim
<<block>> Load_Time_Variation
<<block>> Load_Time_Variation_Static
<<block>> Load_Time_Variation_Dynamic
<<block>> Structural_Load_Type
<<block>> Thermal_Load_Type
<<block>> Intertial_Load_Type
<<block>> Force_Type
<<block>> Moment_Type
<<block>> Heat_Flux_Type
<<block>> Heat_Flow_Rate_Type
<<block>> Acceleration_Type
<<block>> Pressure_Type
<<block>> Temperature_Load_Type
<<block>> Heat_Generation_Rate_Type

Figure 7.38: Load Dimension

7.5.4 Behavior Condition Dimension

This behavior condition dimension is used for categorizing decisions pertaining to behavior conditions in which the behavior of an analysis body system (idealized artifact) is to be computed. This dimension consists of the following five sub-dimensions:

- Behavior_Condition_Discipline_Dim: This sub-dimension is used for categorizing the analysis discipline associated with the behavior condition. Behavior conditions are described in terms of behavior parameters and thus the analysis discipline is decided based on the discipline associated with the behavior parameters. For example, behavior a behavior condition described in terms of displacement (a type of structural behavior parameter) is a structural behavior condition.
- Behavior_Condition_Type_Dim: This sub-dimension is used for categorizing the types of behavior conditions. Two prominent choices are boundary value conditions for boundary value problems, and initial value conditions for initial value problems.
- Behavior_Condition_Application_Space_Dim: This sub-dimension is used for categorizing the geometric application space for behavior conditions, i.e. if the behavior condition is applied to a point analysis feature or a surface analysis feature.
- Behavior_Condition_Variation_Dim: This sub-dimension is used for categorizing the variation of behavior conditions with respect to space and time, represented by the following two sub-dimensions: Behavior_Condition_Space_Variation_Dim and Behavior_Condition_Time_Variation_Dim respectively.
- Behavior_Condition_Parameter_Dim: This sub-dimension is used for categorizing the different types of parameters for quantifying behavior conditions. The categorization is similar to that described in Behavior_Parameter_Dim.

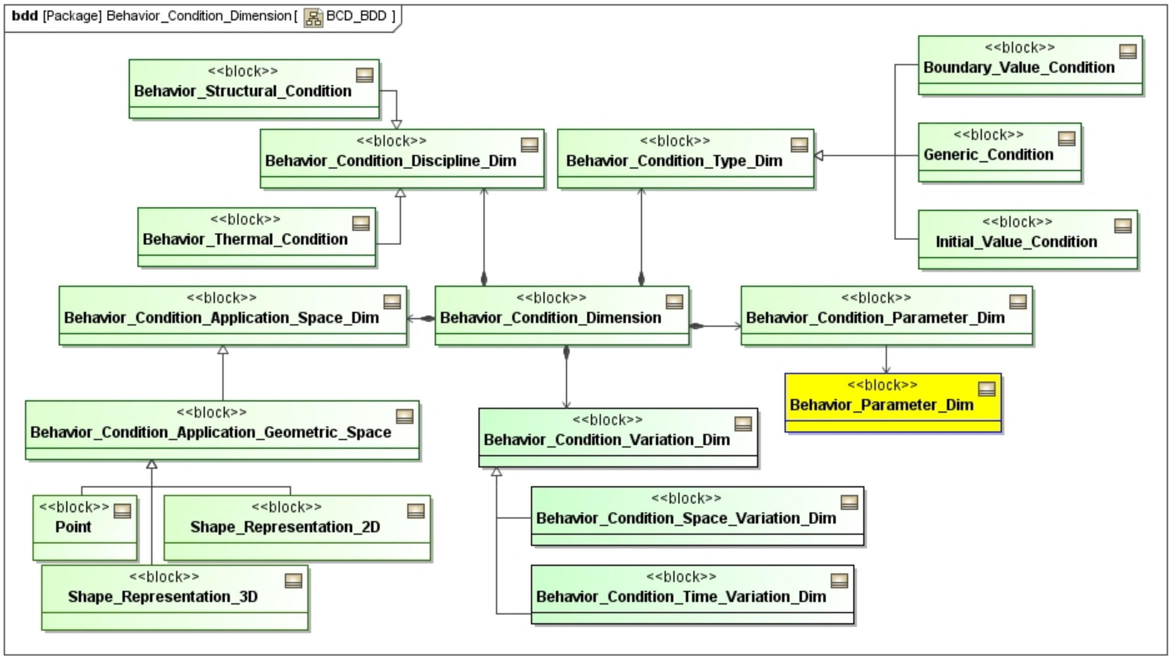

bdd [Package] Behavior_Condition_Dimension [BCD_BDD]
<<block>>
Behavior_Structural_Condition
<<block>>
Boundary_Value_Condition
<<block>>
Behavior_Condition_Discipline_Dim
<<block>>
Behavior_Condition_Type_Dim
<<block>>
Generic_Condition
<<block>>
Behavior_Thermal_Condition
<<block>>
Initial_Value_Condition
<<block>>
Behavior_Condition_Application_Space_Dim
<<block>>
Behavior_Condition_Dimension
<<block>>
Behavior_Condition_Parameter_Dim
<<block>>
Behavior_Parameter_Dim
<<block>>
Behavior_Condition_Application_Geometric_Space
<<block>>
Behavior_Condition_Variation_Dim
<<block>>
Point
<<block>>
Shape_Representation_2D
<<block>>
Behavior_Condition_Space_Variation_Dim
<<block>>
Shape_Representation_3D
<<block>>
Behavior_Condition_Time_Variation_Dim

Figure 7.39: Behavior Condition Dimension

7.6 Summary

In this chapter, the Core Behavior Model (CBM) is presented as a meta-model for representing behavior models of VTMB design alternatives. Five levels of abstractions of behavior models, based on the CBM, are also presented with examples. The ABB Meta-Model that defines the constructs for representing different types of ABBs is also presented in this chapter. The ABB Meta-Model prescribes four foundational aspects of knowledge that must be represented in an ABB. Two of these aspects are described in details with examples in this chapter. The other two aspects concern the transformations applied for composing ABBs, and are presented in the following chapter (Chapter 8). The Core Behavior Model and the ABB Meta-Model are founded on Analysis Knowledge Dimensions that are also presented in this chapter. The Analysis Knowledge Dimensions define the types of decisions taken by analysts in formulating behaviors models and the choices available for each type of decision.

Chapter 8 : BEHAVIOR MODEL FORMULATION METHOD

The focus of this chapter is to present the Behavior Model Formulation Method (BMFM). KCM's Behavior Model Formulation Method prescribes an approach for formulating behavior model structures given idealized design alternatives with varying assembly system topologies and behavior idealization specifications defined by analysts. In this chapter, the Behavior Model Formulation Method is described and its fundamental underpinnings in model transformations and analysis domain theories are presented. In section 8.1, an overview of the Behavior Model Formulation Method is presented, and in section 8.2 the model transformation process used for composing behavior model structures and simulation templates is described in details. The idealization decisions taken by analysts are formally represented as Behavior Model Formulation Specifications, and presented in section 8.3. In section 8.4, the Artifact Model Transformation Library—a library of reusable model transformation rules and patterns—is presented.

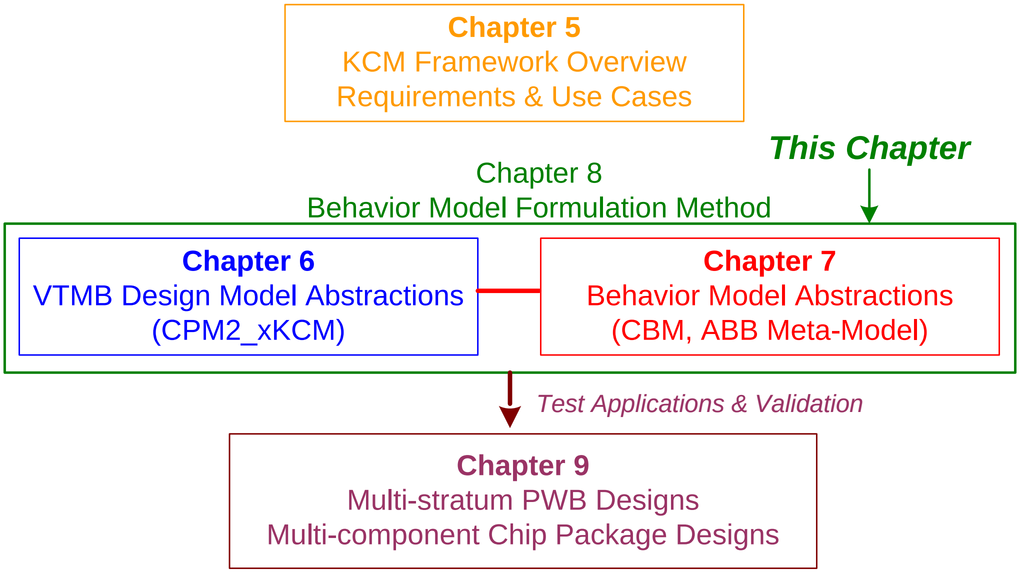

Chapter 5
KCM Framework Overview
Requirements & Use Cases
This Chapter
Chapter 8
Behavior Model Formulation Method
Chapter 6
VTMB Design Model Abstractions
(CPM2_xKCM)
Chapter 7
Behavior Model Abstractions
(CBM, ABB Meta-Model)
Test Applications & Validation
Chapter 9
Multi-stratum PWB Designs
Multi-component Chip Package Designs

Figure 8.1: Behavior Model Formulation Method – focus of this chapter

8.1 Overview

The Behavior Model Formulation Method (BMFM) prescribes a model transformation process for creating behavior model structures. By definition, a model transformation process transforms a source model that confirms to a source meta-model (or schema) to a target model that confirms to a target meta-model (or schema). As shown in the schematic of a model transformation in Figure 8.2 (Czarnecki and Helsen 2006), there are six key elements of a model transformation process:

- *Source meta-model* defines constructs and relationships for defining source models.
- *Target meta-model* defines constructs and relationships for defining target models.
- *Source model* is the input to the model transformation process, and conforms to the source meta-model.
- *Target model* is the output of the model transformation process, and conforms to the target meta-model.
- *Transformation definition* formally states the model transformation process. A transformation definition mainly states: (a) types of entities and relationships in the source model that are of concern or to be transformed to create a target model, (b) transformations that will be executed on these types of source model entities and relationships, and (c) order of execution of transformations. Since a transformation definition is stated in terms of the *types* of entities and relationships, it refers to the source and target meta-models that define these types.
- *Transformation engine* is the software that executes the transformation definition on the source model to create a target model.

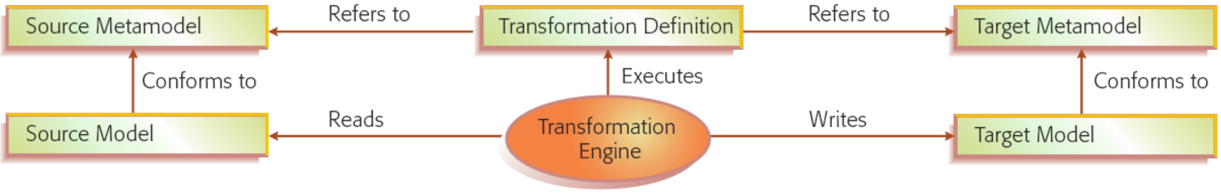


Figure 8.2: Schematic of a model transformation (Czarnecki and Helsen 2006)

Figure 8.3 illustrates the schematics of the model transformation process realized by the BMFM resulting in the automated creation of behavior model structures and simulation templates. Typically model transformations are achieved by creating a target model different from the source model, or by changing the source model itself (in-place transformation). The model transformation process prescribed by the BMFM is an in-place transformation where the source model is not modified but instead additional models are composed and related to the source model. Thus the target model is composed of the source model and new models. This is so because the target model of the BMFM's model transformation process is a simulation template that relates an artifact's design model structure and its behavior model structure; the source model of the BMFM's model transformation process is an artifact's design model structure. In essence, the model transformation process prescribed by BMFM is a model building process. During the model transformation process, an artifact's behavior model structure is created and related to the design model structure, thereby creating a simulation template. While Figure 8.3 illustrates the source and target meta-models and models in the context of BMFM's model transformation process, Figure 8.5 illustrates the source and target meta-models and models in the design and behavior model stack (sections 6.2 and 7.4).

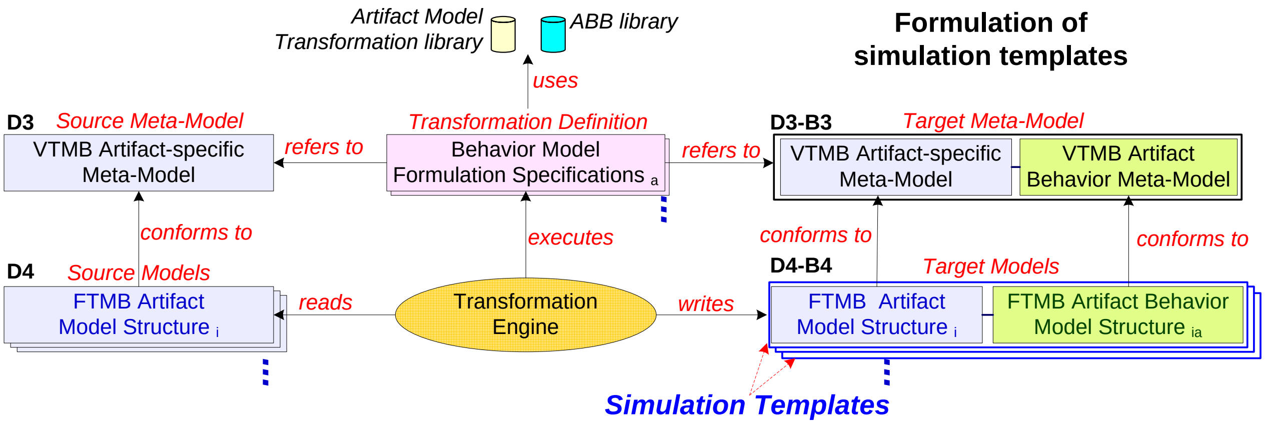


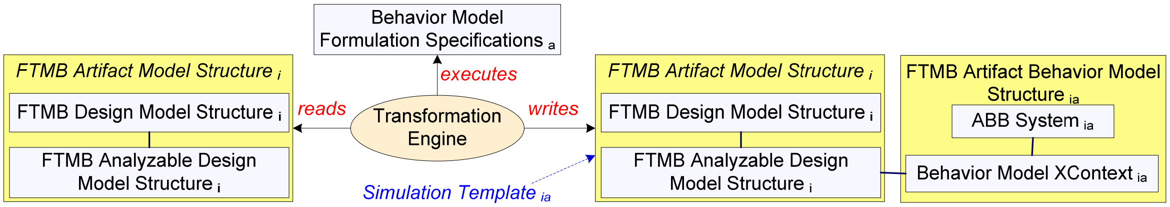
Figure 8.3: Schematic of KCM's Behavior Model Formulation Method (BMFM)

The six key elements of BMFM's model transformation process are as follows:

- *Source meta-model*: The source meta-model of BMFM's model transformation process is a VTMB Artifact-specific Meta-Model—D3 model in the design model stack. As shown in Figure 8.5, this meta-model defines the constructs and relationships for representing design and analyzable design model structures of a family of artifacts with different assembly system topologies, such as a family of multi-stratum printed circuit boards.

- *Target meta-model*: The target meta-model of BMFM's model transformation process is the combined VTMB Artifact-specific Meta-Model (D3) and VTMB Artifact Behavior Meta-Model (B3). As shown in Figure 8.5, this meta-model is used for representing simulation templates for a specific type of analysis for a family of VTMB artifacts.

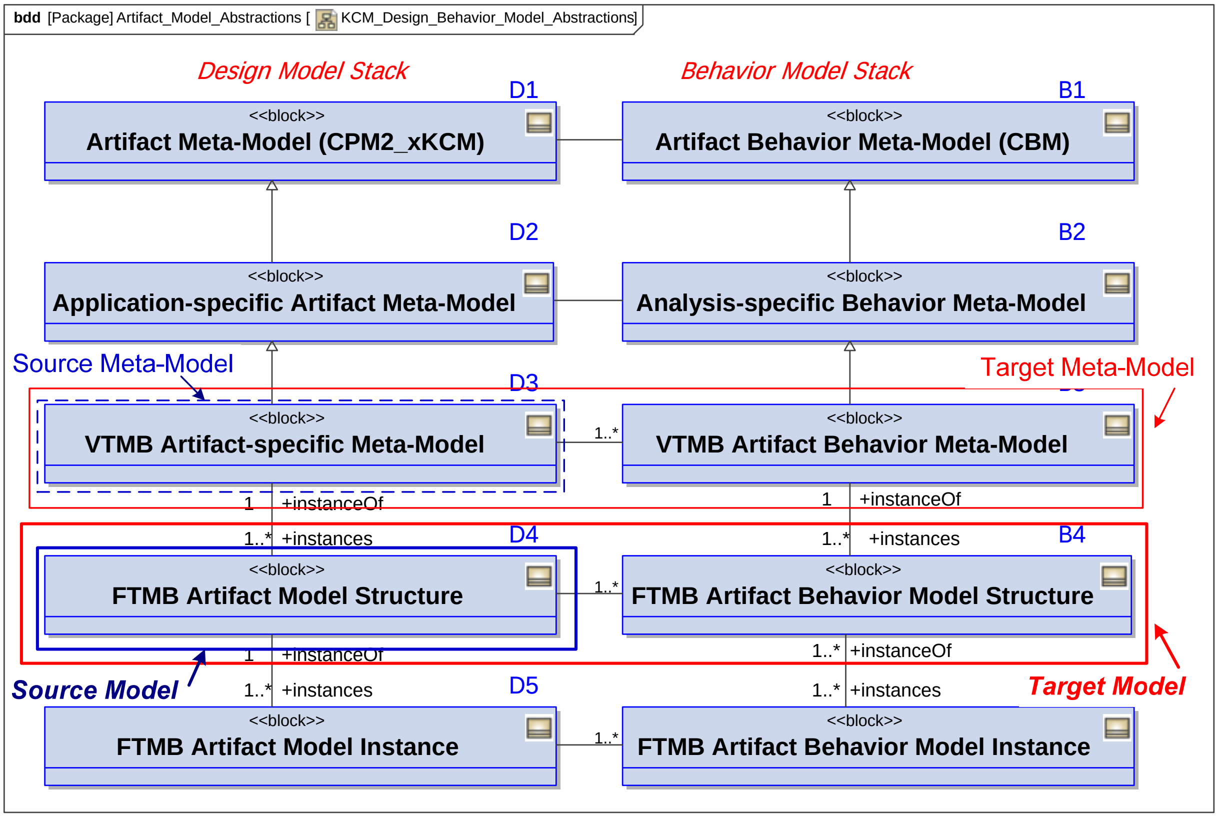
- *Source model*: The source model of BMFM's model transformation process is an FTMB Artifact Model Structure (D4) defined as an instance of the VTMB Artifact Meta-Model (source meta-model). A D4 model represents a family of fixed topology design alternatives, and it consists of the design and analyzable design model structures (Figure 8.4) for these FTMB design alternatives, such as 5-stratum printed circuit boards.



Behavior Model Formulation Specifications a
FTMB Artifact Model Structure i
FTMB Design Model Structure i
FTMB Analyzable Design Model Structure i
reads
executes
Transformation Engine
writes
Simulation Template ia
FTMB Artifact Model Structure i
FTMB Design Model Structure i
FTMB Analyzable Design Model Structure i
FTMB Artifact Behavior Model Structure ia
ABB System ia
Behavior Model XContext ia

Figure 8.4: Detailed view of the source and target models in BMFM

- *Target model*: The target model of BMFM's model transformation process is a simulation template that relates a FTMB artifact model structure to a FTMB behavior model structure. As shown in the detailed view in Figure 8.4, the simulation template uses the behavior model context to relate an ABB system and the analyzable design model structure. As presented in Chapter 7, a behavior model context and an ABB system together define an artifact behavior model structure.



bdd [Package] Artifact_Model_Abstractions [KCM_Design_Behavior_Model_Abstractions]
Design Model Stack
Behavior Model Stack
D1
<<block>>
Artifact Meta-Model (CPM2_xKCM)
B1
<<block>>
Artifact Behavior Meta-Model (CBM)
D2
<<block>>
Application-specific Artifact Meta-Model
B2
<<block>>
Analysis-specific Behavior Meta-Model
Source Meta-Model
Target Meta-Model
D3
<<block>>
VTMB Artifact-specific Meta-Model
1..*
<<block>>
VTMB Artifact Behavior Meta-Model
1
+instanceOf
1
+instanceOf
1..*
+instances
1..*
+instances
D4
B4
<<block>>
FTMB Artifact Model Structure
1..*
<<block>>
FTMB Artifact Behavior Model Structure
1
+instanceOf
1..*
+instanceOf
Source Model
Target Model
1..*
+instances
1..*
+instances
D5
<<block>>
FTMB Artifact Model Instance
1..*
<<block>>
FTMB Artifact Behavior Model Instance

Figure 8.5: Source and target meta-models and models in BMFM - design and behavior model stack view

A FTMB design and analyzable design model structures (source model) are defined as instances of the VTMB Artifact-specific Meta-Model (CPM2_xKCM), and the FTMB behavior model structure is created as an instance of the VTMB Artifact Behavior Meta-Model (CBM). Note that D3 and B3 models are specializations of D1 and B1 models respectively. Thus, the concepts in D1 and B1 may be used as-is or specialized for specific types of artifacts and specific analysis in D3 and B3 respectively. The target meta-model (B3) consists of entities defined in the Core Behavior Model (B1)—especially the definition of behavior model, ABB system, and behavior model context—and the ABB models selected for the specific types of analysis.

- *Transformation definition and Transformation process*: In the BMFM, the transformation definition and process are separate. This allows one to define reusable transformations that can be used by one or more transformation processes. The transformation definitions are building blocks of transformations while the transformation process defines the order in which these transformations are to be executed on the source model. All transformation definitions are stored in a library of model transformations, named *Artifact Model Transformation Library*. The transformation process is known as the *Behavior Model Formulation Specifications*, and it constitutes the behavior idealization decisions taken by analysts. The BMFS is defined in terms of the source and target meta-models, and prescribes the specific transformations from the Artifact Model Transformation Library that will be executed and the order of execution. The BMFS consists of conceptual specifications—idealization decisions—that may be compiled into computable specifications—a set of transformation engine-interpretable instructions that are defined in terms of the pre-existing transformations in the Artifact Model Transformation Library, and are executed by the transformation engine to create the target model.

- *Transformation Engine*: The model transformation process in the BMFM is realized using graph transformations where the source and target meta-models and models are abstracted as graphs and the transformations are abstracted as graph transformations. Hence, BMFM uses a graph transformation engine for model transformations. The

VIATRA graph transformation engine (VIATRA 2007) is used for test cases demonstrated in this dissertation.

As shown in Figure 8.3, BMFM's model transformation process is realized in the following manner:

- The source and target meta-models are defined once for a family of VTMB artifacts (such as printed circuit boards) and for a family of analyses (such as thermo-mechanical analyses).
- For a particular analysis, such as warpage analysis, analysts provide Behavior Model Formulation Specifications (say BMFS_a).
- The source models are defined by designers and are typically derived from parameterized CAD models. A designer may provided a set of FTMB artifact model structures (say FTMB artifact model structure $_i$, FTMB artifact model structure $_j$, and so on)
- During the model transformation process—as illustrated in Figure 8.3 and Figure 8.4—the transformation engine reads a FTMB Artifact Model Structure (say FTMB artifact model structure_i) and executes a Behavior Model Formulation Specification (say BMFS_a) to automatically create a simulation template (say Simulation Template_{ia} that is composed of FTMB Artifact Model Structure_i and Behavior Model Structure_{ia}). For the same BMFS (say BMFS_a), the transformation engine can read several FTMB Artifact Model Structures (say FTMB Artifact Model $\text{Structure}_{i,}$, FTMB Artifact Model Structure_j, and so on) and create corresponding simulation templates (say Simulation Template_{ia}, Simulation Template_{ja}, and so on). Also, for the same FTMB Artifact Model Structure (say FTMB Artifact Model Structure_i), analysts may provide alternate idealizations (say BMFS_b) and automatically create a simulation template (say Simulation Template_{ib}). As shown in Figure 8.4, that model transformation process results in creating a behavior model structure and relating it to an analyzable design model structure via behavior model context entity.

The core advantage of BMFM's model transformation process is to use the same BMFS to transform variable topology analyzable design model structures to create corresponding simulation templates. As an example, Figure 8.6 illustrates how the model transformation process will be realized for creating simulation templates for thermo-mechanical analysis of multi-layered printed circuit boards. A BMFS created by analysts for thermo-mechanical analyses (say Thermo-mechanical BMFS $_{\text{layer-shell}}$) will be executed by the transformation engine on 5-, 6-, 7-layered analyzable PCB model structures (variable topology design alternatives) to create thermo-mechanical simulation templates for 5-, 6-, and 7-layered PCBs. Each simulation template for thermo-mechanical analysis of n-layered PCB will be composed of n-layered analyzable PCB model structure and n-layered laminated shell behavior model structure[16]. In this case, Thermo-mechanical BMFS $_{\text{layer-shell}}$ represents the behavior idealizations—to idealize each layer in the PCB as a shell and to idealize an n-layered PCB as an n-layered laminated shell.

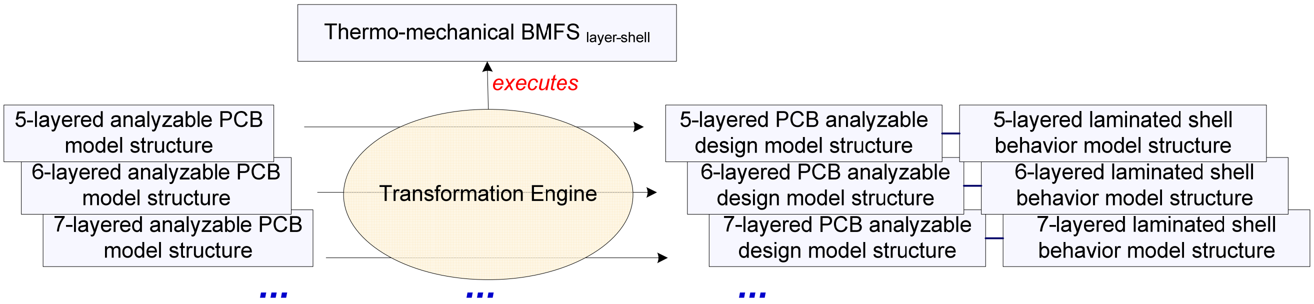

Thermo-mechanical BMFS layer-shell
executes
5-layered analyzable PCB model structure
6-layered analyzable PCB model structure
7-layered analyzable PCB model structure
Transformation Engine
5-layered PCB analyzable design model structure
6-layered PCB analyzable design model structure
7-layered PCB analyzable design model structure
5-layered laminated shell behavior model structure
6-layered laminated shell behavior model structure
7-layered laminated shell behavior model structure
...
...
...

Figure 8.6: Example schematic of KCM's Behavior Model Formulation Method applied to VTMB problems

The Behavior Model Formulation Method addresses VTMB problems because for different desing model structures—each of which represents a set of design alternatives with equivalent assembly system topologies—behavior model structures and simulation templates can automatically be created for the same Behavior Model Formulation Specifications. Additionally, behavior model structures and simulation templates can also be automatically created for different Behavior Model Formulation Specifications and for a given design model structure. The Behavior Model Formulation Method address all types of variations in assembly system topology—number and types of components, features, and interactions—of design alternatives as described in section 2.3. Automated adaptation of simulation templates based on simulation results, as described in ST_Change_Type_3 (section 2.2.2.2), is not demonstrated in the version of the Knowledge Composition Methodology presented in this research, and is recommended for future research. However, the meta-models and formalisms used in the KCM are positioned to address this use case. In the version of KCM presented in this dissertation, analysts may automatically re-formulate

[16] assuming that layers are preserved and not ignored in the idealization specified in BMFS

simulation templates by varying the Behavior Model Formulation Specifications to reflect the new knowledge gained from simulation results.

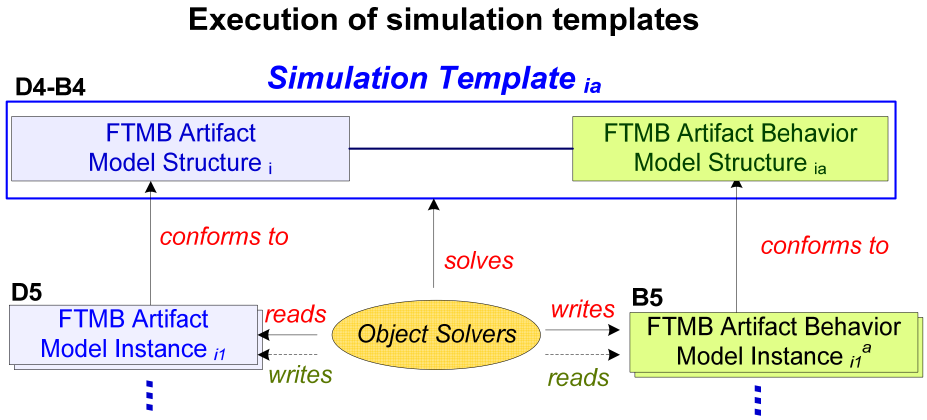

Execution of simulation templates
D4-B4
Simulation Template ia
FTMB Artifact Model Structure i
FTMB Artifact Behavior Model Structure ia
conforms to
solves
conforms to
D5
FTMB Artifact Model Instance i1
reads
writes
Object Solvers
writes
reads
B5
FTMB Artifact Behavior Model Instance i1 a

Figure 8.7: Schematic for the execution of simulation templates

Figure 8.7 illustrates the schematic for the execution of simulation templates. Simulation templates formulated at D4-B4 level using BMFM's model transformation approach can be executed in two scenarios—design verification scenario and design synthesis scenario. In the design verification scenario, design alternatives (D5) with equivalent assembly system topologies are input to a simulation template and corresponding behavior model instances (B5) are formulated. These behavior model instances can then be solved using a specific solution methods and solvers. Note that the primary focus of the KCM is to formulate behavior models independent of a solution method and solver. However, the model transformation approach can be easily extended to include solution method- and solver-specific behavior model structures in simulation templates. For example, a FEA behavior model structure could be included in simulation templates—associated with the FTMB Artifact Behavior Model Structure$_{ia}$—that specifies the element types and mesh specifications for analysis bodies and their interactions. The design synthesis scenario represents the use case where analysts may perform optimization of the analysis body system (represented in a behavior model structure), and intend to update the design model accordingly. In such a scenario, the optimized behavior model instance is input to a simulation template and the corresponding design model instance is formulated. Note that the execution of simulation templates in the design synthesis scenario depends on the nature of mathematical relationships embodied in simulation templates—causal versus non-causal relationships. While the fomer can be executed for different causalities, the latter may require the use of specialized numerical techniques and in some cases may not be pragmatic to solve.

8.2 Composing Behavior Model Structures and Simulation Templates

Given that simulation templates are automatically created from analyzable artifact design model structures and behavior model formulation specifications provided by analysts, it is necessary to understand the different stages of model transformations during this process. This process of model transformation is realized by composing a behavior model structure from analysis building blocks provided by the KCM (Chapter 7), and composing a simulation template from the behavior model structure and design model structure. This composition process is realized in four stages that are described in section 8.2.1. In each stage a specific type of composition is achieved, both for creating behavior model structure and simulation template. In section 8.2.2, the semantics of composing behavior model structures and simulation templates is described. Semantically, the process of composing a behavior model structure and simulation template is a process of deriving equations relating the behavior parameters to the design parameters, and the building blocks of the composition process represent pre-defined equations representing domain theoretic concepts that are used during this derivation. In section 8.2.3, the mechanics of the composition process is presented in terms of graph transformations that are the theoretical foundation of the composition process.

8.2.1 Stages of composition

The model transformation process prescribed by the Behavior Model Formulation Method is a four-stage composition process. Figure 8.8 illustrates these four stages of compositions that occur when a FTMB behavior model structure and simulation template (target model) are created from an FTMB analyzable artifact model structure (source model). The figure shows the source and target models in these four stages of composition. The source model is a FTMB analyzable artifact model that represents an idealized design for analysis purposes. The target model is a FTMB simulate template that includes the source model and a FTMB behavior model structure. Thus, the model transformation process is “in-effect” a model building process that is realized in four stages. During these four stages, the source model is not changed but target model entities are created that relate to the source model. The four stages are as described below:

- *Stage 1 composition*: *Composing analysis bodies and their relationships to analyzable artifacts*

In this composition stage, analysis bodies and their relationships to analyzable artifacts are composed from their respective building blocks based on the Behavior Model Formulation Specifications provided by analysts. As shown in Figure 8.8 and described in the ABB Meta-Model and Core Behavior Model, the building blocks of an analysis body are its features, shape, material behavior, and behavior; and the building blocks of the relationship between an analysis body and an analyzable artifact are relationships between their shapes, material behaviors, and features. The end products of Stage 1 composition are (a) analysis bodies represented as instances of Analysis_Body_ABB[17]), and (b) relationship between analysis bodies and analyzable artifacts, represented as instances of Analyzable_Artifact_ABS_Relationship (see Core Behavior Model for details).

Figure 8.9 illustrates a planar shell analysis body and its relationship to the corresponding analyzable PCB stratum, created at the end of a Stage 1 composition process. The figure is abstracted from the example described in section 7.4.2 where a FTMB (5-shell) thermo-mechanical behavior model structure is created for FTMB (5-stratum) printed circuit boards. The figure shows a composed analysis body and its relationship with an analyzable artifact for a single stratum in the analyzable PCB model.

[17] Note that only the property attribute of an ABB is instantiated; the other three attributes (context, application conditions, and transformations) are static.

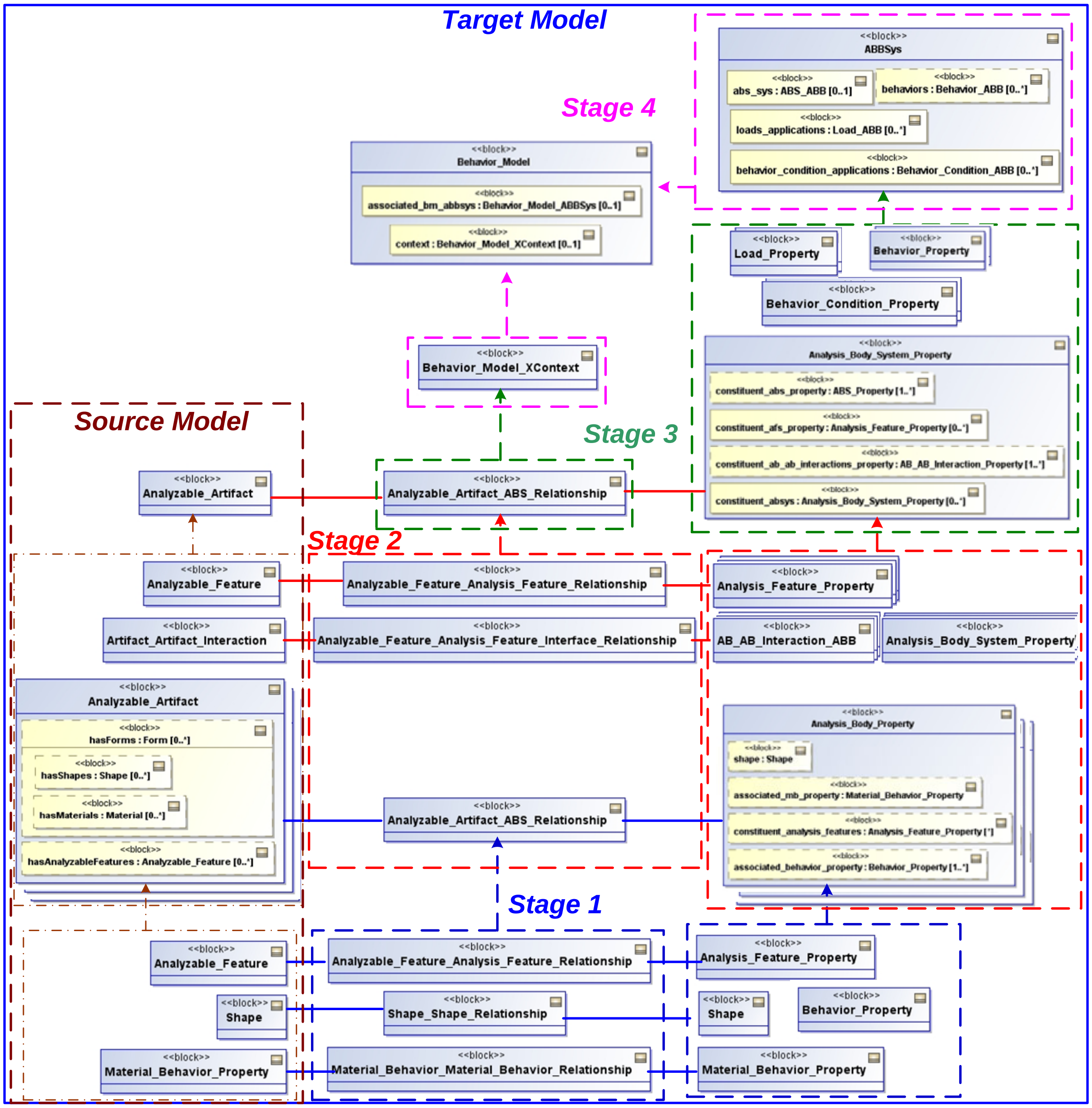

Target Model
Stage 4
<<block>>
ABBSys
<<block>>
abs_sys : ABS_ABB [0..1]
<<block>>
behaviors : Behavior_ABB [0..*]
<<block>>
loads_applications : Load_ABB [0..*]
<<block>>
behavior_condition_applications : Behavior_Condition_ABB [0..*]
<<block>>
Behavior_Model
<<block>>
associated_bm_abbsys : Behavior_Model_ABBSys [0..1]
<<block>>
context : Behavior_Model_XContext [0..1]
<<block>>
Load_Property
<<block>>
Behavior_Property
<<block>>
Behavior_Condition_Property
<<block>>
Analysis_Body_System_Property
<<block>>
constituent_abs_property : ABS_Property [1..*]
<<block>>
constituent_afs_property : Analysis_Feature_Property [0..*]
<<block>>
constituent_ab_ab_interactions_property : AB_AB_Interaction_Property [1..*]
<<block>>
constituent_absys : Analysis_Body_System_Property [0..*]
<<block>>
Behavior_Model_XContext
Stage 3
Source Model
<<block>>
Analyzable_Artifact
<<block>>
Analyzable_Artifact_ABS_Relationship
Stage 2
<<block>>
Analyzable_Feature
<<block>>
Analyzable_Feature_Analysis_Feature_Relationship
<<block>>
Analysis_Feature_Property
<<block>>
Artifact_Artifact_Interaction
<<block>>
Analyzable_Feature_Analysis_Feature_Interface_Relationship
<<block>>
AB_AB_Interaction_ABB
<<block>>
Analysis_Body_System_Property
<<block>>
Analyzable_Artifact
<<block>>
hasForms : Form [0..*]
<<block>>
hasShapes : Shape [0..*]
<<block>>
hasMaterials : Material [0..*]
<<block>>
hasAnalyzableFeatures : Analyzable_Feature [0..*]
<<block>>
Analyzable_Artifact_ABS_Relationship
<<block>>
Analysis_Body_Property
<<block>>
shape : Shape
<<block>>
associated_mb_property : Material_Behavior_Property
<<block>>
constituent_analysis_features : Analysis_Feature_Property [*]
<<block>>
associated_behavior_property : Behavior_Property [1..*]
Stage 1
<<block>>
Analyzable_Feature
<<block>>
Analyzable_Feature_Analysis_Feature_Relationship
<<block>>
Analysis_Feature_Property
<<block>>
Shape
<<block>>
Shape_Shape_Relationship
<<block>>
Shape
<<block>>
Behavior_Property
<<block>>
Material_Behavior_Property
<<block>>
Material_Behavior_Material_Behavior_Relationship
<<block>>
Material_Behavior_Property

Figure 8.8: Stages of composing simulation templates using BMFM

The same structure is repeated for other stratums in the analyzable PCB. In this example, instances of material behavior ABB (LEOTI_1), shape ABB (PS1_Shape), and analysis feature ABB (PS1_PrimSurf and PS1_SecSurf) are created and associated with an instance of analysis body ABB (PS1). In addition, an instance of Shape_Shape_Relationship (SS1), an instance of Material_Behavior_Material_Behavior_Relationship (MBR1), and two instances of Analyzable_Feature_Analysis_Feature_Relationship (AStrat_PShell_1_Prim_Rel and AStrat_PShell_2_Sec_Rel) are created, associated with corresponding entities of the analyzable stratum and planar shell analysis body, and associated with an instance of Analyzable_Artifact_ABS_Relationship (AStrat_PShell1_Rel). Note that instances of specialized ABBs are created during this composition process but for brevity only the parent ABBs are mentioned here. For example, PS1 is an instance of planar shell analysis body ABB which is a special type of analysis body ABB.

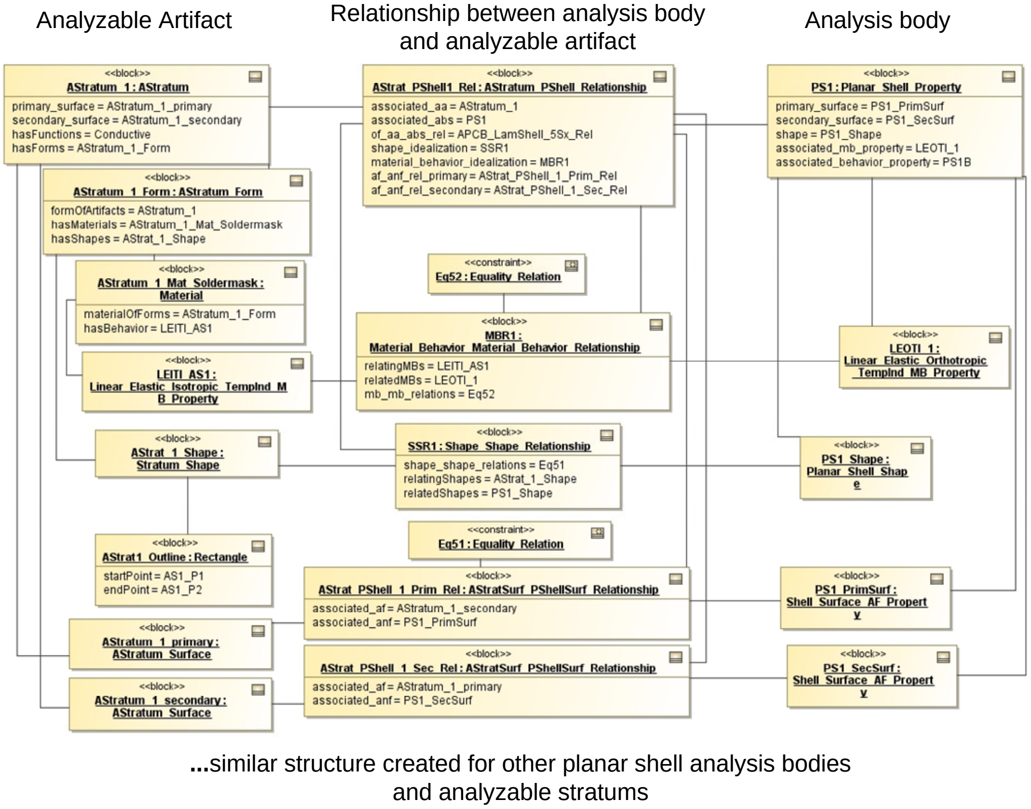

Analyzable Artifact
Relationship between analysis body and analyzable artifact
Analysis body
<<block>>
AStratum 1 : AStratum
primary_surface = AStratum_1_primary
secondary_surface = AStratum_1_secondary
hasFunctions = Conductive
hasForms = AStratum_1_Form
<<block>>
AStratum 1 Form : AStratum Form
formOfArtifacts = AStratum_1
hasMaterials = AStratum_1_Mat_Soldermask
hasShapes = AStrat_1_Shape
<<block>>
AStratum 1 Mat Soldermask : Material
materialOfForms = AStratum_1_Form
hasBehavior = LEITI_AS1
<<block>>
LEITI AS1 : Linear Elastic Isotropic TempInd M B Property
<<block>>
AStrat 1 Shape : Stratum Shape
<<block>>
AStrat1 Outline : Rectangle
startPoint = AS1_P1
endPoint = AS1_P2
<<block>>
AStratum 1 primary : AStratum Surface
<<block>>
AStratum 1 secondary : AStratum Surface
<<block>>
AStrat PShell1 Rel : AStratum PShell Relationship
associated_aa = AStratum_1
associated_abs = PS1
of_aa_abs_rel = APCB_LamShell_5Sx_Rel
shape_idealization = SSR1
material_behavior_idealization = MBR1
af_anf_rel_primary = AStrat_PShell_1_Prim_Rel
af_anf_rel_secondary = AStrat_PShell_1_Sec_Rel
<<constraint>>
Eq52 : Equality Relation
<<block>>
MBR1 : Material Behavior Material Behavior Relationship
relatingMBs = LEITI_AS1
relatedMBs = LEOTI_1
mb_mb_relations = Eq52
<<block>>
SSR1 : Shape Shape Relationship
shape_shape_relations = Eq51
relatingShapes = AStrat_1_Shape
relatedShapes = PS1_Shape
<<constraint>>
Eq51 : Equality Relation
<<block>>
AStrat PShell 1 Prim Rel : AStratSurf PShellSurf Relationship
associated_af = AStratum_1_secondary
associated_anf = PS1_PrimSurf
<<block>>
AStrat PShell 1 Sec Rel : AStratSurf PShellSurf Relationship
associated_af = AStratum_1_primary
associated_anf = PS1_SecSurf
<<block>>
PS1 : Planar Shell Property
primary_surface = PS1_PrimSurf
secondary_surface = PS1_SecSurf
shape = PS1_Shape
associated_mb_property = LEOTI_1
associated_behavior_property = PS1B
<<block>>
LEOTI 1 : Linear Elastic Orthotropic TempInd MB Property
<<block>>
PS1 Shape : Planar Shell Shap e
<<block>>
PS1 PrimSurf : Shell Surface AF Propert y
<<block>>
PS1 SecSurf : Shell Surface AF Propert y
...similar structure created for other planar shell analysis bodies and analyzable stratums

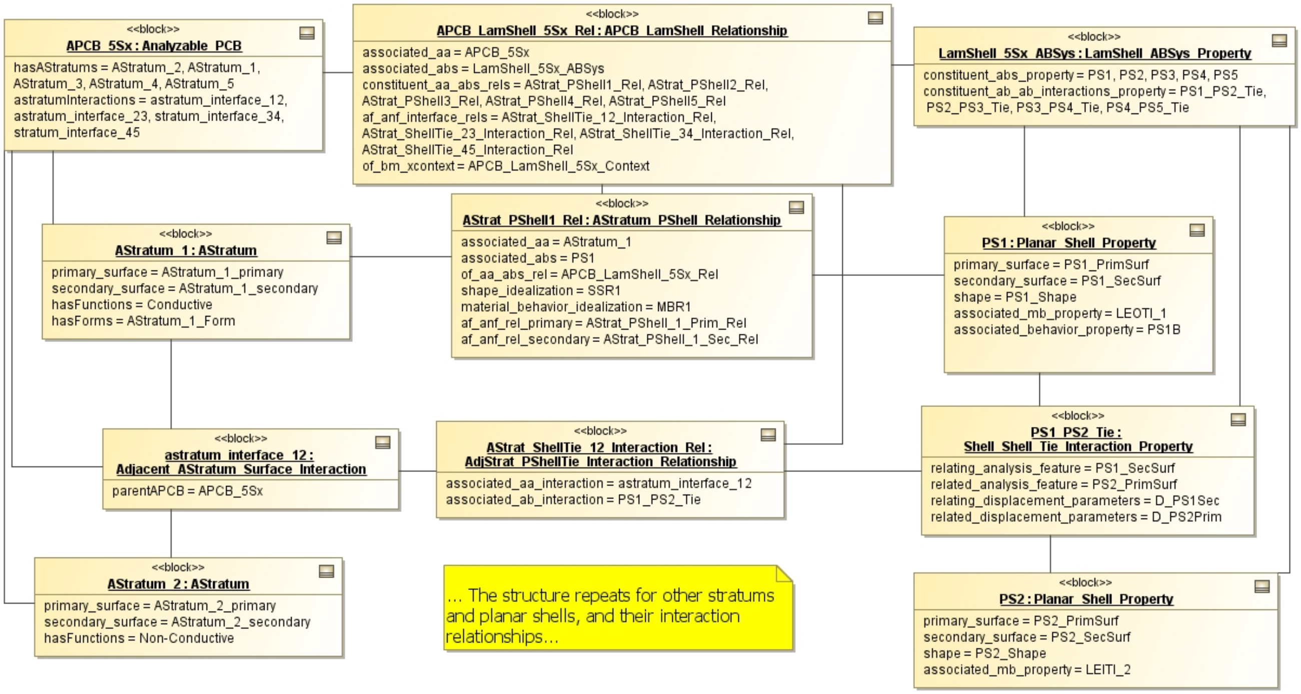
Figure 8.9: Stage 1 Composition: Composing an analysis body and its relationship with analyzable artifacts

Also note that the specialized pre-defined analysis body ABB instantiated here has attributes whose types restricts the shape, feature, and material behavior ABB instances that can be associated with it. For example, the Planar Shell Analysis Body ABB has a shape attribute of type Planar Shell Shape. This allows only instances of Planar Shell Shape to be associated with instances of Planar Shell Analysis Body ABB.

- *Stage 2 composition: Composing analysis body systems and their relationships to analyzable artifacts*

In this composition stage, analysis body systems and their relationships to analyzable artifacts are composed from their respective building blocks based on the Behavior Model Formulation Specifications provided by analysts. As shown in Figure 8.8 and described in the ABB Meta-Model and Core Behavior Model, the building blocks of an analysis body system are its features, constituent analysis bodies and analysis body systems, and interactions between constituent analysis bodies; and the building blocks of the relationship between an analysis body system and an analyzable artifact are relationships between their shapes, material behaviors, features, and their interactions. The end products of Stage 2 composition are (a) analysis body systems represented as instances of Analysis_Body_System_ABB), and (b) relationship between analysis body systems and analyzable artifacts, represented as instances of Analyzable_Artifact_ABS_Relationship (see Core Behavior Model for details).

Figure 8.10 illustrates a laminated shell analysis body system and its relationship to the corresponding analyzable PCB, created at the end of a Stage 2 composition process. The figure is abstracted from the example described in section 7.4.2 where a FTMB (5-shell) thermo-mechanical behavior model structure is created for FTMB (5-stratum) printed circuit boards. The figure shows a composed analysis body system and its relationship with an analyzable artifact. For brevity, only one out of five constituent analysis bodies and one out of four analysis body interactions are shown for the subject analysis body system. In this example, instances of analysis body ABBs (PS1,PS2,...,PS5) created in Stage 1 and analysis body interaction ABBs (PS1_PS2_Tie,...,PS4_PS5_Tie) created in Stage 2 are associated with an instance of analysis body system ABB (LamShell_5Sx_ABSys) created in Stage 2. In addition, 5 instances of Analyzable_Artifact_Analysis_Body_Relationship (AStrat_PShell1_Rel,...,AStrat_PShell5_Rel) created in Stage 1, and four instances of Analyzable_Feature_Analysis_Feature_Interface_Relationship (AStrat_ShellTie_12_Interaction_Rel,...,AStrat_ShellTie_45_Interaction_Rel) created in Stage 2 are associated with an instance of Analyzable_Artifact_ABS_Relationship (APCB_LamShell_5Sx_Rel) created during Stage 2. In this example, the analysis body system does not constitute other analysis body systems (sub-systems). Semantically, in this composition, a laminated shell analysis body system is being composed from the individual shell bodies and the tie interactions among the adjacent shell bodies in the stackup. In addition, a relationship between the laminated shell system and the PCB is being composed from (a) relationships between shell bodies and corresponding idealized stratums on a PCB, and (b) relationships between shell tie interactions and corresponding interfaces between PCB stratums.



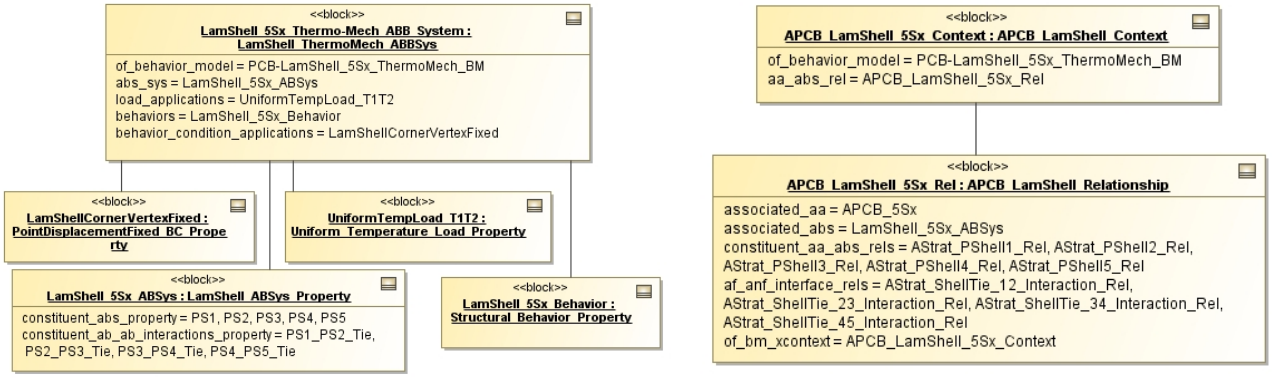
<<block>>
APCB 5Sx : Analyzable PCB
hasAStratums = AStratum_2, AStratum_1, AStratum_3, AStratum_4, AStratum_5
astratumInteractions = astratum_interface_12, astratum_interface_23, stratum_interface_34, stratum_interface_45
<<block>>
APCB LamShell 5Sx Rel : APCB LamShell Relationship
associated_aa = APCB_5Sx
associated_abs = LamShell_5Sx_ABSys
constituent_aa_abs_rels = AStrat_PShell1_Rel, AStrat_PShell2_Rel, AStrat_PShell3_Rel, AStrat_PShell4_Rel, AStrat_PShell5_Rel
af_anf_interface_rels = AStrat_ShellTie_12_Interaction_Rel, AStrat_ShellTie_23_Interaction_Rel, AStrat_ShellTie_34_Interaction_Rel, AStrat_ShellTie_45_Interaction_Rel
of_bm_xcontext = APCB_LamShell_5Sx_Context
<<block>>
LamShell 5Sx ABSys : LamShell ABSys Property
constituent_abs_property = PS1, PS2, PS3, PS4, PS5
constituent_ab_ab_interactions_property = PS1_PS2_Tie, PS2_PS3_Tie, PS3_PS4_Tie, PS4_PS5_Tie
<<block>>
AStratum 1 : AStratum
primary_surface = AStratum_1_primary
secondary_surface = AStratum_1_secondary
hasFunctions = Conductive
hasForms = AStratum_1_Form
<<block>>
AStrat PShell1 Rel : AStratum PShell Relationship
associated_aa = AStratum_1
associated_abs = PS1
of_aa_abs_rel = APCB_LamShell_5Sx_Rel
shape_idealization = SSR1
material_behavior_idealization = MBR1
af_anf_rel_primary = AStrat_PShell_1_Prim_Rel
af_anf_rel_secondary = AStrat_PShell_1_Sec_Rel
<<block>>
PS1 : Planar Shell Property
primary_surface = PS1_PrimSurf
secondary_surface = PS1_SecSurf
shape = PS1_Shape
associated_mb_property = LEOTI_1
associated_behavior_property = PS1B
<<block>>
astratum interface 12 :
Adjacent AStratum Surface Interaction
parentAPCB = APCB_5Sx
<<block>>
AStrat ShellTie 12 Interaction Rel :
AdjStrat PShellTie Interaction Relationship
associated_aa_interaction = astratum_interface_12
associated_ab_interaction = PS1_PS2_Tie
<<block>>
PS1 PS2 Tie :
Shell Shell Tie Interaction Property
relating_analysis_feature = PS1_SecSurf
related_analysis_feature = PS2_PrimSurf
relating_displacement_parameters = D_PS1Sec
related_displacement_parameters = D_PS2Prim
<<block>>
AStratum 2 : AStratum
primary_surface = AStratum_2_primary
secondary_surface = AStratum_2_secondary
hasFunctions = Non-Conductive
... The structure repeats for other stratums and planar shells, and their interaction relationships...
<<block>>
PS2 : Planar Shell Property
primary_surface = PS2_PrimSurf
secondary_surface = PS2_SecSurf
shape = PS2_Shape
associated_mb_property = LEITI_2

Figure 8.10: Stage 2 Composition: Composing an analysis body system and its relationship with analyzable artifact

Note that Stage 2 composition is more intuitive that other stages as it is similar to composition of physical systems where assemblies are composed from parts and the interactions among parts. Composition in KCM is the composition of models that may or may not represent systems that are similar to physical systems. For example, composing an analysis body from its attributes such as shape, features, and material behavior is not intuitively similar to composing a physical system.

- *Stage 3 composition*: *Composing a behavior model ABB system and behavior model context*

In this composition stage, a behavior model ABB system and a behavior model context—relates ABB system to analyzable artifact—are composed from their respective building blocks. As described in the Core Behavior Model, a behavior model ABB system represents the behavior model structure of an analysis body system and a behavior model context is relates the analysis body system to the analyzable artifact. As shown in Figure 8.8 and described in the ABB Meta-Model and Core Behavior Model, the building blocks of a behavior model ABB system are its analysis body system, applied loads, applied behavior conditions, and the set of idealized behaviors that it represents; and the building block of behavior model context is the relationship between the analysis body system in the behavior model ABB and the corresponding analyzable artifact. The end products of Stage 3 composition are (a) behavior model ABB system represented as an instance of Behavior_Model_ABBSys, and (b) behavior mode context represented as an instance of Behavior_Model_XContext.


<<block>>
LamShell_5Sx_Thermo-Mech_ABB_System : LamShell_ThermoMech_ABBSys
of_behavior_model = PCB-LamShell_5Sx_ThermoMech_BM
abs_sys = LamShell_5Sx_ABSys
load_applications = UniformTempLoad_T1T2
behaviors = LamShell_5Sx_Behavior
behavior_condition_applications = LamShellCornerVertexFixed
<<block>>
LamShellCornerVertexFixed : PointDisplacementFixed_BC_Property
<<block>>
UniformTempLoad_T1T2 : Uniform_Temperature_Load_Property
<<block>>
LamShell_5Sx_ABSys : LamShell_ABSys_Property
constituent_abs_property = PS1, PS2, PS3, PS4, PS5
constituent_ab_ab_interactions_property = PS1_PS2_Tie, PS2_PS3_Tie, PS3_PS4_Tie, PS4_PS5_Tie
<<block>>
LamShell_5Sx_Behavior : Structural_Behavior_Property
<<block>>
APCB_LamShell_5Sx_Context : APCB_LamShell_Context
of_behavior_model = PCB-LamShell_5Sx_ThermoMech_BM
aa_abs_rel = APCB_LamShell_5Sx_Rel
<<block>>
APCB_LamShell_5Sx_Rel : APCB_LamShell_Relationship
associated_aa = APCB_5Sx
associated_abs = LamShell_5Sx_ABSys
constituent_aa_abs_rels = AStrat_PShell1_Rel, AStrat_PShell2_Rel, AStrat_PShell3_Rel, AStrat_PShell4_Rel, AStrat_PShell5_Rel
af_anf_interface_rels = AStrat_ShellTie_12_Interaction_Rel, AStrat_ShellTie_23_Interaction_Rel, AStrat_ShellTie_34_Interaction_Rel, AStrat_ShellTie_45_Interaction_Rel
of_bm_xcontext = APCB_LamShell_5Sx_Context

a. Composing behavior model ABB system

b. Composing behavior model context

Figure 8.11: Stage 3 Composition: Composing behavior model ABB system and behavior model context

Figure 8.11 (a and b) illustrate a behavior model ABB system and a behavior model context model respectively, created at the end of Stage 3 composition process. The figure is abstracted from the example described in section 7.4.2 where a FTMB (5-shell) thermo-mechanical behavior model structure is created for a FTMB (5-stratum) printed circuit boards. In this example, an instance of analysis body system created in Stage 2, instance of temperature load ABB (UniformTempLoad_T1T2) created in Stage 3, instance of point displacement fixed boundary condition ABB (LamShellCornerVertexFixed) created in Stage 3, and instance of structural behavior ABB (LamShell_5Sx_Behavior) created in Stage 3 are associated with an instance of behavior model ABB system (LamShell_5Sx_Thermo-Mech_ABB_System) created in Stage 3. Semantically, in this composition, a behavior model ABB system is being composed from the laminated shell analysis body system, uniform temperature load, point displacement fixed boundary condition, and structural behavior parameters and relations. The behavior model context relates the laminated shell analysis body system and the analyzable PCB (idealized PCB design for analysis purposes).

- *Stage 4 composition: Composing behavior model structure and simulation template*

In this composition stage, a behavior model structure is composed from a behavior model ABB system and a behavior model context as shown in Figure 8.8. The end product of Stage 4 composition is a FTMB behavior model structure represented as an instance of Behavior_Model. Figure 8.12 illustrates a FTMB thermo-mechanical behavior model structure (PCB-LamShell_5Sx_ThermoMech_BM) that is composed in this stage from a FTMB behavior model ABB (APCB_LamShell_5Sx_Context) and a behavior model context (LamShell_5Sx_Thermo-Mech_ABB_System) composed in Stage 3.

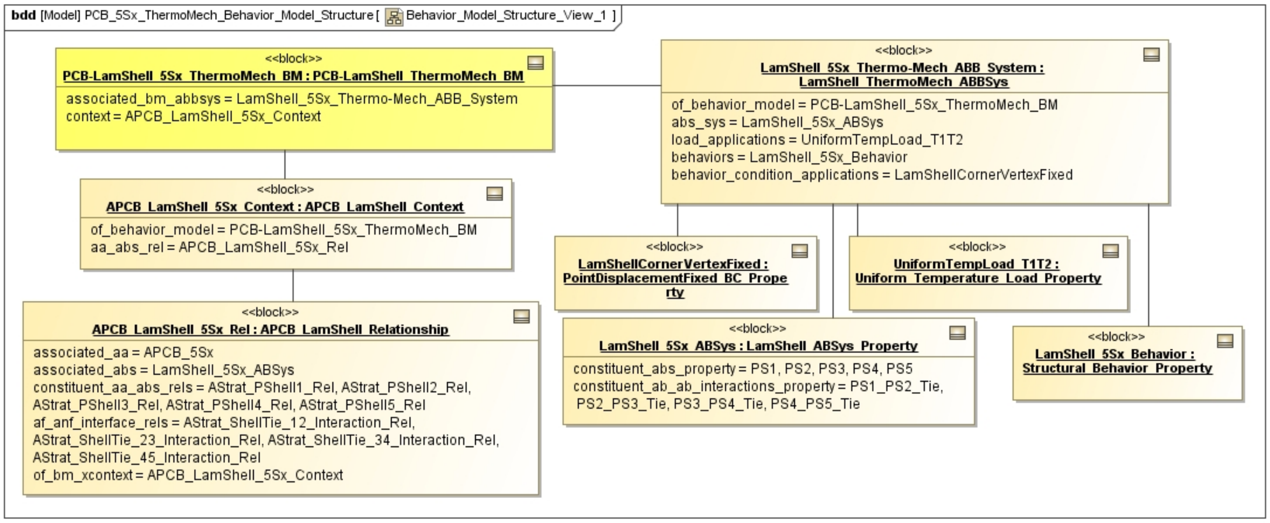

bdd [Model] PCB_5Sx_ThermoMech_Behavior_Model_Structure [Behavior_Model_Structure_View_1]
<<block>>
PCB-LamShell 5Sx ThermoMech BM : PCB-LamShell ThermoMech BM
associated_bm_abbsys = LamShell_5Sx_Thermo-Mech_ABB_System
context = APCB_LamShell_5Sx_Context
<<block>>
LamShell 5Sx Thermo-Mech ABB System : LamShell ThermoMech ABBSys
of_behavior_model = PCB-LamShell_5Sx_ThermoMech_BM
abs_sys = LamShell_5Sx_ABSys
load_applications = UniformTempLoad_T1T2
behaviors = LamShell_5Sx_Behavior
behavior_condition_applications = LamShellCornerVertexFixed
<<block>>
APCB LamShell 5Sx Context : APCB LamShell Context
of_behavior_model = PCB-LamShell_5Sx_ThermoMech_BM
aa_abs_rel = APCB_LamShell_5Sx_Rel
<<block>>
LamShellCornerVertexFixed : PointDisplacementFixed BC Property
<<block>>
UniformTempLoad T1T2 : Uniform Temperature Load Property
<<block>>
APCB LamShell 5Sx Rel : APCB LamShell Relationship
associated_aa = APCB_5Sx
associated_abs = LamShell_5Sx_ABSys
constituent_aa_abs_rels = AStrat_PShell1_Rel, AStrat_PShell2_Rel, AStrat_PShell3_Rel, AStrat_PShell4_Rel, AStrat_PShell5_Rel
af_anf_interface_rels = AStrat_ShellTie_12_Interaction_Rel, AStrat_ShellTie_23_Interaction_Rel, AStrat_ShellTie_34_Interaction_Rel, AStrat_ShellTie_45_Interaction_Rel
of_bm_xcontext = APCB_LamShell_5Sx_Context
<<block>>
LamShell 5Sx ABSys : LamShell ABSys Property
constituent_abs_property = PS1, PS2, PS3, PS4, PS5
constituent_ab_ab_interactions_property = PS1_PS2_Tie, PS2_PS3_Tie, PS3_PS4_Tie, PS4_PS5_Tie
<<block>>
LamShell 5Sx Behavior : Structural Behavior Property

Figure 8.12: Stage 4 Composition: Behavior model structure view

Note that a behavior model is also the root entity (or the central entity) of a simulation template. This is so because a behavior model is composed of behavior model context—represented by Behavior_Model_XContext block—that relates the analysis body system in the ABB system to an analyzable artifact. Note that in composing a FTMB behavior model structure, the simulation template is also composed. Figure 8.13 illustrates the thermo-mechanical behavior simulation template that shows the analyzable artifact (APCB_5Sx) associated with the behavior model context (APCB_LamShell_5Sx_Rel) entity.

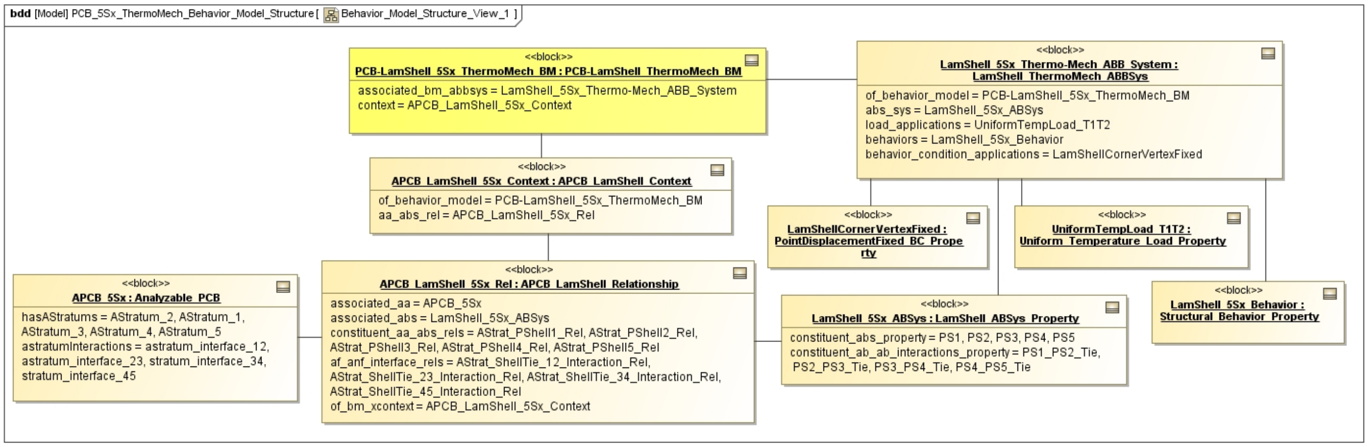

bdd [Model] PCB_5Sx_ThermoMech_Behavior_Model_Structure [Behavior_Model_Structure_View_1]
<<block>>
PCB-LamShell 5Sx ThermoMech BM : PCB-LamShell ThermoMech BM
associated_bm_abbsys = LamShell_5Sx_Thermo-Mech_ABB_System
context = APCB_LamShell_5Sx_Context
<<block>>
LamShell 5Sx Thermo-Mech ABB System : LamShell ThermoMech ABBSys
of_behavior_model = PCB-LamShell_5Sx_ThermoMech_BM
abs_sys = LamShell_5Sx_ABSys
load_applications = UniformTempLoad_T1T2
behaviors = LamShell_5Sx_Behavior
behavior_condition_applications = LamShellCornerVertexFixed
<<block>>
APCB LamShell 5Sx Context : APCB LamShell Context
of_behavior_model = PCB-LamShell_5Sx_ThermoMech_BM
aa_abs_rel = APCB_LamShell_5Sx_Rel
<<block>>
LamShellCornerVertexFixed : PointDisplacementFixed BC Prope rty
<<block>>
UniformTempLoad T1T2 : Uniform Temperature Load Property
<<block>>
APCB 5Sx : Analyzable PCB
hasAStratums = AStratum_2, AStratum_1, AStratum_3, AStratum_4, AStratum_5
astratumInteractions = astratum_interface_12, astratum_interface_23, stratum_interface_34, stratum_interface_45
<<block>>
APCB LamShell 5Sx Rel : APCB LamShell Relationship
associated_aa = APCB_5Sx
associated_abs = LamShell_5Sx_ABSys
constituent_aa_abs_rels = AStrat_PShell1_Rel, AStrat_PShell2_Rel, AStrat_PShell3_Rel, AStrat_PShell4_Rel, AStrat_PShell5_Rel
af_anf_interface_rels = AStrat_ShellTie_12_Interaction_Rel, AStrat_ShellTie_23_Interaction_Rel, AStrat_ShellTie_34_Interaction_Rel, AStrat_ShellTie_45_Interaction_Rel
of_bm_xcontext = APCB_LamShell_5Sx_Context
<<block>>
LamShell 5Sx ABSys : LamShell ABSys Property
constituent_abs_property = PS1, PS2, PS3, PS4, PS5
constituent_ab_ab_interactions_property = PS1_PS2_Tie, PS2_PS3_Tie, PS3_PS4_Tie, PS4_PS5_Tie
<<block>>
LamShell 5Sx Behavior : Structural Behavior Property

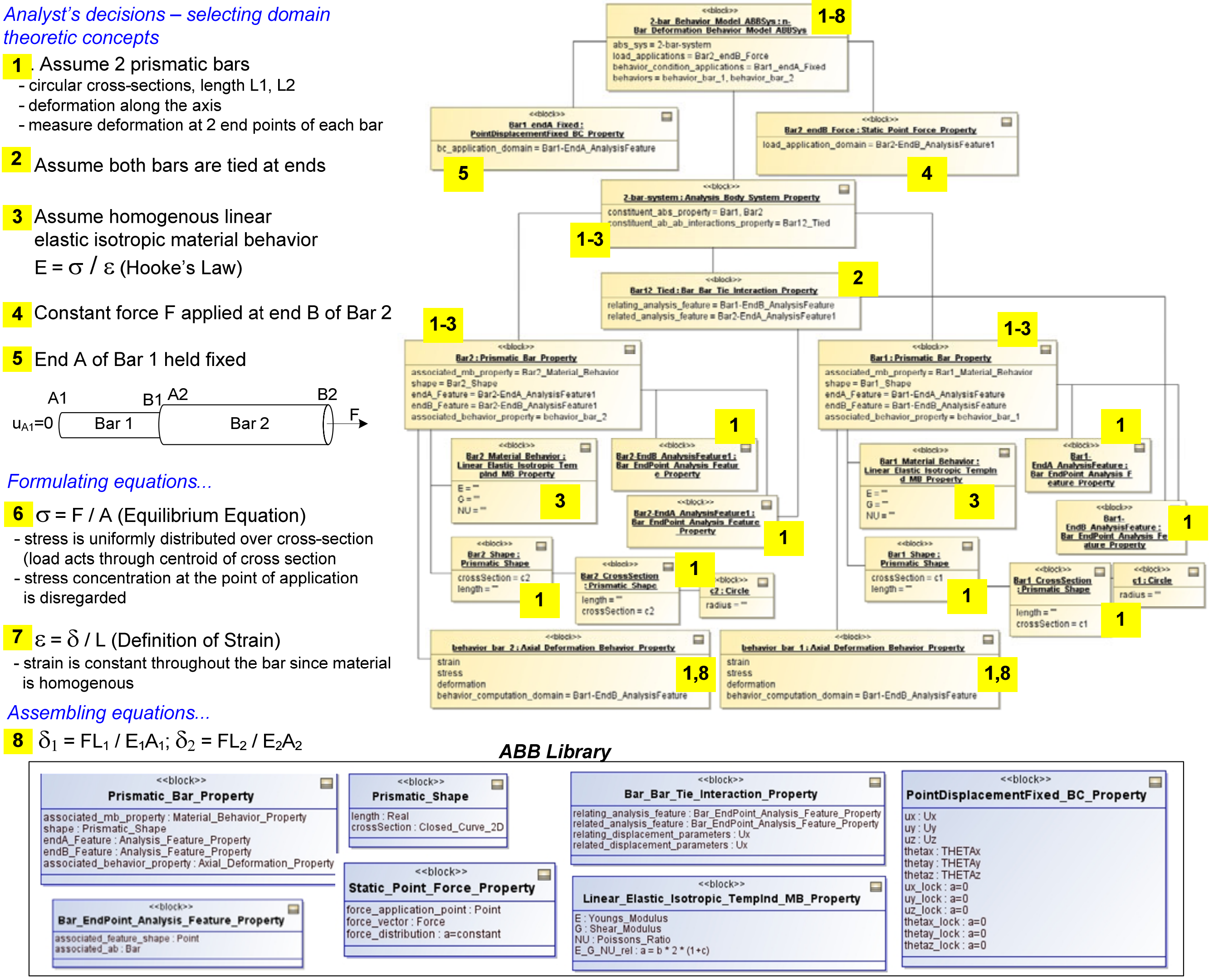
Figure 8.13: Stage 4 Composition: Simulation template view

The four composition stages defined here represent the following two specific characteristics of BMFM's model transformation process: (a) types of composition, and (b) the dependency relation between the types of composition. For example, the Stage 2 composition depends on Stage 1 composition. However, the dependency does not imply that the Stage 1 composition must be completed for all analysis bodies before Stage 2 composition may be initialized, or Stage 2 composition must be completed before Stage 3 composition. Thus, the composition process in different stages may be initialized and run in parallel, although the Stage$_{i+1}$ process cannot finish until Stage$_i$ process has finished.

8.2.2 Semantics of composition

The process of composing simulation templates is similar to the process of deriving behavior relations for a given analysis problem, where behavior relations are analytical formulations of simulation templates—relating design parameters to behavior parameters. In this section, BMFM's model composition process and the traditional process of deriving behavior relations are compared to each other. The intent of this comparison is to establish that the model-based composition of simulation templates is a more formal and structured approach to formulating behavior models, and is fundamentally similar to deriving behavior relations by "assembling" domain theoretic concepts to solve analysis problems.

Figure 8.14 illustrates the comparison between the traditional process of deriving behavior relations and the process of composing behavior model structure. The example illustrated in the figure concerns formulating a behavior model structure to compute the axial deformation of a system of two prismatic bars tied together, with one end of a bar held fixed and a static force is applied at one end of the other bar. The figure shows both the process of deriving behavior relations on the left side, and the behavior model structure (as would be composed using the Behavior Model Formulation Method). The steps in the derivation process and composition process are marked from 1-8. In this comparison, the idealized design model and its relationships to the analysis bodies/system are not shown—only the ABB system is shown for the behavior model structure.



Analyst's decisions – selecting domain theoretic concepts

1 . Assume 2 prismatic bars
- circular cross-sections, length L1, L2
- deformation along the axis
- measure deformation at 2 end points of each bar

2 Assume both bars are tied at ends

3 Assume homogenous linear elastic isotropic material behavior
$E = \sigma / \varepsilon$ (Hooke's Law)

4 Constant force F applied at end B of Bar 2

5 End A of Bar 1 held fixed

Formulating equations...

6 $\sigma = F / A$ (Equilibrium Equation)
- stress is uniformly distributed over cross-section (load acts through centroid of cross section
- stress concentration at the point of application is disregarded

7 $\varepsilon = \delta / L$ (Definition of Strain)
- strain is constant throughout the bar since material is homogenous

Assembling equations...

8 $\delta_1 = FL_1 / E_1A_1$; $\delta_2 = FL_2 / E_2A_2$

<<block>>
2-bar_Behavior_Model_ABBSys : n-Bar_Deformation_Behavior_Model_ABBSys
abs_sys = 2-bar-system
load_applications = Bar2_endB_Force
behavior_condition_applications = Bar1_endA_Fixed
behaviors = behavior_bar_1, behavior_bar_2

<<block>>
Bar1_endA_Fixed : PointDisplacementFixed_BC_Property
bc_application_domain = Bar1-EndA_AnalysisFeature

<<block>>
Bar2_endB_Force : Static_Point_Force_Property
load_application_domain = Bar2-EndB_AnalysisFeature1

<<block>>
2-bar-system : Analysis_Body_System_Property
constituent_abs_property = Bar1, Bar2
constituent_ab_ab_interactions_property = Bar12_Tied

<<block>>
Bar12_Tied : Bar_Bar_Tie_Interaction_Property
relating_analysis_feature = Bar1-EndB_AnalysisFeature
related_analysis_feature = Bar2-EndA_AnalysisFeature1

<<block>>
Bar2 : Prismatic_Bar_Property
associated_mb_property = Bar2_Material_Behavior
shape = Bar2_Shape
endA_Feature = Bar2-EndA_AnalysisFeature1
endB_Feature = Bar2-EndB_AnalysisFeature1
associated_behavior_property = behavior_bar_2

<<block>>
Bar1 : Prismatic_Bar_Property
associated_mb_property = Bar1_Material_Behavior
shape = Bar1_Shape
endA_Feature = Bar1-EndA_AnalysisFeature
endB_Feature = Bar1-EndB_AnalysisFeature
associated_behavior_property = behavior_bar_1

<<block>>
Bar2_Material_Behavior : Linear_Elastic_Isotropic_Templnd_MB_Property
E = ""
G = ""
NU = ""

<<block>>
Bar2-EndB_AnalysisFeature1 : Bar_EndPoint_Analysis_Feature_Property

<<block>>
Bar2-EndA_AnalysisFeature1 : Bar_EndPoint_Analysis_Feature_Property

<<block>>
Bar2_Shape : Prismatic_Shape
crossSection = c2
length = ""

<<block>>
Bar2_CrossSection : Prismatic_Shape
length = ""
crossSection = c2

<<block>>
c2 : Circle
radius = ""

<<block>>
Bar1_Material_Behavior : Linear_Elastic_Isotropic_Templnd_MB_Property
E = ""
G = ""
NU = ""

<<block>>
Bar1-EndA_AnalysisFeature : Bar_EndPoint_Analysis_Feature_Property

<<block>>
Bar1-EndB_AnalysisFeature : Bar_EndPoint_Analysis_Feature_Property

<<block>>
Bar1_Shape : Prismatic_Shape
crossSection = c1
length = ""

<<block>>
Bar1_CrossSection : Prismatic_Shape
length = ""
crossSection = c1

<<block>>
c1 : Circle
radius = ""

<<block>>
behavior_bar_2 : Axial_Deformation_Behavior_Property
strain
stress
deformation
behavior_computation_domain = Bar1-EndB_AnalysisFeature

<<block>>
behavior_bar_1 : Axial_Deformation_Behavior_Property
strain
stress
deformation
behavior_computation_domain = Bar1-EndB_AnalysisFeature

ABB Library

<<block>>
Prismatic_Bar_Property
associated_mb_property : Material_Behavior_Property
shape : Prismatic_Shape
endA_Feature : Analysis_Feature_Property
endB_Feature : Analysis_Feature_Property
associated_behavior_property : Axial_Deformation_Property

<<block>>
Prismatic_Shape
length : Real
crossSection : Closed_Curve_2D

<<block>>
Bar_Bar_Tie_Interaction_Property
relating_analysis_feature : Bar_EndPoint_Analysis_Feature_Property
related_analysis_feature : Bar_EndPoint_Analysis_Feature_Property
relating_displacement_parameters : Ux
related_displacement_parameters : Ux

<<block>>
PointDisplacementFixed_BC_Property
ux : Ux
uy : Uy
uz : Uz
thetax : THETAx
thetay : THETAy
thetaz : THETAz
ux_lock : a=0
uy_lock : a=0
uz_lock : a=0
thetax_lock : a=0
thetay_lock : a=0
thetaz_lock : a=0

<<block>>
Bar_EndPoint_Analysis_Feature_Property
associated_feature_shape : Point
associated_ab : Bar

<<block>>
Static_Point_Force_Property
force_application_point : Point
force_vector : Force
force_distribution : a=constant

<<block>>
Linear_Elastic_Isotropic_TempInd_MB_Property
E : Youngs_Modulus
G : Shear_Modulus
NU : Poissons_Ratio
E_G_NU_rel : a = b * 2 * (1+c)

Figure 8.14: Semantics of composition

All ABBs used for composing the behavior model structure are shown in the ABB library in the figure. Only the property attribute of ABBs is shown (e.g. Prismatic_Bar_Property block that is the type of Prismatic_Bar_ABB.property block).

Steps 1-5 concern the decisions take by analysts and steps 6-8 involve the formulation and assembly of equations for this analysis problem. In step 1, a decision is taken to idealize the behavior of a 2-bar idealized design (or idealized design) as the behavior of a system of two prismatic bars with circular cross-sections and tied end to end. Here, the axial deformation behavior is being studied in particular. This decision corresponds to the instantiation of two prismatic bar analysis body ABBs (Bar1 and Bar2) along with the instantiation of two prismatic shape ABBs (Bar1_Shape and Bar2_Shape) that represent the shape of the two prismatic bars, bar end analysis feature ABBs (Bar1-EndA, Bar1-EndB, Bar2-EndA, Bar2-EndB) that represent the end points of prismatic bars, and axial deformation behavior ABBs (behavior_bar_1 and behavior_bar_2) that represent the behavior parameters to be computed for bar 1 and 2. The tag marked "1" attached to behavior model structure entities (such as Bar1 and Bar2) indicates that these entities are created in step 1. In step 2, a decision is taken to idealize the interaction between the two analysis bars as tied interaction—deformation behavior parameters at Bar1-EndB and Bar2-EndA are equated. In step 3, the constitutive material behavior of both the prismatic bars is idealized as homogenous linear elastic and isotropic. This corresponds to the instantiation of linear elastic isotropic temperature[18] independent material behavior ABB (Bar1_Material_Behavior and Bar2_Material_Behavior). In step 4, a decision is taken to idealize load as static force acting at end B of Bar 2, at the center of the cross section of end B, and in step 5, a decision is taken to assume that end A of Bar 1 is fixed. These decisions correspond to the instantiation of a static force ABB (Bar2_endB_Force) and point displacement fixed boundary condition ABB (Bar1_endA_Fixed).

Steps 6-8 correspond to the formulation and assembly of behavior relations. Behavior relations are formulated based on: (a) Equilibrium equation shown in step 6, (b) Strain definition relation (or displacement relation) as shown in step 7, and (c) Hooke's law material behavior equation shown in step 3. Then, these equations are assembled to define

[18] The temperature independence aspect does not concern the subject analysis problem (since it is not a thermal or thermo-mechanical problem).

deformation behavior of bar 1 and bar 2 as shown in step 8. The formulated equations can be represented as mathematical relations in behavior_bar_1 and behavior_bar_2 entities (for the deformation of each bar) and in 2-bar-system (for the overall deformation of the two bar system).

The assumption decisions made during the derivation process are representative of the selection of ABBs from the ABB library. Each ABB (property) represents the parameters and relations for that ABB, such as the relation between Young's Modulus, Shear Modulus, and Poisson's ratio for the case of linear elastic isotropic temperature independent material behavior ABB shown in the figure. Similarly, the point displacement fixed behavior condition ABB represents the displacement parameters and the constraint equations, such as ux=0, uy=0 and so on.

As in the derivation process, the decisions taken during a step in the model composition process may or may not constrain the choices available for the decisions taken at the next step. For example, the material behavior idealization decision is independent of the analysis body type and shapes (prismatic bar and prismatic shape) and it is also independent of the interaction behavior at the interface of the two bars. Similarly, the interaction behavior is independent of the material behavior of the two bars and the analysis body type and shape. However, just as in the case of the derivation process, some decisions may constraint the subsequent decisions. For example, the decision to idealize the behavior of the designed artifact as a prismatic bar constrains the analysis features and type of shape that can be associated with the bar. In the model composition process, it implies that instances of only specific type of analysis feature ABBs and shape ABBs may be associated with the instances of the analysis feature ABB. A prismatic bar by definition has two end points that are modeled as point features and a prismatic shape associated with it. These constraints are reflected in the definition of the prismatic bar ABB and hence automatically handled during the behavior model composition process. Note that Prismatic_Bar_Property.shape is of type Prismatic_Shape; and Prismatic_Bar_Property.endA_feature and Prismatic_Bar_Property.endB_feature represent the two end features of a prismatic bar and are of type Bar_EndPoint_Analysis_Feature_Property that represents end point feature of a bar.

Note that the domain theoretic principles such as Equilibrium equations, Stress and Strain definitions are not explicitly shown in the behavior model structure in the figure above. They can be represented as behavior relations associated with the 2-bar analysis body system. However, for most multi-body problems, analytical formulations of the system-level behavior relations are not available. Numerical solution techniques need to be employed to solve the problems, such as FEA methods. Most numerical solvers, such as FEA solvers like ABAQUS and ANSYS are "computationally-aware" of the domain theoretic principles such as Equilibrium equations and Hook's Law. Thus, behavior model structures formulated from physics-based principles need to "refer" to the specific principles when transforming ABB systems to solution method models (as prescribed by the MRA simulation template pattern) and not necessarily "represent" the mathematical relationships embodied in these principles. For example, the details of analytical plate theory formulations (Timoshenko and Goodier 1970) may not be necessarily represented in the plate analysis body ABB though the latter may refer to such formulations for the sake of completeness. Albeit, the ABB Meta-Model provides mechanisms to represent such formulations as required, such as Behavior_Property.behavior_parameter_relations for behavior ABBs, and Material_Behavior_Property.mb_parameters_relations for material behavior ABBs, and load_distribution_function for load ABBs, and so on.

8.2.3 Mechanics of composition

As described in the previous sections, the model transformation process prescribed by the Behavior Model Formulation Method is one where behavior model structures and simulation templates are composed[19] in four stages. In this section, the mechanics of this composition process is described. The key computation elements necessary for achieving the composition are described. Figure 8.15 illustrates these computation elements in the backdrop of the schematic of Behavior Model Formulation Method, as described in section 8.1. The model transformation process is computationally realized as graph transformations. The source and target models are represented as graphs and a graph transformation engine creates a target graph for a given source graph. The transformation definitions in the

[19] Here, model composition is regarded as a special type of model transformation.

Artifact Model Transformation Library are represented as graph transformation patterns and rules, and the transformation process defined by the Behavior Model Formulation Specifications is represented as a graph transformation process.

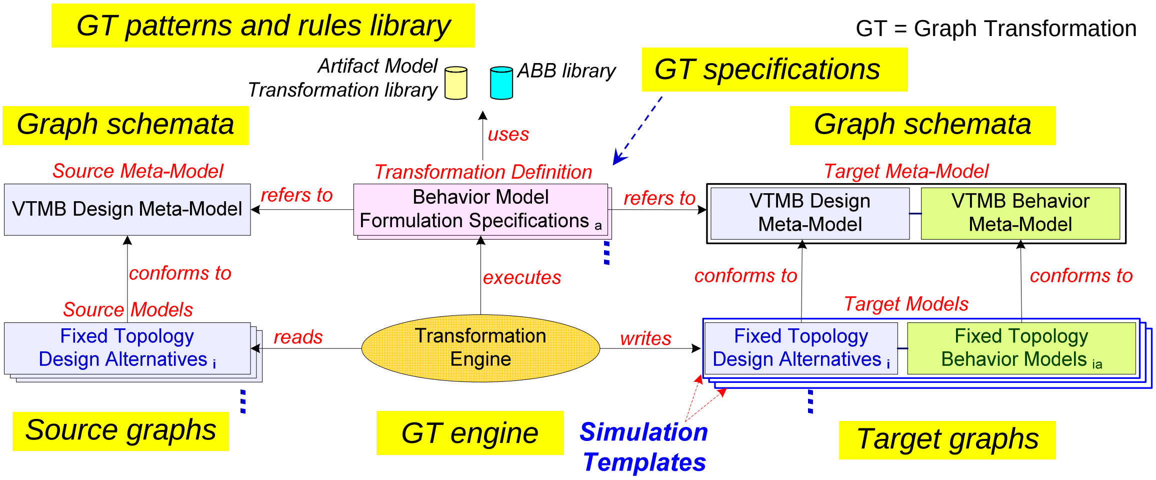


Figure 8.15: BMFM's model transformation process realized as graph transformations (GT)

Source and Target graphs

A ***graph*** G = (V, E) consists of two sets V and E where:

- elements of V are known as **vertices (or nodes)**
- elements of E are known as **edges**
- an edge has 1-2 vertices[20] associated with it (called its end points)

(Gross and Yellen 2003)

The source and target graphs in the Behavior Model Formulation Method are directed, labeled, attributed, and typed graphs (Gross and Yellen 2003), and represented using SysML. In general, labeled, attribute, typed graphs can be thought as formal representations of class models (Andries, Engels et al. 1999; Czarnecki and Helsen 2006). SysML structure models are extensions of UML 2 class models. The nodes in the source and target graphs are represented using SysML instance specifications, and the edges are represented by instance slots. Specifically, the source and target graph are:

- *directed graphs* because slots owned by an instance have values that refer to other instances. Instances owning slots or populating slots are abstracted as graph nodes, and

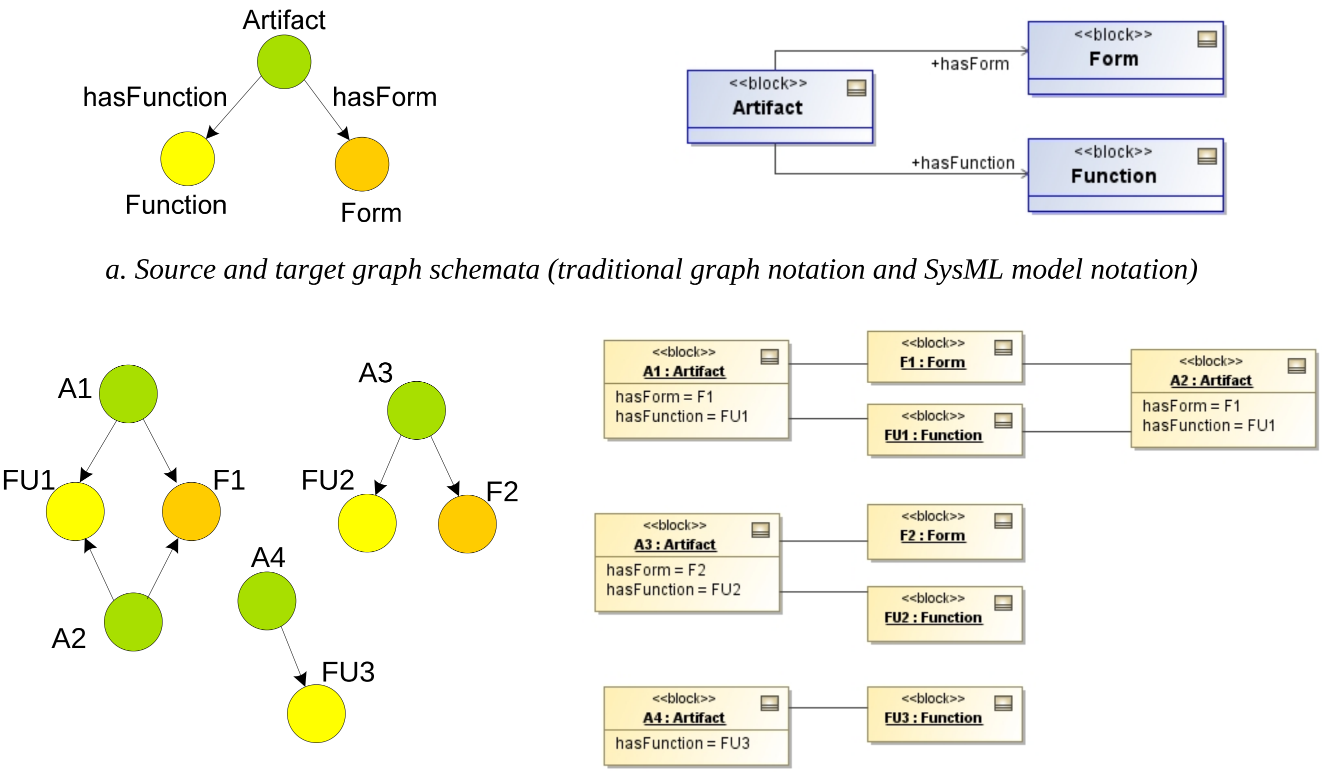
[20] The source and target graphs in this dissertation are not hypergraphs

slots are abstracted as graph edges directed from nodes corresponding to instances owning slots to nodes corresponding to instances populating slots.

- *labeled graphs* because all instances and slots have names. In addition, the names are unique with a given namespace.
- *typed graphs* because instance and slots have types (also known as classifiers). Instances used for populating slots must be of the same type (or subtype) as the slot type.
- *attributed graphs* because slots are attributes of instances. Slots may be of complex type (objects) or primitive type (such as integer and boolean).

The source and target meta-models in the Behavior Model Formulation Method are formalized as SysML-based structure models with different views, such as block definition diagrams, internal block diagrams, and parametric diagrams. In essence, the source and target meta-models are like graph schemata (Ehrig, Engels et al. 1999) for the source and target models (graphs). The nodes in the source and target meta-models (graph schemata) are represented as SysML blocks, and the edges are represented as block properties—part properties, reference properties, and constraint properties. Constraint blocks (classifier for constraint properties) are a special type of block. Other types of edges in the source and target meta-model include generalization relationships between blocks.

Figure 8.16 illustrates the source and target meta-model, and a source model for an Artifact transformation example in both traditional graph notation and in SysML notation—nodes shown as SysML blocks/instances and edges shown as associations/slot references. Figure 8.16a shows the source and target meta-model (same in this case) that represent three blocks—Artifact, Form, and Function, and the relationships between them. From a graph-schemata perspective, the blocks correspond to node types and the association relationships correspond to edge types. Thus, Figure 8.16a shows that source and target graphs instantiated from this graph schemata can have three types of labeled nodes (Artifact, Form, and Function) and two types of edges (hasForm, hasFunction) that originate from Artifact node and end in Form node and Function node respectively. These edges are also attributes of Artifact node, as Artifact.hasForm and Artifact.hasFunction respectively. Figure 8.16b shows a source graph that is an instance of the schemata shown in Figure 8.16a. The figure shows that the source graph has four nodes of type Artifact, two nodes of type Form, and three nodes of type Function. In addition, the edges between these nodes are also shown (though not labeled). The edges are also represented as attributes of each node. For example, the edge from A1 to F1 is represented as value of the attribute A1.hasForm.



a. Source and target graph schemata (traditional graph notation and SysML model notation)

b. Source graph (traditional graph notation and SysML model notation)

Figure 8.16: Source and target graph schemata and a source graph – Artifact transformation example

Note that although the edges in the source graph are not shown[21] as un-directed, they are directed. This is evident from the fact that the attributes hasForm and hasFunction are owned by Artifact. Thus the edges originate from Artifact instance nodes and end in Form and Function instance nodes respectively.

Graph patterns

Graph patterns represent conditions or constraints in a declarative manner, defined on graphs. Patterns are matched against a graph to check if they satisfy the conditions represented by patterns (Varro and Balogh 2007; VIATRA 2007). Fundamentally, pattern matching is a process of finding the occurrence of the graph pattern in a given graph, G. If a

[21] SysML instance specification does not represent the directed edges between instance entities (instance specifications).

graph L is a subgraph of G, it is denoted as L $\subseteq$ G and it implies that the (a) nodes and edges of L are subsets of the nodes and edges of G, (b) source and target mapping for each edge in L coincide with the source and target mappings for each edge in G, and (c) labels of nodes and edges in L coincide with labels of nodes and edges in G (Andries, Engels et al. 1999).

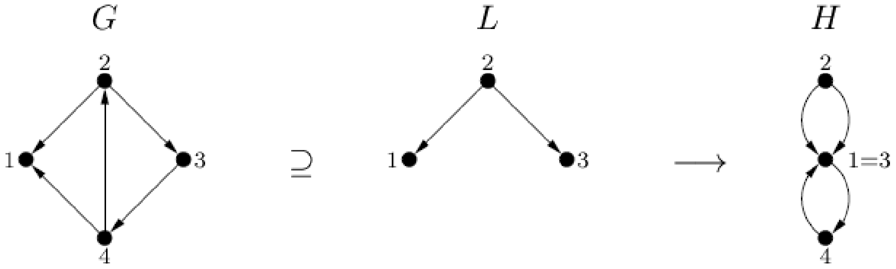


Figure 8.17: Graph L is a subgraph of G and has an occurrence in H (Andries, Engels et al. 1999)

The occurrence of a graph pattern L in a graph G is denoted as *occ:* L → G and implies that (a) there is a mapping which maps the nodes, edges, and labels of L to the nodes, edges and labels in G, (b) for each edge e in L the source of the image of e in G coincides with the image of the source of e in G and the target of the image of e in G coincides with the image of the target of e in G, and (c) for all nodes and edges in L, the label of their images in G coincide with the label of x (Andries, Engels et al. 1999). A bijective mapping is one where (a) each node and edge in L maps to a distinct node and edge in G—injective condition, and (b) all nodes and edges in G have atleast one corresponding node and edge in L—onto condition. If the mapping is bijective, then L and G are isomorphic. In Figure 8.17 above, graph L is a sub-graph of G and has an occurrence in H.

It is known that for the graph pattern matching problem, also known as the sub-graph isomorphism problem, the number of tests that need to be performed to check if a pattern with n nodes matches to a sub-graph in a graph with m nodes requires $O(m^n)$ tests in the worst case (Valiente and Martinez 1997). Research efforts in the past have improved the performance of graph pattern matching algorithms for specific types of graphs. All patterns defined in the Behavior Model Formulation Method (section 8.4) are defined for each type of relationship in the meta-models (CPM2_xKCM and CBM). Hence, all patterns have two nodes. Even for a source graph with large number of nodes, this approach restricts the number of tests that need to be performed in the worst case. In addition, all graph transformation rules defined in the Behavior Model Formulation reuse patterns.

The Behavior Model Formulation Method leverages the VIATRA Textual Command Language (VTCL) to define graph patterns. Figure 8.18 shows a graph pattern (artifact_and_form) for the Artifact transformation example described using VTCL. The pattern checks if there exists a relationship between an artifact and a form. As it can be seen, the pattern has three arguments A, F, and Model_Space that will be bound to nodes in the source graph. The Model_Space argument is used to define the scope of the source model—specifically the package where the source model exists.

```
pattern artifact_and_form(A,F, Model_Space)=
{
     entity(Model_Space);
     Artifact(A) in Model_Space;
     Artifact.hasForm(AF, A, F);
     Form(F) in Model_Space
}
```

Figure 8.18: An example graph pattern represented in VTCL

It is not stated if the arguments are inputs or outputs of the pattern thus making the pattern definition declarative. Thus, the pattern can be used to check for the following conditions or provide the following matches of interest:

- If variable A is bound to an artifact instance (node), the pattern can be used to find the form instance (node) associated with that artifact in the model space.
- If the variable F is bound to a form instance (node), the same pattern can be used to find all artifact instances (nodes) that have the subject form.
- If arguments A and F are bound to an artifact instance and a form instance respectively, the pattern can check if they are associated.
- If none of the arguments are bound to any instances, the pattern can be used to find all Artifact and Form instance pairs that are associated with each other.

Thus, a single graph pattern can be used to realize multiple queries and check for conditions. Typically, it would have taken four conditional statements (IF-ELSE) to realize the four use cases above in a procedural language (such as C, C++, or Java).

In the example above, the artifact_and_form pattern checks for structural conditions only—if two nodes and the relationship between them exist. Patterns can also be used to represent conditions that require checking specific attribute values of matched nodes. For example, patterns may be defined to check if the name, id, or other attribute values match a given value. In addition, patterns can call other functions that may be required to derive certain properties before checking them against a given value. For example, given a rectangle with length and breadth attribute values, a pattern can call a function to compute the area of a rectangle and check against a given value (equal, greater, or less).

To summarize the characteristics and the use cases of graph patterns:

- Graph patterns can be defined to check for structural conditions, such as if a node or an edge in the model graph is of a specific type. Examples of these types of conditions are illustrated in the example above.
- Graph pattern can also be used to check for conditions defined in terms of the attributes values of nodes in a model graph. For example, the check keyword in VIATRA allows for defining conditions that return a boolean value (true/false).
- Graph patterns can call each other using the find keyword. The condition in the caller pattern is satisfied only if the condition in the called pattern is satisfied and the local constructs in the caller patter are satisfied. Patterns can call themselves if certain conditions are satisfied, thus allowing for defining *recursive* patterns.
- Alternate graph patterns can be defined as sub-patterns within a parent pattern, such as by grouping them with the or keyword in VTCL. In this case, the condition in any of the sub-patterns must be satisfied for the condition in the parent pattern to be satisfied.
- Graph patterns can be called in a negative mode, such as by using the neg keyword in VTCL, to return true if the conditions embodied in them are not satisfied.
- If a variable passed to a graph pattern is unbound, graph patterns bind all possible model elements (to that variable) that satisfy the logical condition embodied in the pattern. If all variable passed to a graph pattern are bound to model elements, then the pattern returns true if the model elements bound to the variables satisfy the pattern condition, or false otherwise.
- One can also define the search scope for pattern conditions, such as specifying the namespaces where model elements should be searched.

Graph transformation rule

A ***graph transformation rule*** r = (L, R, App) contains a left-hand side (LHS) graph L, a right-hand side (RHS) graph R, and application conditions App. The application of r to a source (host) graph G replaces occurrence(s) of L in G by R. In general, this is performed by:

- finding occurrence(s) of L in G (also denoted as graph pattern matching)
- checking the application conditions App (such as negative application conditions which prohibit the application of the rule in the presence of certain nodes and edges)
- removing a part of the graph G determined by the occurrence(s) of L yielding the context graph D
- gluing R and the context graph D and obtaining the target (derived) graph H

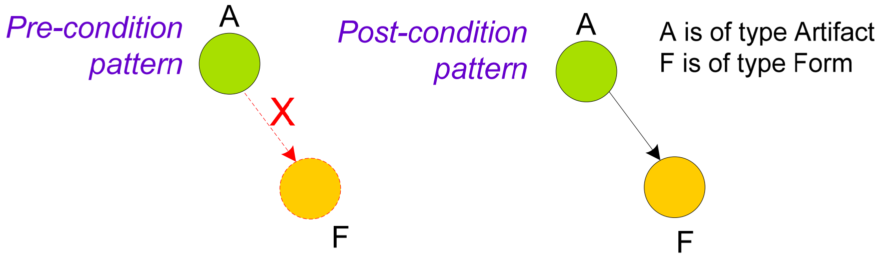
(Varro, Varro et al. 2002; Varro and Balogh 2007; VIATRA 2007)

Although the fundamental idea behind graph transformation rules is the same, graph transformation systems implement them differently and also provide different mechanisms to specify and control transformation rules. Typically, the occurrence of L in G is required to be isomorphic to L. The VIATRA graph transformation system checks for sub-graph isomorphism and provides a mechanism for parallel application of transformation rules (replacement of L with R) to all matches of L in G. This capability is especially relevant for variable topology problems where target model elements can be formulated in parallel for all sub-systems (sub-graphs) in the system model (source graph) that match with the pre-conditions of the idealization decisions.

A graph transformation rule is the atomic unit of model transformation in the Behavior Model Formulation Method. While graph patterns define the logical conditions on model graphs, graph transformation rules define the manipulation of model graphs. In this section, the representation of graph transformations in the context of Behavior Model Formulation Method is presented.

The Behavior Model Formulation Method leverages VIATRA Textual Command Language (VTCL) for representing graph transformation rules. Graph transformation rules represented in VTCL have two parts that are represented as patterns—the pre-condition pattern and a post-condition pattern. The LHS and RHS of a graph transformation rule are embodied in the pre- and post-condition patterns respectively. The application conditions of a graph transformation rule are embodied in patterns that may be called before invoking a transformation rule or included in the pre-condition pattern. The application of a graph transformation rule to a model (say source mode graph) replaces all matches of the pre-condition pattern in the source model graph with the post-condition pattern. The source model graph after the replacement operation is known as the target model graph.

Figure 8.19 illustrates a graph transformation rule with pre-condition and post-condition pattern represented in traditional graph notation and VTCL. The graph transformation rule is used in Artifact transformation example to initialize the form for all artifacts that do not have a form associated with them. The pre-condition pattern represents all artifact instances that do not have a form instance associated with them. Thus, the pre-condition pattern matches all artifact nodes such that for each artifact node there are no edges from the subject artifact node to a form node. For each artifact node matched by the pre-condition pattern, application of the post-condition pattern creates a form node and associates it with the artifact node.



```
// Initialize form for those artifacts in the model space
that do not have a form associated with them
gtrule init_form(in A, in Model_Space, out F, out AF)=
{
     precondition pattern check_artifact_form(A, Model_Space)=
     {
          entity(Model_Space);
          Artifact(A) in Model_Space;
          neg find artifact_xform_patterns.artifact_and_form(A, XF, Model_Space)
     }
     postcondition pattern initialize_form(A, Model_Space, F, AF)=
     {
          entity(Model_Space);
          Artifact(A) in Model_Space;
          Form(F) in Model_Space;
          Artifact.hasForm(AF,A,F);
     }
}
```

Figure 8.19: An example graph transformation rule represented in traditional graph notation and VTCL (used for initializing the form of an artifact in the Artifact transformation example)

In VTCL representation, the rule body begins with the gtrule keyword. The keywords precondition pattern and postcondition pattern along with the curly braces mark the pre-condition and post-condition patterns respectively. Note that pre-condition and post-condition patterns are graph patterns, and hence they may call other pre-defined patterns. In this example, the pre-condition pattern calls the artifact_and_form pattern in a negative mode using neg find keywords.

Graph transformation rules and their pre- and post-condition patterns may also have arguments. The rule arguments are identified as either inputs, outputs, or both. Input arguments are those that can be bound to model elements when the transformation rule is called while the output arguments are those that are bound to model elements as a result of applying the transformation rule and are available to be used in constructs calling the transformation rule, such as ASM rules that call graph transformation rules during the transformation process (described later). Arguments that are identified as both input and output can be pre-bound or bound when the rule is applied. VTCL keywords in, out, and inout are used to identify input, output, and input/output arguments respectively. The Behavior Model Formulation Method uses the following mechanism to create, delete, or preserve model elements when defining graph transformation rules using VTCL:

- For the creation of a new model element, a variable—to which the model element will be bound—should be in the argument and body of the post-condition pattern but not the pre-condition pattern argument or body. The variable may be identified as the output of the graph transformation rule.
- For deleting a model element, a variable—to which the model element will be bound—should be in the pre- and post-condition pattern arguments and pre-condition pattern body but not in the post-condition pattern body.
- For preserving a model element, a variable—to which the model element will be bound—should be in the pre- and post-condition pattern arguments and body.

If a parameter exists in both the pre-condition and post-condition arguments, then the model elements bound to that parameter during pre-condition pattern matching are passed to the post-condition. In the example transformation rule in Figure 8.19, the variables A and Model_Space exist in both the pre-condition and post-condition arguments and body, and hence model elements bound to them are not changed. However, variables F and AF exist only in the post-condition pattern arguments and body, and hence the model elements bound to them are created. There are no variables such that (a) they exist in the pre-condition pattern argument, body, and post-condition pattern argument, and (b) do not exist in the post-condition body, and hence no model elements are deleted when the transformation rule is applied.

As described in section 8.1, the model transformation (composition) process prescribed by the Behavior Model Formulation Method is one where the source model is not altered during the transformation, but instead the target model contains the source model and the additional models. Hence, source model elements are not deleted during this model transformation process. This is so because the Behavior Model Formulation Method uses the graph transformation-based approach to model transformation to synthesize simulation templates.

Graph transformation process

The Behavior Model Formulation Method uses VTCL constructs to define a graph transformation process. The transformation process describes the conditions and order in which the graph transformation rules are applied to the source model graph. In addition to providing constructs to define graph patterns and graph transformation rules, VTCL also provides constructs to define a control structure very similar to conventional programming languages such as C, C++, and Java. The VTCL constructs used for defining this control structure are known as ASM rules, named after Abstract State Machine constructs used in VTCL and similar to conventional programming languages. The ASM rules are similar to methods in object-oriented programming. In essence, the ASM rules in VTCL provide a mechanism to provide explicit scheduling to the model transformations—a pitfall of the graph transformation-based approach in its original form. The purpose of the Behavior Model Formulation Method is to create a specific behavior model structure and simulation template based on the Behavior Model Formulation Specifications—decisions taken by analysts. Hence, there is a specific need for controlling and scheduling the transformation rules. VTCL addresses this need by the virtue of ASM rules. In addition to constructs for calling and scheduling graph transformation rules, VTCL also defines other control structures constructs similar to conventional programming languages, such as an if-else construct.

Graph transformation rules can be called using two specific ASM rule constructs—forall and choose. While the former allows tracking and using all model elements bound to an output argument of a transformation rule, the latter allows tracking and using only one model element (selected non-deterministically). The Behavior Model Formulation Method uses the forall construct only. Figure 8.20 illustrates the forall ASM construct that is used for applying the init_form graph transformation rule (Figure 8.19) to the source model graph (Figure 8.16b).

```
// Call graph transformation rule “init_form”
forall F, AF with apply artifact_xform_rules.init_form(A, Model_Space, F, AF) do
     print("Initialized form");
```

Figure 8.20: VTCL ASM constructs used for defining model transformation process - shows the forall construct used for calling the init_form transformation rule in the Artifact transformation example

Figure 8.21 shows the transformed graph in the Artifact transformation example after the transformation process is executed by the VIATRA graph transformation engine. Specifically, this transformation is achieved by executing the forall ASM rule shown in Figure 8.20. Artifact node A4 in the source graph was the only artifact node that did not have an associated form node—no edges existed from A4 to any form node. After the transformation process execution, a form node F3 and an edge from A4 to F3 has been created. The edge creation is accompanied by the population of attribute A4.hasForm with value F3 (corresponding to the newly created form node object).

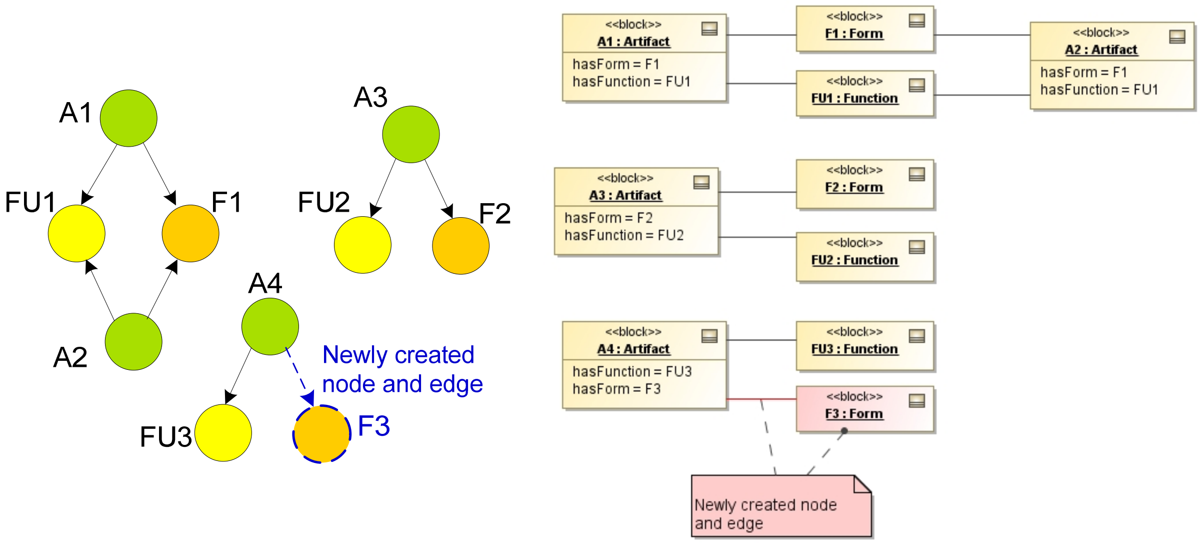
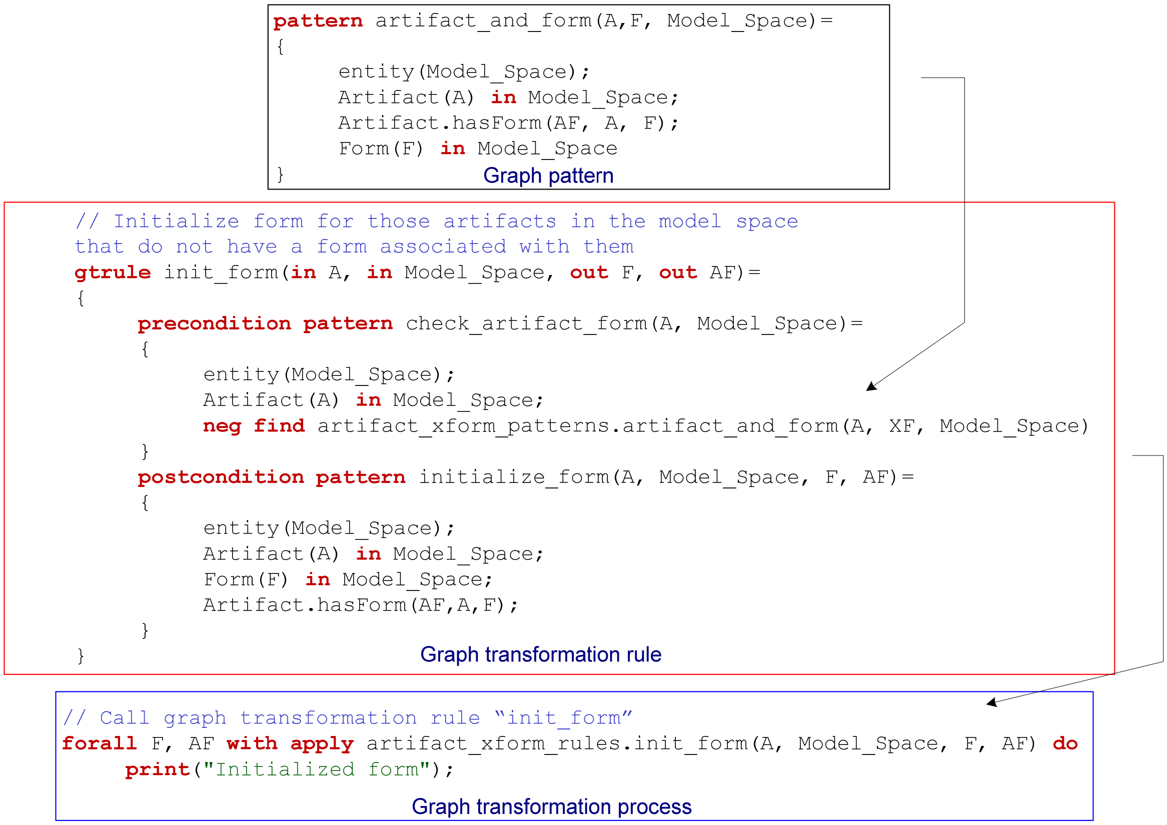


Figure 8.21: Target graph after the graph transformation process executed on the source graph for the Artifact transformation example (traditional graph notation and SysML notation)



```
pattern artifact_and_form(A,F, Model_Space)=
{
     entity(Model_Space);
     Artifact(A) in Model_Space;
     Artifact.hasForm(AF, A, F);
     Form(F) in Model_Space
}
```

Graph pattern

```
// Initialize form for those artifacts in the model space
that do not have a form associated with them
gtrule init_form(in A, in Model_Space, out F, out AF)=
{
     precondition pattern check_artifact_form(A, Model_Space)=
     {
          entity(Model_Space);
          Artifact(A) in Model_Space;
          neg find artifact_xform_patterns.artifact_and_form(A, XF, Model_Space)
     }
     postcondition pattern initialize_form(A, Model_Space, F, AF)=
     {
          entity(Model_Space);
          Artifact(A) in Model_Space;
          Form(F) in Model_Space;
          Artifact.hasForm(AF,A,F);
     }
}
```

Graph transformation rule

```
// Call graph transformation rule “init_form”
forall F, AF with apply artifact_xform_rules.init_form(A, Model_Space, F, AF) do
     print("Initialized form");
```

Graph transformation process

Figure 8.22: Summary of graph transformation approach to model transformations embodied in the Behavior Model Formulation method

To summarize, the key advantages of the graph transformation-based approach to model transformations as embodied in the Behavior Model Formulation Method are as follows:

- Graph patterns provide a mechanism to define conditions and constraints on source graphs in a declarative manner. The same pattern can be used to check if a source graph satisfies a set of conditions as well as to search for model elements that satisfy the conditions. The advantage of using this approach versus using a procedural approach is evident in the multiple use cases that may be addressed by the same pattern—depending upon which pattern arguments are bound to model elements and which are free.
- Graph transformation rules use graph patterns to define the atomic units of model transformations. The rules enable one to model transformations in a declarative rather than a procedural manner—as would be done using conventional procedural programming languages (such as C, C++, or Java). This is achieved by using graph patterns to model the state of sub-graphs before the transformation (pre-condition pattern) and the state of those sub-graphs after the transformation (post-condition pattern). The graph transformation engine can automatically interpret the transformation steps to achieve the final state of the graph.
- ASM rules use procedural programming language-like constructs to explicitly schedule graph transformation rules thereby enabling one to define a model transformation process with assured termination. The existence of a control structures makes it easier to define transformation processes (based on rule-based paradigm) that are testable, maintainable, and reliable (Li 1991).

Figure 8.22 summarizes the graph transformation approach to model transformations as embodied in the Behavior Model Formulation Method. Successful application and scalability of the graph transformation-based approach for complex design models will also depend on the availability of production-strength transformation tools.

The graph transformation approach is core to formulating simulation templates in an effective manner—addressing VTMB variations and idealization variations and efficiently formulating simulation templates. The formulation process is defined in terms of a graph transformation process that can be derived from the idealization specifications provided by analysts—see Behavior Model Formulation Specifications in the next section. Changes in idealization specifications result in changes in the graph transformation process used for formulating simulation templates. The ability to apply transformation rules in parallel to all artifacts (and their features and interactions) that satisfy specific conditions—modeled as graph patterns—enable formulation of simulation templates for VTMB problems. For the same idealization specifications, simulation templates can be automatically re-formulated for families of artifacts with non-equivalent assembly system topologies.

8.3 Behavior Model Formulation Specifications

The Behavior Model Formulation Specifications (BMFS) embody the idealization decisions taken by analysts. BMFS are defined using the Artifact Model Transformation Library and executed by the Transformation Engine to realize the model transformations leading to the creation of behavior model structures and simulation templates. Figure 8.23 shows a detailed view of the BMFS and its relationship with the Artifact Model Transformation Library.

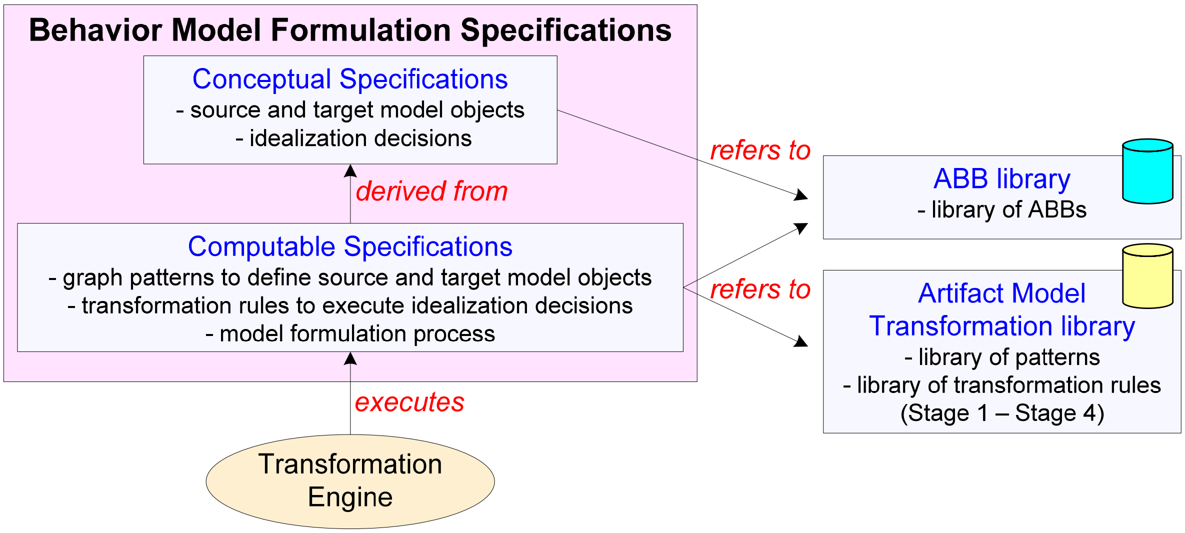

Behavior Model Formulation Specifications
Conceptual Specifications
- source and target model objects
- idealization decisions
derived from
Computable Specifications
- graph patterns to define source and target model objects
- transformation rules to execute idealization decisions
- model formulation process
executes
Transformation Engine
refers to
refers to
ABB library
- library of ABBs
Artifact Model Transformation library
- library of patterns
- library of transformation rules
(Stage 1 – Stage 4)

Figure 8.23: Detailed view of Behavior Model Formulation Specifications

The BMFS can be divided into the following two levels:

- Conceptual Specifications represent the idealization decisions independent of the transformation rules or process used to realize these decisions. Ideally, the same conceptual specifications may be realized by different transformation engines, transformation processes, and sets of transformation rules.

- Computable Specifications represent a transformation process in a syntax that is interpretable and executable by a specific transformation engine. The computable specifications are derived from the conceptual specifications and use transformation patterns and rules defined in the Artifact Model Transformation Library.

8.3.1 Conceptual Specifications

The conceptual specifications represent idealization decisions taken by analysts for all four stages of the composition process (section 8.2.1). In this section, the specific decisions that analysts need to take for each of four composition stages are presented.

Composition Stage 1: In this composition stage, analysis bodies and their relationships to analyzable artifacts are composed from their respective building blocks. The idealization decisions in this composition stage must specifically answer the following questions.

- What *analysis body ABBs* should be used to idealize the behavior of each type of analyzable artifacts?
- What *analysis feature ABBs* should be used for each of these analysis body ABBs, and how are these analysis features ABBs related to the analyzable features of the corresponding analyzable artifact(s)?
- What *shape ABBs* should be used for representing the shape of each of these analysis body ABBs, and how are these shape ABBs related to the shape of the corresponding analyzable artifact(s)?
- What *material behavior ABBs* are used for representing the material behavior of each of these analysis body ABBs, and how are these material behavior objects related to the material behavior of the corresponding analyzable artifact(s)?
- What *behavior ABBs* are used for representing the idealized set of behaviors of these analysis body? The behavior ABBs govern the set of behavior parameters that will be computed for these analysis bodies.

Note that the first question corresponds to the analysis body being composed in Stage 1, and the other four questions correspond to the attributes of analysis body that must be populated during the composition.

Note that material behavior and shape are two types of ABBs that are associated with both an analyzable artifact and an analysis body. An analyzable artifact may have is typically formulated for a large class of analysis problems, and may have multiple material behavior models (and shape models) of different fidelities associated with it. Thus, in answering two of the questions above regarding material behavior ABBs and shape ABBs should be used for analysis bodies, analysts will typically make decisions in three ways as shown in Table 8.1 below. For an analysis body, analysts can select one of the multiple material behavior ABBs (and shape ABBs) associated with the corresponding analyzable artifact(s). This is a special case of idealization where the idealization relationships represent equality. Alternatively, analysts can select a material behavior ABB (or shape ABB) for an analysis body and explicitly specify the idealization relationships between the material behavior ABB (or shape ABB) associated with analyzable artifact(s) and those associated with the analysis body. The third mechanism is when analysts specify conditions for selection or idealization, such as a If-Else condition.

Table 8.1: Modes of taking decisions on material behaviors and shapes of analysis bodies

Select	Selecting a material behavior ABB (or shape ABB) for an analysis body from the list of available material behavior model ABBs (or shape ABBs) associated with an analyzable artifact. Idealization relationships represent equality.
Idealize as ... relations ...	Selecting a material behavior ABB (or shape ABB) for an analysis body and establishing math relations between material behavior ABBs (or shape ABBs) associated with an analysis body and those associated with analyzable artifact(s). Idealization relationships represent these math relations.
If (condition) Select or Idealize... Else Select or Idealize...	Providing a condition for selecting or idealizing one type of material behavior ABB (or shape ABB) versus another type.

In general, conditions may be specified for all decisions taken by analysts in selecting ABBs for composition Stages 1-3. Figure 8.24 below illustrates how conceptual specifications may be represented formally using SysML Parametrics constructs. The figure shows analysts can define the pattern of the idealization relationship between an analyzable artifact and analysis body. When this "pattern" is applied for all analyzable artifacts, then these relationships will be created between all analyzable artifacts and analysis bodies, such as shown in Figure 7.34 in section 7.4.2 for relationships created between all stratums of an analyzable printed circuit board and corresponding planar shell analysis bodies. As an example, Figure 8.24 below illustrates such a pattern. The pattern shows shape idealization relationship (shape_idealization) and material behavior idealization relationship (mb_idealization) created between shape and material behaviors associated with an analyzable artifact and an analysis body. Per CPM2_xKCM Meta-Model (Chapter 6), an analyzable artifact may several forms associated with it; each form may have several shapes and materials associated with it; and each material may have several material behavior models associated with it. The Shape_Shape_Relationship and Material_Behavior_Material_Behavior_Relationship are constraint blocks that embody the mathematical relationships between associated shape and material behavior parameters respectively. Hence, such a pattern can be used to define all three cases in Table 8.1 above. The SysML constraint specifications shape_shape_relations and mb_mb_relations can represent math relations (including conditions).

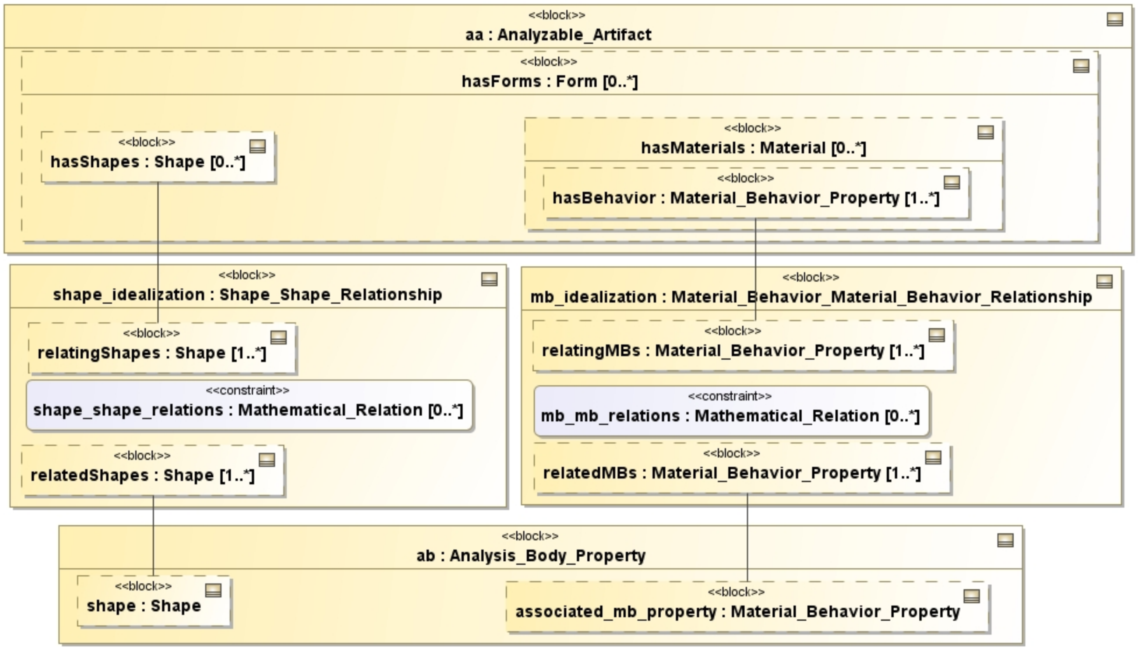

<<block>>
aa : Analyzable_Artifact
<<block>>
hasForms : Form [0..*]
<<block>>
hasShapes : Shape [0..*]
<<block>>
hasMaterials : Material [0..*]
<<block>>
hasBehavior : Material_Behavior_Property [1..*]
<<block>>
shape_idealization : Shape_Shape_Relationship
<<block>>
relatingShapes : Shape [1..*]
<<constraint>>
shape_shape_relations : Mathematical_Relation [0..*]
<<block>>
relatedShapes : Shape [1..*]
<<block>>
mb_idealization : Material_Behavior_Material_Behavior_Relationship
<<block>>
relatingMBs : Material_Behavior_Property [1..*]
<<constraint>>
mb_mb_relations : Mathematical_Relation [0..*]
<<block>>
relatedMBs : Material_Behavior_Property [1..*]
<<block>>
ab : Analysis_Body_Property
<<block>>
shape : Shape
<<block>>
associated_mb_property : Material_Behavior_Property

Figure 8.24: Representation of specifications using SysML Parametrics constructs

Figure 8.25 illustrates a view of the conceptual specifications defined by analysts. The model shown in the figure is a Level 3 VTMB Behavior Model (section 7.4.2). The figure illustrates how an analyzable multi-stratum PCB is idealized. In the context of the idealization questions stated above for Stage 1, the figure shows that an analyzable stratum of the analyzable PCB is idealized as a planar shell analysis body, the primary and secondary surfaces of the analyzable stratum are idealized as primary and secondary surfaces of the planar shells. The figure does not show the shape and material behavior of the analyzable stratums are idealized as the shape and material behavior of the planar shells.

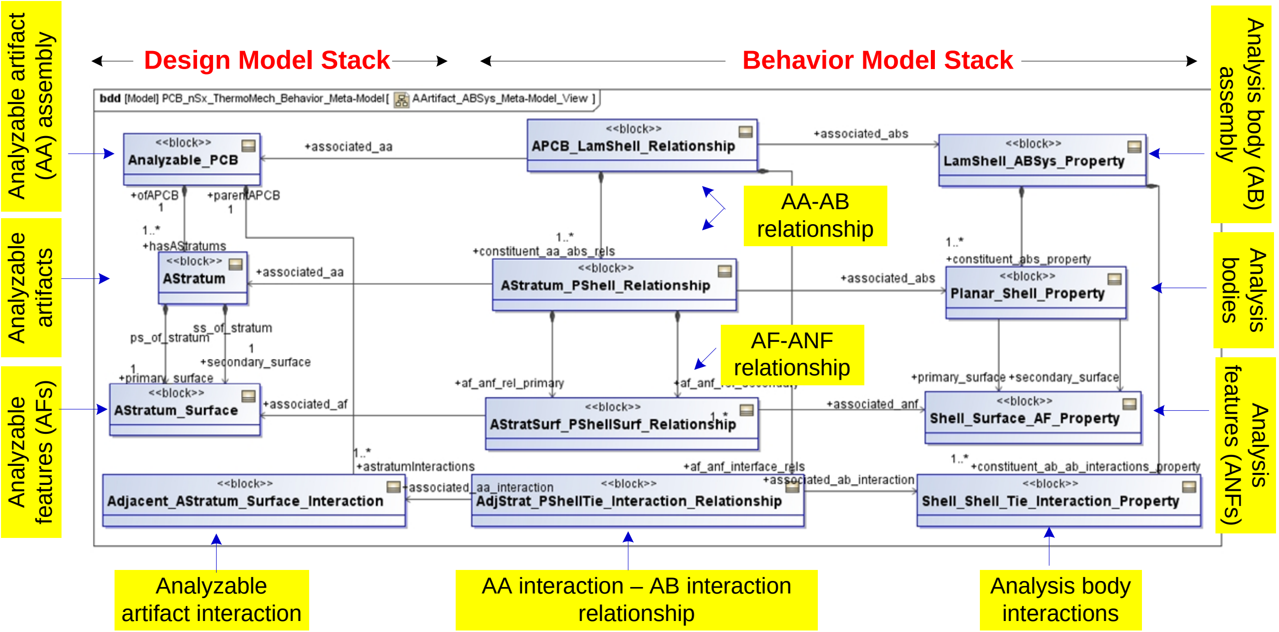

Analyzable artifact (AA) assembly
Analyzable artifacts
Analyzable features (AFs)
Design Model Stack
Behavior Model Stack
Analysis body (AB) assembly
Analysis bodies
Analysis features (ANFs)
bdd [Model] PCB_nSx_ThermoMech_Behavior_Meta-Model [AArtifact_ABSys_Meta-Model_View]
<<block>> Analyzable_PCB
<<block>> APCB_LamShell_Relationship
<<block>> LamShell_ABSys_Property
<<block>> AStratum
<<block>> AStratum_PShell_Relationship
<<block>> Planar_Shell_Property
<<block>> AStratum_Surface
<<block>> AStratSurf_PShellSurf_Relationship
<<block>> Shell_Surface_AF_Property
<<block>> Adjacent_AStratum_Surface_Interaction
<<block>> AdjStrat_PShellTie_Interaction_Relationship
<<block>> Shell_Shell_Tie_Interaction_Property
+associated_aa
+associated_abs
+ofAPCB 1
+parentAPCB 1
1..* +hasAStratums
1..* +constituent_aa_abs_rels
1..* +constituent_abs_property
+associated_aa
+associated_abs
ps_of_stratum
ss_of_stratum
1 +secondary_surface
1 +primary_surface
+af_anf_rel_primary
+af_anf_rel_secondary
+primary_surface +secondary_surface
+associated_af
+associated_anf
1..*
1..* +astratumInteractions
+af_anf_interface_rels
+associated_aa_interaction
+associated_ab_interaction
1..* +constituent_ab_ab_interactions_property
AA-AB relationship
AF-ANF relationship
Analyzable artifact interaction
AA interaction – AB interaction relationship
Analysis body interactions

Figure 8.25: View of the Conceptual Specifications for Stage 1 and 2 compositions - B3 model (PCB_nSx_ThermoMech_Behavior_Meta-Model from section 7.4.2)

Composition Stage 2: In this composition stage, analysis body systems and their relationships with analyzable artifact (assembly) are composed from their respective building blocks. The idealization decisions in this composition stage must specifically answer the following questions:

- What *analysis body system ABBs* are used for representing the idealized behavior of analyzable artifact assemblies, and how are these analysis body systems related to the corresponding analyzable artifact assemblies?
- What *analysis body ABBs* and *analysis body system ABBs* constitute the analysis body system being composed during this stage, and how are they related to the corresponding analyzable artifacts?

- What *analysis body interaction ABBs* are used for representing the behavior of the interaction between the analysis bodies used in composing analysis body systems, and how are these interactions related to the interactions between the corresponding analyzable artifacts?
- What *analysis feature ABBs* should be used to define the analysis features associated with the composed analysis body system?
- What *behavior ABBs* should be used for representing the idealized set of behaviors of the composed analysis body system?

Note that the first question corresponds to the analysis body system being composed in Stage 2, and the other four questions correspond to the attributes of analysis body system that must be populated during the composition.

In the context of the idealization questions stated above for Stage 2 composition, Figure 8.25 illustrates that the analyzable PCB is idealized as a laminated shell analysis body system, and the interaction between the any adjacent stratums of the analyzable PCB are idealized as tie interactions between the planar shell analysis bodies corresponding to the stratums. Note that SysML Parametrics constructs, as shown in Figure 8.24 can be used for formally representing conceptual specifications for Stage 2.

Composition Stage 3: In this composition, a behavior model ABB system is composed from its building blocks, and a behavior model context is created to wrap the relationship between the top level analysis body system and analyzable artifact (assembly). The idealization decisions in this composition stage must specifically answer the following questions:

- What *load ABBs* are used for representing the loads for which the behavior parameters are to be computed?
- What *behavior condition ABBs* are used for representing the behavior conditions for which the behavior parameters are to be computed?
- What *behavior ABBs* are used for representing the behavior parameters to be computed?

Note that SysML Parametrics constructs, as shown in Figure 8.24 can be used for formally representing conceptual specifications for Stage 3, such as selecting load and behavior condition ABBs based on certain conditions defined on analyzable artifact.

Composition Stage 4: In this composition, behavior model structure and simulation template are composed from the behavior model ABB system and behavior model context composed in Stage 3. Except for deciding the behavior model namespace and identifiers, there are no decisions that analysts need to take in this stage. The inputs and outputs of this composition stage are common to all VTMB analysis problems.

8.3.2 Computable Specifications

Computable specifications are model composition instructions that are derived from the conceptual specifications, and interpreted by the model transformation engine. While the conceptual specifications represent the idealization decisions taken by analysts, they do not prescribe a process for model composition. This is so because the idealization decisions are independent of the order in which the model is composed. The computable specifications are executable scripts that define a set of activities that can be executed in series or parallel. Each activity in the script comprises of the following two basic steps.

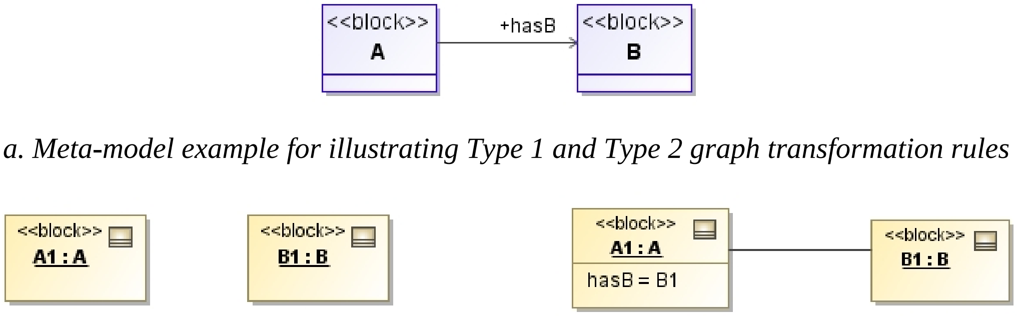
- Invoke pre-defined graph patterns from the Artifact Model Transformation Library to search for model elements in the source model. Graph pattern matches return sub-graphs of the source model graph that satisfy the conditions embodied in the patterns. As an example, for conceptual specifications that state that all stratums in a printed circuit board are to be idealized a shells, the computable specifications include calls to pre-defined graph patterns to search the printed circuit board model space and retrieve all stratums. The conditions specified in invoked patterns may include additional constraints that need to be satisfied by the model elements in the source graph.
- Create new model elements in the target model space by invoking graph transformation rules defined in the Artifact Model Transformation Library. The transformation rules may call pre-defined patterns to check for conditions before creating new model elements. The model elements created by graph transformation rules include both entities (nodes) and edges (relationships between entities).

Besides the basic restrictions posed by the stages of composing simulation templates—composition stage$_{i+1}$ cannot complete until stage$_i$ is completed—the process of formulating simulation templates is not necessarily relevant to an analyst, especially since the computer time taken to generate these templates is of the order of seconds (section 9.5.3.3). Algorithms to derive computable specifications from conceptual specifications would typically be managed by modelers proficient in the language in which the graph transformation process is described (such as VTCL in this case) and conceptual specifications.

In the proof-of-concept software implementation of the Behavior Model Formulation Method, the computable specifications are represented as a graph transformation processing using VTCL, as described in section 8.2.3.

8.4 Artifact Model Transformation Library (AMTL)

The Artifact Model Transformation Library of KCM's Behavior Model Formulation Method provides a repository of graph transformation rules that can be reused for writing Behavior Model Formulation Specifications in the computable form for analysis problems in general. The intent of the transformation library is to provide unit-level transformation rules that are generic for all behavior models, and ABB-specific transformation rules that are used when specific ABBs are used for composing a Behavior Model ABB System. The core of the model transformation method prescribed by the Behavior Model Formulation Method is the creation of simulation templates, and hence the two key types of graph transformation rules in the Artifact Model Transformation Library concern creation of entities, and creation of relationships between entities. Application results for these two types of transformation rules are illustrated using a simple model shown in Figure 8.26. Figure 8.26a shows an example meta-model. The meta-model shows two SysML blocks, A and B, and a relationship between the blocks A.hasB.



a. Meta-model example for illustrating Type 1 and Type 2 graph transformation rules

b. Application result of Type 1 transformation rules – creation of new entity instances in the model space

c. Application result of Type 2 transformation rules – creation of new relationship instances between entity instances in the model space

Figure 8.26: Example model to illustrate Type 1 and 2 graph transformation rules in the Artifact Model Transformation Library

Type 1 transformation rule: In this type of graph transformation rule, a new entity of type A with a given identifier ID is created in a given model space M. As an example, Figure 8.26b illustrates that new instances of entities—A1 as an instance of A and B1 as an instance of B—would be created in this type of transformation though there may be two different transformation rules—one for creating instance an instance of A and one for creating an instance of B. This type of rule corresponds to the creation of a node in the artifact model graph. The schematic of a Type 1 graph transformation rule is as described below. The input parameters ID and M are already bound to entities ID and M while the output parameter is unbound when the rule is called. After the execution of the rule, the output parameter A1 will be bound to entity A1.

Type 1 transformation rule (input: ID, M; output: A1)

Pre-condition

- there exists a model space M
- there does not exist an entity in M with id=ID

Post-condition

- there exists a model space M
- there exists an entity A1 of type A in M with id=ID

The Behavior Model Formulation Method maintains a unique id for the model entities and relationships, and hence the id of an entity is used to check for its existence in a given model space.

Type 2 transformation rule: In this type of graph transformation rule, a new relationship of type A.hasB is created between two given entities of type A and type B respectively in a given model space M. As an example, Figure 8.26c illustrates that Type 2 transformation is used for creating a new relationship between instances, A1.hasB=B1. This type of node corresponds to the creation of an edge in the artifact model graph. The schematic of a Type 2 transformation rule is as described below. The input parameters A1 and B1 are already bound to entities A1 and B1 while the output parameter is unbound when the rule is called. After the execution of the rule, the output parameter A1B1 will be bound to relationship A1B1.

Type 2 transformation rule (input: A1, B1, M; output: A1B1)

Pre-condition

- there exists a model space M
- there exists an entity A1 of type A in M
- there exists an entity B1 of type B in M
- there does not exist a relationship of type A.hasB from A1 to B1
  (or implemented as pattern A_and_B (A1, B1, M) returns false)

Post-condition

- there exists a model space M
- there exists an entity A1 of type A in M
- there exists an entity B1 of type B in M
- there exists a relationship A1B1 of type A.hasB from A1 to B1

In addition to transformation rules Type 1 and Type 2, the Artifact Model Transformation Library also has reusable patterns to check for the existence of entities and relationships, or to search for them in the model space. These patterns are used for Type 2 transformation rules in particular. For example, the last clause in the pre-condition of Type 2 rule could be implemented by calling *pattern_A_and_B* for entities A1 and B1 and checking that the pattern returns *false*. The pattern *pattern_A_and_B* is as shown below:

Pattern Type 1

pattern A_and_B (An, Bn, M)

- there exists a model space M
- there exists an entity An, of type A, in M
- there exists an entity Bn, of type B, in M
- there exists a relationship AnBn, of type A.hasB, in M

Though several types of patterns may be defined using the concept of graph patterns, Pattern Type 1 is commonly used in the Behavior Model Formulation Method. In this type of pattern, a condition is defined to check for the existence of a relationship between two given model elements—as shown in the example above. To make the pattern matching process computationally less expensive, all patterns defined in the Artifact Model Transformation Library are based on the following strategy:

- All patterns defined in Behavior Model Formulation Method are of Type 1—checking for a single relationship in the meta-models (CPM2_xKCM and CBM). Since all pre-defined patterns have 2 nodes, this restricts the number of tests that need to be performed when matching these patterns to sub-graphs in the source graph (Valiente and Martinez 1997).
- More complex patterns are realized by calls to simpler patterns.
- Wherever possible, patterns are invoked on specific sub-sets of the model space. This limits the number of nodes and/or edges in the model space for which pattern matching tests need to be performed.

Sections 8.4.1 to 8.4.4 describe the transformation rules and patterns for composition stages 1 to 4. Composition stages 1-3 consist of rules that are specific to the ABBs being composed and rules that are common to all behavior model structures and simulation templates composed using the Behavior Model Formulation Method. Rules and patterns specific to an ABB are attributes of the ABB itself. The ABB Meta-Model described in section 7.2.1 prescribes four key attributes of an ABB—context, property, application conditions, and transformation rules. When ABBs are instantiated, only their property attribute is populated. The other attributes are static—describe the ABB itself. The first two properties were defined in section 7.2.1 and described for each ABB type in section 7.3. The application conditions and transformation rules for each ABB are modeled as graph patterns and graph transformation rules. The ABB-specific rules and patterns described below are represented as application conditions and transformation rules for that ABB. The representation of both dynamic and static aspects of an ABB is a key distinguishing feature of the ABB Meta-Model with regards to existing approaches. While the static aspects—context and property attributes of an ABB—represent the characteristics of an ABB, the dynamic aspects—application conditions and transformation rules—describe how an ABB is to be used in the context of creating a behavior model.

The entities and relationships created in these transformation rules described below are created in a given model space. For brevity, this is not stated for each rule. The types (classifiers) of instances created in these transformations are entities defined in the Core Behavior Model (section 7.1) and the Core Product Model extended by KCM (Chapter 6).

8.4.1 Stage 1 composition - transformation rules and patterns

The set graph transformation rules for Stage 1 composition consists of rules that are common to the creation of all behavior model structures and simulation templates, and rules that are specific to the ABBs used in a given behavior model structure.

In this composition stage, an analysis body is composed from shape ABB, system ABB, analysis feature ABB, material behavior ABB, and behavior ABBs. In addition, a relationship between the composed analysis body and the corresponding analyzable artifact is created. This relationship is composed from the relationship between the shapes of analyzable artifact and analysis body, relationship between the material behavior of analyzable artifact and analysis body, and relationships between analyzable features and analysis features. The Type 1 and Type 2 transformation rules in the Artifact Model Transformation Library for Stage 3 composition are described below. The name of the rule is followed by a short description of its specific purpose.

Type 1 transformation rules

- Rules for initializing analysis body ABB instances - These transformation rules are specific to the analysis body ABBs used in composing a given behavior model structure. Example of analysis body ABBs are illustrated in Figure 7.10. For example, Beam, Rod, Shell, and Column are different types of structural analysis body ABBs. An initialization rule (Type 1) would exist for each of the analysis body ABBs in the Artifact Model Transformation Library (AMTL). For example, initalize_planar_shell_analysis_body is a rule to initialize an instance of Planar_Shell_Property (type of Planar_Shell_ABB.property), as shown in Figure 7.11. The initialization transformation rule and associated patterns are attributes of the specific analysis body ABBs.
- Rules for initializing shape ABB instances – Similar to transformation rules for initializing analysis body instances, the Artifact Model Transformation Library would contain rules for initializing different types of shape ABBs. For example, initialize_planar_shape is a rule in the AMTL to initialize an instance of Planar_Shape.
- Rules for initializing analysis feature ABB instance - The Artifact Model Transformation Library would contain rules for initializing different types of analysis feature ABBs shown in Figure 7.18. For example, initialize_shell_surface_af is a rule in the AMTL to initialize an instance of Shell_Surface_AF_Property entity shown in the figure.
- Rules for initializing material behavior ABB instance - The Artifact Model Transformation Library would contain rules for initializing different types of material behavior ABBs as shown in Figure 7.12. For example, initialize_linear_elastic_tempind_mb is a rule in the AMTL to initialize an instance of Linear_Elastic_Isotropic_TempInd_MB_Property as shown in Figure 7.13.
- Rules for initializing behavior ABB instance - The Artifact Model Transformation Library would contain rules for initializing different types of behavior ABBs. For example, initialize_structural_behavior is a rule in the AMTL to initialize an instance of Structural_Behavior_Property as shown in Figure 7.17.
- initialize_aa_abs_relationship - used for creating an instance of Analyzable_Artifact_ABS_Relationship for associating an analysis body system with an analyzable artifact. Note that this rule is sued for both Stage 1 and Stage 2 composition.

In Stage 1 composition, relationships are created between analyzable artifact and analysis body.

- initialize_af_anf_relationship - used for creating an instance of Anlyzable_Feature_Analysis_Feature_Relationship for associating an analyzable feature with an analysis feature. Note that this rule is used in both Stage 1 and Stage 2 compositions. In Stage 1 composition, relationships are created between analyzable features and analysis features corresponding to an analyzable artifact and analysis body.
- initialize_shape_shape_relationship - used for creating an instance of Shape_Shape_Relationship that associates two shape instances
- initialize_material_behavior_material_behavior_relationship - used for creating an instance of Material_Behavior_Material_Behavior_Relationship that associates two instances of material behavior ABBs.

Type 2 transformation rules

- Rules for populating attributes of analysis body ABBs – These rules are used for populating attributes of analysis body ABB instances. The Artifact Model Transformation Library would have rules for associating an analysis body ABB instance with (a) shape ABB instance, (b) material behavior ABB instance, (c) analysis feature ABB instance, and (d) behavior ABB instance. Depending upon their specialization, analysis body ABBs may have their own specialized association rules. For example, a planar shell analysis body ABB will have rules to associate its instances with (a) planar shape ABB instances (and not any shape instance), and (b) two planar shell surface analysis feature ABB instances corresponds to its primary and secondary surface respectively. However, a material behavior is not inherent in the definition of a planar shell analysis body ABB, and hence it may use the generic rule that associates an analysis body ABB with a material behavior ABB.
- Rules for populating attributes of Analyzable_Artifact_ABS_Relationship that relates an analyzable artifact with an analysis body system (or analysis body), are described below.
    - associate_aa_abs_rel_with_aa_and_abs – used for creating an instance of the relationships Analyzable_Artifact_ABS_Relationship.associated_aa and Analyzable_Artifact_ABS_Relationship.associated_abs that relate an instance of Analyzable_Artifact_ABS_Relationship to an instance of Analyzable_Artifact and Analysis_Body_System_Property (or Analysis_Body_Property) respectively. This transformation rule associates an analysis body system composed during Stage 2 composition with the corresponding analyzable artifact (assembly).
    - associate_aa_abs_rel_with_af_anf_rel – used for creating an instance of the relationship Analyzable_Artifact_ABS_Relationship.af_anf_rels that relates an instance of Analyzable_Artifact_ABS_Relationship with an instance of Analyzable_Feature_Analysis_Feature_Relationship. Here, the analyzable feature-analysis feature relationship instances are defined between analyzable features of analyzable artifact assembly and analysis body system.
    - associate_aa_abs_rel_to_geom_idealization – used for creating an instance of the relationship Analyzable_Artifact_ABS_Relationship.shape_idealization that relates an

instance of Analyzable_Artifact_ABS_Relationship with an instance of Shape_Shape_Idealization

  - associate_aa_abs_rel_to_mb_idealization - used for creating an instance of the relationship Analyzable_Artifact_ABS_Relationship.material_behavior_idealization that relates an instance of Analyzable_Artifact_ABS_Relationship with an instance of Material_Behavior_Material_Behavior_Idealization
- Rules for populating attributes of Analyzable_Feature_Analysis_Feature_Relationship instance are described below.
  - associate_af_anf_rel_with_af_and_anf – used for creating an instance of each of the following two relationships: Analyzable_Feature_Analysis_Feature_Relationship.associated_af and Analyzable_Feature_Analysis_Feature_Relationship.associated_anf. These relationships relate an instance of Analyzable_Feature_Analysis_Feature_Relationship to an instance of Analzable_Feature and Analysis_Feature_Property respectively.
  - Associate_af_anf_rel_with_shape_idealization - used for creating an instance of the relationship Analyzable_Feature_Analysis_Feature_Relationship.shape_idealization that relates an instance of Analyzable_Feature_Analysis_Feature_Relationship to an instance of Shape_Shape_Relationship
- Rules for populating attributes of Shape_Shape_Relationship instance are described below.
  - associate_ssr_with_relating_shape_and_related_shape – used for creating an instance of each of the following two relationships: Shape_Shape_Relationship.relatedShapes and Shape_Shape_Relationship.relatingShapes. These relationships relate an instance of Shape_Shape_Relationship to an instance of Shape and Shape respectively.
  - associate_ssr_with_idealization_relation – used for creating an instance of the relationships Shape_Shape_Relationship.shape_shape_relations. This relationship associates an instance of Shape_Shape_Relationship to an instance of Mathematical_Relation.
- Rules for populating attributes of Material_Behavior_Material_Behavior_Relationship instance are defined similar to those defined for populating attributes of Shape_Shape_Relationship instances.

Type 1 patterns

- aa_abs_rel_and_aa - used for relationship Analyzable_Artifact_ABS_Relationship.associated_aa
- aa_abs_rel_and_abs – used for relationship Analyzable_Artifact_ABS_Relationship.associated_abs
- aa_abs_rel_and_shape_shape_idealization – used for relationship Analyzable_Artifact_ABS_Relationship.shape_idealization
- aa_absys_rel_and_mb_idealization – used for relationship Analyzable_Artifact_ABS_Relationship.material_behavior_idealization
- aa_abs_rel_and_af_anf_rel – used for relationship Analyzable_Artifact_ABS_Relationship.af_anf_rels
- af_anf_rel_and_af – used for relationship Analyzable_Feature_Analysis_Feature_Relationship.associated_af
- af_anf_rel_and_anf – used for relationship Analyzable_Feature_Analysis_Feature_Relationship.associated_anf

8.4.2 Stage 2 composition – transformation rules and patterns

The set graph transformation rules for Stage 2 composition consists of rules that are common to the creation of all behavior model structures and simulation templates, and rules that are specific to the ABBs used in a given behavior model structure.

In this composition stage, an analysis body system is composed from analysis body ABBs or other analysis body sub-systems, analysis body interaction ABBs, and analysis feature ABBs. Additionally, idealization relationships are created between (i) analyzable artifact (assembly) and the composed analysis body system, (ii) analyzable features and analysis features, and (iii) the interaction between analyzable artifacts and interactions between analysis bodies in the analysis body system.

The Type 1 and Type 2 transformation rules in the Artifact Model Transformation Library for Stage 3 composition are described below. The name of the rule is followed by a short description of its specific purpose.

Type 1 transformation rules

- initalize_AB_System - used for creating an instance of Analysis_Body_System_ABB or any of its specializations
- initialize_aa_abs_relationship – used for creating an instance of Analyzable_Artifact_ABS_Relationship for associating an analysis body system with an analyzable artifact. In contrast to its usage in Stage 1 composition, this rule is used in Stage 2 composition to create relationships between analyzable artifact (assembly) and analysis body system respectively.
- initialize_af_anf_relationship - used for creating an instance of Anlyzable_Feature_Analysis_Feature_Relationship for associating an analyzable feature with an analysis feature. In contrast to its usage in Stage 1 composition, this rule is used in Stage 2 composition to create relationships between analyzable features and analysis features corresponding to analyzable artifact (assembly) and analysis body system respectively.
- initialize_aa_ab_interaction_relationship - used for creating an instance of Anlyzable_Feature_Analysis_Feature_Interface_Relationship for associating an interface between two analyzable artifacts with the corresponding interface between two analysis bodies—either directly or as part of interacting analysis body assemblies

Type 2 transformation rules

- Rules for populating attributes of Analysis_Body_System_ABB or its specializations are described below. Specialized rules may be defined for populating attributes of specializations.
  - associate_absys_with_abs - used for creating an instance of the relationship Analysis_Body_System_Property.constituent_abs_property that relates an instance of Analysis_Body_System_Property[22] and an instance of Analysis_Body_System_Property or Analysis_Body_Property

[22] Note that when ABBs are instantiated, only their property attribute (non-static) is populated. The property attribute of each ABB is of a specific type. For example, `Analysis_Body_ABB.property` is of type `Analysis_Body_Property`.

  - associate_absys_with_abi - used for creating an instance of the relationship Analysis_Body_System_Property.constituent_ab_ab_interactions_property that relates an instance of Analysis_Body_System_Property and an instance of AB_AB_Interaction_Property
  - associate_abs_with_anf - used for creating an instance of the relationship Analysis_Body_System_Property.constituent_afs_property that relates an instance of Analysis_Body_System_Property or an instance of Analysis_Body_Property with an instance of Analysis_Feature_Property
- Rules for populating attributes of Analyzable_Artifact_ABS_Relationship that relates an analyzable artifact with an analysis body system (or analysis body), are described below.
  - associate_aa_abs_rel_with_aa_and_abs – same as described in Stage 1 composition but in Stage 2 composition, this rule associates an analysis body system with the corresponding analyzable artifact (assembly).
  - associate_aa_abs_rel_to_constituent_aa_abs_rel - used for creating an instance of the relationship Analyzable_Artifact_ABS_Relationship.constituent_aa_abs_rels that relates an instance of Analyzable_Artifact_ABS_Relationship with an instance of Analyzable_Artifact_ABS_Relationship.
  - associate_aa_abs_rel_with_af_anf_rel – same as used in Stage 1 composition but in Stage 2 composition, the rule is used to relate an analyzable feature-analysis feature relationship instance with analyzable features of analyzable artifact (assembly) and analysis body system.
  - associate_ aa_abs_rel_with_aa_ab_interaction_rel - used for creating an instance of the relationship Analyzable_Artifact_ABS_Relationship.af_anf_interface_rels that relates an instance of Analyzable_Artifact_ABS_Relationship with an instance of Analyzable_Feature_Analysis_Feature_Relationship.
- Rules for populating attributes of Analyzable_Feature_Analysis_Feature_Relationship instance are same as described in Stage 1 composition except that in Stage 2 composition, they are invoked for associating analyzable features of analyzable artifact (assemblies) and analysis features of analysis body systems.
- Rules for populating attributes of Analyzable_Feature_Analysis_Feature_Interface_Relationship are described below.

- associate_aa_ab_interaction_relationship_to_aaaai_and_ababi – used for relating an instance of the relationship Analyzable_Feature_Analysis_Feature_Interface_Relationship.associated_aa_interaction and Analyzable_Feature_Analysis_Feature_Interface_Relationship.associated_ab_interaction that relate an instance of Analyzable_Feature_Analysis_Feature_Interface_Relationship to an instance of AA_AA_Interaction and AB_AB_Interaction_Property respectively

Type 1 patterns

- aa_abs_rel_and_constituent_aa_abs_rel - used for relationship Analyzable_Artifact_ABS_Relationship.constituent_aa_abs_rels
- aa_abs_rel_and_aa_ab_irel - used for relationship Analyzable_Artifact_ABS_Relationship.af_anf_interface_rels
- absys_and_abs - used for relationship Analysis_Body_System_Property.constituent_abs_property
- absys_and_abi - used for relationship Analysis_Body_System_Property.constituent_ab_ab_interactions_property
- absys_and_anf - used for relationship Analysis_Body_System_Property.constituent_anf_property
- aa_ab_irel_and_aaaai - used for relationship Analyzable_Feature_Analysis_Feature_Interface_Relationship.associated_aa_interaction
- aa_ab_irel_and_ababi - used for relationship Analyzable_Feature_Analysis_Feature_Interface_Relationship.associated_ab_interaction

8.4.3 Stage 3 composition – transformation rules and patterns

The set graph transformation rules for Stage 3 composition consists of rules that are common to the creation of all behavior model structures and simulation templates, and rules that are specific to the ABBs used in a given behavior model structure.

In this composition stage, (a) a behavior model ABB system is composed from analysis body system ABB, load ABBs, behavior condition ABBs, and behavior ABBs, and (b) a behavior model context is composed from the relationship between the analysis body system ABB and an analyzable artifact. A behavior model ABB system represents the idealized behavior of the analyzable artifact. The analyzable artifact is idealized as an analysis body system. The Type 1 and Type 2 transformation rules in the Artifact Model Transformation Library for Stage 3 composition are described below. The name of the rule is followed by a short description of its specific purpose.

Type 1 transformation rules

- initialize_behavior_model_abbsys - used for creating an instance of Behavior_Model_ABBSys
- initialize_behavior_model_xcontext - used for creating an instance of Behavior_Model_XContext
- Rules to initialize load ABB instances –These transformation rules are specific to the load ABBs used in composing a given behavior model structure. Example of load ABBs are illustrated in Figure 7.23. For example, Force, Pressure, Moment, and Temperature are different types of structural load ABBs. An initialization rule (Type 1) would exist for each of the load ABBs in the Artifact Model Transformation Library. For example, initalize_uniform_temp_load is a rule to initialize an instance of Uniform_Temperature_Load_ABB (shown in Figure 7.24).
- Rules to initialize behavior condition ABB instances – Similar to load ABBs, these transformation rules are specific to behavior condition ABBs used in composing a given behavior model structure. Example of behavior condition ABBs are illustrated in Figure 7.25. For example, PointDisplacementFixed_Condition ABB and TemperatureConstant_Condition ABB are two types of behavior condition ABBs. In the former, the displacement is locked at a given point, and in the latter the temperature is held constant. An initialization rule (Type 1) would exist for each of the behavior condition ABBs in the Artifact Model Transformation Library. For example, initalize_point_displacement_fixed_BC is a rule to initialize an instance of PointDisplacementFixed_Condition ABB.

Type 2 transformation rules

- Rules for populating attributes of Behavior_Model_ABBSys instance – These rules are used for populating the attributes of Behavior_Model_ABBSys instance. There are four such rules, one for each attribute of Behavior_Model_ABBSys, as described below:
    - associate_behavior_model_abbsys_with_absys - used for creating an instance of the relationship Behavior_Model_ABBSys.abs_sys that relates an instance of Behavior_Model_ABBSys and an instance of Analysis_Body_System_ABB
    - associate_behavior_model_abbsys_with_load - used for creating an instance of the relationship Behavior_Model_ABBSys.load_applications that relates an instance of Behavior_Model_ABBSys and an instance of Load_ABB
    - associate_behavior_model_abbsys_with_bc - used for creating an instance of the relationship Behavior_Model_ABBSys.behavior_condition_applications that relates an instance of Behavior_Model_ABBSys and an instance of Behavior_Condition_ABB
    - associate_behavior_model_abbsys_with_behavior - used for creating an instance of the relationship Behavior.behaviors that relates an instance of Behavior_Model_ABBSys and an instance of Behavior_ABB
- Rules for populating attribute of Behavior_Model_XContext instance – There is one rule for populating the single attribute of Behavior_Model_XContext, as described below.
    - associate_bmx_context_with_aa_absys_rel - used for creating an instance of the relationship Behavior_Model_XContext.aa_abs_rel that relates an instance of Behavior_Model_ABBSys and an instance of Analyzable_Artifact_ABS_Relationship
- Rules for populating attributes of load ABBs – These rules are used for populating attributes of load ABBs instances in the behavior model structure. There are three attributes of all load ABB properties—load parameters, load application domain, and load distribution function. A rule to populate the application domain attribute of each type of Load_ABB would be defined in the Artifact Model Transformation Library. For example the rule associate_utl_with_vf is used for associating an instance of Uniform_Temperature_Load_ABB with an instance of Volume_Feature_ABB, since temperature is a volume load. Note that load parameters and the distribution function are typically defined for each type of load ABB. However, the specific values in the distribution function may be populated when creating behavior model instance (Level 5

model). For example, for Uniform_Temperature_Load_ABB, the load parameter (temperature) and distribution function (temperature=constant) are inherently decided in the definition of the ABB but the value of the constant may be populated only when creating behavior model instances.

- Rules for populating attributes of behavior condition ABBs – Similar to load ABBs, there are three attributes of all behavior condition ABB properties—behavior condition parameters, application domain, and distribution function. A rule to populate the application domain attribute of behavior condition ABB would be defined in the Artifact Model Transformation Library. For example, the rule associate_pdc_with_pf is used for associating an instance of PointDisplacementFixed_Condition and an instance of Point_Feature_ABB. The behavior condition parameters and distribution function are inherently pre-decided for specific type of behavior condition ABB. For example, the behavior parameters for PointDisplacementFixed_Condition ABB are displacement parameters, and the distribution function is displacement=constant, though the value of the constant may be populated only when creating behavior model instances.

Type 1 patterns

The following Type 1 patterns are defined in the Artifact Model Transformation Library that are typically used for Stage 3 composition

- behavior_model_abbsys_and_absys - used for relationship Behavior_Model_ABBSys.abs_sys
- behavior_model_abbsys_and_load - used for relationship Behavior_Model_ABBSys.load_applications
- behavior_model_abbsys_and_bc - used for relationship Behavior_Model_ABBSys.behavior_condition_applications
- behavior_model_xcontext_and_aa_absys_rel - used for relationship Behavior_Model_XContext.aa_abs_rel

8.4.4 Stage 4 composition – transformation rules and patterns

The graph transformation rules for Stage 4 composition are common to the creation of all behavior model structures and simulation templates formulated using the Behavior

Model Formulation Method. In this composition stage, a behavior model structure and simulation template are composed from a behavior model ABB system—a system of analysis building blocks that represents the idealized behavior of an analyzable artifact, and a behavior model context—relates the analysis body system in the ABB system to the analyzable artifact.

The types (classifiers) of instances created in these transformations are entities defined in the Core Behavior Model (section 7.1). One of the entities, Behavior, is defined in the Core Product Model extended for KCM—CPM2_xKCM (Chapter 6). The Type 1 and Type 2 transformation rules in the Artifact Model Transformation Library for Stage 4 composition are described below. The name of the rule is followed by a short description of its specific purpose.

Type 1 graph transformation rules

- initialize_behavior_model - used for creating an instance of Behavior_Model

Type 2 graph transformation rules

- associate_behavior_model_with_behavior - used for creating an instance of the relationship Behavior.behaviorModels that relates an instance of Behavior and an instance of Behavor_Model
- Rules for populating attributes of Behavior_Model – These rules are used for populating the two attributes of Behavior_Model, and are described below.
  - associate_behavior_model_with_bmabbsys - used for creating an instance of the relationship Behavior_Model.associated_bm_abbsys that relates an instance of Behavior_Model and an instance of Behavor_Model_ABBSys.
  - associate_behavior_model_with_bmxcontext - used for creating an instance of the relationship Behavior.context that relates an instance of Behavior_Model and an instance of Behavor_Model_XContext

Type 1 patterns

The following Type 1 patterns are defined in the Artifact Model Transformation Library that are typically used for Stage 4 composition

- behavior_model_and_behavior_model_abbsys - used for relationship Behavior_Model.associated_bm_abbsys
- behavior_model_and_behavior_model_xcontext - used for relationship Behavior_Model.context

8.4.5 Analyzable artifact model patterns

When composing behavior model structures and simulation templates, entities in the analyzable artifact model need to be unambiguously identified. The Behavior Model Formulation Specifications in both conceptual and computable forms need to explicitly state the analyzable artifacts (or their specific aspects) and the conditions that need to be satisfied before associating behavior model entities. In the Behavior Model Formulation Method, the identification criteria and conditions are formally represented as graph patterns. For example, if stratums of a PCB is idealized as shell, this requires that all stratums of the PCB be unambiguously identified and then transformation rules be executed to initialize shell analysis body ABB and the relationships to the stratums. In addition to identifying analyzable artifacts or their specific aspects, there may be conditions that need to be checked. For example, stratums made of conductive material are idealized to have linear isotropic material behavior, and stratums made of non-conductive material are idealized to have linear orthotropic material behavior.

The Artifact Model Transformation Library would also have patterns for relationships in the analyzable artifact model. Depending upon the variable bindings when the pattern is called, a pattern could be used to search and identify analyzable artifact entities or check for specific conditions. The entities and relationships are specialized for each application domain and hence patterns are created for meta-model defined at Level 2 in the design model stack (section 6.2). Figure 6.7 and Figure 6.8 show the design and analyzable design models for printed circuit boards. As an example, for the relationships in the analyzable PCB model illustrated in Figure 6.8, the following Type 1 patterns are defined in the Artifact Model Transformation Library.

- astratums_and_apwb - used for the relationship Analyzable_PCB.hasStratums
- astratum_interfaces_and_apwb - used for the relationship Analyzable_PCB.astratumInteractions

- astratum_and_surfaces - used for the relationships AStratum.primary_surface and AStratum.secondary_surface
- assi_and_preceding_stratum_surface - used for the relationship Adjacent_AStratum_Surface_Interaction.precedingAstratumSurface
- assi_and_succeding_stratum_surface - used for the relationship Adjacent_AStratum_Surface_Interaction.succeedingAstratumSurface
- astratum_and_form - used for the relationship AStratum.hasForms
- astratum_and_shape - used for the relationship AStratum_Form.hasShapes (where AStratum.hasForms: AStratum_Form)
- astratum_and_material - used for the relationship AStratum_Form.hasMaterial (where AStratum.hasForms: AStratum_Form)
- astratum_and_elec_function - used for the relationship AStratum.hasFunctions

8.5 Summary

To summarize, the Behavior Model Formulation Method (BMFM) of the Knowledge Composition Methodology is presented in this chapter. Specifically, the following aspects of the model transformation process prescribed by BMFM are presented here.

- *Schematics* of the transformation process focuses on the functional components of the transformation framework—source and target meta-models and models, transformation specifications, model transformation library, and the model transformation engine.
- *Stages* of the transformation process focuses on the major steps in which simulation templates are composed from design model structures and idealization decisions.
- *Semantics* of the transformation process relates the process of composing simulation templates to deriving relations between behavior parameters and design parameters. The intent of presenting this aspect is to illustrate that the BMFM is a formal and structured approach to creating simulation templates that embody existing fundamental domain theories. The BMFM provides a computationally effective mechanism to apply existing domain theories and concepts to variable topology multi-body problems.
- *Mechanics* of the transformation process focuses on how the model transformation process is realized as a process of graph transformations.

In addition, pre-defined graph patterns and transformation rules in the Artifact Model Transformation Library are presented.

PART 3: VERIFICATION & VALIDATION, FUTURE WORK, AND CLOSURE

Chapter 9 : TEST CASES

The focus of this chapter is to present test applications of KCM meta-models and methods, and to validate the research hypotheses. In this chapter, test cases are presented to demonstrate different aspects of the Knowledge Composition Methodology. The test cases validate the model composition process prescribed by KCM's Behavior Model Formulation Method. Two families of test cases are presented here. In the first test case family (TCF1) presented in section 9.2, the objective is to generate fixed topology simulation templates for thermo-mechanical analyses of multi-layered printed wiring boards. For the second test case family (TCF2) presented in section 9.3, the objective is to generate fixed topology simulation templates for thermo-mechanical analyses of ball grid array (BGA) chip packages. The test cases demonstrate automated generation of simulation templates for two types of variations: (a) analyzable design model structures with different assembly system topologies (VTMB problems), and (b) idealization decisions taken by analysts.

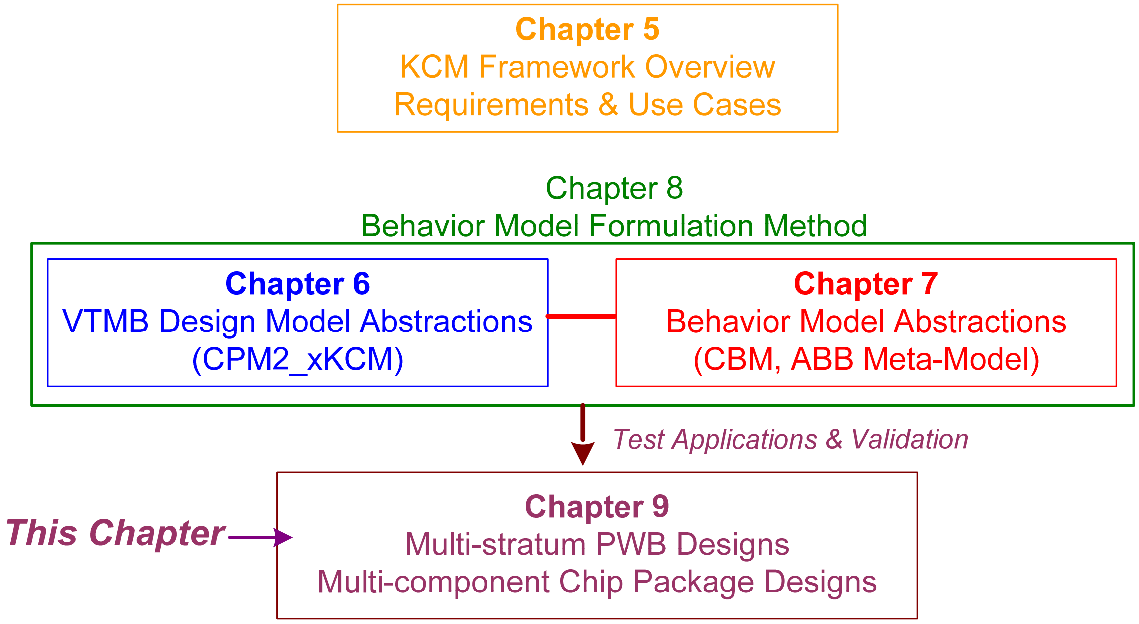
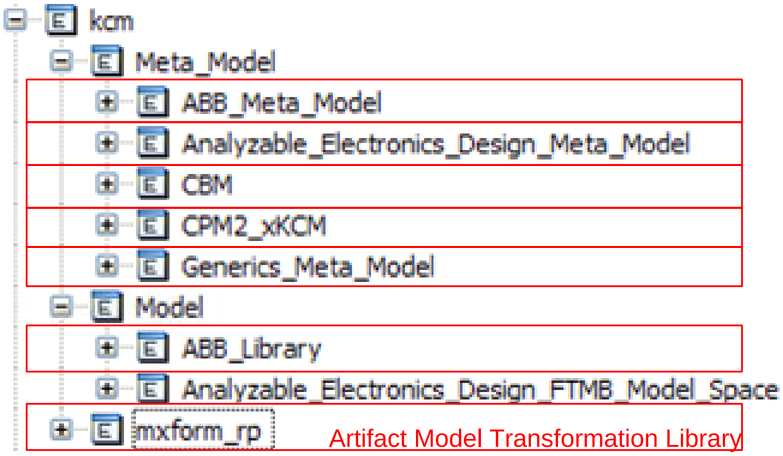

Chapter 5
KCM Framework Overview
Requirements & Use Cases
Chapter 8
Behavior Model Formulation Method
Chapter 6
VTMB Design Model Abstractions
(CPM2_xKCM)
Chapter 7
Behavior Model Abstractions
(CBM, ABB Meta-Model)
Test Applications & Validation
This Chapter
Chapter 9
Multi-stratum PWB Designs
Multi-component Chip Package Designs

Figure 9.1: Applications and Validation of KCM meta-models and methods

9.1 Models in VIATRA Model Transformation Framework

The test cases presented in this chapter are implemented using VIATRA model transformation framework. For all test cases, the model space in this framework is pre-loaded with KCM meta-models and libraries. Figure 9.2 illustrates the KCM model space in the VIATRA model transformation framework. The following meta-models, models libraries, and model transformation libraries are pre-loaded for execution the model transformations prescribed by KCM's Behavior Model Formulation Method. The meta-models and model libraries presented here are implementations of the KCM meta-models and models in VIATRA Textual Metamodeling Language, and the model transformations presented here are implementations of KCM's Artifact Model Transformation Library in VIATRA Textual Command Language (VTCL).



- kcm
  - Meta_Model
    - ABB_Meta_Model
    - Analyzable_Electronics_Design_Meta_Model
    - CBM
    - CPM2_xKCM
    - Generics_Meta_Model
  - Model
    - ABB_Library
    - Analyzable_Electronics_Design_FTMB_Model_Space
  - mxform_rp

Artifact Model Transformation Library

Figure 9.2: KCM meta-models, models, and model transformation libraries shown in the KCM model space of the VIATRA model transformation framework

- Meta-Models
  - CPM2_xKCM is the implementation of the CPM2_xKCM meta-model (section 6.1) in VTML.
  - CBM is the implementation of the Core Behavior Model (CBM – section 7.1) in VTML.
  - ABB_Meta_Model is the implementation of the ABB Meta-Model (section 7.2) in VTML.
  - Generics_Meta_Model is the implementation of KCM's Generic Meta-Model for representing geometry, math relations, and other constructs that are used by all meta-models and models.
  - Analyzable_Electronics_Design_Meta-Model is a VTMB electronics artifact-specific meta-model. It is a Level 3 model in the design model stack and contains design and analyzable design meta-models for representing electronics artifacts of varying

assembly system topologies. The Analyzable_Electronics_Design_Meta-Model is the meta-model for representing both types of electronics artifacts—printed wiring boards and ball grid array chip packages—used in the test cases described in this chapter. In section 6.2, a sub-set of this meta-model for representing design and analyzable design aspects of printed wiring boards was presented.

- Models
  - ABB_Library is the implementation of KCM's Analysis Building Block Library (section 7.3) in VTML.
  - Analyzable_Electronics_Design_FTMB_Model_Space is a model space for fixed topology multi-body analyzable design model structures—Level 4 models in the design model stack.
- Model transformations
  - mxform_rp is the implementations of KCM's Artifact Model Transformation Library (section 8.4) in VTCL.

9.2 Test Case Family 1 (TCF1): Thermo-mechanical Analysis of Multi-Layered Printed Wiring Boards

A printed wiring board[23] is an electronic artifact that transmits signals between components mounted on it via conductive pathways (traces) originating from / terminating in other conductive features (lands). A bare printed wiring board has no packaged components on it. A PCB with mounted components (such as chip packages) is also known as a printed wiring assembly (PWA/PCA). The mechanical function of a PWB is to support the electronic circuitry laid out in multiple stratums of the PWB. Figure 9.3 illustrates the 2D layout and through-thickness stackup of a typical PWB. A PWB consists of a stackup of materials as shown in the through-thickness view. Each layer of material is known as a stratum. A stackup is made of alternatively electrically conductive and non-conductive stratums. Conductive stratums have conductive features such as lands and traces as shown in the planar layout view. Vias and through-holes are openings in the stackup from one conductive layer to another—primarily meant to provide electrical connections across

[23] Also known as printed circuit board (PCB)

stratums. The through-thickness view shows the structure of the stackup—same for different thicknesses of the stratums.

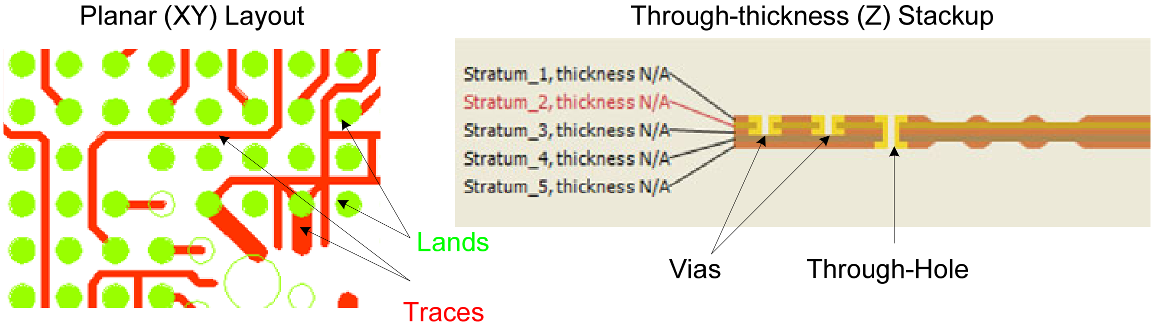
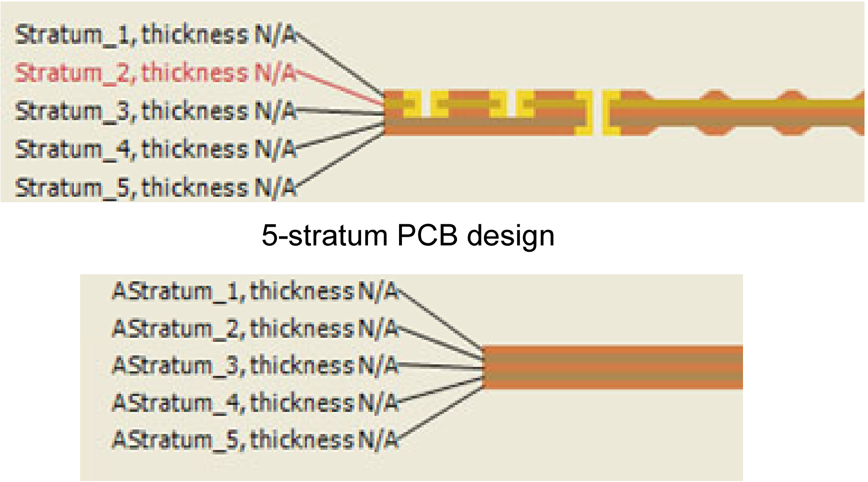


Figure 9.3: A typical Printed Wiring Board design (shown here with 5 stratums)

In this section, simulation templates shall be automatically generated using the Behavior Model Formulation Method for PWBs with different number of stratums, and for different behavior idealization decisions. These simulation templates are to be used for thermo-mechanical analyses of printed wiring boards. Specifically, the objective of creating these simulation templates is to compute both out-of-plane and in-plane deformation of printed wiring boards for different temperature loads. This type of analysis is required to simulate the deformation of printed wiring boards when components are assembled on their surface, or when a printed wiring board is being manufactured in a sequential lamination process in which heating and cooling result in different materials on a PWB to expand and contract differently owing to mismatches in their coefficient of thermal expansions. The deformation of printed wiring boards leads to mis-registration between component terminals and the conductive footprints on the PCB where they are supposed to mount, leading to acute reliability problems (Zwemer, Bajaj et al. 2004; Bajaj, Peak et al. 2006).

In the Behavior Model Formulation Method, the source models are fixed topology analyzable design model structures—Level 4 models in the design model stack. If a design is to be analyzed as-is (including all features), then the analyzable design model structure is same as the design model structure. For the specific case of simulation templates generated for thermo-mechanical analyses of printed wiring boards in the test cased presented here, the design and analyzable design are different. In the analyzable design model, the stratums are idealized as homogenous. This is a fairly common idealization in different types of analyses of printed wiring boards, especially when global behaviors are of interest to analysts. Figure 9.4 illustrates both design and analyzable design for a 5-stratum PWB. In the analyzable design, the design layers (conductive stratums) are idealized as uniform and homogenous as opposed to having specific conductive features such as lands and traces in the design model.

The analyzable design model structure is the starting point of the test cases demonstrated here. Sections 9.2.1 and 9.2.2 illustrate two different behavior idealizations—BMFS^1 and BMFS^2 respectively. For each BMFS, simulation templates are created for two analyzable PWB design alternatives—one with 5 stratums and one with 9 stratums. The analyzable PWB design alternatives with 5-stratums and 9-stratums have non-equivalent assembly system topologies due to differences in number of assembly components (5 versus 9), and differences in number of interactions between components (4 versus 8).


Stratum_1, thickness N/A
Stratum_2, thickness N/A
Stratum_3, thickness N/A
Stratum_4, thickness N/A
Stratum_5, thickness N/A

5-stratum PCB design

AStratum_1, thickness N/A
AStratum_2, thickness N/A
AStratum_3, thickness N/A
AStratum_4, thickness N/A
AStratum_5, thickness N/A

5-stratum analyzable PCB design

Figure 9.4: 5-stratum PCB – design and analyzable design views

Table 9.1 shows the four fixed topology simulation templates that would be auto-generated for combinations of 2 different Behavior Model Formulation Specifications and 2 analyzable design model structures with different assembly system topologies.

Table 9.1: Simulation templates created for thermo-mechanical analysis of PWBs

	Analyzable Design Model Structures	
	5-stratum analyzable PCB	*9-stratum analyzable PCB*
BMFS^1	Simulation Template $_{5}^{1}$	Simulation Template $_{9}^{1}$
BMFS^2	Simulation Template $_{5}^{2}$	Simulation Template $_{9}^{2}$

9.2.1 Behavior Model Formulation Specifications 1 (BMFS^1)

In this section, simulation templates auto-generated for idealization decisions embodied in BMFS^1 are presented. First, the conceptual specifications in BMFS^1 are presented. Then, fixed topology simulation templates auto-generated for two analyzable PWB design model structures with different assembly system topologies are presented in sections 9.2.1.1 and 9.2.1.2 respectively. The conceptual specifications in BMFS^1 are summarized in Table 9.2 below. Note that the conceptual specifications are presented here using the select and idealize constructs described in section 8.3.1.

Table 9.2: Conceptual specifications (BMFS^1) for thermo-mechanical analyses of multi-stratum PCBs

Conceptual specifications for Stage 1 composition	
Entities in analyzable PCB design model	*Entities in Multi-shell analysis body system (as instances of ABBs stated below)*
Analyzable stratum	***Idealize as*** Planar shell analysis body ABB
Planar shape	***Select*** Planar shape ABB
Linear elastic isotropic temperature-independent material behavior Linear elastic orthotropic temperature-independent material behavior …	***Select*** Linear elastic isotropic temperature independent material behavior ABB
Analyzable features	Analysis features
Primary surface (planar surface feature)	***Idealize as*** Planar surface feature ABB
Secondary surface (planar surface features)	***Idealize as*** Planar surface feature ABB
Conceptual specifications for Stage 2 composition	
Analyzable PCB	***Idealize as*** Multi-shell analysis body system
Analyzable stratum	***Idealize as*** Planar shell analysis body ABB
Adjacent stratum surface interaction	***Idealize as*** Shell-shell tie interaction ABB (perfectly bonded shell-shell interaction)
Analyzable features	Analysis features
Volume of analyzable PCB	***Idealize as*** Volume feature ABB
Mid-pt of bottom soldermask stratum	***Idealize as*** Point feature ABB
Conceptual specifications for Stage 3 composition	
Heating a PCB	***Idealize as*** Uniform temperature load ABB ***associated with*** Volume feature ABB instance corresponding the volume of the analyzable PCB
PCB held fixed at mid-pt of the bottom soldermask stratum	***Idealize as*** Point displacement constant behavior condition ABB ***associated with*** Point feature ABB instance corresponding to mid-pt of bottom soldermask

The conceptual specifications are summarized for Stages 1-3 of the composition process. Stage 4 is creation of high-level behavior model entities that are common to all behavior model structures formulated using KCM. Figure 9.5 illustrates specifications for idealization relationships between an analyzable stratum in the analyzable PWB design model structure and the corresponding planar shell analysis body in the behavior model structure (to be created). Only assembly system topology-related aspects of the specifications are shown.

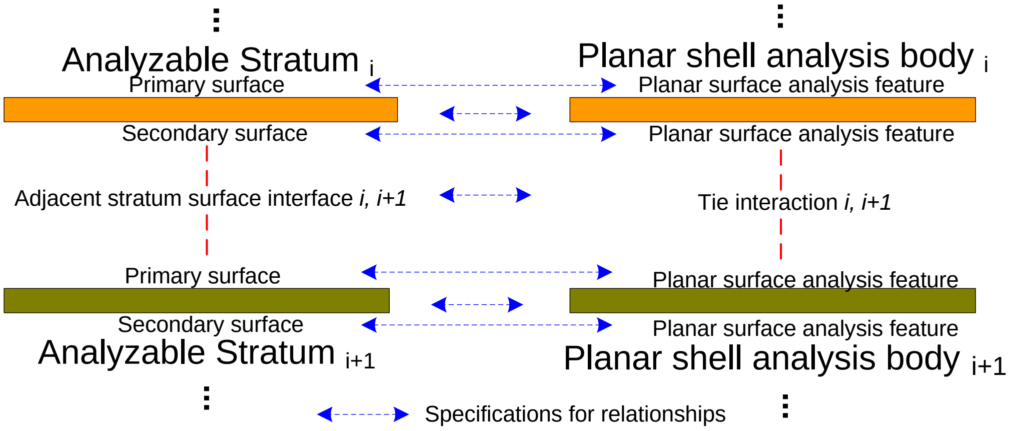

Analyzable Stratum i
Primary surface
Secondary surface
Adjacent stratum surface interface i, i+1
Primary surface
Secondary surface
Analyzable Stratum i+1
Planar shell analysis body i
Planar surface analysis feature
Planar surface analysis feature
Tie interaction i, i+1
Planar surface analysis feature
Planar surface analysis feature
Planar shell analysis body i+1
Specifications for relationships

Figure 9.5: Specifications for relationships between analyzable stratum and planar shell analysis bodies (only assembly system topology-related aspects are shown)

Stage 1 composition concerns the idealizations at the level of a single analysis body. In BMFS[1], a stratum of an analyzable PCB is idealized as a planar shell analysis body as shown in Figure 9.5. Thus, the mechanical behavior of a stratum is idealized as the mechanical behavior of a shell, and the shape of the stratum is idealized as a planar shell shape—a thin prismatic shape where the outline and thickness of the shape is same as the outline and thickness of the analyzable stratum. The material behavior of all analyzable stratums is idealized to be linear, elastic, isotropic, and temperature independent. The primary and secondary surfaces of the analyzable stratums are idealized as planar surface analysis features.

Stage 2 composition concerns idealizations at the level of multiple analysis bodies in the context of an analysis body system. In BMFS[1], an analyzable PWB is idealized as a multi-shell system composed of a stack of planar shell analysis bodies—each body corresponding to an analyzable stratum. Since the thickness of each planar shell analysis body is small, the multi-shell system itself behavior as a laminated shell. The interactions between the analyzable stratums, also known as adjacent stratum surface interfaces, are relationships between the secondary surface of the preceding stratum and the primary surface of the succeeding stratum. In BMFS^1, an adjacent stratum surface interface is idealized as a tie interaction between the corresponding planar shell analysis bodies. The planar surface analysis features of two adjacent planar shell analysis bodies participate in a tie interaction. The behavior of a tie interaction is same as if adjacent planar shell analysis bodies were perfectly bonded. Hence, the displacement (translational and rotational components) is continuous across the interface of adjacent planar shell analysis bodies. In addition, two new analysis features are defined at the multi-shell analysis body system level. These are (i) volume feature ABB instance corresponding to the volume of an analyzable PCB, and (ii) point feature ABB instance corresponding to the mid-point of the bottom analyzable stratum (corresponding to the soldermask stratum in the PWB design model).

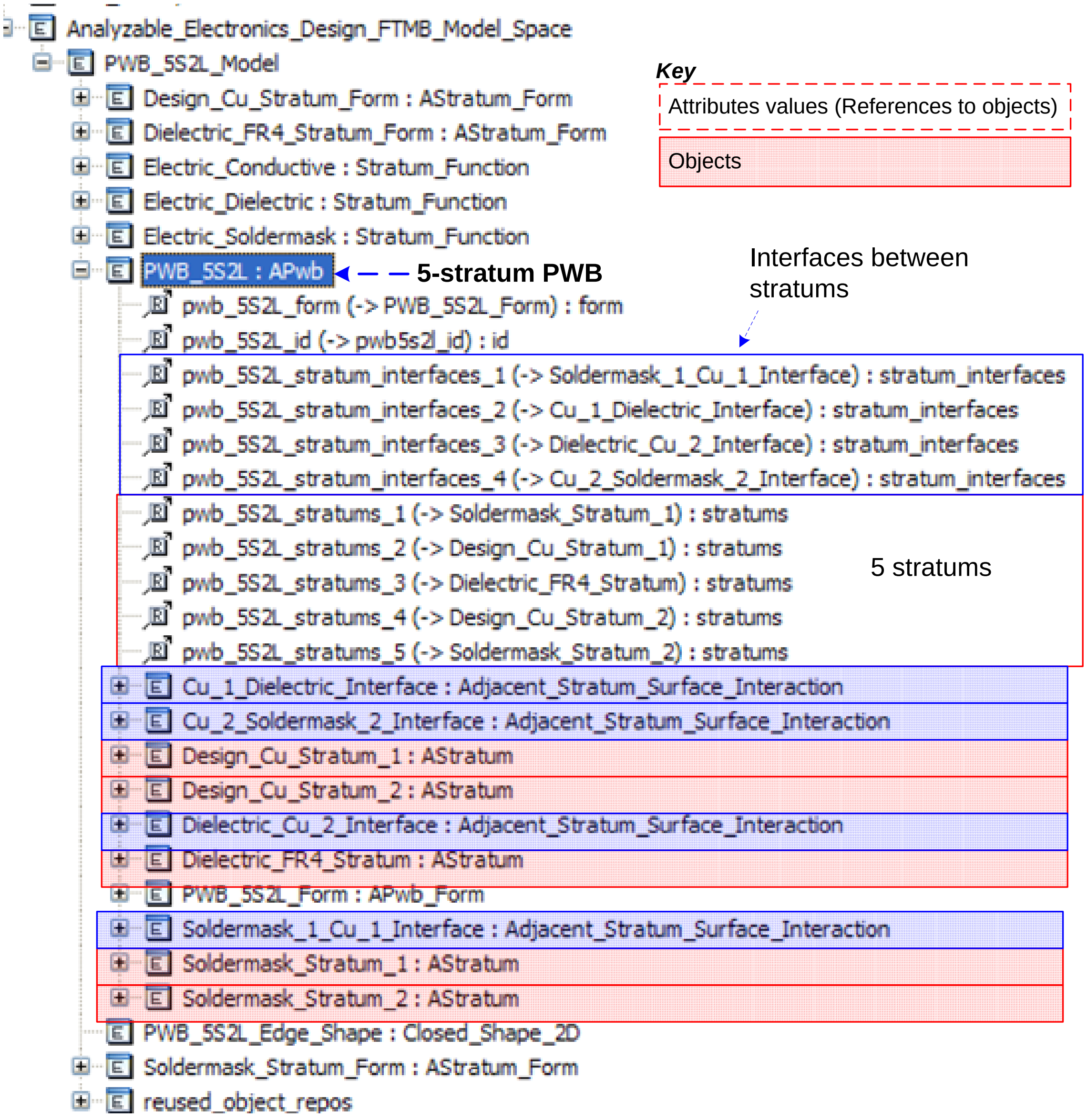
Stage 3 composition concerns the idealizations of loads and behavior conditions in which the behavior of the analysis body system is to be computed. The process of heating a PCB, say for mounting components, is idealized as a uniform temperature load—uniform increase in temperature from a reference value to a target value. In addition, the load is idealized to be uniform through the volume of the entire multi-shell analysis body system. The behavior condition for this analysis is to hold the mid-point of the bottom analyzable stratum as fixed. This corresponds to locking all degrees of freedom at that point in the analysis body system. This behavior condition is realized by the use of point-displacement-fixed ABB instance that embodies the displacement constraint, and associating it with the point feature ABB instance corresponding to the mid-point of the bottom planar shell analysis body. Per the idealization specifications in Figure 9.5, the bottom planar shell analysis body corresponds to the bottom analyzable stratum.

9.2.1.1 Simulation Template $_5^1$: Simulation template for 5-stratum analyzable PWB design model structure and BMFS^1

In this section a fixed topology simulation template auto-generated for 5-stratum analyzable PWB design model structure and BMFS^1 is presented. The source model in the model transformation shown here is PWB_5S2L—a 5-stratum, 2-layer[24] analyzable PWB

[24] Conductive stratums are known as layers. In this example, the analyzable PWB has 5 stratums and 2 layers.

design model structure, shown in Figure 9.6 in the VIATRA model space. PWB_5S2L is an instance of the Analyzable_Electronics_Design_Meta-Model (section 6.2) that is pre-loaded in the VIATRA model space. The source model shown here is a Level 4 model in the design model stack, and same as the PCB_5Sx model illustrated in Figure 6.10.

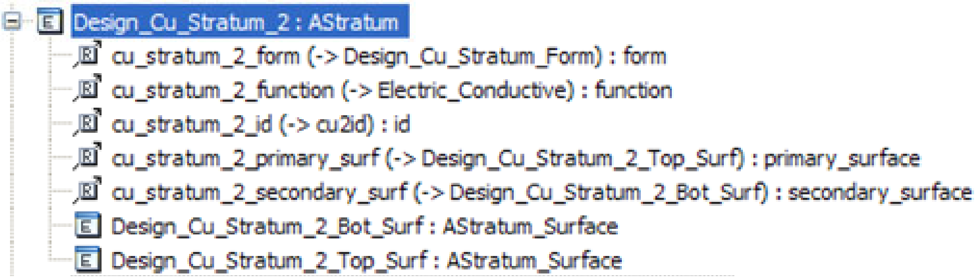


```
Analyzable_Electronics_Design_FTMB_Model_Space
  PWB_5S2L_Model
    Design_Cu_Stratum_Form : AStratum_Form
    Dielectric_FR4_Stratum_Form : AStratum_Form
    Electric_Conductive : Stratum_Function
    Electric_Dielectric : Stratum_Function
    Electric_Soldermask : Stratum_Function
    PWB_5S2L : APwb
      pwb_5S2L_form (-> PWB_5S2L_Form) : form
      pwb_5S2L_id (-> pwb5s2l_id) : id
      pwb_5S2L_stratum_interfaces_1 (-> Soldermask_1_Cu_1_Interface) : stratum_interfaces
      pwb_5S2L_stratum_interfaces_2 (-> Cu_1_Dielectric_Interface) : stratum_interfaces
      pwb_5S2L_stratum_interfaces_3 (-> Dielectric_Cu_2_Interface) : stratum_interfaces
      pwb_5S2L_stratum_interfaces_4 (-> Cu_2_Soldermask_2_Interface) : stratum_interfaces
      pwb_5S2L_stratums_1 (-> Soldermask_Stratum_1) : stratums
      pwb_5S2L_stratums_2 (-> Design_Cu_Stratum_1) : stratums
      pwb_5S2L_stratums_3 (-> Dielectric_FR4_Stratum) : stratums
      pwb_5S2L_stratums_4 (-> Design_Cu_Stratum_2) : stratums
      pwb_5S2L_stratums_5 (-> Soldermask_Stratum_2) : stratums
      Cu_1_Dielectric_Interface : Adjacent_Stratum_Surface_Interaction
      Cu_2_Soldermask_2_Interface : Adjacent_Stratum_Surface_Interaction
      Design_Cu_Stratum_1 : AStratum
      Design_Cu_Stratum_2 : AStratum
      Dielectric_Cu_2_Interface : Adjacent_Stratum_Surface_Interaction
      Dielectric_FR4_Stratum : AStratum
      PWB_5S2L_Form : APwb_Form
      Soldermask_1_Cu_1_Interface : Adjacent_Stratum_Surface_Interaction
      Soldermask_Stratum_1 : AStratum
      Soldermask_Stratum_2 : AStratum
    PWB_5S2L_Edge_Shape : Closed_Shape_2D
    Soldermask_Stratum_Form : AStratum_Form
    reused_object_repos
```

Key
Attributes values (References to objects)
Objects
5-stratum PWB
Interfaces between stratums
5 stratums

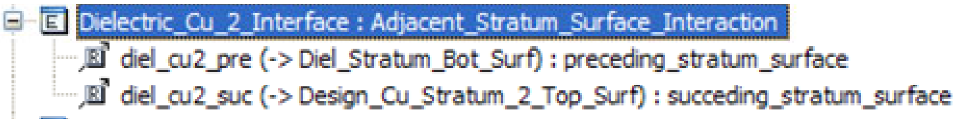
Figure 9.6: PWB_5S2L has 5 analyzable stratums and 4 stratum interfaces

In the figure above, the objects in the model space denoted with an icon with letter E are entities, and objects denoted with an icon with letter R and an arrow (→) are attributes of the containing entity. In VTML, attributes are modeled as relationships and hence the letter R is used o denote them in the VIATRA model space. As shown in Figure 9.6, PWB_5S2L has 5 analyzable stratums and 4 stratum interfaces. In the figure, the entities highlighted using dashed lines are the show the attribute values of the PWB_5S2L entity. These values refer to the 5 analyzable stratum objects and 4 stratum interfaces entities as shown in the model space. Figure 9.7 illustrates an analyzable stratum object and its attribute values, and Figure 9.8 illustrates a stratum interface object and its attribute values—preceding and succeeding stratum surfaces.



```
Design_Cu_Stratum_2 : AStratum
    cu_stratum_2_form (-> Design_Cu_Stratum_Form) : form
    cu_stratum_2_function (-> Electric_Conductive) : function
    cu_stratum_2_id (-> cu2id) : id
    cu_stratum_2_primary_surf (-> Design_Cu_Stratum_2_Top_Surf) : primary_surface
    cu_stratum_2_secondary_surf (-> Design_Cu_Stratum_2_Bot_Surf) : secondary_surface
    Design_Cu_Stratum_2_Bot_Surf : AStratum_Surface
    Design_Cu_Stratum_2_Top_Surf : AStratum_Surface
```

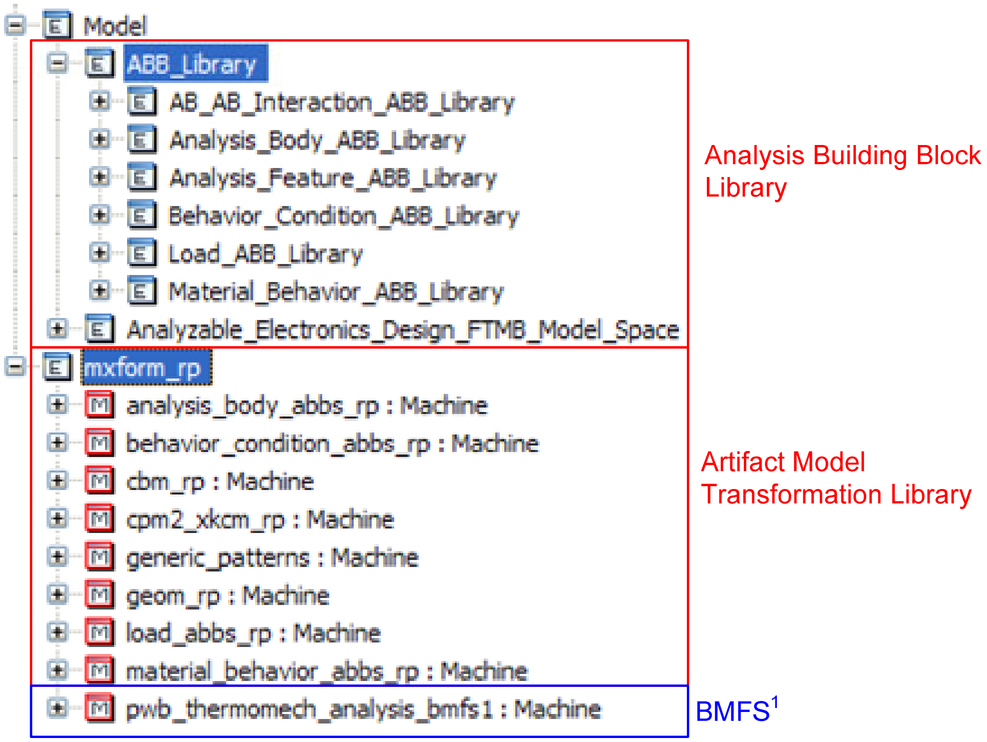
Figure 9.7: Stratum entity example



```
Dielectric_Cu_2_Interface : Adjacent_Stratum_Surface_Interaction
    diel_cu2_pre (-> Diel_Stratum_Bot_Surf) : preceding_stratum_surface
    diel_cu2_suc (-> Design_Cu_Stratum_2_Top_Surf) : succeding_stratum_surface
```

Figure 9.8: Stratum interface entity example

Once the FTMB analyzable design model (PWB_5S2L) is available in the model space, BMFS[1] can be loaded and executed to auto-generate fixed topology simulation template. Figure 9.9 illustrates the ABB Library, Artifact Model Transformation Library, and BMFS[1] (computable specification) in the VIATRA model space. The ABB Library and Artifact Model Transformation Library are common to all behavior model structures and simulation templates formulated using the Behavior Model Formulation Method, but the Behavior Model Formulation Specifications that embody the idealization decisions typically differ from one analysis to another. Figure 9.10 illustrates the execution of BMFS[1] in the VIATRA model space—right click on model space entry and select Run from the menu.



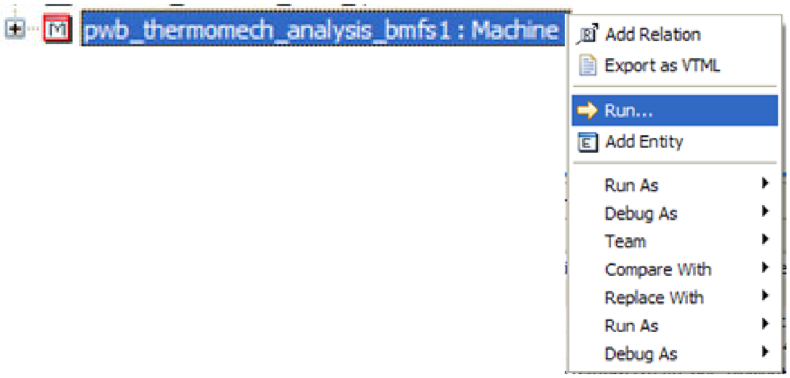
```
Model
  ABB_Library
    AB_AB_Interaction_ABB_Library
    Analysis_Body_ABB_Library
    Analysis_Feature_ABB_Library
    Behavior_Condition_ABB_Library
    Load_ABB_Library
    Material_Behavior_ABB_Library
  Analyzable_Electronics_Design_FTMB_Model_Space
mxform_rp
  analysis_body_abbs_rp : Machine
  behavior_condition_abbs_rp : Machine
  cbm_rp : Machine
  cpm2_xkcm_rp : Machine
  generic_patterns : Machine
  geom_rp : Machine
  load_abbs_rp : Machine
  material_behavior_abbs_rp : Machine
  pwb_thermomech_analysis_bmfs1 : Machine
```

Analysis Building Block Library

Artifact Model Transformation Library

BMFS[1]

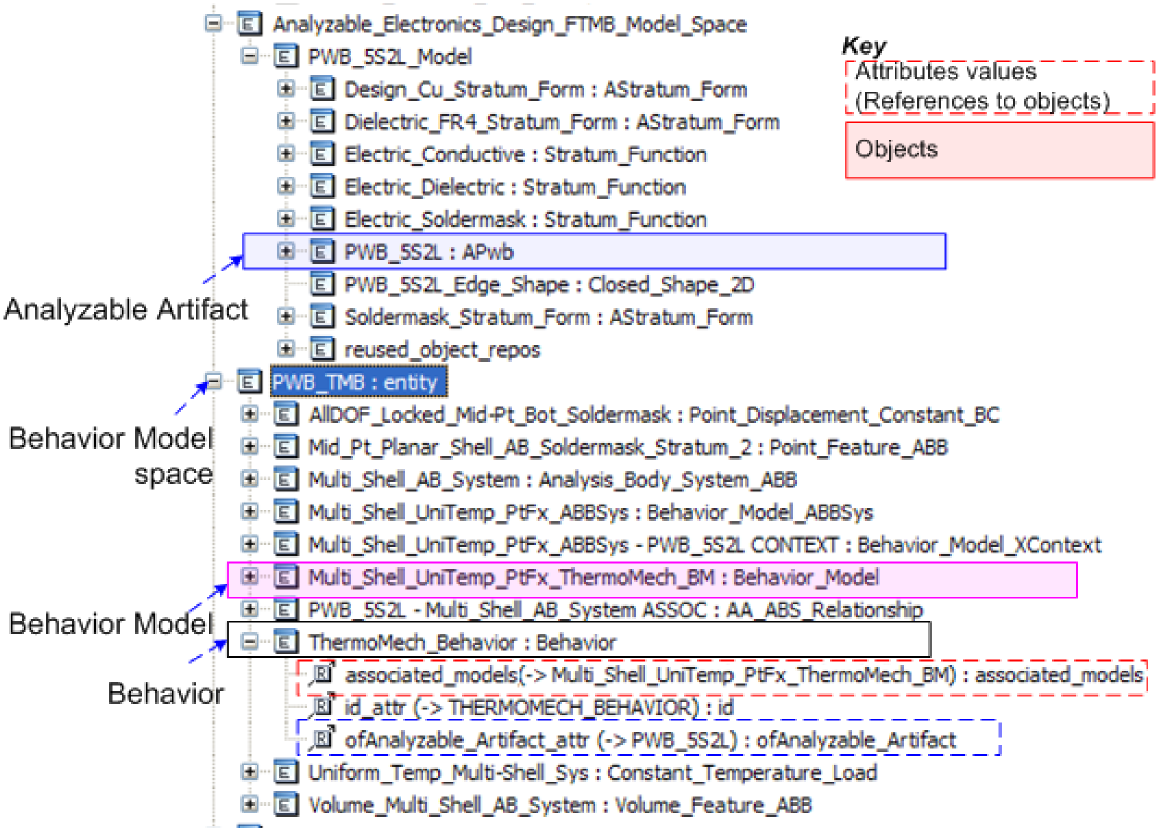
Figure 9.9: VIATRA model space showing ABB Library, AMTL, and BMFS[1]



pwb_thermomech_analysis_bmfs1 : Machine

Figure 9.10: Executing BMFS[1] in VIATRA model space.

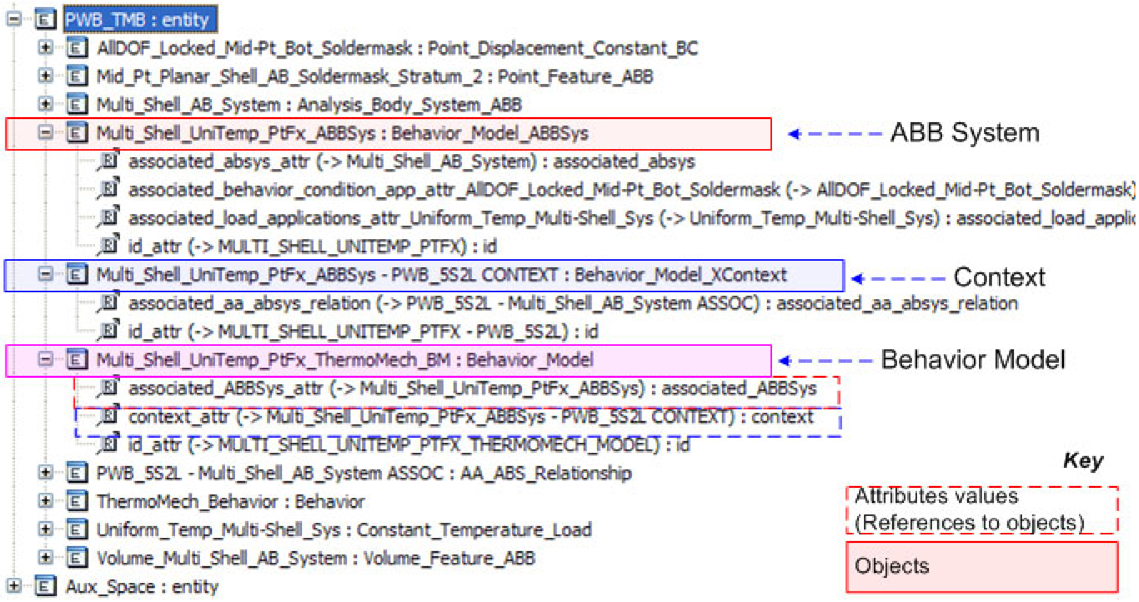
The Behavior Model space shown in Figure 9.11 shows the simulation template entities auto-created by executing BMFS[1] (computable specifications). Figure 9.11 illustrates the relationship between an analyzable artifact, its behaviors, and behavior models used for computing those behaviors. This is one of the core concepts in CPM2_xKCM. Note that two attributes of the entity ThermoMech_Behavior relate PWB_5S2L (analyzable artifact) and Multi_Shell_UniTemp_PtDx_ThermoMech_BM (behavior model). Similarly, other thermo-mechanical behavior models of different fidelities can be associated with the entity ThermoMech_Behavior.



```
Analyzable_Electronics_Design_FTMB_Model_Space
  PWB_5S2L_Model
    Design_Cu_Stratum_Form : AStratum_Form
    Dielectric_FR4_Stratum_Form : AStratum_Form
    Electric_Conductive : Stratum_Function
    Electric_Dielectric : Stratum_Function
    Electric_Soldermask : Stratum_Function
    PWB_5S2L : APwb
    PWB_5S2L_Edge_Shape : Closed_Shape_2D
    Soldermask_Stratum_Form : AStratum_Form
    reused_object_repos
  PWB_TMB : entity
    AllDOF_Locked_Mid-Pt_Bot_Soldermask : Point_Displacement_Constant_BC
    Mid_Pt_Planar_Shell_AB_Soldermask_Stratum_2 : Point_Feature_ABB
    Multi_Shell_AB_System : Analysis_Body_System_ABB
    Multi_Shell_UniTemp_PtFx_ABBSys : Behavior_Model_ABBSys
    Multi_Shell_UniTemp_PtFx_ABBSys - PWB_5S2L CONTEXT : Behavior_Model_XContext
    Multi_Shell_UniTemp_PtFx_ThermoMech_BM : Behavior_Model
    PWB_5S2L - Multi_Shell_AB_System ASSOC : AA_ABS_Relationship
    ThermoMech_Behavior : Behavior
      associated_models(-> Multi_Shell_UniTemp_PtFx_ThermoMech_BM) : associated_models
      id_attr (-> THERMOMECH_BEHAVIOR) : id
      ofAnalyzable_Artifact_attr (-> PWB_5S2L) : ofAnalyzable_Artifact
    Uniform_Temp_Multi-Shell_Sys : Constant_Temperature_Load
    Volume_Multi_Shell_AB_System : Volume_Feature_ABB
```

Figure 9.11: Simulation template automatically created using

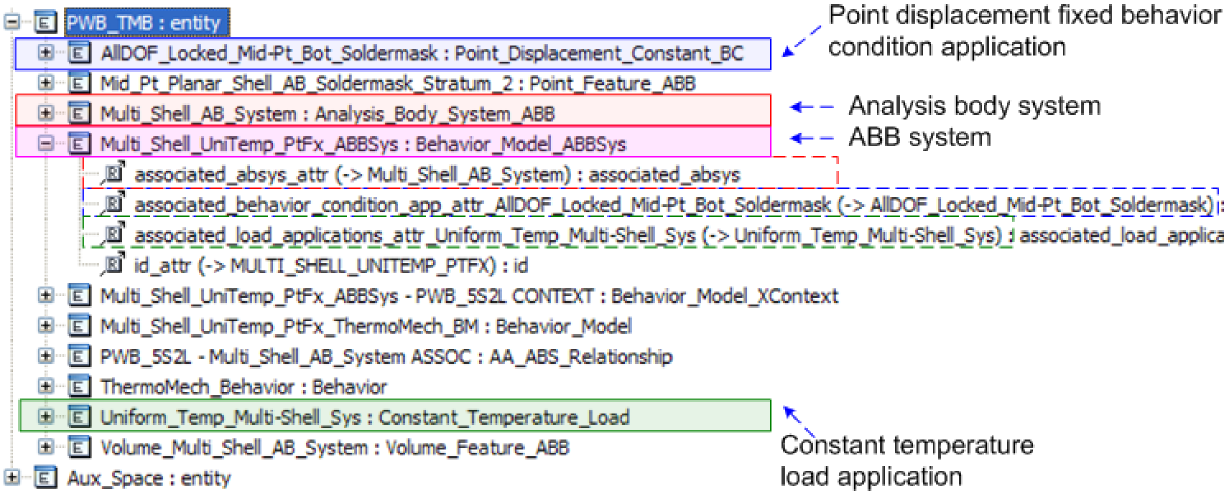
Behavior Model Formulation Method



```
PWB_TMB : entity
  AllDOF_Locked_Mid-Pt_Bot_Soldermask : Point_Displacement_Constant_BC
  Mid_Pt_Planar_Shell_AB_Soldermask_Stratum_2 : Point_Feature_ABB
  Multi_Shell_AB_System : Analysis_Body_System_ABB
  Multi_Shell_UniTemp_PtFx_ABBSys : Behavior_Model_ABBSys
    associated_absys_attr (-> Multi_Shell_AB_System) : associated_absys
    associated_behavior_condition_app_attr_AllDOF_Locked_Mid-Pt_Bot_Soldermask (-> AllDOF_Locked_Mid-Pt_Bot_Soldermask)
    associated_load_applications_attr_Uniform_Temp_Multi-Shell_Sys (-> Uniform_Temp_Multi-Shell_Sys) : associated_load_applic
    id_attr (-> MULTI_SHELL_UNITEMP_PTFX) : id
  Multi_Shell_UniTemp_PtFx_ABBSys - PWB_5S2L CONTEXT : Behavior_Model_XContext
    associated_aa_absys_relation (-> PWB_5S2L - Multi_Shell_AB_System ASSOC) : associated_aa_absys_relation
    id_attr (-> MULTI_SHELL_UNITEMP_PTFX - PWB_5S2L) : id
  Multi_Shell_UniTemp_PtFx_ThermoMech_BM : Behavior_Model
    associated_ABBSys_attr (-> Multi_Shell_UniTemp_PtFx_ABBSys) : associated_ABBSys
    context_attr (-> Multi_Shell_UniTemp_PtFx_ABBSys - PWB_5S2L CONTEXT) : context
    id_attr (-> MULTI_SHELL_UNITEMP_PTFX_THERMOMECH_MODEL) : id
  PWB_5S2L - Multi_Shell_AB_System ASSOC : AA_ABS_Relationship
  ThermoMech_Behavior : Behavior
  Uniform_Temp_Multi-Shell_Sys : Constant_Temperature_Load
  Volume_Multi_Shell_AB_System : Volume_Feature_ABB
Aux_Space : entity
```

Figure 9.12: Results of Stage 4 composition

Figure 9.12 illustrates the entities and relationships created in the simulation template at the end of Stage 4 of the composition process. The figure shows a behavior model, behavior model ABB system, and context entities during the process. The attribute values of the behavior model entity refer to the ABB system and context entities. Figure 9.13 illustrates the behavior model ABB system entity and its attribute values creating at the end of Stage 3 of the composition process. The figure shows the ABB system consists of the multi-shell analysis body system, point displacement behavior condition, and a constant temperature load.



```
PWB_TMB : entity
    AllDOF_Locked_Mid-Pt_Bot_Soldermask : Point_Displacement_Constant_BC
    Mid_Pt_Planar_Shell_AB_Soldermask_Stratum_2 : Point_Feature_ABB
    Multi_Shell_AB_System : Analysis_Body_System_ABB
    Multi_Shell_UniTemp_PtFx_ABBSys : Behavior_Model_ABBSys
        associated_absys_attr (-> Multi_Shell_AB_System) : associated_absys
        associated_behavior_condition_app_attr_AllDOF_Locked_Mid-Pt_Bot_Soldermask (-> AllDOF_Locked_Mid-Pt_Bot_Soldermask) :
        associated_load_applications_attr_Uniform_Temp_Multi-Shell_Sys (-> Uniform_Temp_Multi-Shell_Sys) : associated_load_applica
        id_attr (-> MULTI_SHELL_UNITEMP_PTFX) : id
    Multi_Shell_UniTemp_PtFx_ABBSys - PWB_5S2L CONTEXT : Behavior_Model_XContext
    Multi_Shell_UniTemp_PtFx_ThermoMech_BM : Behavior_Model
    PWB_5S2L - Multi_Shell_AB_System ASSOC : AA_ABS_Relationship
    ThermoMech_Behavior : Behavior
    Uniform_Temp_Multi-Shell_Sys : Constant_Temperature_Load
    Volume_Multi_Shell_AB_System : Volume_Feature_ABB
Aux_Space : entity
```

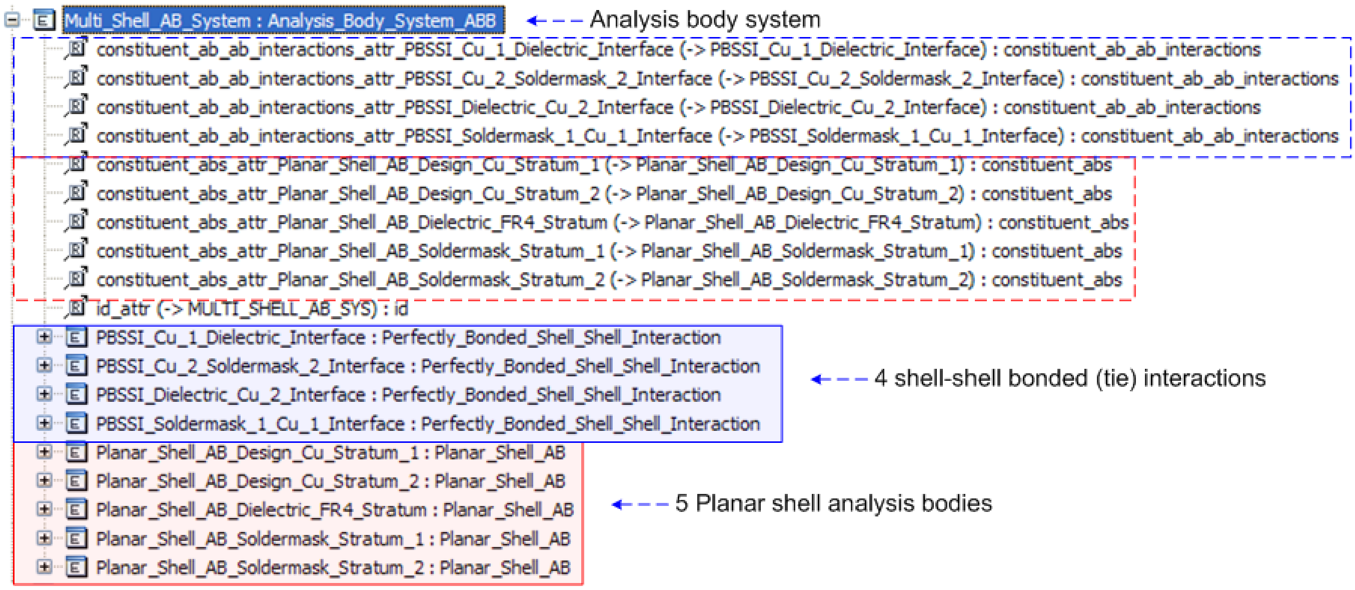
Point displacement fixed behavior condition application

Analysis body system

ABB system

Constant temperature load application

Figure 9.13: Behavior Model ABB System created at the end of Stage 3 composition



```
Multi_Shell_AB_System : Analysis_Body_System_ABB
    constituent_ab_ab_interactions_attr_PBSSI_Cu_1_Dielectric_Interface (-> PBSSI_Cu_1_Dielectric_Interface) : constituent_ab_ab_interactions
    constituent_ab_ab_interactions_attr_PBSSI_Cu_2_Soldermask_2_Interface (-> PBSSI_Cu_2_Soldermask_2_Interface) : constituent_ab_ab_interactions
    constituent_ab_ab_interactions_attr_PBSSI_Dielectric_Cu_2_Interface (-> PBSSI_Dielectric_Cu_2_Interface) : constituent_ab_ab_interactions
    constituent_ab_ab_interactions_attr_PBSSI_Soldermask_1_Cu_1_Interface (-> PBSSI_Soldermask_1_Cu_1_Interface) : constituent_ab_ab_interactions
    constituent_abs_attr_Planar_Shell_AB_Design_Cu_Stratum_1 (-> Planar_Shell_AB_Design_Cu_Stratum_1) : constituent_abs
    constituent_abs_attr_Planar_Shell_AB_Design_Cu_Stratum_2 (-> Planar_Shell_AB_Design_Cu_Stratum_2) : constituent_abs
    constituent_abs_attr_Planar_Shell_AB_Dielectric_FR4_Stratum (-> Planar_Shell_AB_Dielectric_FR4_Stratum) : constituent_abs
    constituent_abs_attr_Planar_Shell_AB_Soldermask_Stratum_1 (-> Planar_Shell_AB_Soldermask_Stratum_1) : constituent_abs
    constituent_abs_attr_Planar_Shell_AB_Soldermask_Stratum_2 (-> Planar_Shell_AB_Soldermask_Stratum_2) : constituent_abs
    id_attr (-> MULTI_SHELL_AB_SYS) : id
    PBSSI_Cu_1_Dielectric_Interface : Perfectly_Bonded_Shell_Shell_Interaction
    PBSSI_Cu_2_Soldermask_2_Interface : Perfectly_Bonded_Shell_Shell_Interaction
    PBSSI_Dielectric_Cu_2_Interface : Perfectly_Bonded_Shell_Shell_Interaction
    PBSSI_Soldermask_1_Cu_1_Interface : Perfectly_Bonded_Shell_Shell_Interaction
    Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_AB
    Planar_Shell_AB_Design_Cu_Stratum_2 : Planar_Shell_AB
    Planar_Shell_AB_Dielectric_FR4_Stratum : Planar_Shell_AB
    Planar_Shell_AB_Soldermask_Stratum_1 : Planar_Shell_AB
    Planar_Shell_AB_Soldermask_Stratum_2 : Planar_Shell_AB
```

Analysis body system

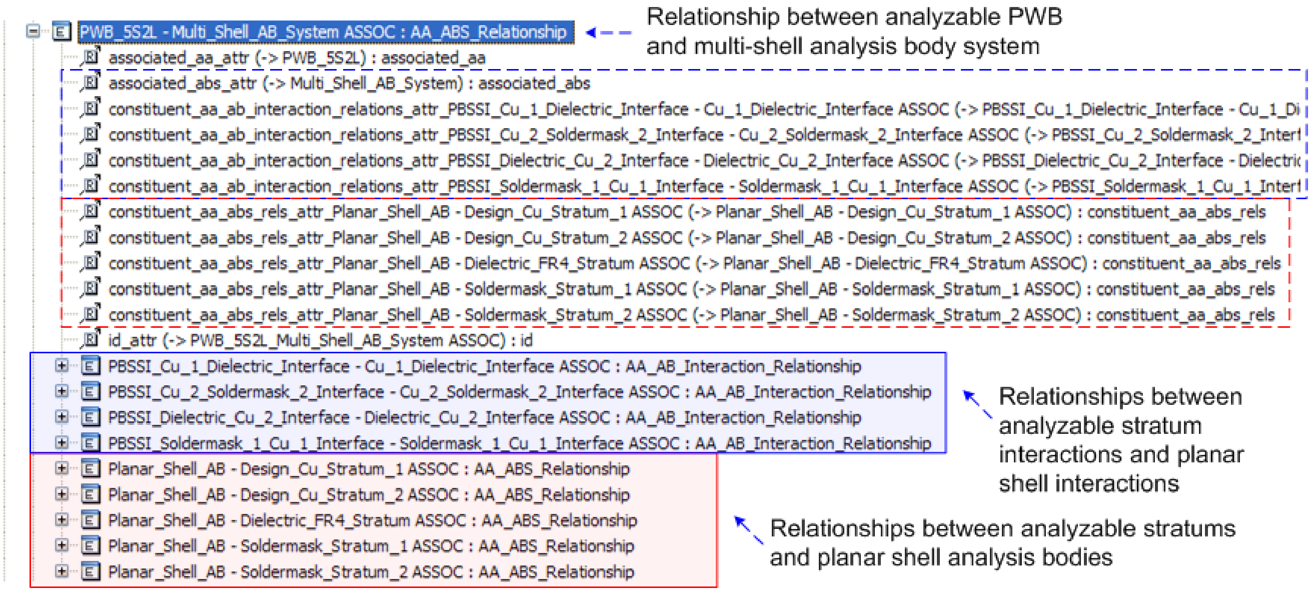
4 shell-shell bonded (tie) interactions

5 Planar shell analysis bodies

Figure 9.14: Analysis body system created at the end of Stage 2 composition

Figure 9.14 above illustrates the multi-shell analysis body system created at the end of Stage 2 composition process.

Note that the VIATRA framework orders attributes in an alphabetical order, and hence attributes corresponding to planar shell analysis bodies do not show the stackup order of these bodies in the multi-shell analysis body system. Figure 9.14 illustrates the attributes of the multi-shell analysis body system that refer to the 5 planar shell analysis bodies and 4 shell-shell tie interactions automatically created during the Stage 2 composition. In addition to the analysis bodies and their interactions, the relationships between the 5-stratum analyzable PWB and 5-shell analysis body system is also automatically created at the end of Stage 2 composition. Figure 9.15 below illustrates the five relationships created between analyzable stratums and planar shell analysis bodies—1 relationships for each pair, and the four relationships created between the analyzable stratum interfaces and the tie interactions between planar shell analysis bodies—1 relationship for each pair. These relationships realize the specifications illustrated in Figure 9.5.

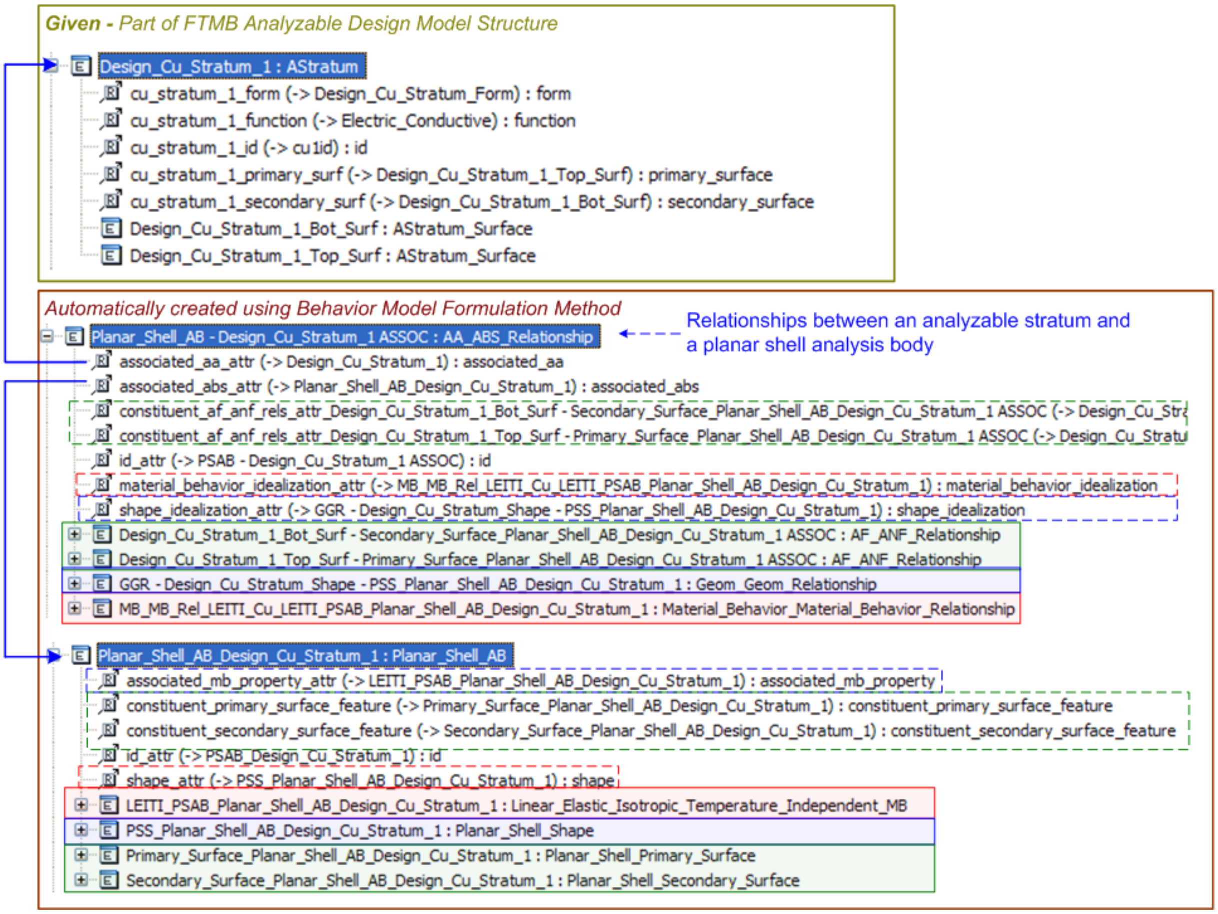


```
PWB_5S2L - Multi_Shell_AB_System ASSOC : AA_ABS_Relationship
    associated_aa_attr (-> PWB_5S2L) : associated_aa
    associated_abs_attr (-> Multi_Shell_AB_System) : associated_abs
    constituent_aa_ab_interaction_relations_attr_PBSSI_Cu_1_Dielectric_Interface - Cu_1_Dielectric_Interface ASSOC (-> PBSSI_Cu_1_Dielectric_Interface - Cu_1_Di
    constituent_aa_ab_interaction_relations_attr_PBSSI_Cu_2_Soldermask_2_Interface - Cu_2_Soldermask_2_Interface ASSOC (-> PBSSI_Cu_2_Soldermask_2_Interf
    constituent_aa_ab_interaction_relations_attr_PBSSI_Dielectric_Cu_2_Interface - Dielectric_Cu_2_Interface ASSOC (-> PBSSI_Dielectric_Cu_2_Interface - Dielectric
    constituent_aa_ab_interaction_relations_attr_PBSSI_Soldermask_1_Cu_1_Interface - Soldermask_1_Cu_1_Interface ASSOC (-> PBSSI_Soldermask_1_Cu_1_Interf
    constituent_aa_abs_rels_attr_Planar_Shell_AB - Design_Cu_Stratum_1 ASSOC (-> Planar_Shell_AB - Design_Cu_Stratum_1 ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_Planar_Shell_AB - Design_Cu_Stratum_2 ASSOC (-> Planar_Shell_AB - Design_Cu_Stratum_2 ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_Planar_Shell_AB - Dielectric_FR4_Stratum ASSOC (-> Planar_Shell_AB - Dielectric_FR4_Stratum ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_Planar_Shell_AB - Soldermask_Stratum_1 ASSOC (-> Planar_Shell_AB - Soldermask_Stratum_1 ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_Planar_Shell_AB - Soldermask_Stratum_2 ASSOC (-> Planar_Shell_AB - Soldermask_Stratum_2 ASSOC) : constituent_aa_abs_rels
    id_attr (-> PWB_5S2L_Multi_Shell_AB_System ASSOC) : id
  PBSSI_Cu_1_Dielectric_Interface - Cu_1_Dielectric_Interface ASSOC : AA_AB_Interaction_Relationship
  PBSSI_Cu_2_Soldermask_2_Interface - Cu_2_Soldermask_2_Interface ASSOC : AA_AB_Interaction_Relationship
  PBSSI_Dielectric_Cu_2_Interface - Dielectric_Cu_2_Interface ASSOC : AA_AB_Interaction_Relationship
  PBSSI_Soldermask_1_Cu_1_Interface - Soldermask_1_Cu_1_Interface ASSOC : AA_AB_Interaction_Relationship
  Planar_Shell_AB - Design_Cu_Stratum_1 ASSOC : AA_ABS_Relationship
  Planar_Shell_AB - Design_Cu_Stratum_2 ASSOC : AA_ABS_Relationship
  Planar_Shell_AB - Dielectric_FR4_Stratum ASSOC : AA_ABS_Relationship
  Planar_Shell_AB - Soldermask_Stratum_1 ASSOC : AA_ABS_Relationship
  Planar_Shell_AB - Soldermask_Stratum_2 ASSOC : AA_ABS_Relationship
```

Relationship between analyzable PWB and multi-shell analysis body system
Relationships between analyzable stratum interactions and planar shell interactions
Relationships between analyzable stratums and planar shell analysis bodies

Figure 9.15: Relationship between analyzable PWB and multi-shell analysis body system

A relationship between an analyzable artifact and an analysis body consists of relationships between their shapes, material behaviors, and analysis features. Hence, for every relationship between an analyzable stratum and a planar shell analysis body, several sub-relationships are also automatically created at the end of Stage 1 composition. Figure 9.16 illustrates these sub-relationships for an analyzable stratum and an analysis body. The entity Planar_Shell_AB – Design_Cu_Stratum_1_ASSOC represents the relationship between an analyzable stratum (specifically Design_Cu_Stratum_1) and the corresponding planar shell analysis body. The attribute values of this relationship refer to the relationships between (i) their primary and secondary features represented by entities of type AF_ANF_Relationship, (ii) their shapes represented by entity of type Geom_Geom_Relationship, and (iii) their material behaviors represented by entity of type Material_Behavior_Material_Behavior_Relationship.



Given - Part of FTMB Analyzable Design Model Structure

```
Design_Cu_Stratum_1 : AStratum
    cu_stratum_1_form (-> Design_Cu_Stratum_Form) : form
    cu_stratum_1_function (-> Electric_Conductive) : function
    cu_stratum_1_id (-> cu1id) : id
    cu_stratum_1_primary_surf (-> Design_Cu_Stratum_1_Top_Surf) : primary_surface
    cu_stratum_1_secondary_surf (-> Design_Cu_Stratum_1_Bot_Surf) : secondary_surface
    Design_Cu_Stratum_1_Bot_Surf : AStratum_Surface
    Design_Cu_Stratum_1_Top_Surf : AStratum_Surface
```

Automatically created using Behavior Model Formulation Method

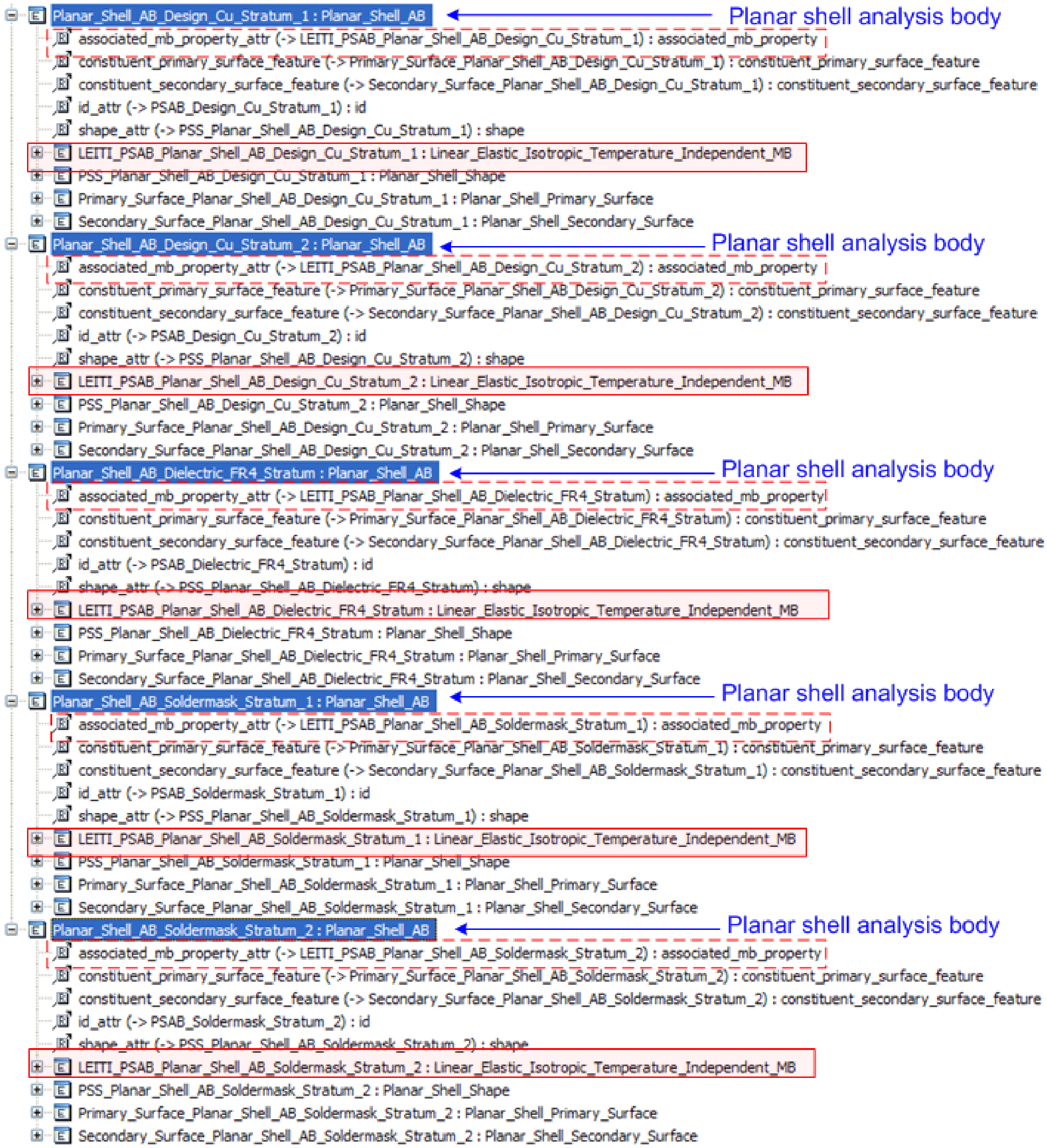
Relationships between an analyzable stratum and a planar shell analysis body

```
Planar_Shell_AB - Design_Cu_Stratum_1 ASSOC : AA_ABS_Relationship
    associated_aa_attr (-> Design_Cu_Stratum_1) : associated_aa
    associated_abs_attr (-> Planar_Shell_AB_Design_Cu_Stratum_1) : associated_abs
    constituent_af_anf_rels_attr_Design_Cu_Stratum_1_Bot_Surf - Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1 ASSOC (-> Design_Cu_Str
    constituent_af_anf_rels_attr_Design_Cu_Stratum_1_Top_Surf - Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1 ASSOC (-> Design_Cu_Stratu
    id_attr (-> PSAB - Design_Cu_Stratum_1 ASSOC) : id
    material_behavior_idealization_attr (-> MB_MB_Rel_LEITI_Cu_LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_1) : material_behavior_idealization
    shape_idealization_attr (-> GGR - Design_Cu_Stratum_Shape - PSS_Planar_Shell_AB_Design_Cu_Stratum_1) : shape_idealization
    Design_Cu_Stratum_1_Bot_Surf - Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1 ASSOC : AF_ANF_Relationship
    Design_Cu_Stratum_1_Top_Surf - Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1 ASSOC : AF_ANF_Relationship
    GGR - Design_Cu_Stratum_Shape - PSS_Planar_Shell_AB_Design_Cu_Stratum_1 : Geom_Geom_Relationship
    MB_MB_Rel_LEITI_Cu_LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_1 : Material_Behavior_Material_Behavior_Relationship

Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_AB
    associated_mb_property_attr (-> LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_1) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1) : constituent_primary_surface_feature
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1) : constituent_secondary_surface_feature
    id_attr (-> PSAB_Design_Cu_Stratum_1) : id
    shape_attr (-> PSS_Planar_Shell_AB_Design_Cu_Stratum_1) : shape
    LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_1 : Linear_Elastic_Isotropic_Temperature_Independent_MB
    PSS_Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_Shape
    Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_Primary_Surface
    Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_Secondary_Surface
```

Figure 9.16: Relationship between an analyzable stratum and analysis body created in Stage 1 composition

Figure 9.17 illustrates the planar shell analysis bodies created at end of Stage 1 composition. Each planar shell analysis body is an instance of Planar Shell Analysis Body ABB. The attributes of each planar shell analysis body is populated with other ABB instances. The shape attribute is populated by Planar Shape ABB instance, the material behavior attribute is populated by Linear Elastic Isotropic Temperature Independent Material Behavior ABB instance (highlighted in the figure), and the primary and secondary analysis feature attributes are populated by Planar Surface Analysis Feature ABB instances (Planar Shell Primary Surface is a special type of Planar Surface Analysis Feature ABB). *Note that the figure shows the planar shell analysis body entities and not their occurrence in the multi-shell system.*



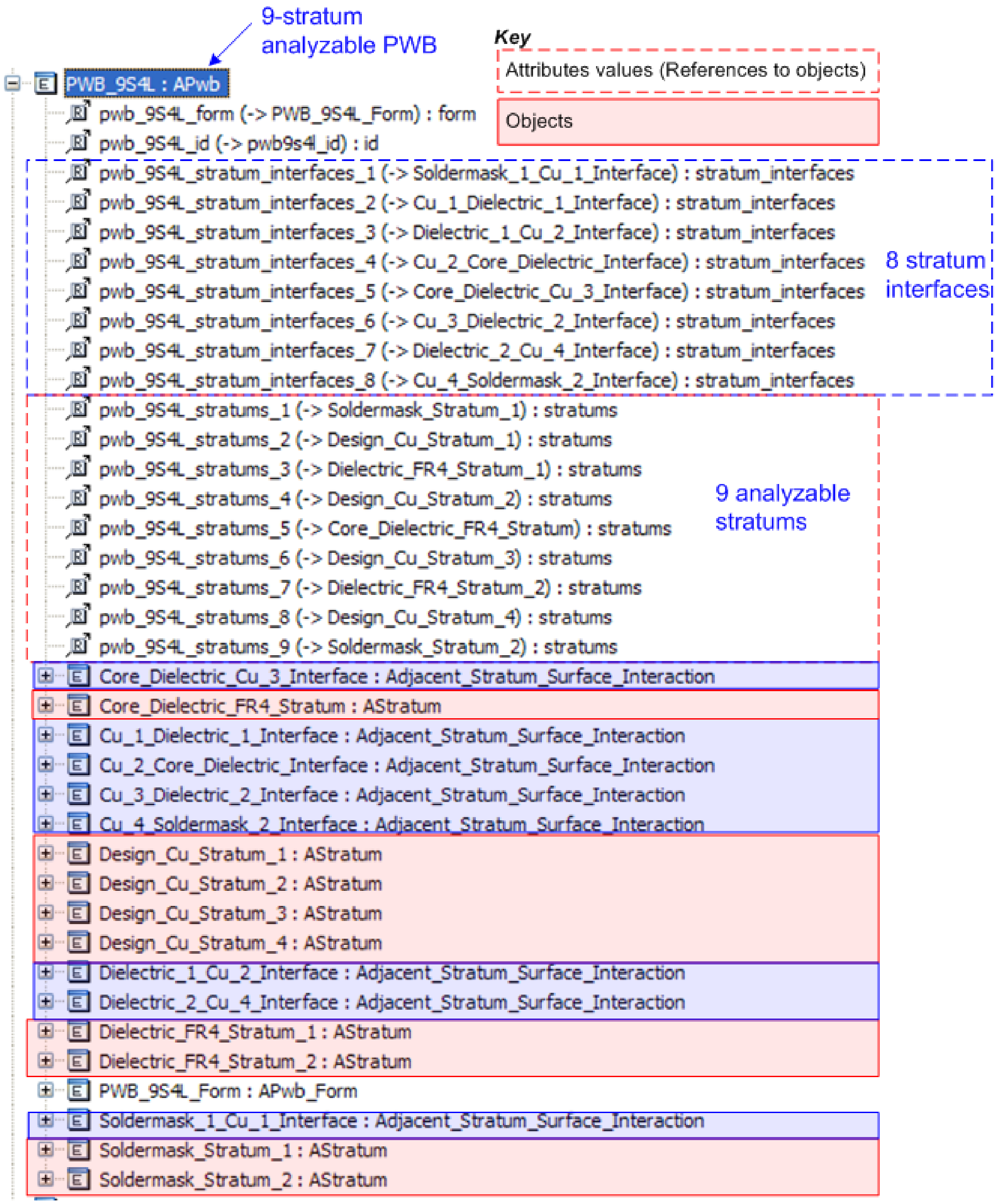
```
Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_AB
    associated_mb_property_attr (-> LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_1) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1) : constituent_primary_surface_feature
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1) : constituent_secondary_surface_feature
    id_attr (-> PSAB_Design_Cu_Stratum_1) : id
    shape_attr (-> PSS_Planar_Shell_AB_Design_Cu_Stratum_1) : shape
  LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_1 : Linear_Elastic_Isotropic_Temperature_Independent_MB
  PSS_Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_Shape
  Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_Primary_Surface
  Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_Secondary_Surface
Planar_Shell_AB_Design_Cu_Stratum_2 : Planar_Shell_AB
    associated_mb_property_attr (-> LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_2) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_2) : constituent_primary_surface_feature
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_2) : constituent_secondary_surface_feature
    id_attr (-> PSAB_Design_Cu_Stratum_2) : id
    shape_attr (-> PSS_Planar_Shell_AB_Design_Cu_Stratum_2) : shape
  LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_2 : Linear_Elastic_Isotropic_Temperature_Independent_MB
  PSS_Planar_Shell_AB_Design_Cu_Stratum_2 : Planar_Shell_Shape
  Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_2 : Planar_Shell_Primary_Surface
  Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_2 : Planar_Shell_Secondary_Surface
Planar_Shell_AB_Dielectric_FR4_Stratum : Planar_Shell_AB
    associated_mb_property_attr (-> LEITI_PSAB_Planar_Shell_AB_Dielectric_FR4_Stratum) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Dielectric_FR4_Stratum) : constituent_primary_surface_feature
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Dielectric_FR4_Stratum) : constituent_secondary_surface_feature
    id_attr (-> PSAB_Dielectric_FR4_Stratum) : id
    shape_attr (-> PSS_Planar_Shell_AB_Dielectric_FR4_Stratum) : shape
  LEITI_PSAB_Planar_Shell_AB_Dielectric_FR4_Stratum : Linear_Elastic_Isotropic_Temperature_Independent_MB
  PSS_Planar_Shell_AB_Dielectric_FR4_Stratum : Planar_Shell_Shape
  Primary_Surface_Planar_Shell_AB_Dielectric_FR4_Stratum : Planar_Shell_Primary_Surface
  Secondary_Surface_Planar_Shell_AB_Dielectric_FR4_Stratum : Planar_Shell_Secondary_Surface
Planar_Shell_AB_Soldermask_Stratum_1 : Planar_Shell_AB
    associated_mb_property_attr (-> LEITI_PSAB_Planar_Shell_AB_Soldermask_Stratum_1) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Soldermask_Stratum_1) : constituent_primary_surface_feature
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_1) : constituent_secondary_surface_feature
    id_attr (-> PSAB_Soldermask_Stratum_1) : id
    shape_attr (-> PSS_Planar_Shell_AB_Soldermask_Stratum_1) : shape
  LEITI_PSAB_Planar_Shell_AB_Soldermask_Stratum_1 : Linear_Elastic_Isotropic_Temperature_Independent_MB
  PSS_Planar_Shell_AB_Soldermask_Stratum_1 : Planar_Shell_Shape
  Primary_Surface_Planar_Shell_AB_Soldermask_Stratum_1 : Planar_Shell_Primary_Surface
  Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_1 : Planar_Shell_Secondary_Surface
Planar_Shell_AB_Soldermask_Stratum_2 : Planar_Shell_AB
    associated_mb_property_attr (-> LEITI_PSAB_Planar_Shell_AB_Soldermask_Stratum_2) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Soldermask_Stratum_2) : constituent_primary_surface_feature
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2) : constituent_secondary_surface_feature
    id_attr (-> PSAB_Soldermask_Stratum_2) : id
    shape_attr (-> PSS_Planar_Shell_AB_Soldermask_Stratum_2) : shape
  LEITI_PSAB_Planar_Shell_AB_Soldermask_Stratum_2 : Linear_Elastic_Isotropic_Temperature_Independent_MB
  PSS_Planar_Shell_AB_Soldermask_Stratum_2 : Planar_Shell_Shape
  Primary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 : Planar_Shell_Primary_Surface
  Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 : Planar_Shell_Secondary_Surface
```

Planar shell analysis body
Planar shell analysis body
Planar shell analysis body
Planar shell analysis body
Planar shell analysis body

Figure 9.17:Planar shell analysis bodies created in Stage 1 composition

9.2.1.2 Simulation Template ${}_9{}^1$: Simulation template for 9-stratum PWB design model structure and BMFS^1

In this section, the simulation template automatically created for the same Behavior Model Formulation Specifications (BMFS^1) as in the previous section but for a 9-stratum analyzable PWB design is presented.

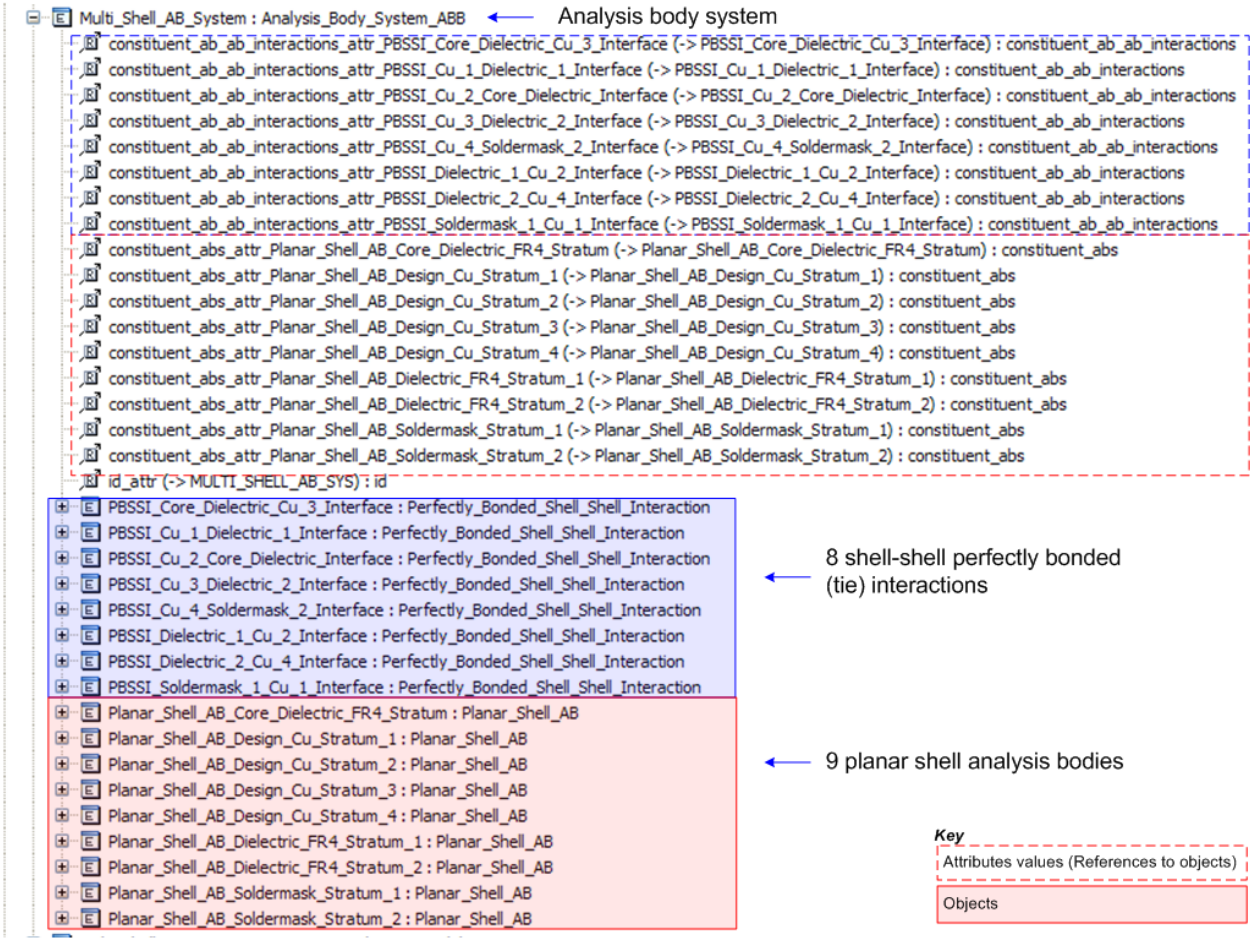


PWB_9S4L : APwb
pwb_9S4L_form (-> PWB_9S4L_Form) : form
pwb_9S4L_id (-> pwb9s4l_id) : id
pwb_9S4L_stratum_interfaces_1 (-> Soldermask_1_Cu_1_Interface) : stratum_interfaces
pwb_9S4L_stratum_interfaces_2 (-> Cu_1_Dielectric_1_Interface) : stratum_interfaces
pwb_9S4L_stratum_interfaces_3 (-> Dielectric_1_Cu_2_Interface) : stratum_interfaces
pwb_9S4L_stratum_interfaces_4 (-> Cu_2_Core_Dielectric_Interface) : stratum_interfaces
pwb_9S4L_stratum_interfaces_5 (-> Core_Dielectric_Cu_3_Interface) : stratum_interfaces
pwb_9S4L_stratum_interfaces_6 (-> Cu_3_Dielectric_2_Interface) : stratum_interfaces
pwb_9S4L_stratum_interfaces_7 (-> Dielectric_2_Cu_4_Interface) : stratum_interfaces
pwb_9S4L_stratum_interfaces_8 (-> Cu_4_Soldermask_2_Interface) : stratum_interfaces
pwb_9S4L_stratums_1 (-> Soldermask_Stratum_1) : stratums
pwb_9S4L_stratums_2 (-> Design_Cu_Stratum_1) : stratums
pwb_9S4L_stratums_3 (-> Dielectric_FR4_Stratum_1) : stratums
pwb_9S4L_stratums_4 (-> Design_Cu_Stratum_2) : stratums
pwb_9S4L_stratums_5 (-> Core_Dielectric_FR4_Stratum) : stratums
pwb_9S4L_stratums_6 (-> Design_Cu_Stratum_3) : stratums
pwb_9S4L_stratums_7 (-> Dielectric_FR4_Stratum_2) : stratums
pwb_9S4L_stratums_8 (-> Design_Cu_Stratum_4) : stratums
pwb_9S4L_stratums_9 (-> Soldermask_Stratum_2) : stratums
Core_Dielectric_Cu_3_Interface : Adjacent_Stratum_Surface_Interaction
Core_Dielectric_FR4_Stratum : AStratum
Cu_1_Dielectric_1_Interface : Adjacent_Stratum_Surface_Interaction
Cu_2_Core_Dielectric_Interface : Adjacent_Stratum_Surface_Interaction
Cu_3_Dielectric_2_Interface : Adjacent_Stratum_Surface_Interaction
Cu_4_Soldermask_2_Interface : Adjacent_Stratum_Surface_Interaction
Design_Cu_Stratum_1 : AStratum
Design_Cu_Stratum_2 : AStratum
Design_Cu_Stratum_3 : AStratum
Design_Cu_Stratum_4 : AStratum
Dielectric_1_Cu_2_Interface : Adjacent_Stratum_Surface_Interaction
Dielectric_2_Cu_4_Interface : Adjacent_Stratum_Surface_Interaction
Dielectric_FR4_Stratum_1 : AStratum
Dielectric_FR4_Stratum_2 : AStratum
PWB_9S4L_Form : APwb_Form
Soldermask_1_Cu_1_Interface : Adjacent_Stratum_Surface_Interaction
Soldermask_Stratum_1 : AStratum
Soldermask_Stratum_2 : AStratum

Figure 9.18: PWB_9S4L has 9 analyzable stratums and 8 stratum interfaces

Figure 9.18 illustrates PWB_9S4L—a 9-stratum, 4-layer analyzable PWB design model structure—in the VIATRA model space. Like PWB_5S2L presented in the previous section, PWB_9S4L is also a Level 4 model in the design model stack, and is an instance of the Analyzable_Electronics_Design_Meta-Model (section 6.2).

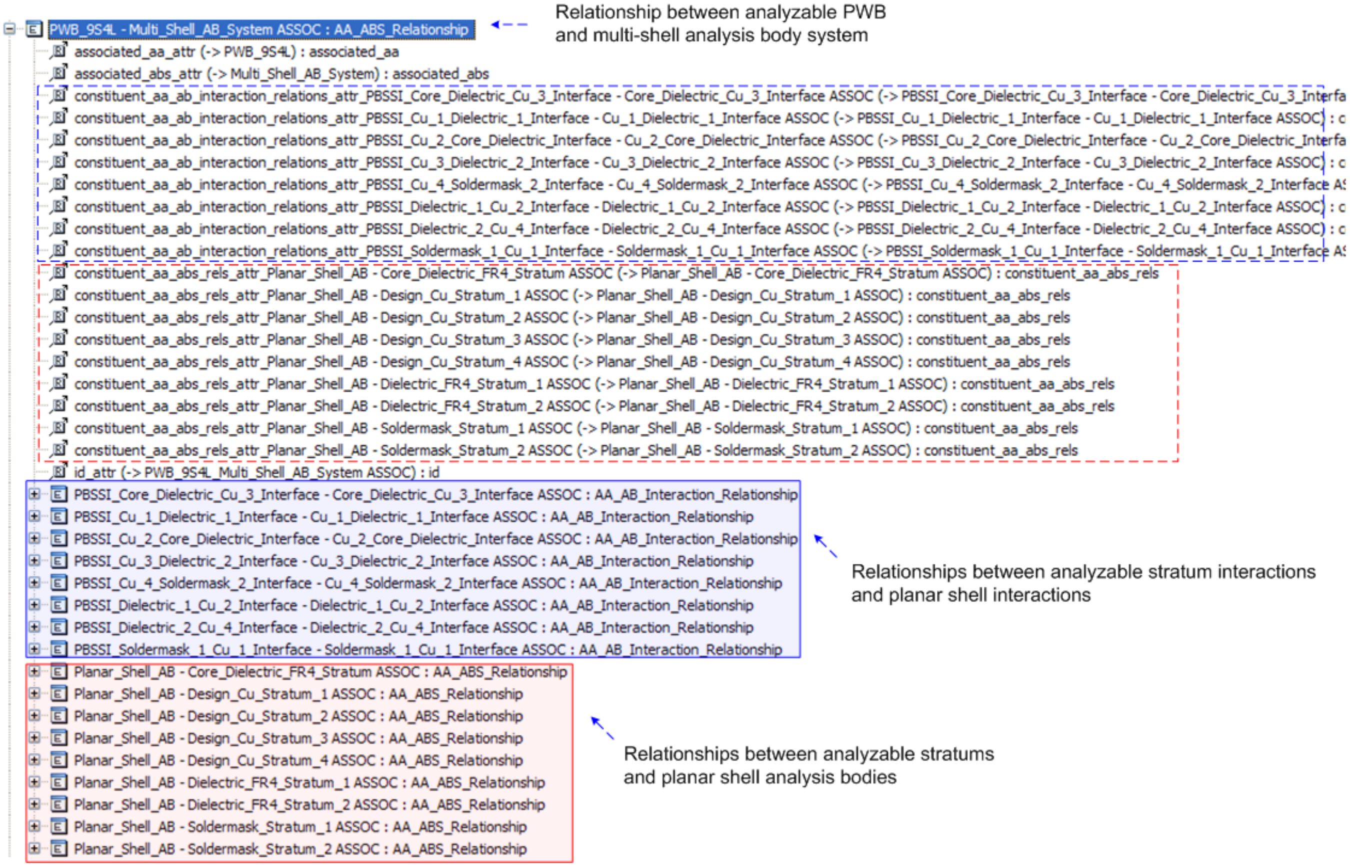
Only those aspects of the simulation template are presented here that are different for the 9-stratum analyzable PWB design. Simulation template entities and relationships created in composition Stages 1, 3 and 4 are the same for Simulation Template $_5^1$ and Simulation Template $_9^1$. However, results of composition Stage 2 are different. This is so because the changes in assembly system topology due to changes in the number of analyzable artifacts and their interactions (as in this case) affects the number of analysis bodies and their interactions in the analysis body system—composed in Stage 2 composition.


Analysis body system
8 shell-shell perfectly bonded (tie) interactions
9 planar shell analysis bodies
Key
Attributes values (References to objects)
Objects

```
Multi_Shell_AB_System : Analysis_Body_System_ABB
  constituent_ab_ab_interactions_attr_PBSSI_Core_Dielectric_Cu_3_Interface (-> PBSSI_Core_Dielectric_Cu_3_Interface) : constituent_ab_ab_interactions
  constituent_ab_ab_interactions_attr_PBSSI_Cu_1_Dielectric_1_Interface (-> PBSSI_Cu_1_Dielectric_1_Interface) : constituent_ab_ab_interactions
  constituent_ab_ab_interactions_attr_PBSSI_Cu_2_Core_Dielectric_Interface (-> PBSSI_Cu_2_Core_Dielectric_Interface) : constituent_ab_ab_interactions
  constituent_ab_ab_interactions_attr_PBSSI_Cu_3_Dielectric_2_Interface (-> PBSSI_Cu_3_Dielectric_2_Interface) : constituent_ab_ab_interactions
  constituent_ab_ab_interactions_attr_PBSSI_Cu_4_Soldermask_2_Interface (-> PBSSI_Cu_4_Soldermask_2_Interface) : constituent_ab_ab_interactions
  constituent_ab_ab_interactions_attr_PBSSI_Dielectric_1_Cu_2_Interface (-> PBSSI_Dielectric_1_Cu_2_Interface) : constituent_ab_ab_interactions
  constituent_ab_ab_interactions_attr_PBSSI_Dielectric_2_Cu_4_Interface (-> PBSSI_Dielectric_2_Cu_4_Interface) : constituent_ab_ab_interactions
  constituent_ab_ab_interactions_attr_PBSSI_Soldermask_1_Cu_1_Interface (-> PBSSI_Soldermask_1_Cu_1_Interface) : constituent_ab_ab_interactions
  constituent_abs_attr_Planar_Shell_AB_Core_Dielectric_FR4_Stratum (-> Planar_Shell_AB_Core_Dielectric_FR4_Stratum) : constituent_abs
  constituent_abs_attr_Planar_Shell_AB_Design_Cu_Stratum_1 (-> Planar_Shell_AB_Design_Cu_Stratum_1) : constituent_abs
  constituent_abs_attr_Planar_Shell_AB_Design_Cu_Stratum_2 (-> Planar_Shell_AB_Design_Cu_Stratum_2) : constituent_abs
  constituent_abs_attr_Planar_Shell_AB_Design_Cu_Stratum_3 (-> Planar_Shell_AB_Design_Cu_Stratum_3) : constituent_abs
  constituent_abs_attr_Planar_Shell_AB_Design_Cu_Stratum_4 (-> Planar_Shell_AB_Design_Cu_Stratum_4) : constituent_abs
  constituent_abs_attr_Planar_Shell_AB_Dielectric_FR4_Stratum_1 (-> Planar_Shell_AB_Dielectric_FR4_Stratum_1) : constituent_abs
  constituent_abs_attr_Planar_Shell_AB_Dielectric_FR4_Stratum_2 (-> Planar_Shell_AB_Dielectric_FR4_Stratum_2) : constituent_abs
  constituent_abs_attr_Planar_Shell_AB_Soldermask_Stratum_1 (-> Planar_Shell_AB_Soldermask_Stratum_1) : constituent_abs
  constituent_abs_attr_Planar_Shell_AB_Soldermask_Stratum_2 (-> Planar_Shell_AB_Soldermask_Stratum_2) : constituent_abs
  id_attr (-> MULTI_SHELL_AB_SYS) : id
  PBSSI_Core_Dielectric_Cu_3_Interface : Perfectly_Bonded_Shell_Shell_Interaction
  PBSSI_Cu_1_Dielectric_1_Interface : Perfectly_Bonded_Shell_Shell_Interaction
  PBSSI_Cu_2_Core_Dielectric_Interface : Perfectly_Bonded_Shell_Shell_Interaction
  PBSSI_Cu_3_Dielectric_2_Interface : Perfectly_Bonded_Shell_Shell_Interaction
  PBSSI_Cu_4_Soldermask_2_Interface : Perfectly_Bonded_Shell_Shell_Interaction
  PBSSI_Dielectric_1_Cu_2_Interface : Perfectly_Bonded_Shell_Shell_Interaction
  PBSSI_Dielectric_2_Cu_4_Interface : Perfectly_Bonded_Shell_Shell_Interaction
  PBSSI_Soldermask_1_Cu_1_Interface : Perfectly_Bonded_Shell_Shell_Interaction
  Planar_Shell_AB_Core_Dielectric_FR4_Stratum : Planar_Shell_AB
  Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_AB
  Planar_Shell_AB_Design_Cu_Stratum_2 : Planar_Shell_AB
  Planar_Shell_AB_Design_Cu_Stratum_3 : Planar_Shell_AB
  Planar_Shell_AB_Design_Cu_Stratum_4 : Planar_Shell_AB
  Planar_Shell_AB_Dielectric_FR4_Stratum_1 : Planar_Shell_AB
  Planar_Shell_AB_Dielectric_FR4_Stratum_2 : Planar_Shell_AB
  Planar_Shell_AB_Soldermask_Stratum_1 : Planar_Shell_AB
  Planar_Shell_AB_Soldermask_Stratum_2 : Planar_Shell_AB
```

Figure 9.19: Multi-shell analysis body system created in composition Stage 2 for Simulation Template$_9^1$

Figure 9.18 illustrates the analysis body system automatically created in Stage 2 of the composition process for Simulation Template $_9{}^1$. The figure shows 9 planar shell analysis bodies and 8 shell-shell perfectly bonded (tie) interactions created as components of multi-shell analysis body system at the end of composition Stage 2. The 9 analysis bodies correspond to the 9 analyzable stratums, and the 8 tie interactions correspond to the 8 stratum interfaces in PWB_9S4L.



```
PWB_9S4L - Multi_Shell_AB_System ASSOC : AA_ABS_Relationship
    associated_aa_attr (-> PWB_9S4L) : associated_aa
    associated_abs_attr (-> Multi_Shell_AB_System) : associated_abs
    constituent_aa_ab_interaction_relations_attr_PBSSI_Core_Dielectric_Cu_3_Interface - Core_Dielectric_Cu_3_Interface ASSOC (-> PBSSI_Core_Dielectric_Cu_3_Interface - Core_Dielectric_Cu_3_Interfa
    constituent_aa_ab_interaction_relations_attr_PBSSI_Cu_1_Dielectric_1_Interface - Cu_1_Dielectric_1_Interface ASSOC (-> PBSSI_Cu_1_Dielectric_1_Interface - Cu_1_Dielectric_1_Interface ASSOC) : c
    constituent_aa_ab_interaction_relations_attr_PBSSI_Cu_2_Core_Dielectric_Interface - Cu_2_Core_Dielectric_Interface ASSOC (-> PBSSI_Cu_2_Core_Dielectric_Interface - Cu_2_Core_Dielectric_Interfa
    constituent_aa_ab_interaction_relations_attr_PBSSI_Cu_3_Dielectric_2_Interface - Cu_3_Dielectric_2_Interface ASSOC (-> PBSSI_Cu_3_Dielectric_2_Interface - Cu_3_Dielectric_2_Interface ASSOC) : c
    constituent_aa_ab_interaction_relations_attr_PBSSI_Cu_4_Soldermask_2_Interface - Cu_4_Soldermask_2_Interface ASSOC (-> PBSSI_Cu_4_Soldermask_2_Interface - Cu_4_Soldermask_2_Interface A
    constituent_aa_ab_interaction_relations_attr_PBSSI_Dielectric_1_Cu_2_Interface - Dielectric_1_Cu_2_Interface ASSOC (-> PBSSI_Dielectric_1_Cu_2_Interface - Dielectric_1_Cu_2_Interface ASSOC) : c
    constituent_aa_ab_interaction_relations_attr_PBSSI_Dielectric_2_Cu_4_Interface - Dielectric_2_Cu_4_Interface ASSOC (-> PBSSI_Dielectric_2_Cu_4_Interface - Dielectric_2_Cu_4_Interface ASSOC) : c
    constituent_aa_ab_interaction_relations_attr_PBSSI_Soldermask_1_Cu_1_Interface - Soldermask_1_Cu_1_Interface ASSOC (-> PBSSI_Soldermask_1_Cu_1_Interface - Soldermask_1_Cu_1_Interface A
    constituent_aa_abs_rels_attr_Planar_Shell_AB - Core_Dielectric_FR4_Stratum ASSOC (-> Planar_Shell_AB - Core_Dielectric_FR4_Stratum ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_Planar_Shell_AB - Design_Cu_Stratum_1 ASSOC (-> Planar_Shell_AB - Design_Cu_Stratum_1 ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_Planar_Shell_AB - Design_Cu_Stratum_2 ASSOC (-> Planar_Shell_AB - Design_Cu_Stratum_2 ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_Planar_Shell_AB - Design_Cu_Stratum_3 ASSOC (-> Planar_Shell_AB - Design_Cu_Stratum_3 ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_Planar_Shell_AB - Design_Cu_Stratum_4 ASSOC (-> Planar_Shell_AB - Design_Cu_Stratum_4 ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_Planar_Shell_AB - Dielectric_FR4_Stratum_1 ASSOC (-> Planar_Shell_AB - Dielectric_FR4_Stratum_1 ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_Planar_Shell_AB - Dielectric_FR4_Stratum_2 ASSOC (-> Planar_Shell_AB - Dielectric_FR4_Stratum_2 ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_Planar_Shell_AB - Soldermask_Stratum_1 ASSOC (-> Planar_Shell_AB - Soldermask_Stratum_1 ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_Planar_Shell_AB - Soldermask_Stratum_2 ASSOC (-> Planar_Shell_AB - Soldermask_Stratum_2 ASSOC) : constituent_aa_abs_rels
    id_attr (-> PWB_9S4L_Multi_Shell_AB_System ASSOC) : id
  PBSSI_Core_Dielectric_Cu_3_Interface - Core_Dielectric_Cu_3_Interface ASSOC : AA_AB_Interaction_Relationship
  PBSSI_Cu_1_Dielectric_1_Interface - Cu_1_Dielectric_1_Interface ASSOC : AA_AB_Interaction_Relationship
  PBSSI_Cu_2_Core_Dielectric_Interface - Cu_2_Core_Dielectric_Interface ASSOC : AA_AB_Interaction_Relationship
  PBSSI_Cu_3_Dielectric_2_Interface - Cu_3_Dielectric_2_Interface ASSOC : AA_AB_Interaction_Relationship
  PBSSI_Cu_4_Soldermask_2_Interface - Cu_4_Soldermask_2_Interface ASSOC : AA_AB_Interaction_Relationship
  PBSSI_Dielectric_1_Cu_2_Interface - Dielectric_1_Cu_2_Interface ASSOC : AA_AB_Interaction_Relationship
  PBSSI_Dielectric_2_Cu_4_Interface - Dielectric_2_Cu_4_Interface ASSOC : AA_AB_Interaction_Relationship
  PBSSI_Soldermask_1_Cu_1_Interface - Soldermask_1_Cu_1_Interface ASSOC : AA_AB_Interaction_Relationship
  Planar_Shell_AB - Core_Dielectric_FR4_Stratum ASSOC : AA_ABS_Relationship
  Planar_Shell_AB - Design_Cu_Stratum_1 ASSOC : AA_ABS_Relationship
  Planar_Shell_AB - Design_Cu_Stratum_2 ASSOC : AA_ABS_Relationship
  Planar_Shell_AB - Design_Cu_Stratum_3 ASSOC : AA_ABS_Relationship
  Planar_Shell_AB - Design_Cu_Stratum_4 ASSOC : AA_ABS_Relationship
  Planar_Shell_AB - Dielectric_FR4_Stratum_1 ASSOC : AA_ABS_Relationship
  Planar_Shell_AB - Dielectric_FR4_Stratum_2 ASSOC : AA_ABS_Relationship
  Planar_Shell_AB - Soldermask_Stratum_1 ASSOC : AA_ABS_Relationship
  Planar_Shell_AB - Soldermask_Stratum_2 ASSOC : AA_ABS_Relationship
```

Relationship between analyzable PWB
and multi-shell analysis body system

Relationships between analyzable stratum interactions
and planar shell interactions

Relationships between analyzable stratums
and planar shell analysis bodies

Figure 9.20: Relationship between analyzable PWB design and multi-shell analysis body system created in composition Stage 2

Figure 9.20 illustrates the relationship between PWB_9S4L and the multi-shell analysis body system created at the end of Stage 2 composition process. Note that in this case nine analyzable stratum–planar shell analysis body relationships have been created (one for each pair), and eight relationships have been created between analyzable stratum interfaces and analysis body tie interactions (one for each pair). Hence for the same BMFS, the Behavior Model Formulation Method can be used to automatically compose simulation templates for design alternatives with non-equivalent assembly system topologies.

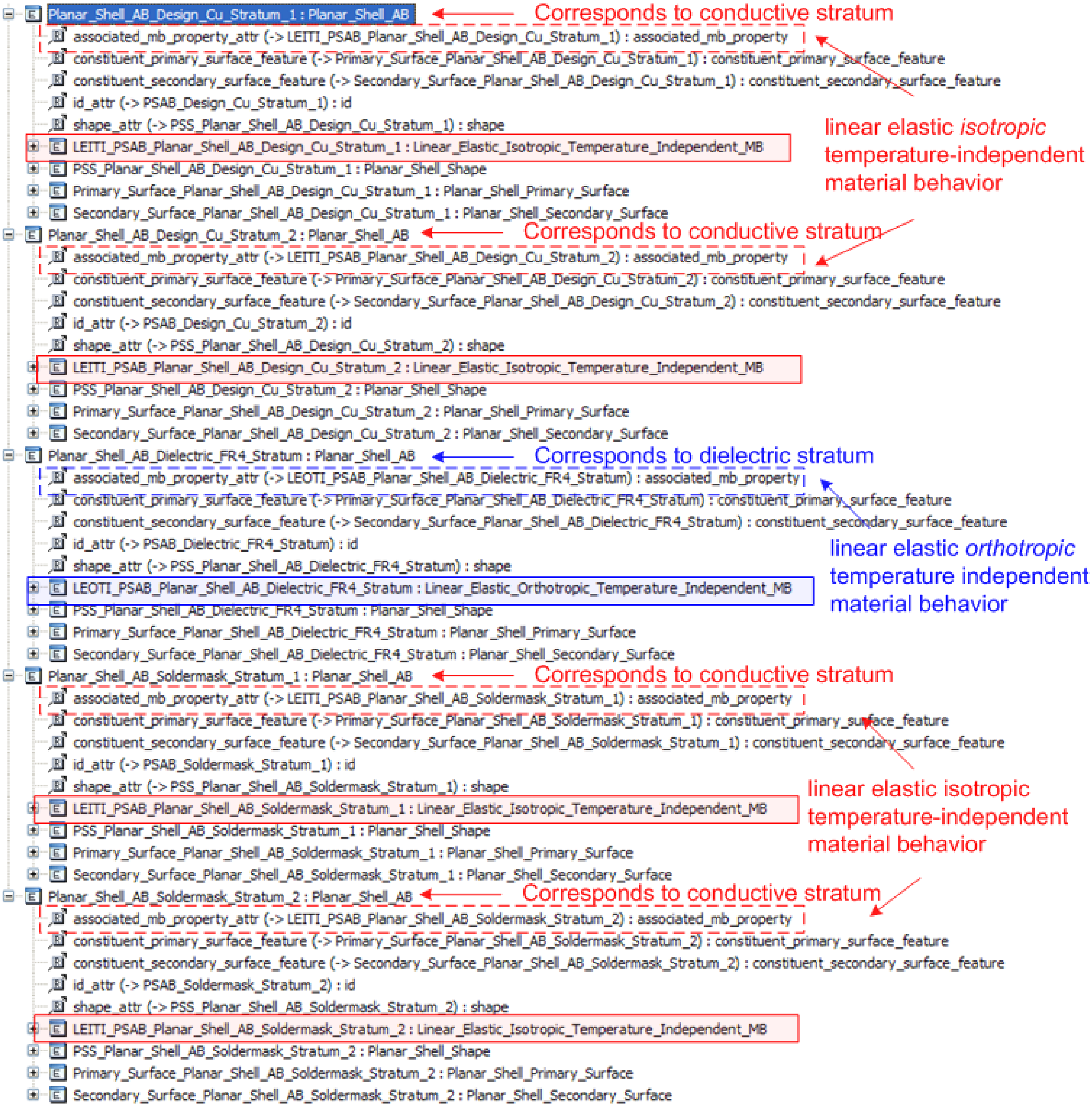
9.2.2 Behavior Model Formulation Specifications 2 (BMFS^2)

In this section, simulation templates automatically generated for the second set of Behavior Model Formulation Specifications (BMFS^2) are presented. In BMFS^2, the material behavior idealization decisions are changed as compared to BMFS^1. Instead of idealizing the material behavior of all analyzable stratums as linear, elastic, isotropic, and temperature independent (as in BMFS^1), the following conditions is used to select the material behavior ABB to be associated with a planar shell analysis body associated with an analyzable stratum:

```
If (electrical function of an analyzable stratum is CONDUCTIVE or SOLDERMASK)
   Select linear elastic isotropic temperature-independent material behavior ABB
Else if (electrical function is DIELECTRIC)
   Select linear elastic orthotropic temperature independent material behavior ABB
```

These idealization decisions are reflected in both the conceptual and computable Behavior Model Formulation Specifications. Note that these idealization decisions are at the level of individual analysis bodies since material behavior is an attribute of an analysis body. Thus, the new idealizations in BMFS^2 affect results of the Stage 1 composition only. Hence, only results of the Stage 1 composition are presented below. Figure 9.21 and Figure 9.22 below illustrate the planar shell analysis bodies created for PWB_5S4L and PWB_9S4L with the new idealization decisions embodied in BMFS^2. The figures show that the planar shell analysis bodies corresponding to conductive and soldermask stratums are associated with instances of Linear Elastic Isotropic Temperature-independent Material Behavior ABB, and those associated with dielectric stratums are associated with instances of Linear Elastic Orthotropic Temperature-independent Material Behavior ABB during composition Stage 1. The figures clearly illustrate that with changes in idealization decisions, simulation templates can be easily and automatically generated for design alternatives with different assembly system topologies.

9.2.2.1 Simulation Template $_5^2$: Simulation template for 5-stratum PWB design model structure and BMFS^2



```
Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_AB
    associated_mb_property_attr (-> LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_1) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1) : constituent_primary_surface_feature
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1) : constituent_secondary_surface_feature
    id_attr (-> PSAB_Design_Cu_Stratum_1) : id
    shape_attr (-> PSS_Planar_Shell_AB_Design_Cu_Stratum_1) : shape
  + LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_1 : Linear_Elastic_Isotropic_Temperature_Independent_MB
  + PSS_Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_Shape
  + Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_Primary_Surface
  + Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_Secondary_Surface
Planar_Shell_AB_Design_Cu_Stratum_2 : Planar_Shell_AB
    associated_mb_property_attr (-> LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_2) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_2) : constituent_primary_surface_feature
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_2) : constituent_secondary_surface_feature
    id_attr (-> PSAB_Design_Cu_Stratum_2) : id
    shape_attr (-> PSS_Planar_Shell_AB_Design_Cu_Stratum_2) : shape
  + LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_2 : Linear_Elastic_Isotropic_Temperature_Independent_MB
  + PSS_Planar_Shell_AB_Design_Cu_Stratum_2 : Planar_Shell_Shape
  + Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_2 : Planar_Shell_Primary_Surface
  + Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_2 : Planar_Shell_Secondary_Surface
Planar_Shell_AB_Dielectric_FR4_Stratum : Planar_Shell_AB
    associated_mb_property_attr (-> LEOTI_PSAB_Planar_Shell_AB_Dielectric_FR4_Stratum) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Dielectric_FR4_Stratum) : constituent_primary_surface_feature
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Dielectric_FR4_Stratum) : constituent_secondary_surface_feature
    id_attr (-> PSAB_Dielectric_FR4_Stratum) : id
    shape_attr (-> PSS_Planar_Shell_AB_Dielectric_FR4_Stratum) : shape
  + LEOTI_PSAB_Planar_Shell_AB_Dielectric_FR4_Stratum : Linear_Elastic_Orthotropic_Temperature_Independent_MB
  + PSS_Planar_Shell_AB_Dielectric_FR4_Stratum : Planar_Shell_Shape
  + Primary_Surface_Planar_Shell_AB_Dielectric_FR4_Stratum : Planar_Shell_Primary_Surface
  + Secondary_Surface_Planar_Shell_AB_Dielectric_FR4_Stratum : Planar_Shell_Secondary_Surface
Planar_Shell_AB_Soldermask_Stratum_1 : Planar_Shell_AB
    associated_mb_property_attr (-> LEITI_PSAB_Planar_Shell_AB_Soldermask_Stratum_1) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Soldermask_Stratum_1) : constituent_primary_surface_feature
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_1) : constituent_secondary_surface_feature
    id_attr (-> PSAB_Soldermask_Stratum_1) : id
    shape_attr (-> PSS_Planar_Shell_AB_Soldermask_Stratum_1) : shape
  + LEITI_PSAB_Planar_Shell_AB_Soldermask_Stratum_1 : Linear_Elastic_Isotropic_Temperature_Independent_MB
  + PSS_Planar_Shell_AB_Soldermask_Stratum_1 : Planar_Shell_Shape
  + Primary_Surface_Planar_Shell_AB_Soldermask_Stratum_1 : Planar_Shell_Primary_Surface
  + Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_1 : Planar_Shell_Secondary_Surface
Planar_Shell_AB_Soldermask_Stratum_2 : Planar_Shell_AB
    associated_mb_property_attr (-> LEITI_PSAB_Planar_Shell_AB_Soldermask_Stratum_2) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Soldermask_Stratum_2) : constituent_primary_surface_feature
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2) : constituent_secondary_surface_feature
    id_attr (-> PSAB_Soldermask_Stratum_2) : id
    shape_attr (-> PSS_Planar_Shell_AB_Soldermask_Stratum_2) : shape
  + LEITI_PSAB_Planar_Shell_AB_Soldermask_Stratum_2 : Linear_Elastic_Isotropic_Temperature_Independent_MB
  + PSS_Planar_Shell_AB_Soldermask_Stratum_2 : Planar_Shell_Shape
  + Primary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 : Planar_Shell_Primary_Surface
  + Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 : Planar_Shell_Secondary_Surface
```

Corresponds to conductive stratum

linear elastic *isotropic* temperature-independent material behavior

Corresponds to conductive stratum

Corresponds to dielectric stratum

linear elastic *orthotropic* temperature independent material behavior

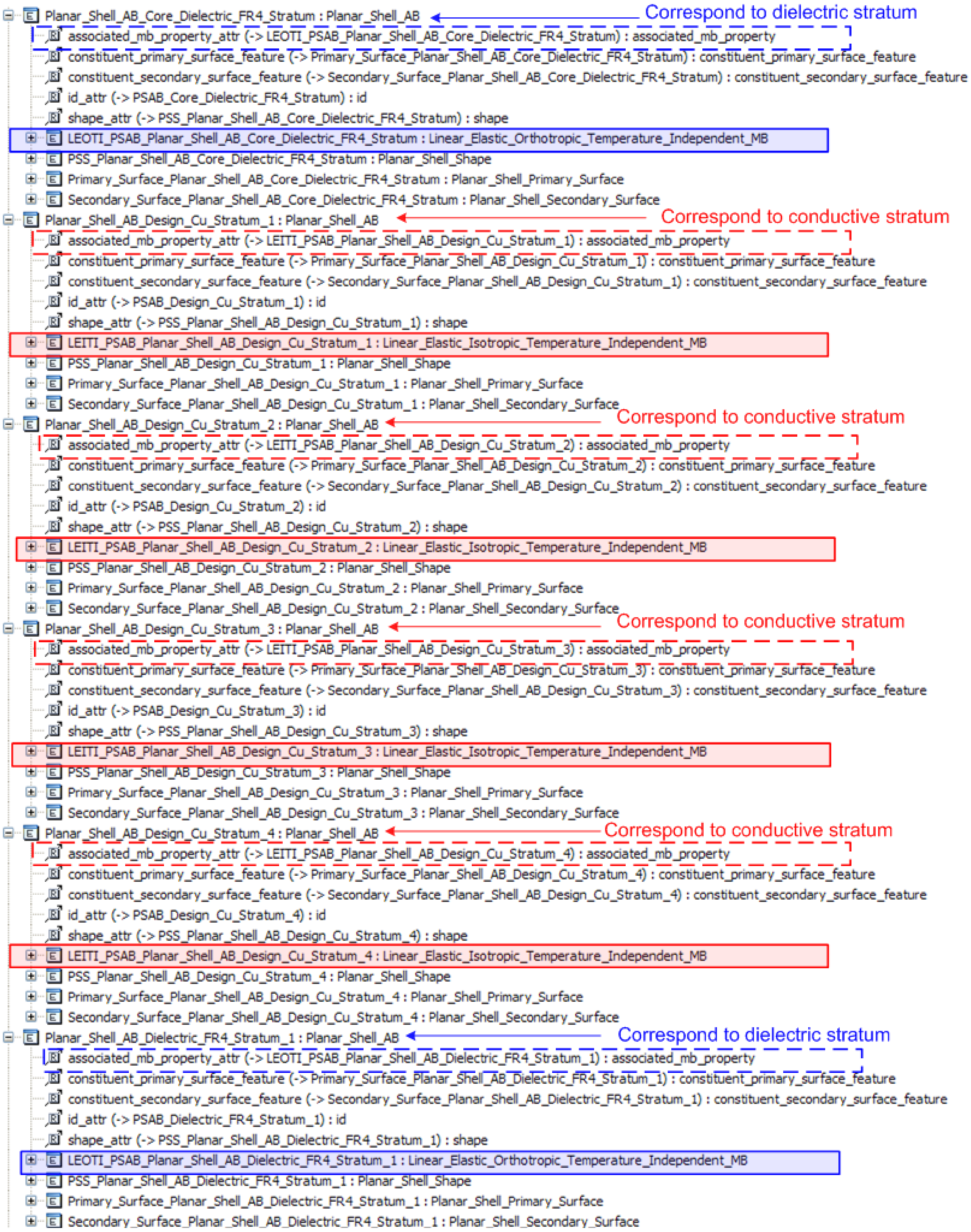
Corresponds to conductive stratum

linear elastic isotropic temperature-independent material behavior

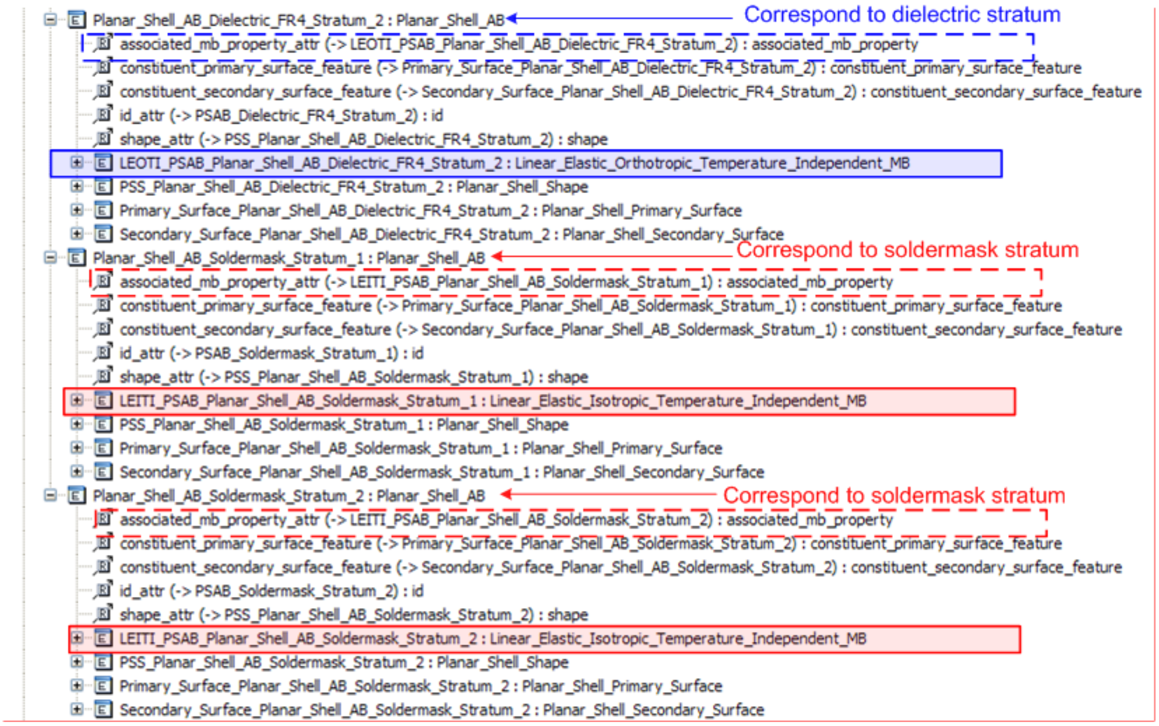
Corresponds to conductive stratum

Figure 9.21: Planar shell analysis bodies created for BMFS^2 and PWB_5S4L in composition Stage 1

9.2.2.2 Simulation Template $_9^2$: Simulation template for 9-stratum PWB design model structure and BMFS^2



```
Planar_Shell_AB_Core_Dielectric_FR4_Stratum : Planar_Shell_AB        Correspond to dielectric stratum
    associated_mb_property_attr (-> LEOTI_PSAB_Planar_Shell_AB_Core_Dielectric_FR4_Stratum) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Core_Dielectric_FR4_Stratum) : constituent_primary_surface_feature
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Core_Dielectric_FR4_Stratum) : constituent_secondary_surface_feature
    id_attr (-> PSAB_Core_Dielectric_FR4_Stratum) : id
    shape_attr (-> PSS_Planar_Shell_AB_Core_Dielectric_FR4_Stratum) : shape
    LEOTI_PSAB_Planar_Shell_AB_Core_Dielectric_FR4_Stratum : Linear_Elastic_Orthotropic_Temperature_Independent_MB
    PSS_Planar_Shell_AB_Core_Dielectric_FR4_Stratum : Planar_Shell_Shape
    Primary_Surface_Planar_Shell_AB_Core_Dielectric_FR4_Stratum : Planar_Shell_Primary_Surface
    Secondary_Surface_Planar_Shell_AB_Core_Dielectric_FR4_Stratum : Planar_Shell_Secondary_Surface
Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_AB        Correspond to conductive stratum
    associated_mb_property_attr (-> LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_1) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1) : constituent_primary_surface_feature
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1) : constituent_secondary_surface_feature
    id_attr (-> PSAB_Design_Cu_Stratum_1) : id
    shape_attr (-> PSS_Planar_Shell_AB_Design_Cu_Stratum_1) : shape
    LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_1 : Linear_Elastic_Isotropic_Temperature_Independent_MB
    PSS_Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_Shape
    Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_Primary_Surface
    Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_Secondary_Surface
Planar_Shell_AB_Design_Cu_Stratum_2 : Planar_Shell_AB        Correspond to conductive stratum
    associated_mb_property_attr (-> LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_2) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_2) : constituent_primary_surface_feature
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_2) : constituent_secondary_surface_feature
    id_attr (-> PSAB_Design_Cu_Stratum_2) : id
    shape_attr (-> PSS_Planar_Shell_AB_Design_Cu_Stratum_2) : shape
    LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_2 : Linear_Elastic_Isotropic_Temperature_Independent_MB
    PSS_Planar_Shell_AB_Design_Cu_Stratum_2 : Planar_Shell_Shape
    Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_2 : Planar_Shell_Primary_Surface
    Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_2 : Planar_Shell_Secondary_Surface
Planar_Shell_AB_Design_Cu_Stratum_3 : Planar_Shell_AB        Correspond to conductive stratum
    associated_mb_property_attr (-> LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_3) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_3) : constituent_primary_surface_feature
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_3) : constituent_secondary_surface_feature
    id_attr (-> PSAB_Design_Cu_Stratum_3) : id
    shape_attr (-> PSS_Planar_Shell_AB_Design_Cu_Stratum_3) : shape
    LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_3 : Linear_Elastic_Isotropic_Temperature_Independent_MB
    PSS_Planar_Shell_AB_Design_Cu_Stratum_3 : Planar_Shell_Shape
    Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_3 : Planar_Shell_Primary_Surface
    Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_3 : Planar_Shell_Secondary_Surface
Planar_Shell_AB_Design_Cu_Stratum_4 : Planar_Shell_AB        Correspond to conductive stratum
    associated_mb_property_attr (-> LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_4) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_4) : constituent_primary_surface_feature
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_4) : constituent_secondary_surface_feature
    id_attr (-> PSAB_Design_Cu_Stratum_4) : id
    shape_attr (-> PSS_Planar_Shell_AB_Design_Cu_Stratum_4) : shape
    LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_4 : Linear_Elastic_Isotropic_Temperature_Independent_MB
    PSS_Planar_Shell_AB_Design_Cu_Stratum_4 : Planar_Shell_Shape
    Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_4 : Planar_Shell_Primary_Surface
    Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_4 : Planar_Shell_Secondary_Surface
Planar_Shell_AB_Dielectric_FR4_Stratum_1 : Planar_Shell_AB        Correspond to dielectric stratum
    associated_mb_property_attr (-> LEOTI_PSAB_Planar_Shell_AB_Dielectric_FR4_Stratum_1) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Dielectric_FR4_Stratum_1) : constituent_primary_surface_feature
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Dielectric_FR4_Stratum_1) : constituent_secondary_surface_feature
    id_attr (-> PSAB_Dielectric_FR4_Stratum_1) : id
    shape_attr (-> PSS_Planar_Shell_AB_Dielectric_FR4_Stratum_1) : shape
    LEOTI_PSAB_Planar_Shell_AB_Dielectric_FR4_Stratum_1 : Linear_Elastic_Orthotropic_Temperature_Independent_MB
    PSS_Planar_Shell_AB_Dielectric_FR4_Stratum_1 : Planar_Shell_Shape
    Primary_Surface_Planar_Shell_AB_Dielectric_FR4_Stratum_1 : Planar_Shell_Primary_Surface
    Secondary_Surface_Planar_Shell_AB_Dielectric_FR4_Stratum_1 : Planar_Shell_Secondary_Surface
```



```
Planar_Shell_AB_Dielectric_FR4_Stratum_2 : Planar_Shell_AB
    associated_mb_property_attr (-> LEOTI_PSAB_Planar_Shell_AB_Dielectric_FR4_Stratum_2) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Dielectric_FR4_Stratum_2) : constituent_primary_surface_feature
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Dielectric_FR4_Stratum_2) : constituent_secondary_surface_feature
    id_attr (-> PSAB_Dielectric_FR4_Stratum_2) : id
    shape_attr (-> PSS_Planar_Shell_AB_Dielectric_FR4_Stratum_2) : shape
    LEOTI_PSAB_Planar_Shell_AB_Dielectric_FR4_Stratum_2 : Linear_Elastic_Orthotropic_Temperature_Independent_MB
    PSS_Planar_Shell_AB_Dielectric_FR4_Stratum_2 : Planar_Shell_Shape
    Primary_Surface_Planar_Shell_AB_Dielectric_FR4_Stratum_2 : Planar_Shell_Primary_Surface
    Secondary_Surface_Planar_Shell_AB_Dielectric_FR4_Stratum_2 : Planar_Shell_Secondary_Surface
Planar_Shell_AB_Soldermask_Stratum_1 : Planar_Shell_AB
    associated_mb_property_attr (-> LEITI_PSAB_Planar_Shell_AB_Soldermask_Stratum_1) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Soldermask_Stratum_1) : constituent_primary_surface_feature
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_1) : constituent_secondary_surface_feature
    id_attr (-> PSAB_Soldermask_Stratum_1) : id
    shape_attr (-> PSS_Planar_Shell_AB_Soldermask_Stratum_1) : shape
    LEITI_PSAB_Planar_Shell_AB_Soldermask_Stratum_1 : Linear_Elastic_Isotropic_Temperature_Independent_MB
    PSS_Planar_Shell_AB_Soldermask_Stratum_1 : Planar_Shell_Shape
    Primary_Surface_Planar_Shell_AB_Soldermask_Stratum_1 : Planar_Shell_Primary_Surface
    Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_1 : Planar_Shell_Secondary_Surface
Planar_Shell_AB_Soldermask_Stratum_2 : Planar_Shell_AB
    associated_mb_property_attr (-> LEITI_PSAB_Planar_Shell_AB_Soldermask_Stratum_2) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Soldermask_Stratum_2) : constituent_primary_surface_feature
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2) : constituent_secondary_surface_feature
    id_attr (-> PSAB_Soldermask_Stratum_2) : id
    shape_attr (-> PSS_Planar_Shell_AB_Soldermask_Stratum_2) : shape
    LEITI_PSAB_Planar_Shell_AB_Soldermask_Stratum_2 : Linear_Elastic_Isotropic_Temperature_Independent_MB
    PSS_Planar_Shell_AB_Soldermask_Stratum_2 : Planar_Shell_Shape
    Primary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 : Planar_Shell_Primary_Surface
    Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 : Planar_Shell_Secondary_Surface
```

Correspond to dielectric stratum

Correspond to soldermask stratum

Correspond to soldermask stratum

Figure 9.22: Planar shell analysis bodies created for BMFS^2 and PWB_9S4L in composition Stage 1

9.3 Test Case Family 2 (TCF2): Thermo-mechanical Analysis of Ball Grid Array (BGA) Chip Packages

A ball grid array (BGA) chip package is a surface mount electronic package that interconnects with a printed wiring board via balls of solder arranged in a grid on the bottom surface of the package. In general, an electronic chip package embodies integrated circuits (ICs). The solder balls on the bottom surface of a BGA[25] are meant to conduct electrical signals between the IC and the PWB on which the BGA is mounted. BGAs are commonly used today in most electronics devices, such has handhelds and computers. Figure 9.23 (left) shows snapshots of BGAs used for consumer electronics and microprocessors, and Figure 9.23 (right) shows a three-dimensional CAD model of an idealized BGA assembly—mold around the silicon chip is not shown. Figure 9.24 shows assembled and exploded views of an idealized BGA assembly mounted on a PWB. Figure 9.25 shows a cross-sectional view of a BGA assembly.

[25] Ball grid array chip packages are referred as BGAs for brevity

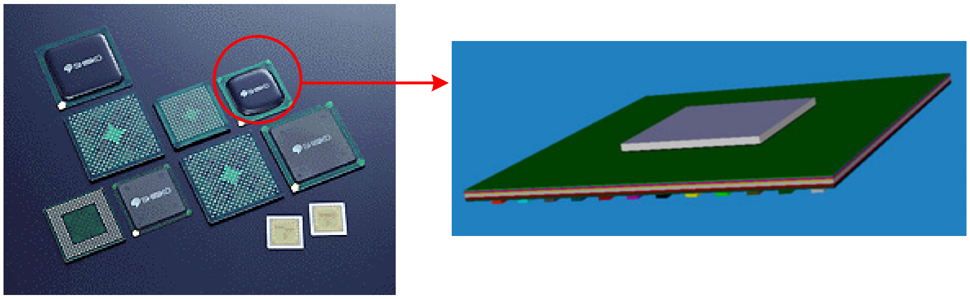
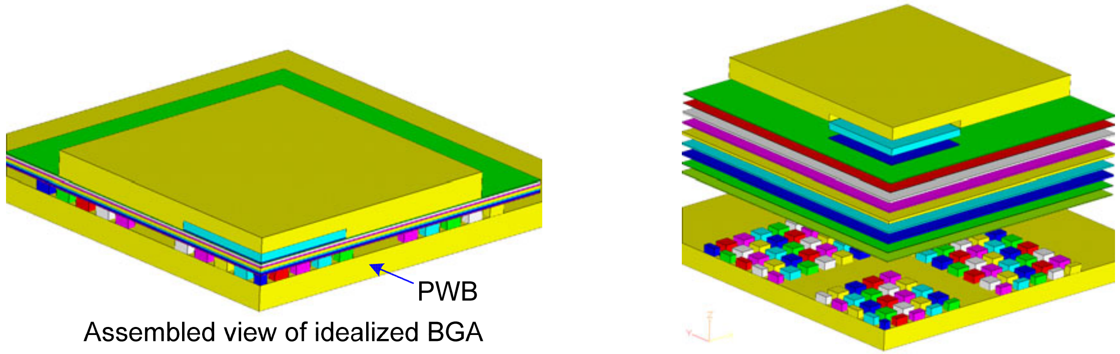


Photo: www.shinko.co.jp

Figure 9.23: Ball grid array (BGA) chip packages (left) and 3D CAD models of idealized BGAs (right)



Assembled view of idealized BGA mounted on PWB

Exploded view of idealized BGA mounted on PWB

Figure 9.24: Assembled and exploded views of an idealized BGA mounted on a PWB

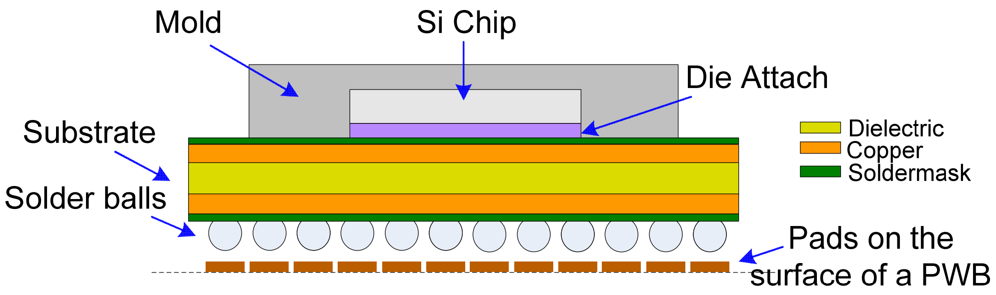


Figure 9.25: Cross-sectional view showing components of an idealized BGA chip package assembly

Figure 9.25 also shows the key components of a BGA assembly in the context of VTMB analysis problems presented in this section. The idealized BGA assembly shown in the figure consists of the following components:

- Substrate is a multi-layered structure similar to a PWB that embodies other electronic functions supporting the IC
- Solder balls are ball-shaped solder material structures that connect the chip package to a PWB, both electrically and mechanically. Solder balls are arranged in a grid on the

bottom surface of a BGA, and interface with the conductive pads on the surface of a PWB when the BGA is mounted.

- Si Chip is a silicon die that houses the integrated circuit.
- Die Attach is a mechanical adhesive that binds the chip to the substrate.
- Mold is an enclosing to protect the chip.

Note that the design of a BGA is more complicated and variant than described in the idealized view above. The idealized design presented above is the basis for analyzable design models used in this section for demonstrating the Behavior Model Formulation Method.

In this section, the Behavior Model Formulation Method is used to automatically generate simulation templates for thermo-mechanical analysis of BGAs. Thermo-mechanical issues lead to reliability problems for BGAs. The heat from the surrounding regions or that generated from the chip causes different materials in a BGA assembly to expand and contract differently due to mismatches in their coefficient of thermal expansions leading to deformation of the BGA assembly and reliability issues resulting thereof. In this section two analyzable BGA design models are considered—one with 16 solder bodies and one with 36[26] solder bodies. Two different Behavior Model Formulation Specifications are used for generating simulation templates for the two BGA assemblies that have non-equivalent assembly system topology. Table 9.3 below shows the four simulation templates that will be automatically created by the combination of two variable topology BGA alternatives and two Behavior Model Formulation Specifications. These simulation templates are presented in sections 9.3.1 and 9.3.2.

Table 9.3: Simulation templates created for thermo-mechanical analysis of BGAs

	Analyzable Design Model Structures	
	16-solder ball analyzable BGA	*36-solder ball analyzable BGA*
BMFS^1	Simulation Template $_{16}{}^1$	Simulation Template $_{36}{}^1$
BMFS^2	Simulation Template $_{16}{}^2$	Simulation Template $_{36}{}^2$

[26] Note that the number of solder balls may well be over 100 for a complex BGA. The low number of solder balls shown here are purely for demonstration of VTMB aspects of the Behavior Model Formulation Method.

The objective of the simulation templates generated here is to compute the deformation of a BGA assembly when it is uniformly heated, either due to the heat generated from the chip or the heat from the surroundings—as in an assembly process.

9.3.1 Behavior Model Formulation Specifications 1 (BMFS^1)

The conceptual specifications for BMFS^1 for all composition stages are stated in Table 9.4. In addition, Figure 9.26 illustrates the idealization decisions to create a Multi-Shell-Solid analysis body system corresponding to an analyzable BGA design model structure. The chip substrate is idealized in the same manner as the PWB in BMFS^2 in section 9.2.2. The chip, mold, and solder balls are idealized as generic solid analysis bodies with no shape idealizations—analysis body has the same shape as the analyzable artifact, and with linear elastic isotropic temperature-independent material behavior. The bottom surface of the chip mates with the die attach and the outer surface of the chip mates with the bottom (inner) surface of the mold. All these features are idealized as generic analysis features (instances of analysis feature ABB). The die attach is modeled as a planar shell analysis body ABB and its top (primary) and bottom (secondary) features are idealized as instances of planar surface analysis feature ABB—same as in the case for primary and secondary surfaces of all stratums in the chip substrate. The solder ball is also idealized as a generic solid with linear elastic isotropic temperature-independent material behavior. The solder ball is shaped as a truncated sphere with two truncation features—top and bottom—that connect it with the chip substrate and PWB respectively.

For Stage 2 composition, all interfaces are idealized as tie interactions. The idealized BGA corresponds to a Multi-Shell-Solid analysis body system as shown in Figure 9.26 and stated in Table 9.4.

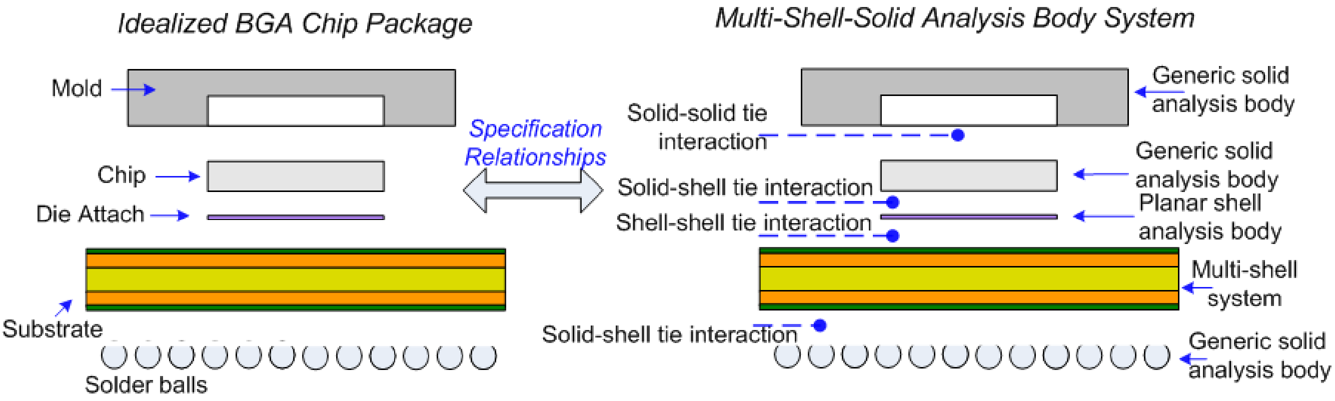


Figure 9.26: BMFS[1] relationship specifications between idealized BGA and Multi-Shell-Solid analysis body system

Table 9.4: Conceptual specifications (BMFS[1]) for thermomechanical analysis of multi-component BGAs

Conceptual specifications for Stage 1 composition	
Entities in analyzable BGA design model	*Entities in Multi-shell-solid analysis body system (as instances of ABBs stated below)*
Analyzable stratum (of **chip substrate**)	***Idealize as*** Planar shell analysis body ABB
Shapes Planar shape …	***Select*** Planar shape ABB
Material Behaviors Linear elastic isotropic temperature-independent material behavior Linear elastic orthotropic temperature-independent material behavior	***If***(stratum function is conductive or soldermask) ***Select*** Linear elastic isotropic temperature independent material behavior ABB ***Else If***(stratum function is dielectric) ***Select*** Linear elastic orthotropic temperature independent material behavior ABB
Analyzable features	Analysis features
Primary surface	***Idealize as*** Planar surface feature ABB
Secondary surface	***Idealize as*** Planar surface feature ABB
Si Chip	***Idealize as*** Generic solid ABB
Shape Cuboid …	***Idealize as*** Cuboid shape ABB
Material behaviors Linear elastic isotropic temperature independent material behavior ABB	***Select*** Linear elastic isotropic temperature independent material behavior ABB
Analyzable features	Analysis features
Bottom surface	***Idealize as*** Analysis feature ABB
Outer surface	***Idealize as*** Analysis feature ABB
Mold	***Idealize as*** Generic solid ABB
Shape 3D shape representation	***Select*** 3D shape representation ABB
Material behaviors	***Select*** Linear elastic isotropic temperature

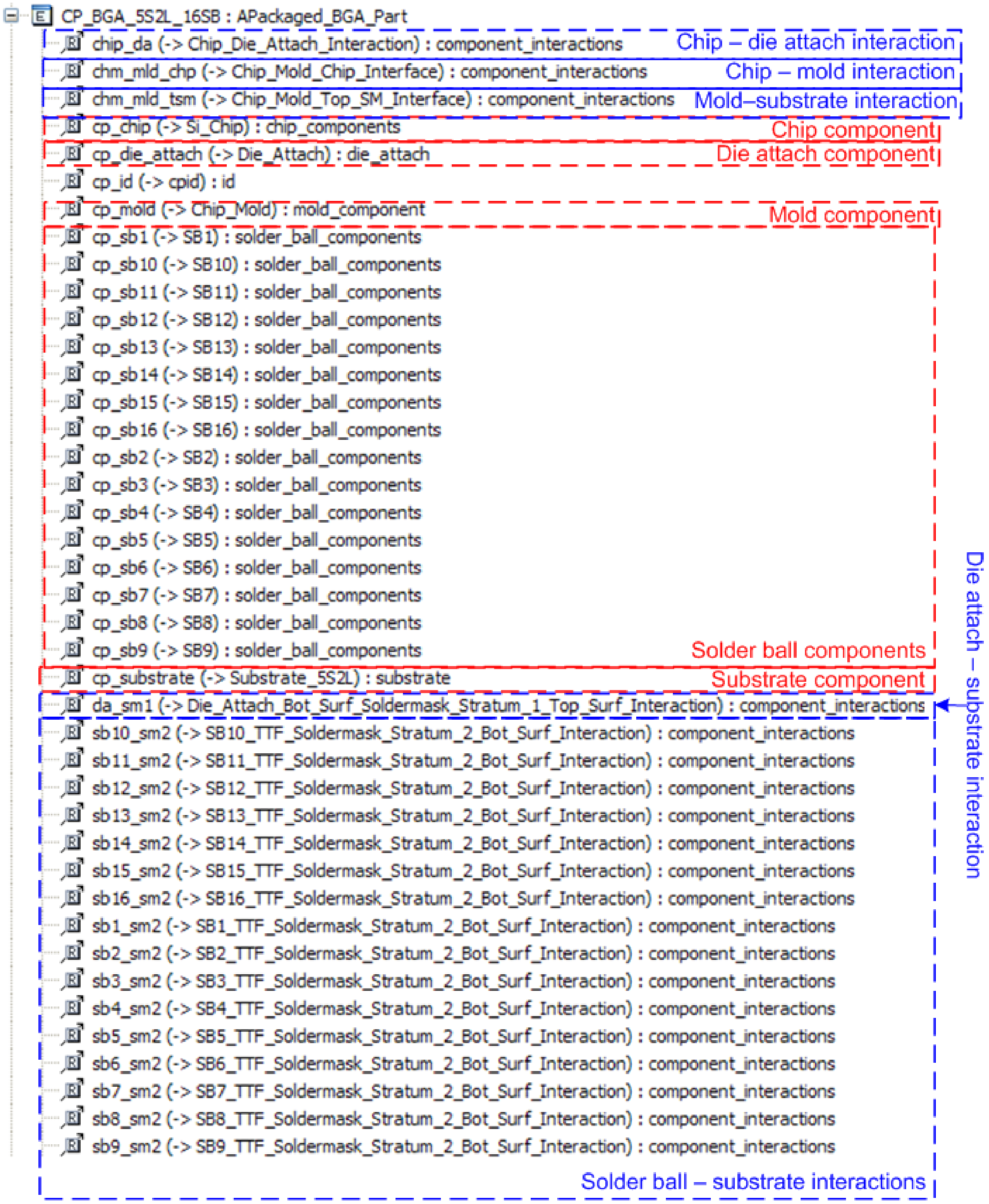
Linear elastic isotropic temperature independent material behavior ABB …	independent material behavior ABB
Analyzable features	Analysis features
Bottom surface	***Idealize as*** Analysis feature ABB
Die Attach	***Idealize as*** Planar shell analysis body ABB
Shapes Planar shape	***Select*** Planar shape ABB
Material behaviors Linear elastic isotropic temperature independent material behavior ABB …	***Select*** Linear elastic isotropic temperature independent material behavior ABB
Analyzable features	Analysis features
Primary surface	***Idealize as*** Planar surface feature ABB
Secondary surface	***Idealize as*** Planar surface feature ABB
Solder Ball	***Idealize as*** Generic solid ABB
Shape Truncated sphere …	***Select*** Truncated sphere shape ABB
Material behaviors Linear elastic isotropic temperature independent material behavior ABB	***Select*** Linear elastic isotropic temperature independent material behavior ABB
Analyzable features	Analysis features
Top truncation feature	***Idealize as*** Analysis feature ABB
Bottom truncation feature	***Idealize as*** Analysis feature ABB
Conceptual specifications for Stage 2 composition	
Analyzable BGA	***Idealize as*** Multi-Shell-Solid Analysis Body System
Chip Substrate	***Idealize as*** Multi-shell analysis body system
Analyzable stratum	***Idealize as*** Planar shell analysis body ABB
Stratum interfaces	***Idealize as*** Shell-shell tie interaction ABB
Mold-Chip interface	***Idealize as*** Solid-solid tie interaction ABB
Mold-Substrate interface	***Idealize as*** Solid-shell tie interaction ABB
Chip-Die Attach interface	***Idealize as*** Solid-shell tie interaction ABB
Die Attach-Substrate interface	***Idealize as*** Shell-shell tie interaction ABB
Substrate-Solder Ball interface	***Idealize as*** Shell-Solid tie interaction ABB
Analyzable features	Analysis features
Volume of analyzable PCB	***Idealize as*** Volume feature ABB
Mid-pt of bottom soldermask stratum of BGA substrate	***Idealize as*** Point feature ABB
Conceptual specifications for Stage 3 composition	
Heating a BGA	***Idealize as*** Uniform temperature load ABB ***associated with*** Volume feature ABB instance corresponding the volume of the analyzable BGA
BGA held fixed at mid-pt of the bottom soldermask stratum of the substrate	***Idealize as*** Point displacement constant behavior condition ABB ***associated with*** Point feature ABB instance corresponding to mid-pt of bottom soldermask of the BGA substrate

Stage 3 composition concerns the idealizations of loads and behavior conditions in which the behavior of the Multi-Shell-Solid analysis body system corresponding to an idealized BGA. A uniform temperature load is used for idealizing the thermal load on a BGA when it is heated (due to the heat generated from the chip or during the assembly process). The behavior condition for this analysis is to hold the mid-point of the bottom analyzable stratum as fixed. This corresponds to locking all degrees of freedom at that point in the analysis body system. This behavior condition is realized by the use of point-displacement-fixed ABB instance that embodies the displacement constraint, and associating it with the point feature ABB instance corresponding to the mid-point of the bottom planar shell analysis body.

Note that the conceptual specifications have been presented here in a tabulated form. The intent here is to describe the types of idealization decisions taken by analysts when developing conceptual specifications. Conceptual specifications are represented more formally using SysML Parametrics constructs as shown in Figure 8.24 of section 8.3.1.

9.3.1.1 Simulation Template $_{16}^{1}$: Simulation template for 16-solder ball analyzable BGA model structure and BMFS^{1}

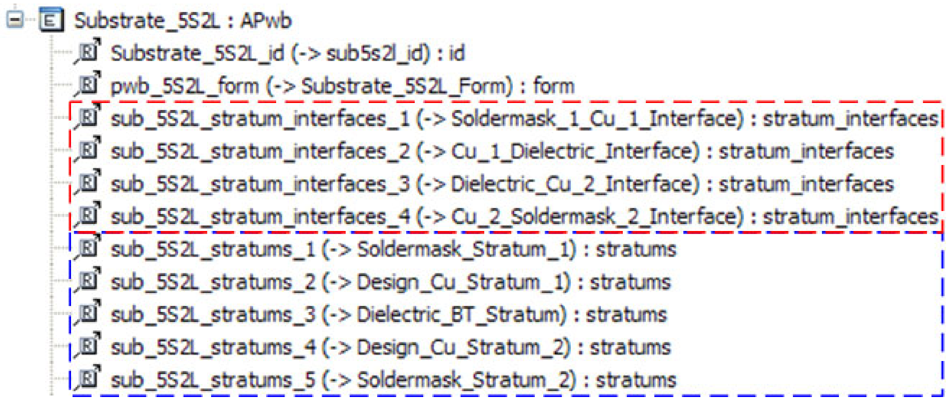
Figure 9.27 illustrates CP_BGA_5S2L_16SB—analyzable design model structure for a 16-solder ball BGA.



```
CP_BGA_5S2L_16SB : APackaged_BGA_Part
    chip_da (-> Chip_Die_Attach_Interaction) : component_interactions
    chm_mld_chp (-> Chip_Mold_Chip_Interface) : component_interactions
    chm_mld_tsm (-> Chip_Mold_Top_SM_Interface) : component_interactions
    cp_chip (-> Si_Chip) : chip_components
    cp_die_attach (-> Die_Attach) : die_attach
    cp_id (-> cpid) : id
    cp_mold (-> Chip_Mold) : mold_component
    cp_sb1 (-> SB1) : solder_ball_components
    cp_sb10 (-> SB10) : solder_ball_components
    cp_sb11 (-> SB11) : solder_ball_components
    cp_sb12 (-> SB12) : solder_ball_components
    cp_sb13 (-> SB13) : solder_ball_components
    cp_sb14 (-> SB14) : solder_ball_components
    cp_sb15 (-> SB15) : solder_ball_components
    cp_sb16 (-> SB16) : solder_ball_components
    cp_sb2 (-> SB2) : solder_ball_components
    cp_sb3 (-> SB3) : solder_ball_components
    cp_sb4 (-> SB4) : solder_ball_components
    cp_sb5 (-> SB5) : solder_ball_components
    cp_sb6 (-> SB6) : solder_ball_components
    cp_sb7 (-> SB7) : solder_ball_components
    cp_sb8 (-> SB8) : solder_ball_components
    cp_sb9 (-> SB9) : solder_ball_components
    cp_substrate (-> Substrate_5S2L) : substrate
    da_sm1 (-> Die_Attach_Bot_Surf_Soldermask_Stratum_1_Top_Surf_Interaction) : component_interactions
    sb10_sm2 (-> SB10_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
    sb11_sm2 (-> SB11_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
    sb12_sm2 (-> SB12_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
    sb13_sm2 (-> SB13_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
    sb14_sm2 (-> SB14_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
    sb15_sm2 (-> SB15_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
    sb16_sm2 (-> SB16_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
    sb1_sm2 (-> SB1_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
    sb2_sm2 (-> SB2_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
    sb3_sm2 (-> SB3_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
    sb4_sm2 (-> SB4_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
    sb5_sm2 (-> SB5_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
    sb6_sm2 (-> SB6_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
    sb7_sm2 (-> SB7_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
    sb8_sm2 (-> SB8_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
    sb9_sm2 (-> SB9_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
```

Chip – die attach interaction
Chip – mold interaction
Mold–substrate interaction
Chip component
Die attach component
Mold component
Solder ball components
Substrate component
Die attach – substrate interaction
Solder ball – substrate interactions

Figure 9.27: 16-solder body analyzable BGA design model structure (CP_BGA_5S2L_16SB)

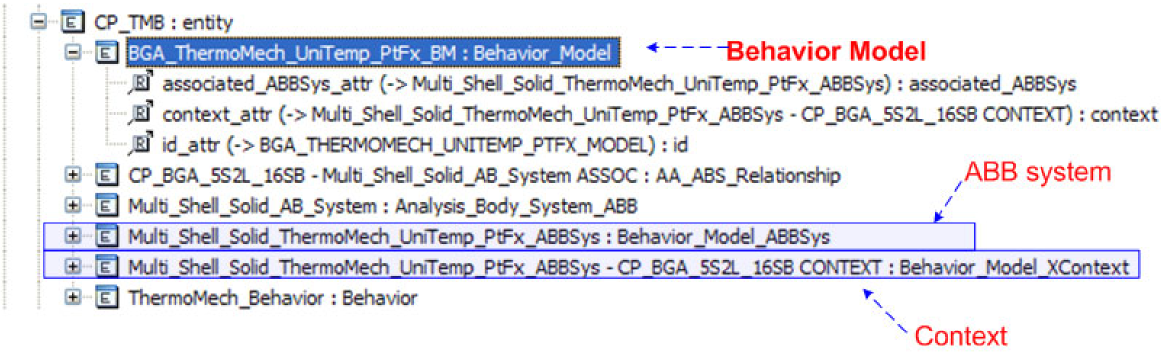


```
Substrate_5S2L : APwb
    Substrate_5S2L_id (-> sub5s2l_id) : id
    pwb_5S2L_form (-> Substrate_5S2L_Form) : form
    sub_5S2L_stratum_interfaces_1 (-> Soldermask_1_Cu_1_Interface) : stratum_interfaces
    sub_5S2L_stratum_interfaces_2 (-> Cu_1_Dielectric_Interface) : stratum_interfaces
    sub_5S2L_stratum_interfaces_3 (-> Dielectric_Cu_2_Interface) : stratum_interfaces
    sub_5S2L_stratum_interfaces_4 (-> Cu_2_Soldermask_2_Interface) : stratum_interfaces
    sub_5S2L_stratums_1 (-> Soldermask_Stratum_1) : stratums
    sub_5S2L_stratums_2 (-> Design_Cu_Stratum_1) : stratums
    sub_5S2L_stratums_3 (-> Dielectric_BT_Stratum) : stratums
    sub_5S2L_stratums_4 (-> Design_Cu_Stratum_2) : stratums
    sub_5S2L_stratums_5 (-> Soldermask_Stratum_2) : stratums
```

Figure 9.28: 5-stratum, 2-layer analyzable chip substrate design model structure (Substrate_5S2L)

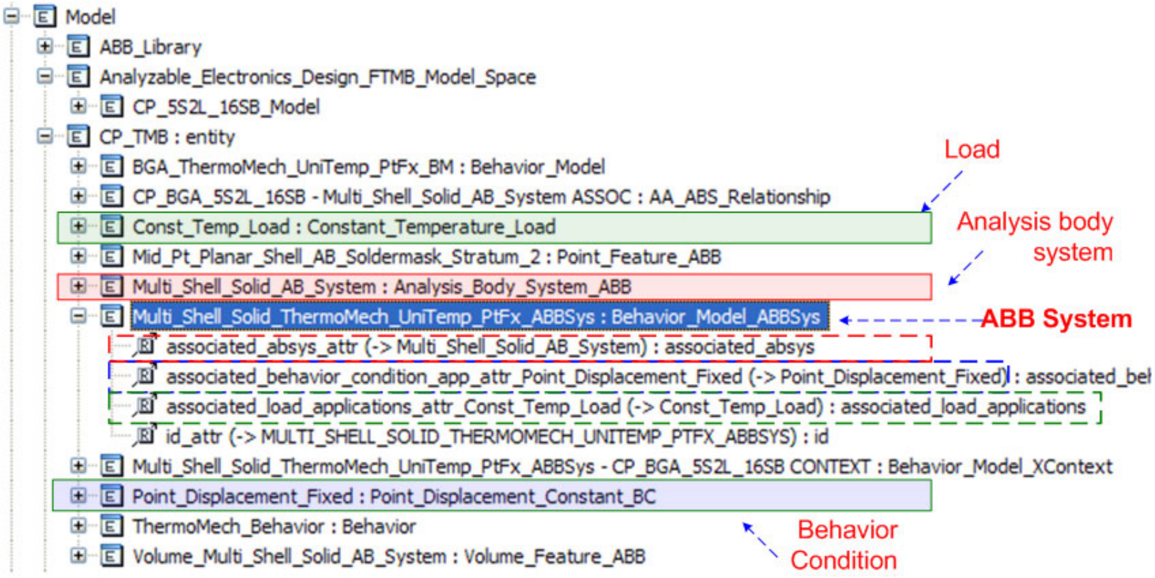
Figure 9.27 shows 16 solder ball components and their interactions with the substrate (one interaction for each solder ball component). The figure also shows the chip, mold, die attach, and substrate components, and their interactions. Figure 9.28 shows Substrate_5S2L—analyzable design model structure for the chip substrate. The 5 stratum components (2 layers) and the 4 interfaces between them are shown in the figure.

After the execution of BMFS[1], a thermo-mechanical behavior model for the 16-solder ball analyzable BGA is created. Figure 9.29 illustrates this behavior model (BGA_ThermoMech_UniTempp_PtFx_BM) and its associated ABB system and context created at the end of Stage 4 composition. Figure 9.30 illustrates the ABB system created at the end of Stage 3 composition. The ABB system consists of an analysis body system (Multi_Shell_Solid_Analysis_Body_System), a constant-temperature load applied to the entire volume of the analysis body system, and a point-displacement-fixed behavior condition applied to the mid-point of the bottom surface of the last planar shell analysis body (corresponding to the last soldermask layer) in the analysis body system corresponding to the substrate.



```
CP_TMB : entity
  BGA_ThermoMech_UniTemp_PtFx_BM : Behavior_Model
    associated_ABBSys_attr (-> Multi_Shell_Solid_ThermoMech_UniTemp_PtFx_ABBSys) : associated_ABBSys
    context_attr (-> Multi_Shell_Solid_ThermoMech_UniTemp_PtFx_ABBSys - CP_BGA_5S2L_16SB CONTEXT) : context
    id_attr (-> BGA_THERMOMECH_UNITEMP_PTFX_MODEL) : id
  CP_BGA_5S2L_16SB - Multi_Shell_Solid_AB_System ASSOC : AA_ABS_Relationship
  Multi_Shell_Solid_AB_System : Analysis_Body_System_ABB
  Multi_Shell_Solid_ThermoMech_UniTemp_PtFx_ABBSys : Behavior_Model_ABBSys
  Multi_Shell_Solid_ThermoMech_UniTemp_PtFx_ABBSys - CP_BGA_5S2L_16SB CONTEXT : Behavior_Model_XContext
  ThermoMech_Behavior : Behavior
```

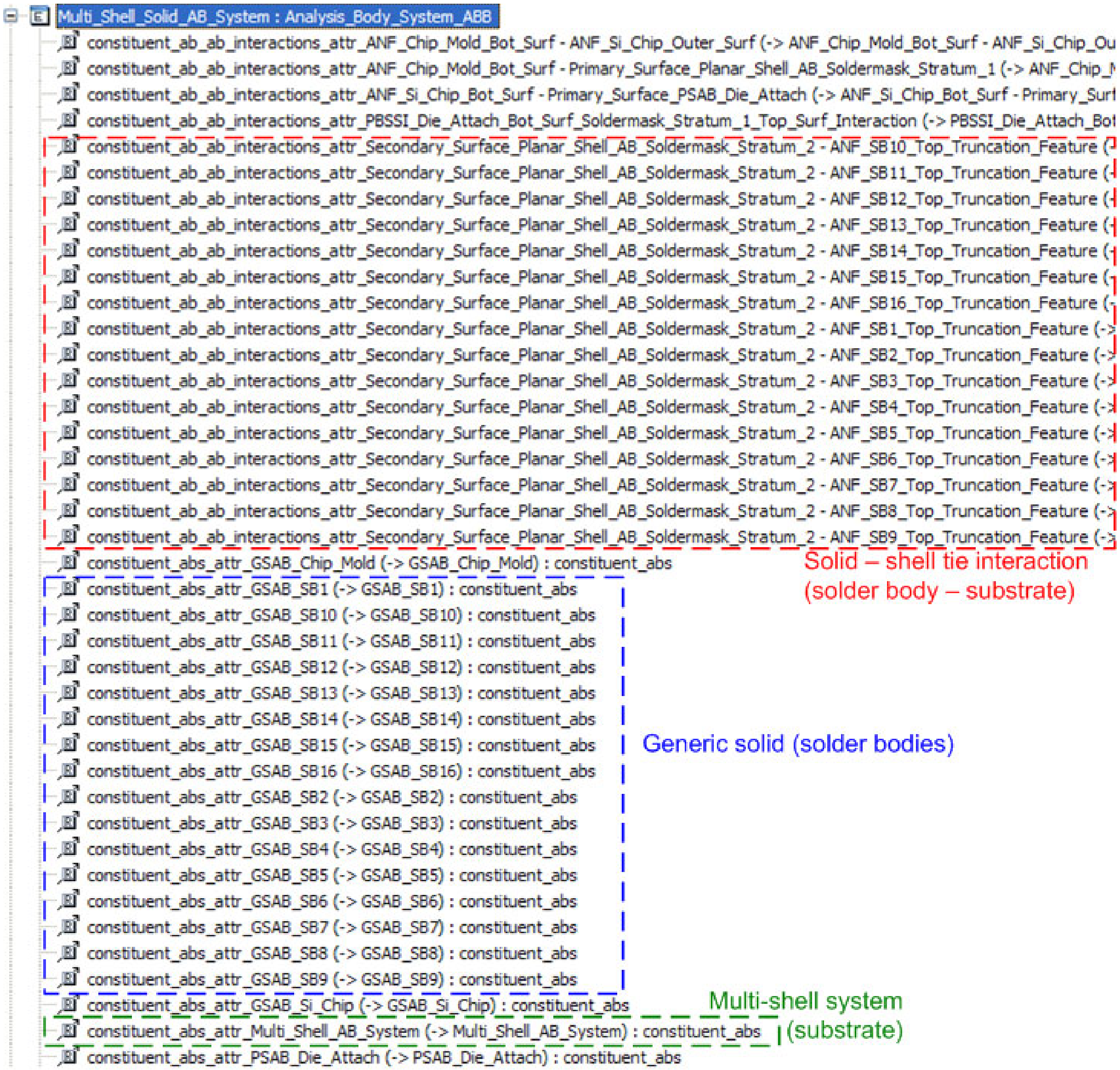
Figure 9.29: Behavior Model structure created for CP_BGA_5S2L_16SB and BMFS[1] in composition Stage 4



```
Model
  ABB_Library
  Analyzable_Electronics_Design_FTMB_Model_Space
    CP_5S2L_16SB_Model
  CP_TMB : entity
    BGA_ThermoMech_UniTemp_PtFx_BM : Behavior_Model
    CP_BGA_5S2L_16SB - Multi_Shell_Solid_AB_System ASSOC : AA_ABS_Relationship
    Const_Temp_Load : Constant_Temperature_Load
    Mid_Pt_Planar_Shell_AB_Soldermask_Stratum_2 : Point_Feature_ABB
    Multi_Shell_Solid_AB_System : Analysis_Body_System_ABB
    Multi_Shell_Solid_ThermoMech_UniTemp_PtFx_ABBSys : Behavior_Model_ABBSys
      associated_absys_attr (-> Multi_Shell_Solid_AB_System) : associated_absys
      associated_behavior_condition_app_attr_Point_Displacement_Fixed (-> Point_Displacement_Fixed) : associated_bel
      associated_load_applications_attr_Const_Temp_Load (-> Const_Temp_Load) : associated_load_applications
      id_attr (-> MULTI_SHELL_SOLID_THERMOMECH_UNITEMP_PTFX_ABBSYS) : id
    Multi_Shell_Solid_ThermoMech_UniTemp_PtFx_ABBSys - CP_BGA_5S2L_16SB CONTEXT : Behavior_Model_XContext
    Point_Displacement_Fixed : Point_Displacement_Constant_BC
    ThermoMech_Behavior : Behavior
    Volume_Multi_Shell_Solid_AB_System : Volume_Feature_ABB
```

Figure 9.30: ABB system created for CP_BGA_5S2L_16SB and BMFS[1] in composition Stage 3

Figure 9.31 illustrates the Multi-Shell-Solid analysis body system—corresponding to the analyzable BGA—created in Stage 2 of the composition process. The figure shows the 16 generic solid analysis body components corresponding to the solder balls in the analyzable BGA design assembly, and the 16 solid-shell tie interactions between these analysis bodies and the last planar shell analysis body—corresponding to the bottom soldermask layer—of the substrate analysis body system. The figure also shows the multi-shell analysis body system—corresponding to the chip substrate created in Stage 2 composition.



```
Multi_Shell_Solid_AB_System : Analysis_Body_System_ABB
  constituent_ab_ab_interactions_attr_ANF_Chip_Mold_Bot_Surf - ANF_Si_Chip_Outer_Surf (-> ANF_Chip_Mold_Bot_Surf - ANF_Si_Chip_Ou
  constituent_ab_ab_interactions_attr_ANF_Chip_Mold_Bot_Surf - Primary_Surface_Planar_Shell_AB_Soldermask_Stratum_1 (-> ANF_Chip_M
  constituent_ab_ab_interactions_attr_ANF_Si_Chip_Bot_Surf - Primary_Surface_PSAB_Die_Attach (-> ANF_Si_Chip_Bot_Surf - Primary_Surf
  constituent_ab_ab_interactions_attr_PBSSI_Die_Attach_Bot_Surf_Soldermask_Stratum_1_Top_Surf_Interaction (-> PBSSI_Die_Attach_Bot
  constituent_ab_ab_interactions_attr_Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB10_Top_Truncation_Feature (
  constituent_ab_ab_interactions_attr_Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB11_Top_Truncation_Feature (
  constituent_ab_ab_interactions_attr_Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB12_Top_Truncation_Feature (
  constituent_ab_ab_interactions_attr_Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB13_Top_Truncation_Feature (
  constituent_ab_ab_interactions_attr_Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB14_Top_Truncation_Feature (
  constituent_ab_ab_interactions_attr_Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB15_Top_Truncation_Feature (
  constituent_ab_ab_interactions_attr_Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB16_Top_Truncation_Feature (
  constituent_ab_ab_interactions_attr_Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB1_Top_Truncation_Feature (->
  constituent_ab_ab_interactions_attr_Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB2_Top_Truncation_Feature (->
  constituent_ab_ab_interactions_attr_Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB3_Top_Truncation_Feature (->
  constituent_ab_ab_interactions_attr_Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB4_Top_Truncation_Feature (->
  constituent_ab_ab_interactions_attr_Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB5_Top_Truncation_Feature (->
  constituent_ab_ab_interactions_attr_Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB6_Top_Truncation_Feature (->
  constituent_ab_ab_interactions_attr_Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB7_Top_Truncation_Feature (->
  constituent_ab_ab_interactions_attr_Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB8_Top_Truncation_Feature (->
  constituent_ab_ab_interactions_attr_Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB9_Top_Truncation_Feature (->
  constituent_abs_attr_GSAB_Chip_Mold (-> GSAB_Chip_Mold) : constituent_abs
  constituent_abs_attr_GSAB_SB1 (-> GSAB_SB1) : constituent_abs
  constituent_abs_attr_GSAB_SB10 (-> GSAB_SB10) : constituent_abs
  constituent_abs_attr_GSAB_SB11 (-> GSAB_SB11) : constituent_abs
  constituent_abs_attr_GSAB_SB12 (-> GSAB_SB12) : constituent_abs
  constituent_abs_attr_GSAB_SB13 (-> GSAB_SB13) : constituent_abs
  constituent_abs_attr_GSAB_SB14 (-> GSAB_SB14) : constituent_abs
  constituent_abs_attr_GSAB_SB15 (-> GSAB_SB15) : constituent_abs
  constituent_abs_attr_GSAB_SB16 (-> GSAB_SB16) : constituent_abs
  constituent_abs_attr_GSAB_SB2 (-> GSAB_SB2) : constituent_abs
  constituent_abs_attr_GSAB_SB3 (-> GSAB_SB3) : constituent_abs
  constituent_abs_attr_GSAB_SB4 (-> GSAB_SB4) : constituent_abs
  constituent_abs_attr_GSAB_SB5 (-> GSAB_SB5) : constituent_abs
  constituent_abs_attr_GSAB_SB6 (-> GSAB_SB6) : constituent_abs
  constituent_abs_attr_GSAB_SB7 (-> GSAB_SB7) : constituent_abs
  constituent_abs_attr_GSAB_SB8 (-> GSAB_SB8) : constituent_abs
  constituent_abs_attr_GSAB_SB9 (-> GSAB_SB9) : constituent_abs
  constituent_abs_attr_GSAB_Si_Chip (-> GSAB_Si_Chip) : constituent_abs
  constituent_abs_attr_Multi_Shell_AB_System (-> Multi_Shell_AB_System) : constituent_abs
  constituent_abs_attr_PSAB_Die_Attach (-> PSAB_Die_Attach) : constituent_abs
```

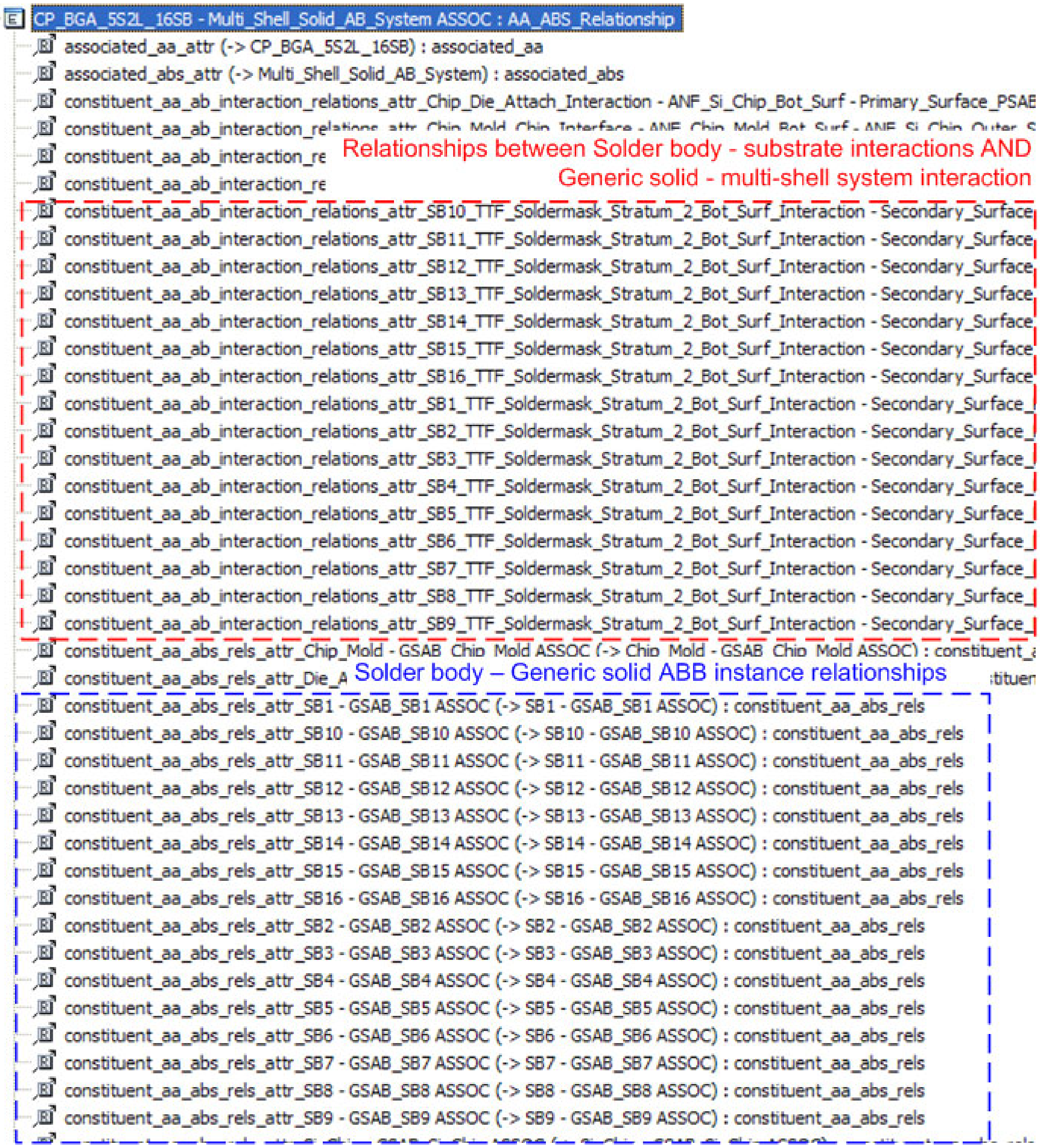
Solid – shell tie interaction
(solder body – substrate)

Generic solid (solder bodies)

Multi-shell system
(substrate)

Figure 9.31: Analysis body system created for CP_BGA_5S2L_16SB and BMFS[1] in composition Stage 2

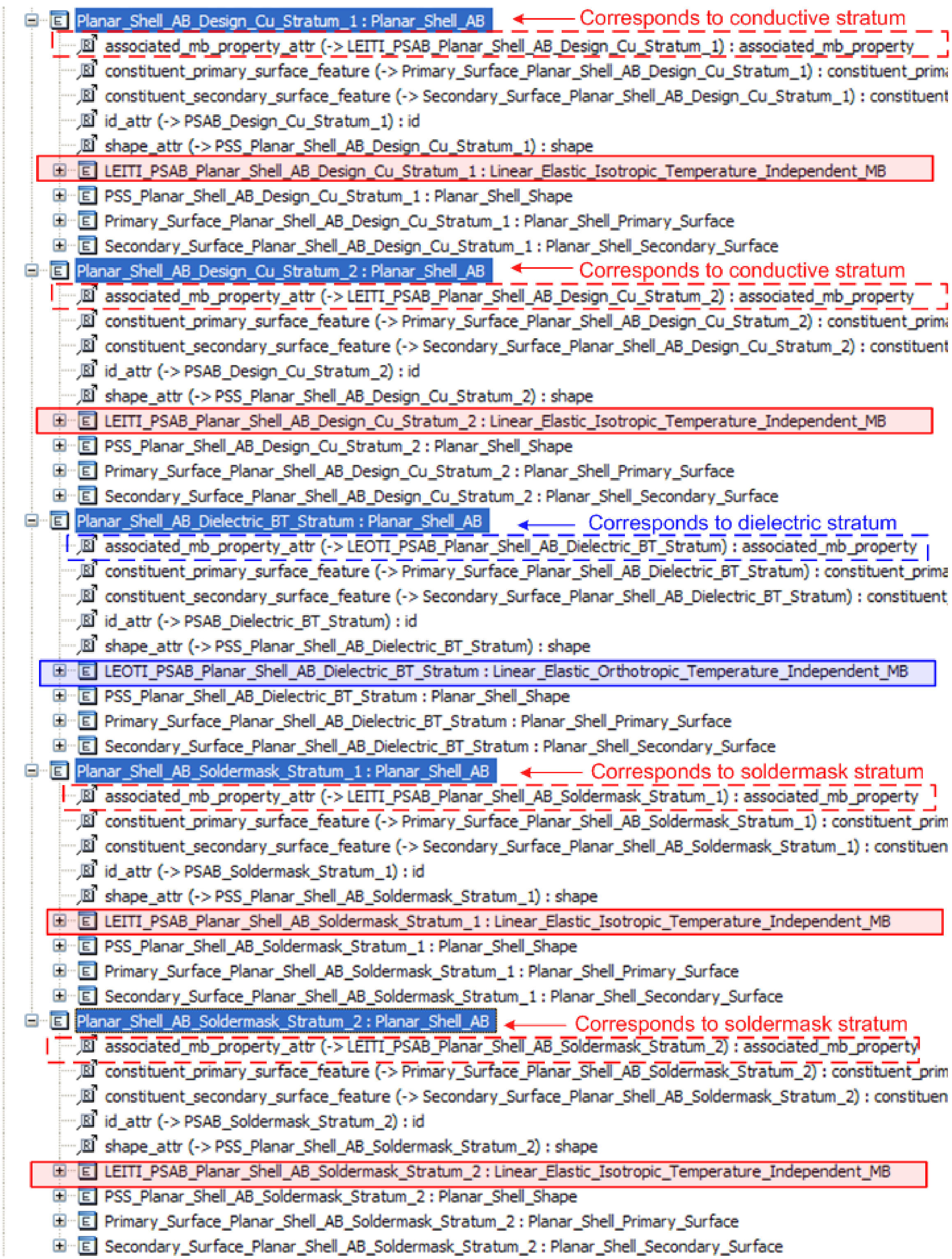
Figure 9.32 illustrates 16 association relationships created between generic solid analysis bodies (corresponding to solder balls) in the multi-shell-solid analysis body system and the solder balls in the analyzable BGA design assembly in composition Stage 2. The figure also shows the 16 analyzable artifact-analysis body interaction relationships between (a) interface between these generic solid bodies and the bottom surface of the last planar shell analysis body in the multi-shell analysis body system (corresponding to the chip substrate) and (b) interface between the solder balls and the bottom surface of the last stratum (soldermask) of the analyzable chip substrate.


Relationships between Solder body - substrate interactions AND
Generic solid - multi-shell system interaction
Solder body – Generic solid ABB instance relationships

```
CP_BGA_5S2L_16SB - Multi_Shell_Solid_AB_System ASSOC : AA_ABS_Relationship
    associated_aa_attr (-> CP_BGA_5S2L_16SB) : associated_aa
    associated_abs_attr (-> Multi_Shell_Solid_AB_System) : associated_abs
    constituent_aa_ab_interaction_relations_attr_Chip_Die_Attach_Interaction - ANF_Si_Chip_Bot_Surf - Primary_Surface_PSAE
    constituent_aa_ab_interaction_relations_attr_Chip_Mold_Chip_Interface - ANF_Chip_Mold_Bot_Surf - ANF_Si_Chip_Outer_S
    constituent_aa_ab_interaction_re
    constituent_aa_ab_interaction_re
    constituent_aa_ab_interaction_relations_attr_SB10_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction - Secondary_Surface
    constituent_aa_ab_interaction_relations_attr_SB11_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction - Secondary_Surface
    constituent_aa_ab_interaction_relations_attr_SB12_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction - Secondary_Surface
    constituent_aa_ab_interaction_relations_attr_SB13_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction - Secondary_Surface
    constituent_aa_ab_interaction_relations_attr_SB14_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction - Secondary_Surface
    constituent_aa_ab_interaction_relations_attr_SB15_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction - Secondary_Surface
    constituent_aa_ab_interaction_relations_attr_SB16_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction - Secondary_Surface
    constituent_aa_ab_interaction_relations_attr_SB1_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction - Secondary_Surface_
    constituent_aa_ab_interaction_relations_attr_SB2_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction - Secondary_Surface_
    constituent_aa_ab_interaction_relations_attr_SB3_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction - Secondary_Surface_
    constituent_aa_ab_interaction_relations_attr_SB4_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction - Secondary_Surface_
    constituent_aa_ab_interaction_relations_attr_SB5_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction - Secondary_Surface_
    constituent_aa_ab_interaction_relations_attr_SB6_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction - Secondary_Surface_
    constituent_aa_ab_interaction_relations_attr_SB7_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction - Secondary_Surface_
    constituent_aa_ab_interaction_relations_attr_SB8_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction - Secondary_Surface_
    constituent_aa_ab_interaction_relations_attr_SB9_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction - Secondary_Surface_
    constituent_aa_abs_rels_attr_Chip_Mold - GSAB_Chip_Mold ASSOC (-> Chip_Mold - GSAB_Chip_Mold ASSOC) : constituent_a
    constituent_aa_abs_rels_attr_Die_A                                                                 stituen
    constituent_aa_abs_rels_attr_SB1 - GSAB_SB1 ASSOC (-> SB1 - GSAB_SB1 ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_SB10 - GSAB_SB10 ASSOC (-> SB10 - GSAB_SB10 ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_SB11 - GSAB_SB11 ASSOC (-> SB11 - GSAB_SB11 ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_SB12 - GSAB_SB12 ASSOC (-> SB12 - GSAB_SB12 ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_SB13 - GSAB_SB13 ASSOC (-> SB13 - GSAB_SB13 ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_SB14 - GSAB_SB14 ASSOC (-> SB14 - GSAB_SB14 ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_SB15 - GSAB_SB15 ASSOC (-> SB15 - GSAB_SB15 ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_SB16 - GSAB_SB16 ASSOC (-> SB16 - GSAB_SB16 ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_SB2 - GSAB_SB2 ASSOC (-> SB2 - GSAB_SB2 ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_SB3 - GSAB_SB3 ASSOC (-> SB3 - GSAB_SB3 ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_SB4 - GSAB_SB4 ASSOC (-> SB4 - GSAB_SB4 ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_SB5 - GSAB_SB5 ASSOC (-> SB5 - GSAB_SB5 ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_SB6 - GSAB_SB6 ASSOC (-> SB6 - GSAB_SB6 ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_SB7 - GSAB_SB7 ASSOC (-> SB7 - GSAB_SB7 ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_SB8 - GSAB_SB8 ASSOC (-> SB8 - GSAB_SB8 ASSOC) : constituent_aa_abs_rels
    constituent_aa_abs_rels_attr_SB9 - GSAB_SB9 ASSOC (-> SB9 - GSAB_SB9 ASSOC) : constituent_aa_abs_rels
```

Figure 9.32: Relationship between analyzable BGA design model structure (CP_BGA_5S2L_16SB) and Multi-Shell-Solid analysis body system for BMFS^1 in composition Stage 2

Figure 9.33 illustrates the material behavior of five planar shell analysis bodies in the multi-shell analysis body system corresponding to the chip substrate. Per the idealization decisions in BMFS^1, the planar shell analysis bodies corresponding to conductive and soldermask stratums have an isotropic material behavior as opposed to orthotropic material behavior for the body corresponding to the dielectric stratum.



```
Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_AB
    associated_mb_property_attr (-> LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_1) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1) : constituent_prima
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1) : constituent
    id_attr (-> PSAB_Design_Cu_Stratum_1) : id
    shape_attr (-> PSS_Planar_Shell_AB_Design_Cu_Stratum_1) : shape
    LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_1 : Linear_Elastic_Isotropic_Temperature_Independent_MB
    PSS_Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_Shape
    Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_Primary_Surface
    Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_Secondary_Surface
Planar_Shell_AB_Design_Cu_Stratum_2 : Planar_Shell_AB
    associated_mb_property_attr (-> LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_2) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_2) : constituent_prima
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_2) : constituent
    id_attr (-> PSAB_Design_Cu_Stratum_2) : id
    shape_attr (-> PSS_Planar_Shell_AB_Design_Cu_Stratum_2) : shape
    LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_2 : Linear_Elastic_Isotropic_Temperature_Independent_MB
    PSS_Planar_Shell_AB_Design_Cu_Stratum_2 : Planar_Shell_Shape
    Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_2 : Planar_Shell_Primary_Surface
    Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_2 : Planar_Shell_Secondary_Surface
Planar_Shell_AB_Dielectric_BT_Stratum : Planar_Shell_AB
    associated_mb_property_attr (-> LEOTI_PSAB_Planar_Shell_AB_Dielectric_BT_Stratum) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Dielectric_BT_Stratum) : constituent_prima
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Dielectric_BT_Stratum) : constituent
    id_attr (-> PSAB_Dielectric_BT_Stratum) : id
    shape_attr (-> PSS_Planar_Shell_AB_Dielectric_BT_Stratum) : shape
    LEOTI_PSAB_Planar_Shell_AB_Dielectric_BT_Stratum : Linear_Elastic_Orthotropic_Temperature_Independent_MB
    PSS_Planar_Shell_AB_Dielectric_BT_Stratum : Planar_Shell_Shape
    Primary_Surface_Planar_Shell_AB_Dielectric_BT_Stratum : Planar_Shell_Primary_Surface
    Secondary_Surface_Planar_Shell_AB_Dielectric_BT_Stratum : Planar_Shell_Secondary_Surface
Planar_Shell_AB_Soldermask_Stratum_1 : Planar_Shell_AB
    associated_mb_property_attr (-> LEITI_PSAB_Planar_Shell_AB_Soldermask_Stratum_1) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Soldermask_Stratum_1) : constituent_prim
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_1) : constituen
    id_attr (-> PSAB_Soldermask_Stratum_1) : id
    shape_attr (-> PSS_Planar_Shell_AB_Soldermask_Stratum_1) : shape
    LEITI_PSAB_Planar_Shell_AB_Soldermask_Stratum_1 : Linear_Elastic_Isotropic_Temperature_Independent_MB
    PSS_Planar_Shell_AB_Soldermask_Stratum_1 : Planar_Shell_Shape
    Primary_Surface_Planar_Shell_AB_Soldermask_Stratum_1 : Planar_Shell_Primary_Surface
    Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_1 : Planar_Shell_Secondary_Surface
Planar_Shell_AB_Soldermask_Stratum_2 : Planar_Shell_AB
    associated_mb_property_attr (-> LEITI_PSAB_Planar_Shell_AB_Soldermask_Stratum_2) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Soldermask_Stratum_2) : constituent_prim
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2) : constituen
    id_attr (-> PSAB_Soldermask_Stratum_2) : id
    shape_attr (-> PSS_Planar_Shell_AB_Soldermask_Stratum_2) : shape
    LEITI_PSAB_Planar_Shell_AB_Soldermask_Stratum_2 : Linear_Elastic_Isotropic_Temperature_Independent_MB
    PSS_Planar_Shell_AB_Soldermask_Stratum_2 : Planar_Shell_Shape
    Primary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 : Planar_Shell_Primary_Surface
    Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 : Planar_Shell_Secondary_Surface
```

Corresponds to conductive stratum

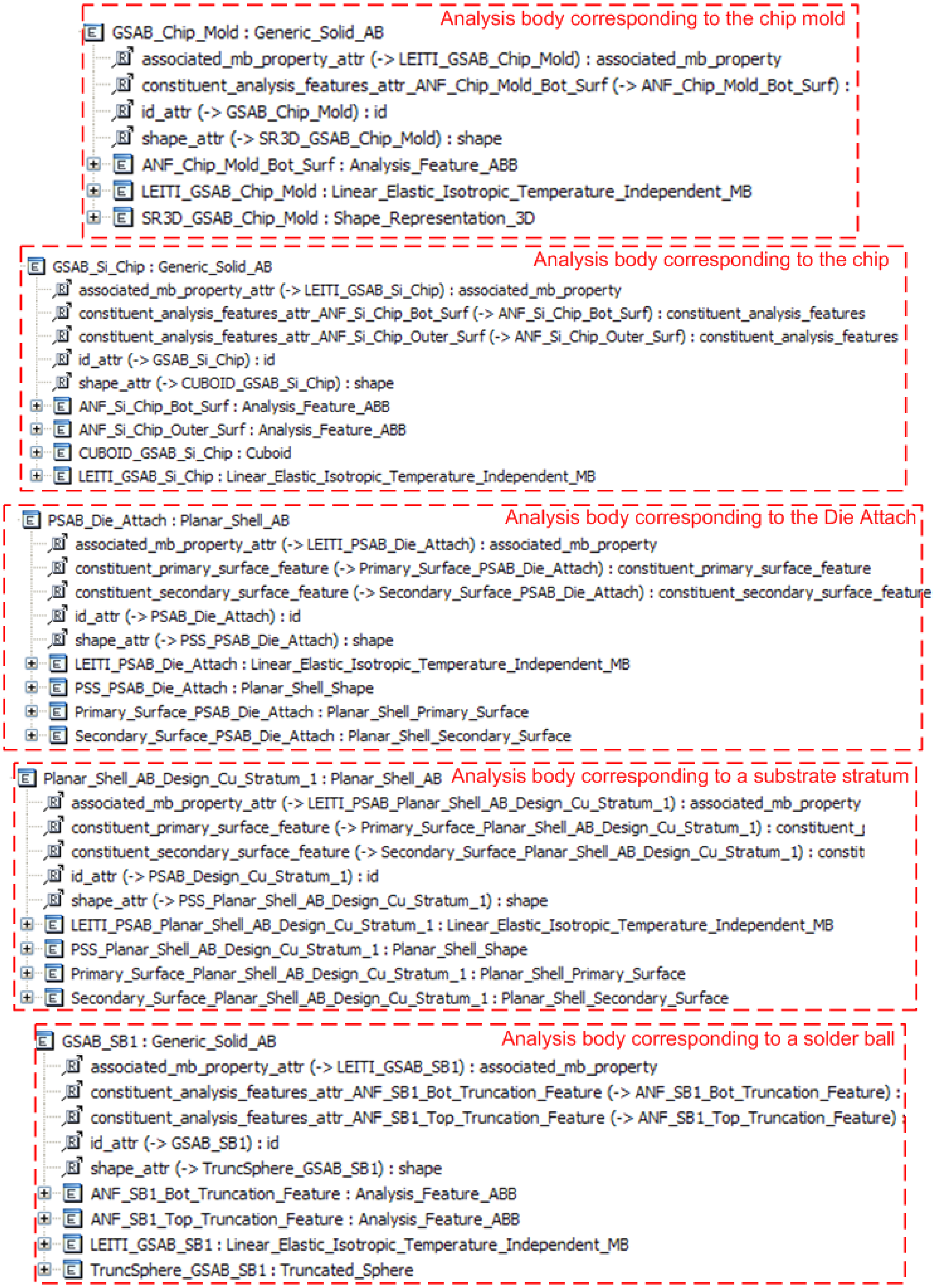
Corresponds to conductive stratum

Corresponds to dielectric stratum

Corresponds to soldermask stratum

Corresponds to soldermask stratum

Figure 9.33: Material behavior of analysis bodies corresponding to substrate stratums



Analysis body corresponding to the chip mold

```
GSAB_Chip_Mold : Generic_Solid_AB
    associated_mb_property_attr (-> LEITI_GSAB_Chip_Mold) : associated_mb_property
    constituent_analysis_features_attr_ANF_Chip_Mold_Bot_Surf (-> ANF_Chip_Mold_Bot_Surf) :
    id_attr (-> GSAB_Chip_Mold) : id
    shape_attr (-> SR3D_GSAB_Chip_Mold) : shape
  + ANF_Chip_Mold_Bot_Surf : Analysis_Feature_ABB
  + LEITI_GSAB_Chip_Mold : Linear_Elastic_Isotropic_Temperature_Independent_MB
  + SR3D_GSAB_Chip_Mold : Shape_Representation_3D
```

Analysis body corresponding to the chip

```
GSAB_Si_Chip : Generic_Solid_AB
    associated_mb_property_attr (-> LEITI_GSAB_Si_Chip) : associated_mb_property
    constituent_analysis_features_attr_ANF_Si_Chip_Bot_Surf (-> ANF_Si_Chip_Bot_Surf) : constituent_analysis_features
    constituent_analysis_features_attr_ANF_Si_Chip_Outer_Surf (-> ANF_Si_Chip_Outer_Surf) : constituent_analysis_features
    id_attr (-> GSAB_Si_Chip) : id
    shape_attr (-> CUBOID_GSAB_Si_Chip) : shape
  + ANF_Si_Chip_Bot_Surf : Analysis_Feature_ABB
  + ANF_Si_Chip_Outer_Surf : Analysis_Feature_ABB
  + CUBOID_GSAB_Si_Chip : Cuboid
  + LEITI_GSAB_Si_Chip : Linear_Elastic_Isotropic_Temperature_Independent_MB
```

Analysis body corresponding to the Die Attach

```
PSAB_Die_Attach : Planar_Shell_AB
    associated_mb_property_attr (-> LEITI_PSAB_Die_Attach) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_PSAB_Die_Attach) : constituent_primary_surface_feature
    constituent_secondary_surface_feature (-> Secondary_Surface_PSAB_Die_Attach) : constituent_secondary_surface_feature
    id_attr (-> PSAB_Die_Attach) : id
    shape_attr (-> PSS_PSAB_Die_Attach) : shape
  + LEITI_PSAB_Die_Attach : Linear_Elastic_Isotropic_Temperature_Independent_MB
  + PSS_PSAB_Die_Attach : Planar_Shell_Shape
  + Primary_Surface_PSAB_Die_Attach : Planar_Shell_Primary_Surface
  + Secondary_Surface_PSAB_Die_Attach : Planar_Shell_Secondary_Surface
```

Analysis body corresponding to a substrate stratum

```
Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_AB
    associated_mb_property_attr (-> LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_1) : associated_mb_property
    constituent_primary_surface_feature (-> Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1) : constituent_
    constituent_secondary_surface_feature (-> Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1) : constit
    id_attr (-> PSAB_Design_Cu_Stratum_1) : id
    shape_attr (-> PSS_Planar_Shell_AB_Design_Cu_Stratum_1) : shape
  + LEITI_PSAB_Planar_Shell_AB_Design_Cu_Stratum_1 : Linear_Elastic_Isotropic_Temperature_Independent_MB
  + PSS_Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_Shape
  + Primary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_Primary_Surface
  + Secondary_Surface_Planar_Shell_AB_Design_Cu_Stratum_1 : Planar_Shell_Secondary_Surface
```

Analysis body corresponding to a solder ball

```
GSAB_SB1 : Generic_Solid_AB
    associated_mb_property_attr (-> LEITI_GSAB_SB1) : associated_mb_property
    constituent_analysis_features_attr_ANF_SB1_Bot_Truncation_Feature (-> ANF_SB1_Bot_Truncation_Feature) :
    constituent_analysis_features_attr_ANF_SB1_Top_Truncation_Feature (-> ANF_SB1_Top_Truncation_Feature) :
    id_attr (-> GSAB_SB1) : id
    shape_attr (-> TruncSphere_GSAB_SB1) : shape
  + ANF_SB1_Bot_Truncation_Feature : Analysis_Feature_ABB
  + ANF_SB1_Top_Truncation_Feature : Analysis_Feature_ABB
  + LEITI_GSAB_SB1 : Linear_Elastic_Isotropic_Temperature_Independent_MB
  + TruncSphere_GSAB_SB1 : Truncated_Sphere
```

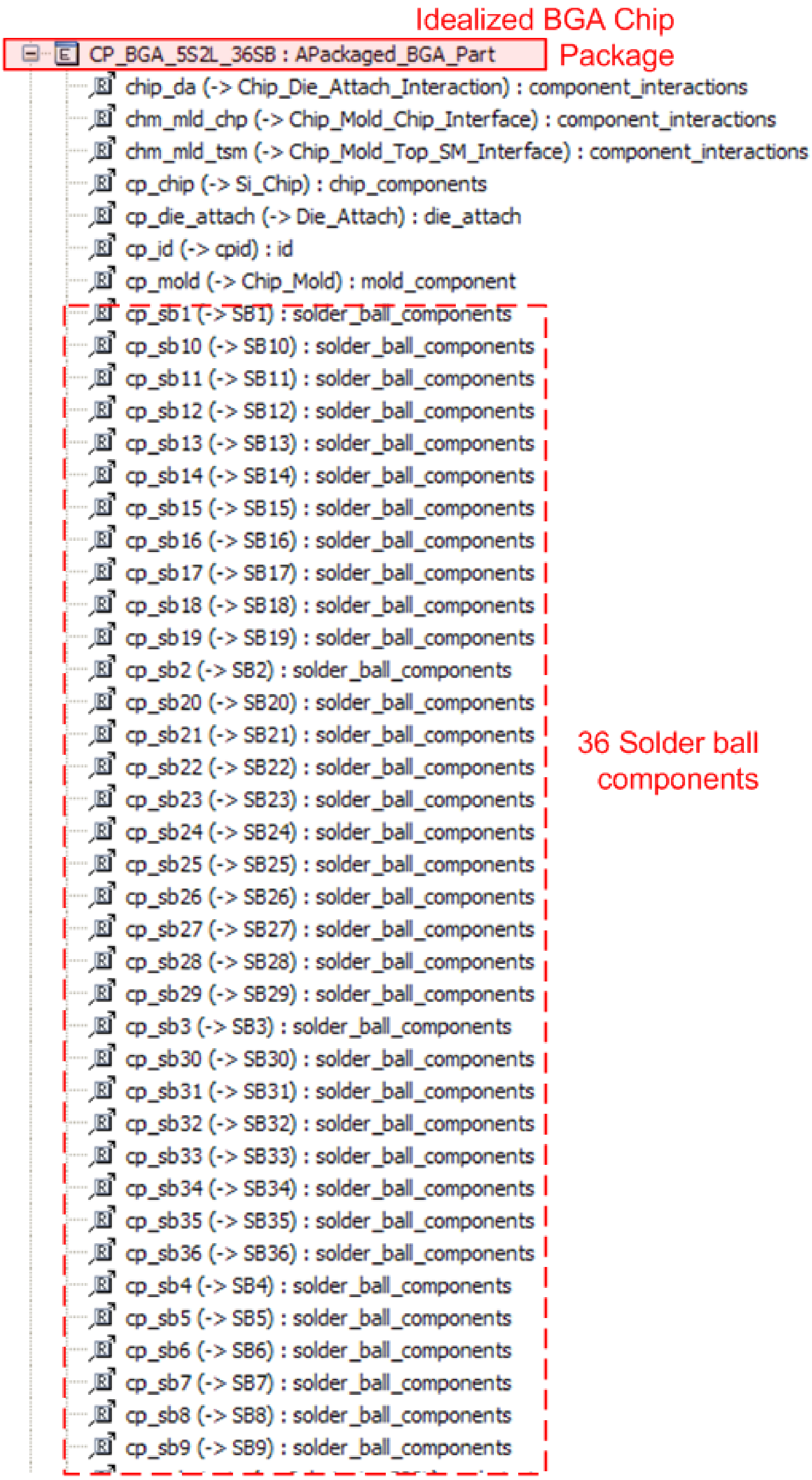
Figure 9.34: Types of analysis bodies created for CP_BGA_5S2L_16SB and BMFS[1] *in composition Stage 1*

Figure 9.34 illustrates the different types of analysis bodies, corresponding to components in the analyzable BGA assembly, created in composition Stage 1. The number of each type of analysis body is shown in Figure 9.31. For example, 16 generic solid analysis bodies (corresponding to 16 solder balls) were created in Stage 1 of the composition process. The intent of Figure 9.34 is to illustrate the different types of entities created for each analysis body. The figure shows shape, material behavior, and analysis features created for analysis bodies corresponding to mold, chip, die attach, substrate stratum, and solder balls. In addition, associations between shape, material behavior, and analysis features of each analysis body and the corresponding component in the analyzable BGA assembly are also created during Stage 1 composition.

9.3.1.2 Simulation Template $_{36}^{1}$: Simulation template for 36-solder ball analyzable BGA model structure and BMFS^{1}

In this section, the simulation template automatically created for an analyzable BGA design model structure with 36 solder balls and for idealization decisions embodied in BMFS^{1} is presented. The analyzable BGA assembly with 36 solder balls has a non-equivalent assembly system topology as compared to the analyzable BGA assembly with 16 solder balls. For the same BMFS, change in assembly system topology of the design alternative affects results of composition Stages 1 and 2 only. If the topology variation is only due to changes in the number of artifacts in the design alternative assembly, the number of analysis bodies created during Stage 1 composition changes. In addition, the analysis body system composed in Stage 2 has a different number of analysis body components and their interactions.

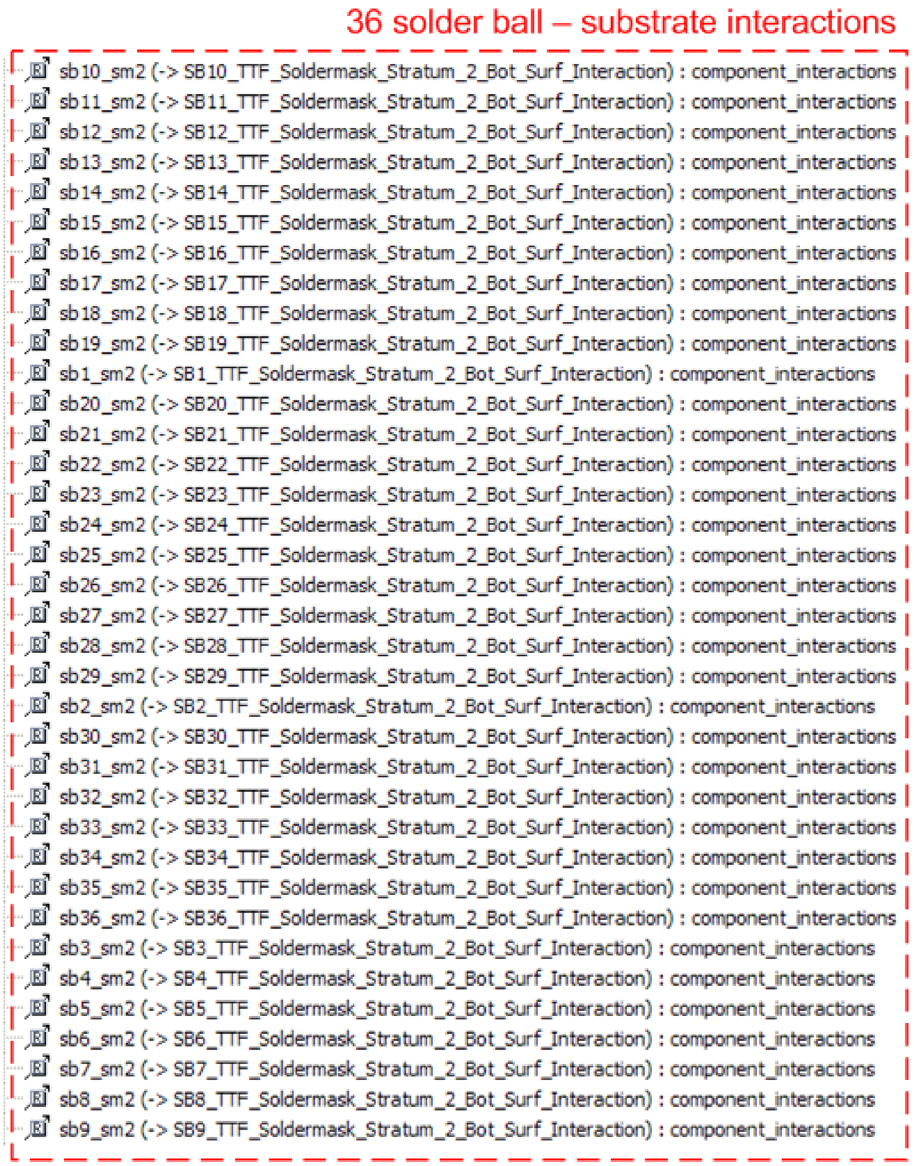
Figure 9.35 illustrates CP_BGA_5S2L_36SB—an analyzable BGA design model structure with 36 solder balls. The figure shows the 36 solder ball components and the 36 interactions between the solder balls and the bottom stratum of the chip substrate—one interaction per solder ball. Note that in this example BGA assembly, the number of solder balls (and associated interactions) is the only change compared to the 16 solder ball BGA example illustrated in the previous section.



Idealized BGA Chip Package

```
CP_BGA_5S2L_36SB : APackaged_BGA_Part
    chip_da (-> Chip_Die_Attach_Interaction) : component_interactions
    chm_mld_chp (-> Chip_Mold_Chip_Interface) : component_interactions
    chm_mld_tsm (-> Chip_Mold_Top_SM_Interface) : component_interactions
    cp_chip (-> Si_Chip) : chip_components
    cp_die_attach (-> Die_Attach) : die_attach
    cp_id (-> cpid) : id
    cp_mold (-> Chip_Mold) : mold_component
    cp_sb1 (-> SB1) : solder_ball_components
    cp_sb10 (-> SB10) : solder_ball_components
    cp_sb11 (-> SB11) : solder_ball_components
    cp_sb12 (-> SB12) : solder_ball_components
    cp_sb13 (-> SB13) : solder_ball_components
    cp_sb14 (-> SB14) : solder_ball_components
    cp_sb15 (-> SB15) : solder_ball_components
    cp_sb16 (-> SB16) : solder_ball_components
    cp_sb17 (-> SB17) : solder_ball_components
    cp_sb18 (-> SB18) : solder_ball_components
    cp_sb19 (-> SB19) : solder_ball_components
    cp_sb2 (-> SB2) : solder_ball_components
    cp_sb20 (-> SB20) : solder_ball_components
    cp_sb21 (-> SB21) : solder_ball_components
    cp_sb22 (-> SB22) : solder_ball_components
    cp_sb23 (-> SB23) : solder_ball_components
    cp_sb24 (-> SB24) : solder_ball_components
    cp_sb25 (-> SB25) : solder_ball_components
    cp_sb26 (-> SB26) : solder_ball_components
    cp_sb27 (-> SB27) : solder_ball_components
    cp_sb28 (-> SB28) : solder_ball_components
    cp_sb29 (-> SB29) : solder_ball_components
    cp_sb3 (-> SB3) : solder_ball_components
    cp_sb30 (-> SB30) : solder_ball_components
    cp_sb31 (-> SB31) : solder_ball_components
    cp_sb32 (-> SB32) : solder_ball_components
    cp_sb33 (-> SB33) : solder_ball_components
    cp_sb34 (-> SB34) : solder_ball_components
    cp_sb35 (-> SB35) : solder_ball_components
    cp_sb36 (-> SB36) : solder_ball_components
    cp_sb4 (-> SB4) : solder_ball_components
    cp_sb5 (-> SB5) : solder_ball_components
    cp_sb6 (-> SB6) : solder_ball_components
    cp_sb7 (-> SB7) : solder_ball_components
    cp_sb8 (-> SB8) : solder_ball_components
    cp_sb9 (-> SB9) : solder_ball_components
```

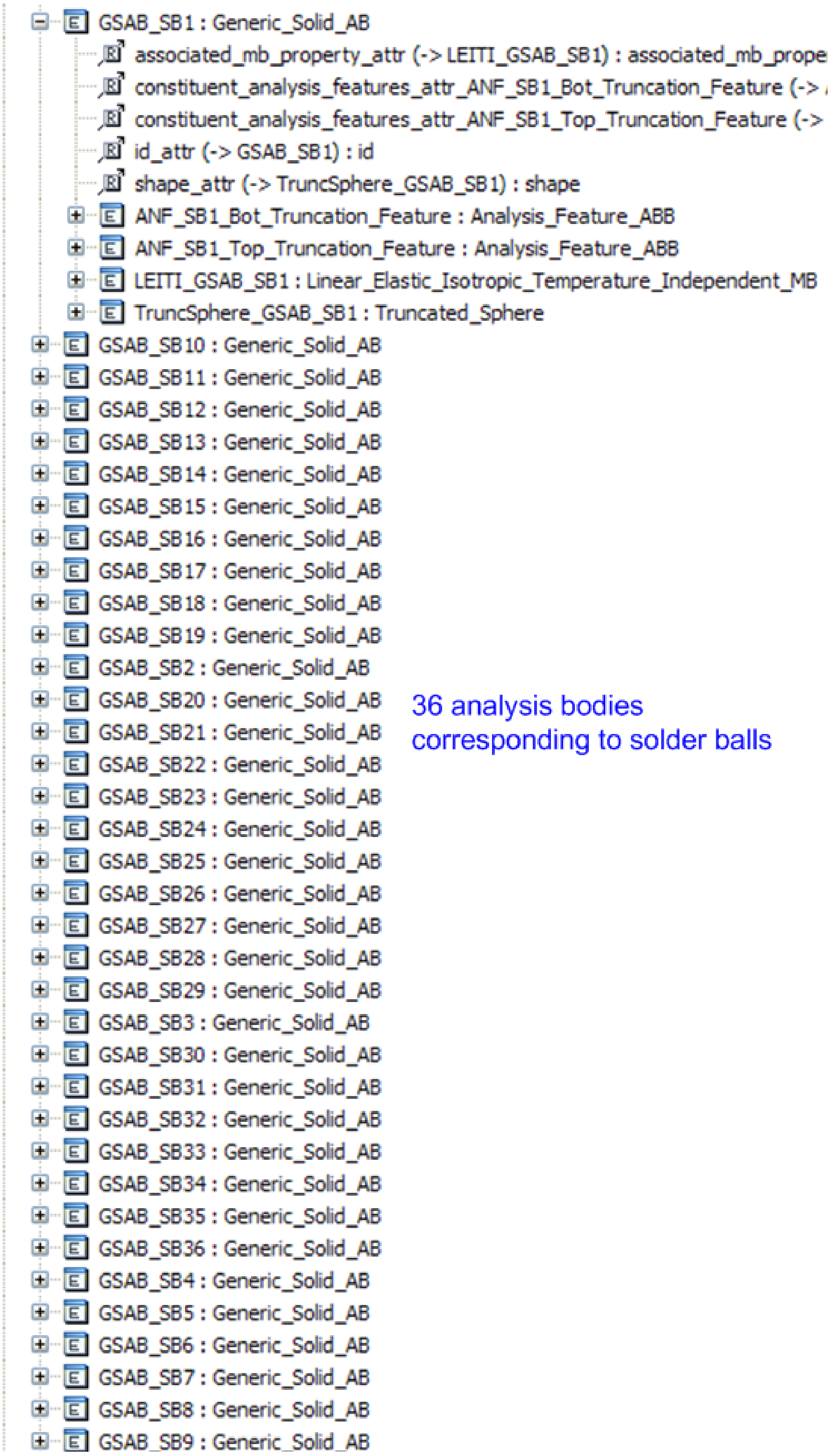
36 Solder ball components



```
sb10_sm2 (-> SB10_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb11_sm2 (-> SB11_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb12_sm2 (-> SB12_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb13_sm2 (-> SB13_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb14_sm2 (-> SB14_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb15_sm2 (-> SB15_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb16_sm2 (-> SB16_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb17_sm2 (-> SB17_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb18_sm2 (-> SB18_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb19_sm2 (-> SB19_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb1_sm2 (-> SB1_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb20_sm2 (-> SB20_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb21_sm2 (-> SB21_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb22_sm2 (-> SB22_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb23_sm2 (-> SB23_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb24_sm2 (-> SB24_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb25_sm2 (-> SB25_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb26_sm2 (-> SB26_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb27_sm2 (-> SB27_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb28_sm2 (-> SB28_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb29_sm2 (-> SB29_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb2_sm2 (-> SB2_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb30_sm2 (-> SB30_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb31_sm2 (-> SB31_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb32_sm2 (-> SB32_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb33_sm2 (-> SB33_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb34_sm2 (-> SB34_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb35_sm2 (-> SB35_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb36_sm2 (-> SB36_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb3_sm2 (-> SB3_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb4_sm2 (-> SB4_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb5_sm2 (-> SB5_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb6_sm2 (-> SB6_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb7_sm2 (-> SB7_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb8_sm2 (-> SB8_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
sb9_sm2 (-> SB9_TTF_Soldermask_Stratum_2_Bot_Surf_Interaction) : component_interactions
```

Figure 9.35: 36-solder body analyzable BGA design model structure (CP_BGA_5S2L_36SB)

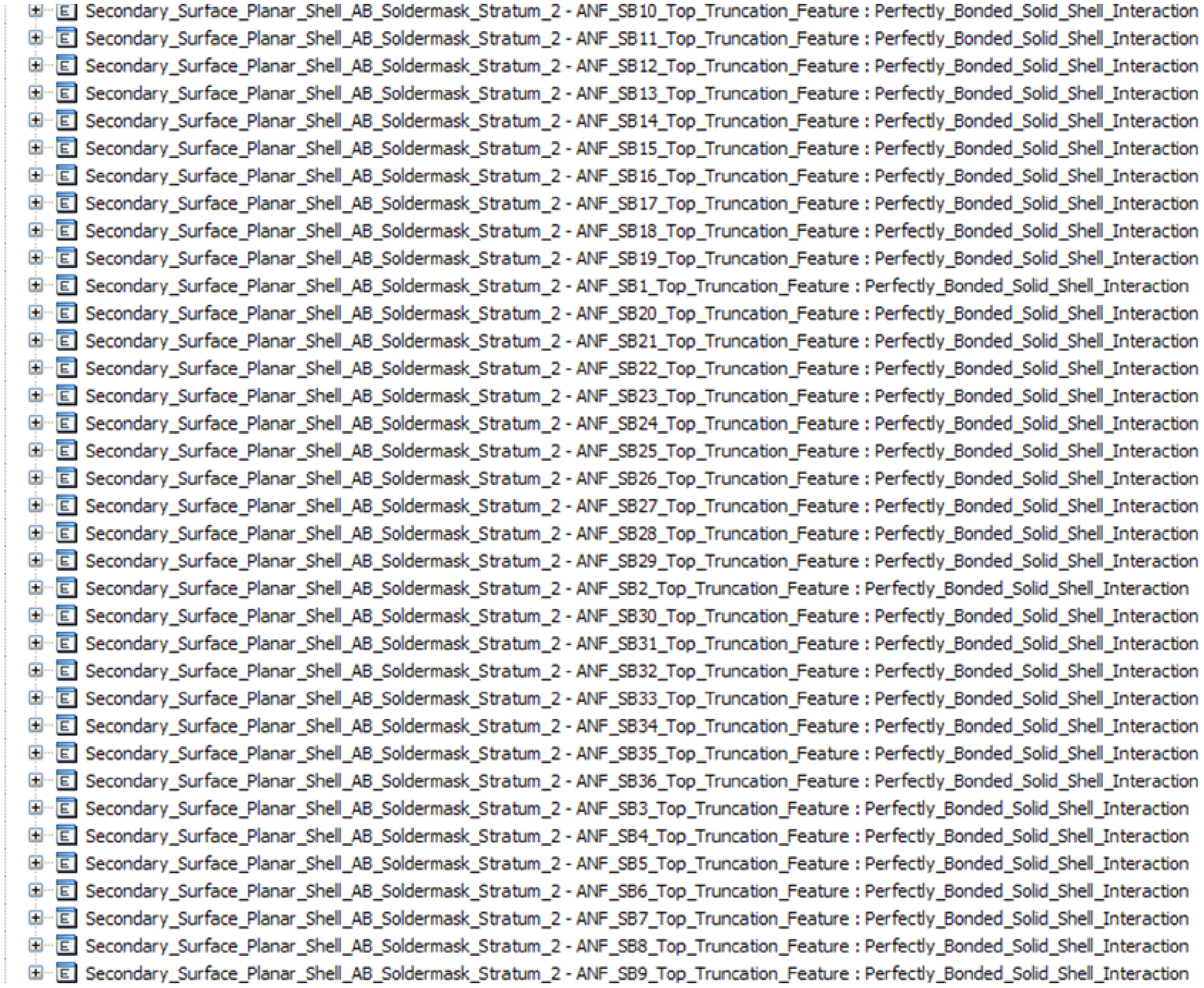
Figure 9.36 illustrates the 36 analysis bodies created as a result of executing BMFS[1] on analyzable BGA design model structure with 36 solder balls. The figure also shows the 36 tie interactions created between these 36 analysis bodies and the planar shell analysis body corresponding to the last (bottom) stratum of the chip substrate.



```
GSAB_SB1 : Generic_Solid_AB
    associated_mb_property_attr (-> LEITI_GSAB_SB1) : associated_mb_prope
    constituent_analysis_features_attr_ANF_SB1_Bot_Truncation_Feature (->
    constituent_analysis_features_attr_ANF_SB1_Top_Truncation_Feature (->
    id_attr (-> GSAB_SB1) : id
    shape_attr (-> TruncSphere_GSAB_SB1) : shape
    ANF_SB1_Bot_Truncation_Feature : Analysis_Feature_ABB
    ANF_SB1_Top_Truncation_Feature : Analysis_Feature_ABB
    LEITI_GSAB_SB1 : Linear_Elastic_Isotropic_Temperature_Independent_MB
    TruncSphere_GSAB_SB1 : Truncated_Sphere
GSAB_SB10 : Generic_Solid_AB
GSAB_SB11 : Generic_Solid_AB
GSAB_SB12 : Generic_Solid_AB
GSAB_SB13 : Generic_Solid_AB
GSAB_SB14 : Generic_Solid_AB
GSAB_SB15 : Generic_Solid_AB
GSAB_SB16 : Generic_Solid_AB
GSAB_SB17 : Generic_Solid_AB
GSAB_SB18 : Generic_Solid_AB
GSAB_SB19 : Generic_Solid_AB
GSAB_SB2 : Generic_Solid_AB
GSAB_SB20 : Generic_Solid_AB
GSAB_SB21 : Generic_Solid_AB
GSAB_SB22 : Generic_Solid_AB
GSAB_SB23 : Generic_Solid_AB
GSAB_SB24 : Generic_Solid_AB
GSAB_SB25 : Generic_Solid_AB
GSAB_SB26 : Generic_Solid_AB
GSAB_SB27 : Generic_Solid_AB
GSAB_SB28 : Generic_Solid_AB
GSAB_SB29 : Generic_Solid_AB
GSAB_SB3 : Generic_Solid_AB
GSAB_SB30 : Generic_Solid_AB
GSAB_SB31 : Generic_Solid_AB
GSAB_SB32 : Generic_Solid_AB
GSAB_SB33 : Generic_Solid_AB
GSAB_SB34 : Generic_Solid_AB
GSAB_SB35 : Generic_Solid_AB
GSAB_SB36 : Generic_Solid_AB
GSAB_SB4 : Generic_Solid_AB
GSAB_SB5 : Generic_Solid_AB
GSAB_SB6 : Generic_Solid_AB
GSAB_SB7 : Generic_Solid_AB
GSAB_SB8 : Generic_Solid_AB
GSAB_SB9 : Generic_Solid_AB
```

36 analysis bodies
corresponding to solder balls

36 tie interactions between analysis bodies (corresponding to solder balls) and planar shell analysis body (corresponding to the bottom stratum of the substrate)



```
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB10_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB11_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB12_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB13_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB14_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB15_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB16_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB17_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB18_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB19_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB1_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB20_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB21_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB22_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB23_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB24_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB25_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB26_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB27_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB28_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB29_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB2_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB30_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB31_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB32_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB33_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB34_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB35_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB36_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB3_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB4_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB5_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB6_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB7_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB8_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
Secondary_Surface_Planar_Shell_AB_Soldermask_Stratum_2 - ANF_SB9_Top_Truncation_Feature : Perfectly_Bonded_Solid_Shell_Interaction
```

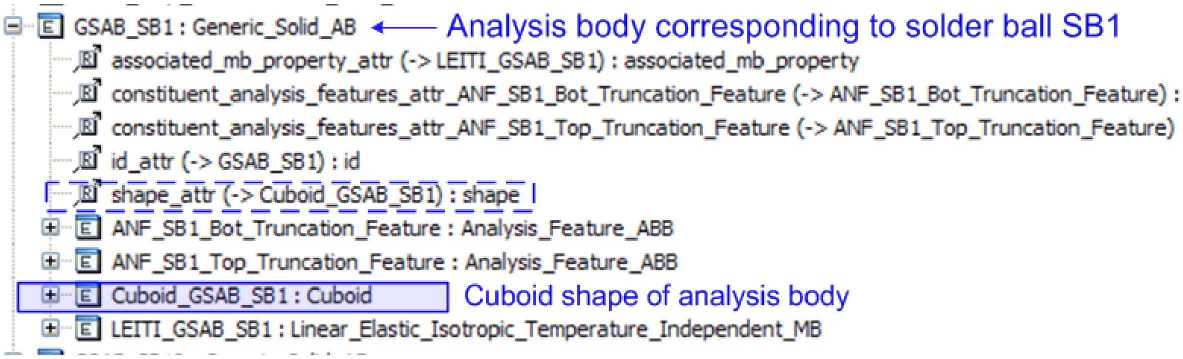
Figure 9.36: Analysis bodies and tie interactions corresponding to 36 solder balls in CP_BGA_5S2L_36 SB

9.3.2 Behavior Model Formulation Specifications 2 (BMFS^2)

In BMFS^2, an alternate idealization is prescribed for the solder balls. In contrast with BMFS^1, the shape of a solder ball is to be idealized as a cuboid in BMFS^2. This type of idealization is common for thermo-mechanical analyses of a BGA when the global behavior of the package is to be computed (Zeng 2004). The shape transformation relations may vary from (a) creating a cuboid whose height is same as the height of the truncated sphere (solder ball shape), and whose length and width are same as the diameters of the sphere, to (b) creating a cuboid those height is same as the height of the truncated sphere (solder ball shape), and whose length and width are equal and computed such that volume of the cuboid is same as the volume of the truncated sphere. The affect of this idealization change in

BMFS^2 is seen at composition Stage 1 where analysis bodies corresponding to solder balls are created.

Figure 9.37 illustrates the shape of the analysis body corresponding to solder ball SB1, created in Stage 1 composition after executing BMFS^2. The figure shows the entity representing the cuboid shape of this analysis body.

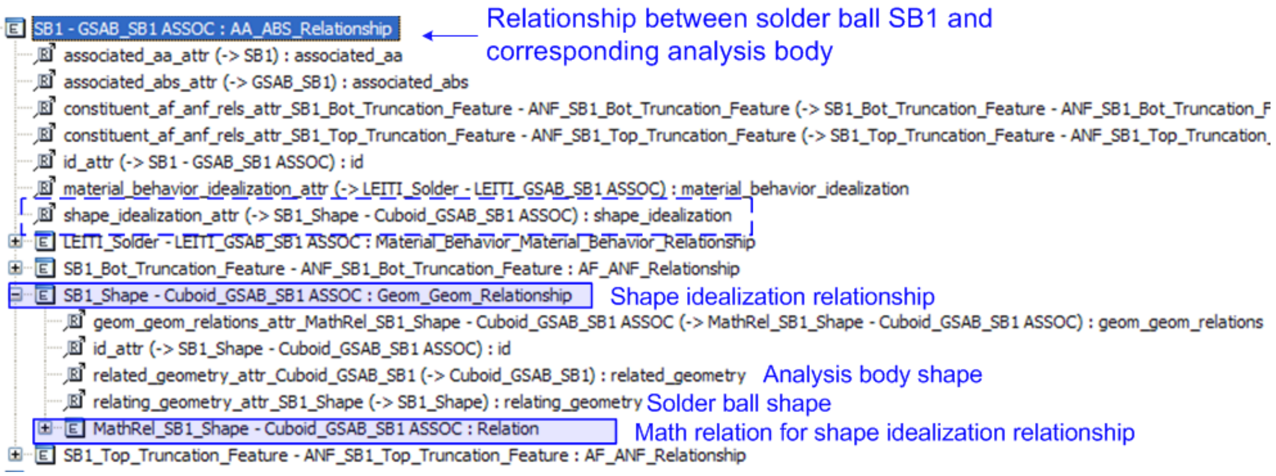


```
GSAB_SB1 : Generic_Solid_AB
    associated_mb_property_attr (-> LEITI_GSAB_SB1) : associated_mb_property
    constituent_analysis_features_attr_ANF_SB1_Bot_Truncation_Feature (-> ANF_SB1_Bot_Truncation_Feature) :
    constituent_analysis_features_attr_ANF_SB1_Top_Truncation_Feature (-> ANF_SB1_Top_Truncation_Feature)
    id_attr (-> GSAB_SB1) : id
    shape_attr (-> Cuboid_GSAB_SB1) : shape
    ANF_SB1_Bot_Truncation_Feature : Analysis_Feature_ABB
    ANF_SB1_Top_Truncation_Feature : Analysis_Feature_ABB
    Cuboid_GSAB_SB1 : Cuboid
    LEITI_GSAB_SB1 : Linear_Elastic_Isotropic_Temperature_Independent_MB
```

Analysis body corresponding to solder ball SB1
Cuboid shape of analysis body

Figure 9.37: Analysis body corresponding to solder ball SB1 has a cuboid shape

Figure 9.38 shows the analyzable artifact-analysis body relationship between solder ball SB1 and the corresponding analysis body. As shown, the shape idealization relationship is an attribute of the analyzable artifact-analysis body relationship, and it represents the math relations embodying the shape idealization transformations.



```
SB1 - GSAB_SB1 ASSOC : AA_ABS_Relationship
    associated_aa_attr (-> SB1) : associated_aa
    associated_abs_attr (-> GSAB_SB1) : associated_abs
    constituent_af_anf_rels_attr_SB1_Bot_Truncation_Feature - ANF_SB1_Bot_Truncation_Feature (-> SB1_Bot_Truncation_Feature - ANF_SB1_Bot_Truncation_F
    constituent_af_anf_rels_attr_SB1_Top_Truncation_Feature - ANF_SB1_Top_Truncation_Feature (-> SB1_Top_Truncation_Feature - ANF_SB1_Top_Truncation_
    id_attr (-> SB1 - GSAB_SB1 ASSOC) : id
    material_behavior_idealization_attr (-> LEITI_Solder - LEITI_GSAB_SB1 ASSOC) : material_behavior_idealization
    shape_idealization_attr (-> SB1_Shape - Cuboid_GSAB_SB1 ASSOC) : shape_idealization
    LEITI_Solder - LEITI_GSAB_SB1 ASSOC : Material_Behavior_Material_Behavior_Relationship
    SB1_Bot_Truncation_Feature - ANF_SB1_Bot_Truncation_Feature : AF_ANF_Relationship
    SB1_Shape - Cuboid_GSAB_SB1 ASSOC : Geom_Geom_Relationship
        geom_geom_relations_attr_MathRel_SB1_Shape - Cuboid_GSAB_SB1 ASSOC (-> MathRel_SB1_Shape - Cuboid_GSAB_SB1 ASSOC) : geom_geom_relations
        id_attr (-> SB1_Shape - Cuboid_GSAB_SB1 ASSOC) : id
        related_geometry_attr_Cuboid_GSAB_SB1 (-> Cuboid_GSAB_SB1) : related_geometry
        relating_geometry_attr_SB1_Shape (-> SB1_Shape) : relating_geometry
        MathRel_SB1_Shape - Cuboid_GSAB_SB1 ASSOC : Relation
    SB1_Top_Truncation_Feature - ANF_SB1_Top_Truncation_Feature : AF_ANF_Relationship
```

Relationship between solder ball SB1 and corresponding analysis body
Shape idealization relationship
Analysis body shape
Solder ball shape
Math relation for shape idealization relationship

Figure 9.38: Shape idealization relationship between truncated sphere shape of SB1 and cuboid shape of the corresponding analysis body

In this case, no specific math relations are specified but the entity Relation (type of math relation shown in the figure below) can represent the parameters and the math relations among these parameters—similar to a constraint blocks in SysML.

9.3.2.1 Simulation Template $_{16}{}^{2}$: Simulation template for 16-solder ball analyzable BGA model structure and BMFS^{2}

The simulation template generated using BMFS^{2} for CPM_BGA_5S2L_16SB—analyzable BGA design with 16 solder balls—is same as shown for BMFS^{1} (section 9.3.1.1) except for the shape attribute of all analysis bodies corresponding to solder balls.

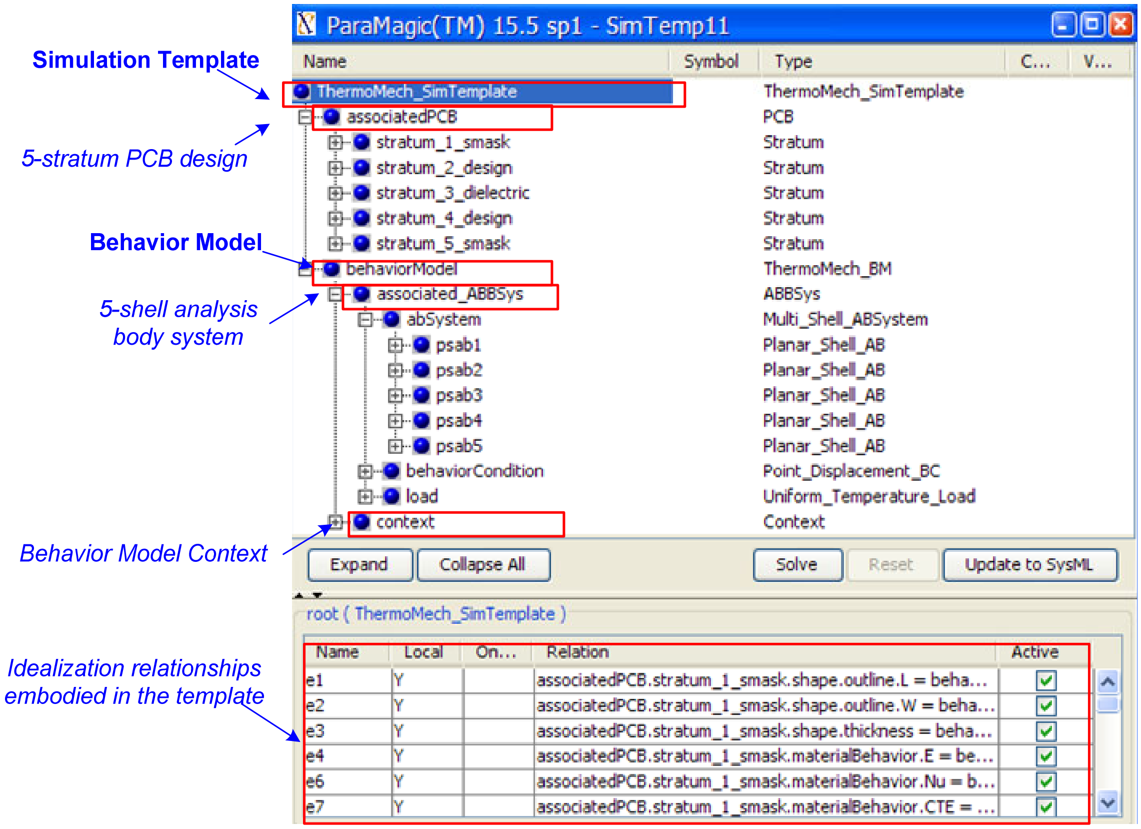
9.3.2.2 Simulation Template $_{36}{}^{2}$: Simulation template for 36-solder ball analyzable BGA model structure and BMFS^{2}

The simulation template generated using BMFS^{2} for CPM_BGA_5S2L_36SB—analyzable BGA design with 36 solder balls—is same as shown for BMFS^{1} (section 9.3.1.2) except for the shape attribute of all analysis bodies corresponding to solder balls.

9.4 Execution of Simulation Templates

In this section, the value of a single simulation template in performing trade studies on design alternatives is demonstrated. The simulation template shown here corresponds to the simulation template $ST_{5}{}^{1}$ for thermo-mechanical analysis of 5-stratum PWBs per the idealization decisions in BMFS^{1} (section 9.2.1.1). The execution of this simulation template in two different causalities—design verification and synthesis scenarios—is shown here. In the design verification scenario, the values of design parameters are given and the values of the analysis body parameters are computed. In the design synthesis scenario, the values of analysis body parameters are given and the values of design parameters are computed. The design synthesis scenario represents the case where analysts have optimized the performance of the analysis body system and intend to update the design based on these values. In each scenario, the simulation template can be used to solve for different values of the "given" parameters to compute corresponding values of the "target" parameters.

Figure 9.39 below illustrates a high-level tree view of the simulation template that would be automatically created using the Behavior Model Formulation Method. In this figure, the simulation template has been loaded in ParaMagic^{TM}—an object solver that can execute math relationships for multiple causalities. The figure shows that the simulation template is composed of the analyzable artifact structure (5-stratum PWBs) and a behavior model structure (for thermo-mechanical analysis of PWBs). The 5 stratums of the PWB and the 5 planar shell analysis bodies (psab1-5) are shown in the figure. In addition, the idealization relationships embodied in the simulation template can be seen in lower part of ParaMagic browser.



Simulation Template

5-stratum PCB design

Behavior Model

5-shell analysis body system

Behavior Model Context

Idealization relationships embodied in the template

ParaMagic(TM) 15.5 sp1 - SimTemp11

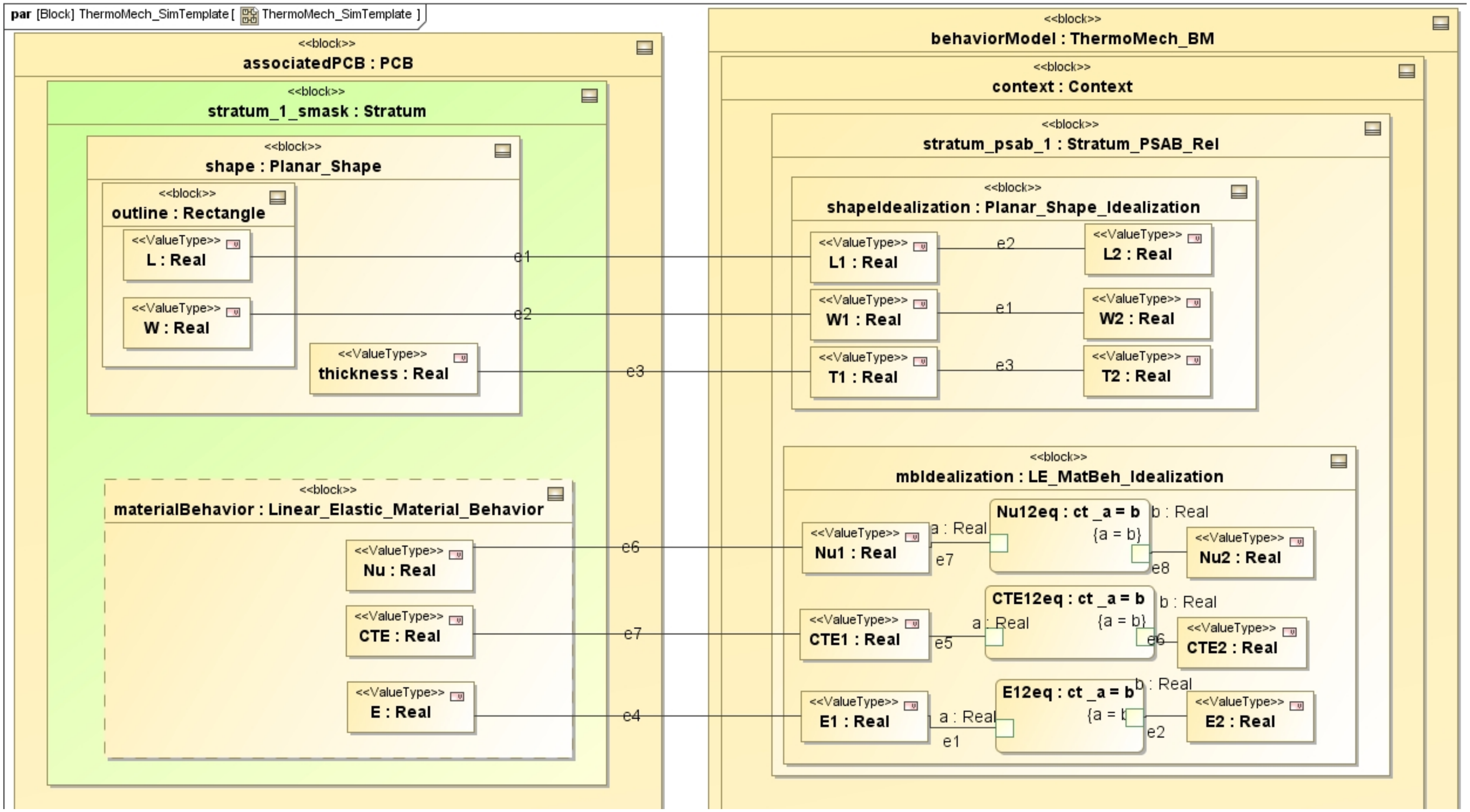
Name	Symbol	Type	C...	V...
ThermoMech_SimTemplate		ThermoMech_SimTemplate		
associatedPCB		PCB		
stratum_1_smask		Stratum		
stratum_2_design		Stratum		
stratum_3_dielectric		Stratum		
stratum_4_design		Stratum		
stratum_5_smask		Stratum		
behaviorModel		ThermoMech_BM		
associated_ABBSys		ABBSys		
abSystem		Multi_Shell_ABSystem		
psab1		Planar_Shell_AB		
psab2		Planar_Shell_AB		
psab3		Planar_Shell_AB		
psab4		Planar_Shell_AB		
psab5		Planar_Shell_AB		
behaviorCondition		Point_Displacement_BC		
load		Uniform_Temperature_Load		
context		Context		

root (ThermoMech_SimTemplate)

Name	Local	On...	Relation	Active
e1	Y		associatedPCB.stratum_1_smask.shape.outline.L = beha...	☑
e2	Y		associatedPCB.stratum_1_smask.shape.outline.W = beha...	☑
e3	Y		associatedPCB.stratum_1_smask.shape.thickness = beha...	☑
e4	Y		associatedPCB.stratum_1_smask.materialBehavior.E = be...	☑
e6	Y		associatedPCB.stratum_1_smask.materialBehavior.Nu = b...	☑
e7	Y		associatedPCB.stratum_1_smask.materialBehavior.CTE = ...	☑

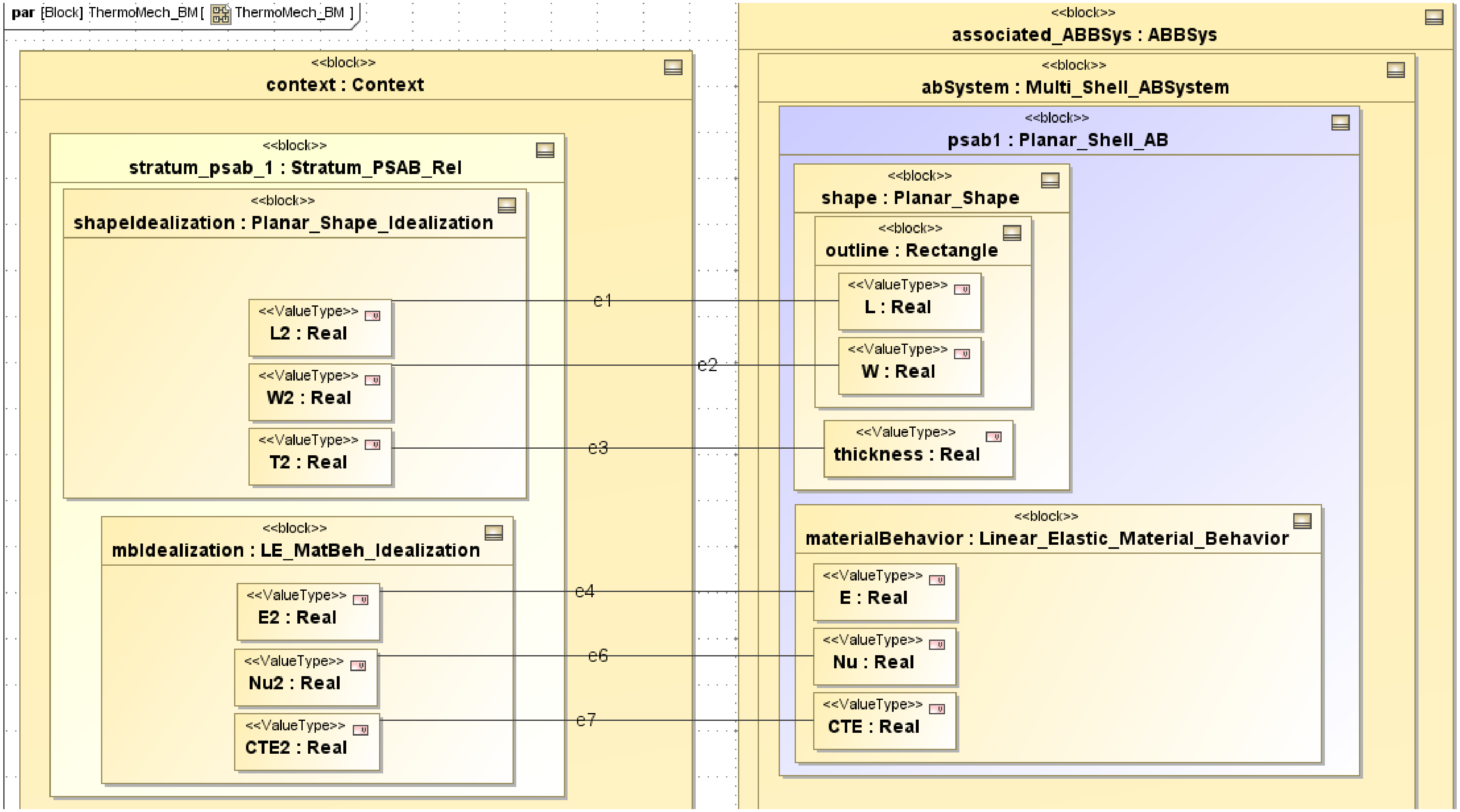
Figure 9.39: Analyzable Artifact (PCB) and Behavior Model Context

Figure 9.40 and Figure 9.41 illustrate shape and material behavior idealization relationships for a single stratum and analysis body in this simulation template (SysML block definition diagram view). Note that the same structure is repeated for all stratums and analysis bodies in the simulation template.



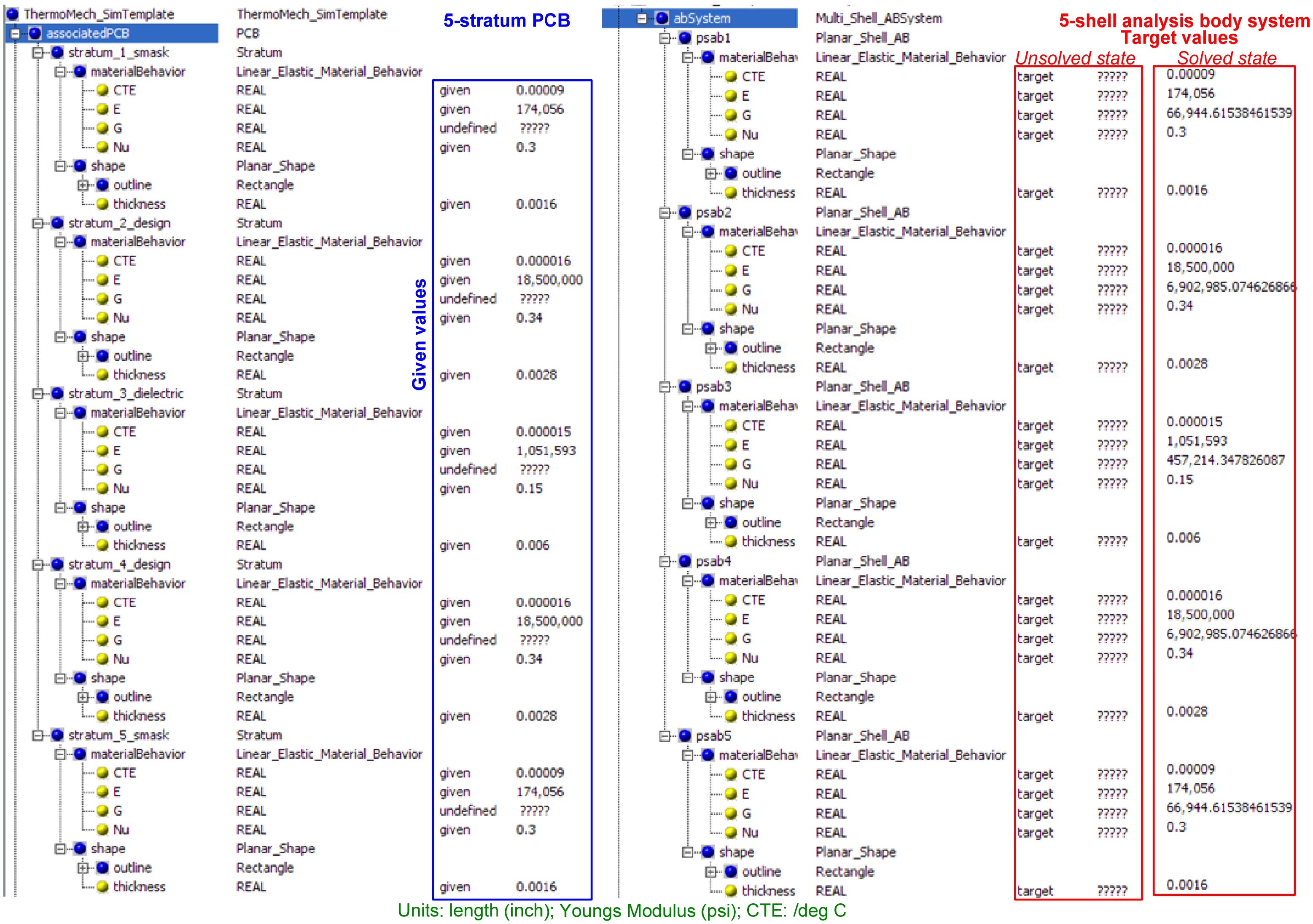
…similarly for 4 other stratums…

Figure 9.40: Analyzable artifact and behavior model context relationships for a single stratum - SysML block definition diagram view



…similarly for 4 other analysis bodies…

Figure 9.41: Behavior model context and analysis body relationships for a single analysis body (corresponding to a single PWB stratum) SysML block definition diagram view



		Given values	
ThermoMech_SimTemplate	ThermoMech_SimTemplate		
associatedPCB	PCB		
stratum_1_smask	Stratum		
materialBehavior	Linear_Elastic_Material_Behavior		
CTE	REAL	given	0.00009
E	REAL	given	174,056
G	REAL	undefined	?????
Nu	REAL	given	0.3
shape	Planar_Shape		
outline	Rectangle		
thickness	REAL	given	0.0016
stratum_2_design	Stratum		
materialBehavior	Linear_Elastic_Material_Behavior		
CTE	REAL	given	0.000016
E	REAL	given	18,500,000
G	REAL	undefined	?????
Nu	REAL	given	0.34
shape	Planar_Shape		
outline	Rectangle		
thickness	REAL	given	0.0028
stratum_3_dielectric	Stratum		
materialBehavior	Linear_Elastic_Material_Behavior		
CTE	REAL	given	0.000015
E	REAL	given	1,051,593
G	REAL	undefined	?????
Nu	REAL	given	0.15
shape	Planar_Shape		
outline	Rectangle		
thickness	REAL	given	0.006
stratum_4_design	Stratum		
materialBehavior	Linear_Elastic_Material_Behavior		
CTE	REAL	given	0.000016
E	REAL	given	18,500,000
G	REAL	undefined	?????
Nu	REAL	given	0.34
shape	Planar_Shape		
outline	Rectangle		
thickness	REAL	given	0.0028
stratum_5_smask	Stratum		
materialBehavior	Linear_Elastic_Material_Behavior		
CTE	REAL	given	0.00009
E	REAL	given	174,056
G	REAL	undefined	?????
Nu	REAL	given	0.3
shape	Planar_Shape		
outline	Rectangle		
thickness	REAL	given	0.0016

		Unsolved state		Solved state
abSystem	Multi_Shell_ABSystem			
psab1	Planar_Shell_AB			
materialBeha	Linear_Elastic_Material_Behavior			
CTE	REAL	target	?????	0.00009
E	REAL	target	?????	174,056
G	REAL	target	?????	66,944.61538461539
Nu	REAL	target	?????	0.3
shape	Planar_Shape			
outline	Rectangle			
thickness	REAL	target	?????	0.0016
psab2	Planar_Shell_AB			
materialBeha	Linear_Elastic_Material_Behavior			
CTE	REAL	target	?????	0.000016
E	REAL	target	?????	18,500,000
G	REAL	target	?????	6,902,985.074626866
Nu	REAL	target	?????	0.34
shape	Planar_Shape			
outline	Rectangle			
thickness	REAL	target	?????	0.0028
psab3	Planar_Shell_AB			
materialBeha	Linear_Elastic_Material_Behavior			
CTE	REAL	target	?????	0.000015
E	REAL	target	?????	1,051,593
G	REAL	target	?????	457,214.347826087
Nu	REAL	target	?????	0.15
shape	Planar_Shape			
outline	Rectangle			
thickness	REAL	target	?????	0.006
psab4	Planar_Shell_AB			
materialBeha	Linear_Elastic_Material_Behavior			
CTE	REAL	target	?????	0.000016
E	REAL	target	?????	18,500,000
G	REAL	target	?????	6,902,985.074626866
Nu	REAL	target	?????	0.34
shape	Planar_Shape			
outline	Rectangle			
thickness	REAL	target	?????	0.0028
psab5	Planar_Shell_AB			
materialBeha	Linear_Elastic_Material_Behavior			
CTE	REAL	target	?????	0.00009
E	REAL	target	?????	174,056
G	REAL	target	?????	66,944.61538461539
Nu	REAL	target	?????	0.3
shape	Planar_Shape			
outline	Rectangle			
thickness	REAL	target	?????	0.0016

Units: length (inch); Youngs Modulus (psi); CTE: /deg C

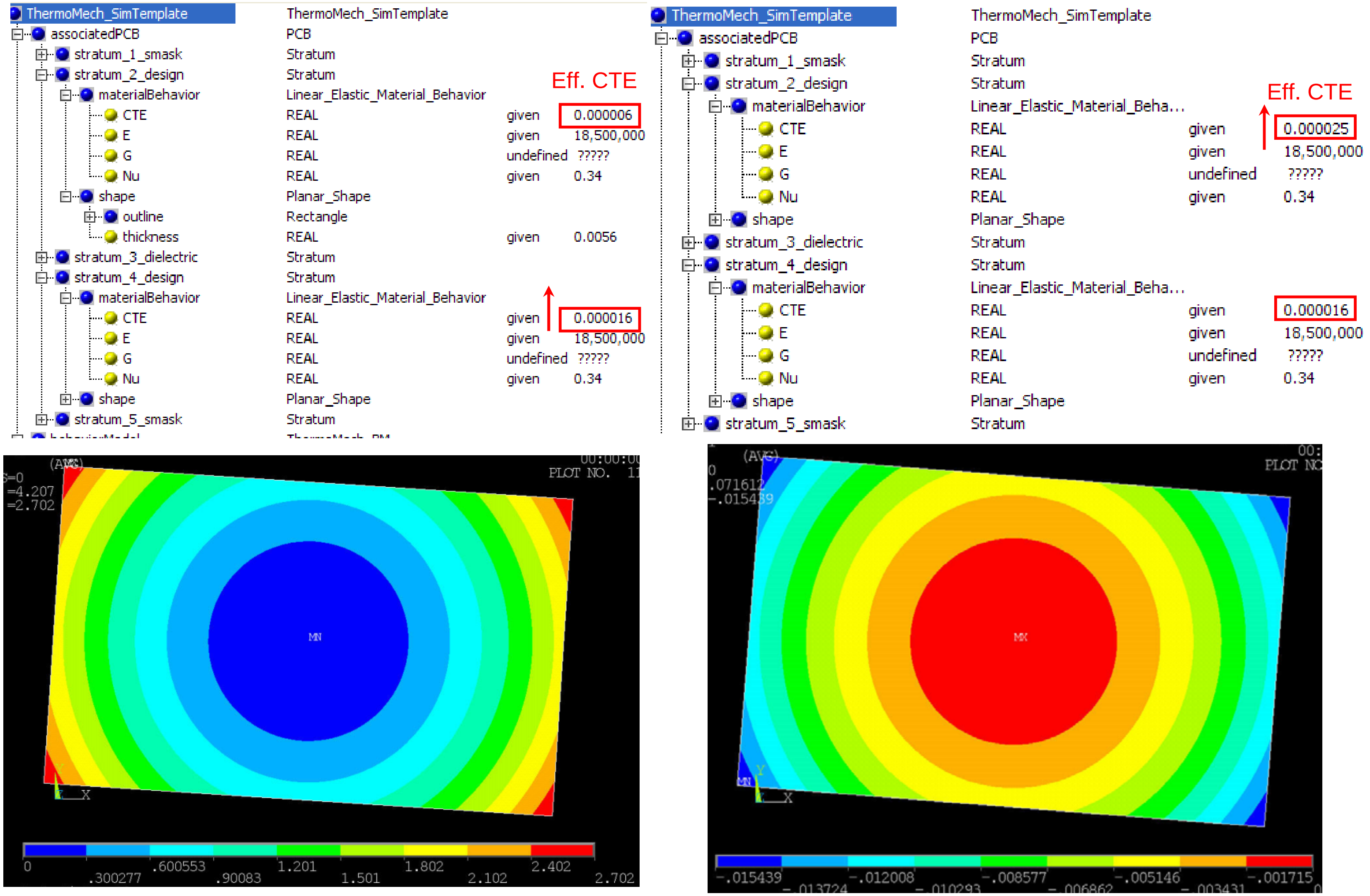
Figure 9.42: Design verification scenario - Analysis body parameters computed from design parameters

Figure 9.40 illustrates the connections between a stratum in the PWB design model structure to the analyzable artifact—analysis body relationship in the Behavior Model Context; and Figure 9.41 illustrates the connections between analyzable artifact—analysis body relationship to the analysis body in the analysis body system.

Figure 9.42 above shows an expanded tree view of the simulation template in ParaMagic browser for a single PWB design model instance. For a given PWB design model structure in the simulation template, multiple instances may be defined (corresponding to different values of parameters). The figure shows the given values of shape and material behavior parameters for all 5 stratums in the PWB. The figure also shows that the shape and material behavior parameter values for the 5-stratum analysis body system are targets. When the simulation template is executed (using ParaMagic) to solve for values of the analysis body system parameters, they are computed as shown in the figure as target values in the solved state. ParaMagic uses Mathematica to solve for the idealization relationships embodied in the simulation template. Note that the computed values of the target parameters are the same as given parameters because the idealization relationships in this simulation template equated the material behavior and shape parameters of stratums to those of planar shell analysis bodies. More complex idealization relationships can be embodied in the SysML constraint blocks shown in Figure 9.40 and Figure 9.41.

For the design verification scenario, shape and material behavior parameters of planar shell analysis bodies are computed for different values of shape and material behavior parameters of PWB design stratums. This corresponds to formulating behavior model instances (B5 models) for design model instances (D5 models). Figure 9.43 illustrates executions of the simulation template ST_5^1 (illustrated in Figure 9.40, Figure 9.41, and Figure 9.42) for two design model instances. In the first design model instance—left hand side of the Figure 9.43—the effective co-efficient of thermal expansion (CTE) of the bottom design layer (stratum_4_design) is higher than that of the top design layer (stratum_2_design), other aspects of the stackup remaining balanced. Hence, when the PWB is heated from 25^{o}C - 250^{o}C, the bottom design layer expands more than the top design layer, resulting in a bowl-shaped deformation of the PWB (concave when viewed from the top). In the second design instance—right hand side of the Figure 9.43—the effective co-efficient of thermal expansion of the top design layer (stratum_2_design) is higher than that of the bottom design layer (stratum_4_design), other aspects of the stackup remaining balanced. Hence, when the PWB is heated from 25°C - 250°C, the top design layer expands more than the bottom design layer, resulting in a dome-shaped deformation of the PWB (convex when viewed from the top).

Figure 9.44 illustrates the execution of the same simulation template (ST_5^1) in the design synthesis scenario. Here, a design model instance (D5 model) is automatically created from a given behavior model instance (B5 model) using the simulation template. This scenario represents the use case where an analyst optimizes the shape and/or material behavior properties of a multi-shell system to minimize the out-of-plane deformation. The optimal multi-shell system (represented as a behavior model instance) is then used to derive a PWB design instance.



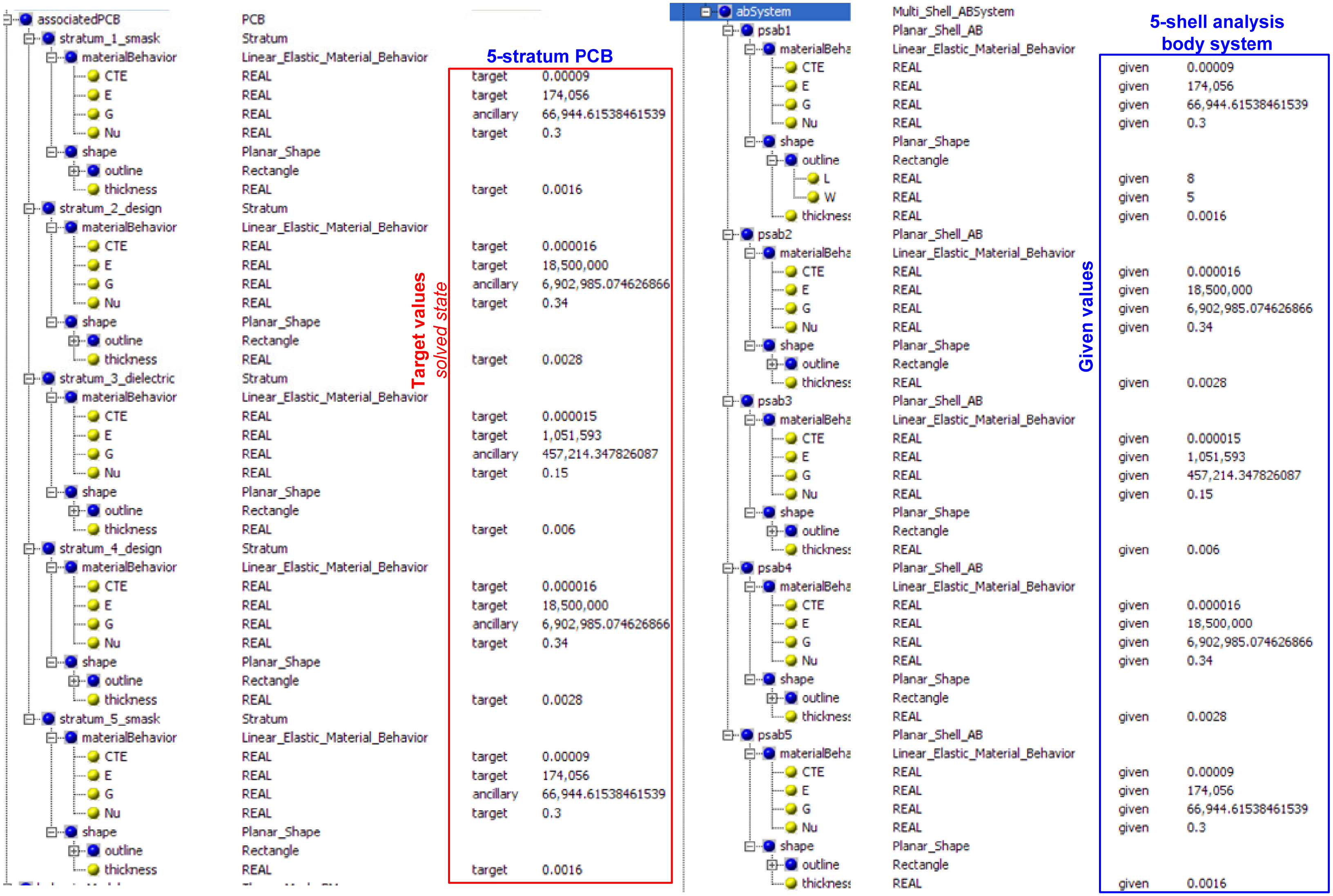
ThermoMech_SimTemplate	ThermoMech_SimTemplate		
associatedPCB	PCB		
stratum_1_smask	Stratum		
stratum_2_design	Stratum		
materialBehavior	Linear_Elastic_Material_Behavior		
CTE	REAL	given	0.000006
E	REAL	given	18,500,000
G	REAL	undefined	?????
Nu	REAL	given	0.34
shape	Planar_Shape		
outline	Rectangle		
thickness	REAL	given	0.0056
stratum_3_dielectric	Stratum		
stratum_4_design	Stratum		
materialBehavior	Linear_Elastic_Material_Behavior		
CTE	REAL	given	0.000016
E	REAL	given	18,500,000
G	REAL	undefined	?????
Nu	REAL	given	0.34
shape	Planar_Shape		
stratum_5_smask	Stratum		

Eff. CTE

ThermoMech_SimTemplate	ThermoMech_SimTemplate		
associatedPCB	PCB		
stratum_1_smask	Stratum		
stratum_2_design	Stratum		
materialBehavior	Linear_Elastic_Material_Beha...		
CTE	REAL	given	0.000025
E	REAL	given	18,500,000
G	REAL	undefined	?????
Nu	REAL	given	0.34
shape	Planar_Shape		
stratum_3_dielectric	Stratum		
stratum_4_design	Stratum		
materialBehavior	Linear_Elastic_Material_Beha...		
CTE	REAL	given	0.000016
E	REAL	given	18,500,000
G	REAL	undefined	?????
Nu	REAL	given	0.34
shape	Planar_Shape		
stratum_5_smask	Stratum		

Eff. CTE

Figure 9.43: Design verification scenario – Behavior model instances (B5) formulated from design model instances (D5), and solved using FEA



5-stratum PCB

Target values
solved state

```
associatedPCB                  PCB
  stratum_1_smask              Stratum
    materialBehavior           Linear_Elastic_Material_Behavior
      CTE                      REAL                               target     0.00009
      E                        REAL                               target     174,056
      G                        REAL                               ancillary  66,944.61538461539
      Nu                       REAL                               target     0.3
    shape                      Planar_Shape
      outline                  Rectangle
      thickness                REAL                               target     0.0016
  stratum_2_design             Stratum
    materialBehavior           Linear_Elastic_Material_Behavior
      CTE                      REAL                               target     0.000016
      E                        REAL                               target     18,500,000
      G                        REAL                               ancillary  6,902,985.074626866
      Nu                       REAL                               target     0.34
    shape                      Planar_Shape
      outline                  Rectangle
      thickness                REAL                               target     0.0028
  stratum_3_dielectric         Stratum
    materialBehavior           Linear_Elastic_Material_Behavior
      CTE                      REAL                               target     0.000015
      E                        REAL                               target     1,051,593
      G                        REAL                               ancillary  457,214.347826087
      Nu                       REAL                               target     0.15
    shape                      Planar_Shape
      outline                  Rectangle
      thickness                REAL                               target     0.006
  stratum_4_design             Stratum
    materialBehavior           Linear_Elastic_Material_Behavior
      CTE                      REAL                               target     0.000016
      E                        REAL                               target     18,500,000
      G                        REAL                               ancillary  6,902,985.074626866
      Nu                       REAL                               target     0.34
    shape                      Planar_Shape
      outline                  Rectangle
      thickness                REAL                               target     0.0028
  stratum_5_smask              Stratum
    materialBehavior           Linear_Elastic_Material_Behavior
      CTE                      REAL                               target     0.00009
      E                        REAL                               target     174,056
      G                        REAL                               ancillary  66,944.61538461539
      Nu                       REAL                               target     0.3
    shape                      Planar_Shape
      outline                  Rectangle
      thickness                REAL                               target     0.0016
```

5-shell analysis body system

Given values

```
abSystem                 Multi_Shell_ABSystem
  psab1                  Planar_Shell_AB
    materialBeha         Linear_Elastic_Material_Behavior
      CTE                REAL                               given   0.00009
      E                  REAL                               given   174,056
      G                  REAL                               given   66,944.61538461539
      Nu                 REAL                               given   0.3
    shape                Planar_Shape
      outline            Rectangle
        L                REAL                               given   8
        W                REAL                               given   5
      thicknes           REAL                               given   0.0016
  psab2                  Planar_Shell_AB
    materialBeha         Linear_Elastic_Material_Behavior
      CTE                REAL                               given   0.000016
      E                  REAL                               given   18,500,000
      G                  REAL                               given   6,902,985.074626866
      Nu                 REAL                               given   0.34
    shape                Planar_Shape
      outline            Rectangle
      thicknes           REAL                               given   0.0028
  psab3                  Planar_Shell_AB
    materialBeha         Linear_Elastic_Material_Behavior
      CTE                REAL                               given   0.000015
      E                  REAL                               given   1,051,593
      G                  REAL                               given   457,214.347826087
      Nu                 REAL                               given   0.15
    shape                Planar_Shape
      outline            Rectangle
      thicknes           REAL                               given   0.006
  psab4                  Planar_Shell_AB
    materialBeha         Linear_Elastic_Material_Behavior
      CTE                REAL                               given   0.000016
      E                  REAL                               given   18,500,000
      G                  REAL                               given   6,902,985.074626866
      Nu                 REAL                               given   0.34
    shape                Planar_Shape
      outline            Rectangle
      thicknes           REAL                               given   0.0028
  psab5                  Planar_Shell_AB
    materialBeha         Linear_Elastic_Material_Behavior
      CTE                REAL                               given   0.00009
      E                  REAL                               given   174,056
      G                  REAL                               given   66,944.61538461539
      Nu                 REAL                               given   0.3
    shape                Planar_Shape
      outline            Rectangle
      thicknes           REAL                               given   0.0016
```

Figure 9.44: Design synthesis scenario – Design parameters computed from analysis body parameters

The automated creation of behavior model instances corresponding to different design model instances using a simulation template demonstrates the value of simulation templates in performing trade studies over fixed topology design alternatives. The capability of the Behavior Model Formulation Method to automatically create simulation templates for variable topology design alternatives greatly enhances the effectiveness of analysis problem formulation process. The FEA results also validate the completeness of information represented by the KCM meta-models. If the information were incomplete, solution method-specific models (such as FEA models) could not have been solved. The behavior model instances formulated by executing simulation templates are independent of the solution method (FEA in this case), and can be solved using different solution methods and solvers.

9.5 Validation of Research Hypotheses

In this section, the primary and secondary research hypotheses are validated using results obtained for two classes of analysis problems using KCM's Behavior Model Formulation Method. First, both the secondary research hypotheses are discussed. The validation of the primary research hypothesis depends upon the validation of the secondary research hypotheses. Capabilities of specific aspects of the test results and KCM components in the context of each hypothesis are highlighted. Then, a description of the effectiveness of the Knowledge Composition Methodology for analysis problem formulation is presented. The focus of this description is to present how KCM answers the primary research question in general, including addressing the two research gaps identified in section 2.4.2.

9.5.1 Validation of Secondary Research Hypothesis 1

The secondary research question SRQ1 and the corresponding hypothesis presented in Chapter 4 are stated below.

***Secondary Research Question 1 (SRQ1)**: How can we formalize an ABB such that it can be reused for composing simulation templates?*
Hypothesis (SRH1): We can formalize an ABB such that it can be reused for composing simulation templates by: ▪ using a non-causal, declarative formalism to describe the concept and the knowledge represented by an ABB ▪ using a model transformation-based formalism to describe the method for using an ABB when composing simulation templates

The validation approach for SRH1 is founded on selecting formalism for representing ABBs, and demonstrating that ABBs represented in this formalism can be reused for composing simulation templates.

The ABB Meta-Model (section 7.2) of the Knowledge Composition Methodology provides the formalism for representing analysis building blocks (ABBs). It defines the nature of knowledge represented in ABBs. It defines four aspects of this knowledge—context, property, application conditions, and application transforms. The first two aspects represent the concept and the knowledge embodied in an ABB, and the second two aspects represent the conditions and model transformations associated with using an ABB for composing simulation templates. The ABB Meta-Model presented in section 7.2 specifically describes the context and property attributes of 9 different types of ABBs, such as analysis body ABBs and load ABBs. The ABB library presented in section 7.3 shows examples of each of different types of ABBs. SysML blocks (extensions of UML classes) provide a non-causal and declarative formalism to describe the concept embodied in an ABB.

The other two aspects of the knowledge embodied in an ABB (application conditions and application transforms) are represented using graph patterns and graph transformation rules respectively. The Artifact Model Transformation Library presented in section 8.4 defines Type 1 graph transformation rules for creating ABB instances, and

Type 2 graph transformation rule for associating an ABB instance with other parts of a simulation template. The composition of a simulation template is presented in four stages and the types of ABBs participating in each composition stage are presented in section 8.2.1.

Simulation templates automatically created for test case families 1 and 2 (sections 9.2 and 9.3 respectively) demonstrate that ABBs defined using the ABB Meta-Model can be used for composing simulation templates. For each test case family, four simulation templates are automatically created for variation of VTMB design alternatives and idealization decisions. Depending upon the idealization decisions, the simulation templates reuse the same ABB definitions (including the patterns and transformation rules for each ABB). For example, analysis body ABBs (representing planar shell analysis body), material behavior ABBs (representing isotropic and orthotropic material behaviors), temperature load ABB, and behavior conditions ABBs defined in the ABB Library and the Artifact Model Transformation Library are used for all 8 simulation templates created in the test case families 1 and 2. As shown in Figure 9.2, the ABB Library and Artifact Model Transformation Library are pre-loaded in the KCM model space in the VIATRA model transformation framework before the simulation templates are automatically created.

9.5.2 Validation of Secondary Research Hypothesis 2

The secondary research question SRQ2 and the corresponding hypothesis presented in Chapter 4 are stated below.

***Secondary Research Question 2 (SRQ2)**:* How can we systematically and automatically compose simulation templates from ABBs?

Hypothesis (SRH2): We can systematically and automatically compose simulation templates from ABBs by:

- representing idealization decisions in terms of specific ABBs to be used in composing simulation templates and the conditions for using these ABBs
- formalizing the process of composing simulation templates as a model transformation process that automatically creates simulation templates for VTMB design alternatives and idealization decisions

The Behavior Model Formulation Method defines Behavior Model Formulation Specifications (BMFS) for representing the idealization decisions taken by analysts in Figure 8.4. The conceptual specifications in BMFS are used by analysts in terms of the ABBs used for each composition stage, including conditions that need to be satisfied for using specific ABBs. The computable specifications are in the form of a script for explicitly scheduling the graph transformation rules for composing simulation templates. The computable specifications are derived from the conceptual specifications.

In both test case families (TCF1 and TCF2), two different conceptual specifications ($BMFS^1$ and $BMFS^2$) are defined for composing simulation templates. These conceptual specifications are defined in terms of the ABBs used for composing simulation templates and conditions for using each ABB. For example, in $BMFS^1$ of test case family 1, all stratums of a PCB are to be idealized as planar shell analysis bodies. This is realized by creating an instance of planar shell analysis body ABB for each PCB stratum. The conditions for using ABBs may be existential (all stratums are idealized as planar shell analysis bodies), or based on values of certain properties of design objects—the material behavior of stratums with conductive function is idealized as linear, elastic, isotropic, and temperature independent.

The Behavior Model Formulation Method presented in Chapter 8 prescribes a model transformation process based on graph transformations for automatically composing simulation templates for VTMB design alternatives and idealization decisions. Simulation templates automatically created for both test case families (TCF1 and TCF2) validate the capability of the Behavior Model Formulation Method in creating simulation templates. In TCF1, design model structures PWB_5S2L and PWB_9S4L represent two families of PWB design alternatives. Design alternatives in one family are topologically non-equivalent to the design alternatives in the other family. For two sets of idealization decisions ($BMFS^1$ and $BMFS^2$) and two design model structures, the Behavior Model Formulation Method automatically generates four different simulation templates—one for each combination of BMFS and design model structure. Similarly in TCF2, design model structures CP_BGA_5S2L_16SB and CP_BGA_5S2L_36SB represent two families of BGA chip package design alternatives such that design alternatives in one family are topologically non-equivalent to design alternatives in the other family. For two sets of idealization decisions (BMFS^1 and BMFS^2) and two design model structures, the Behavior Model Formulation Method automatically generates four different simulation templates—one for each combination of BMFS and design model structure.

9.5.3 Validation of Primary Research Hypotheses

Validation results for the secondary research hypotheses above also validate the primary research hypothesis indirectly. In this section, a summary of results from test case families (TCFs) 1 and 2 is presented in support of the primary research hypothesis. The intent of this section is to describe the effectiveness of KCM's Behavior Model Formulation Method in formulating analysis problems.

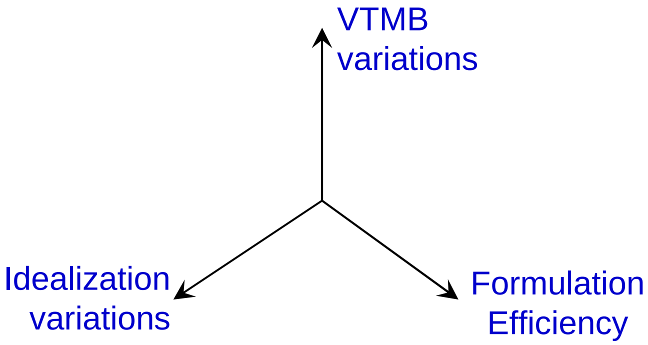
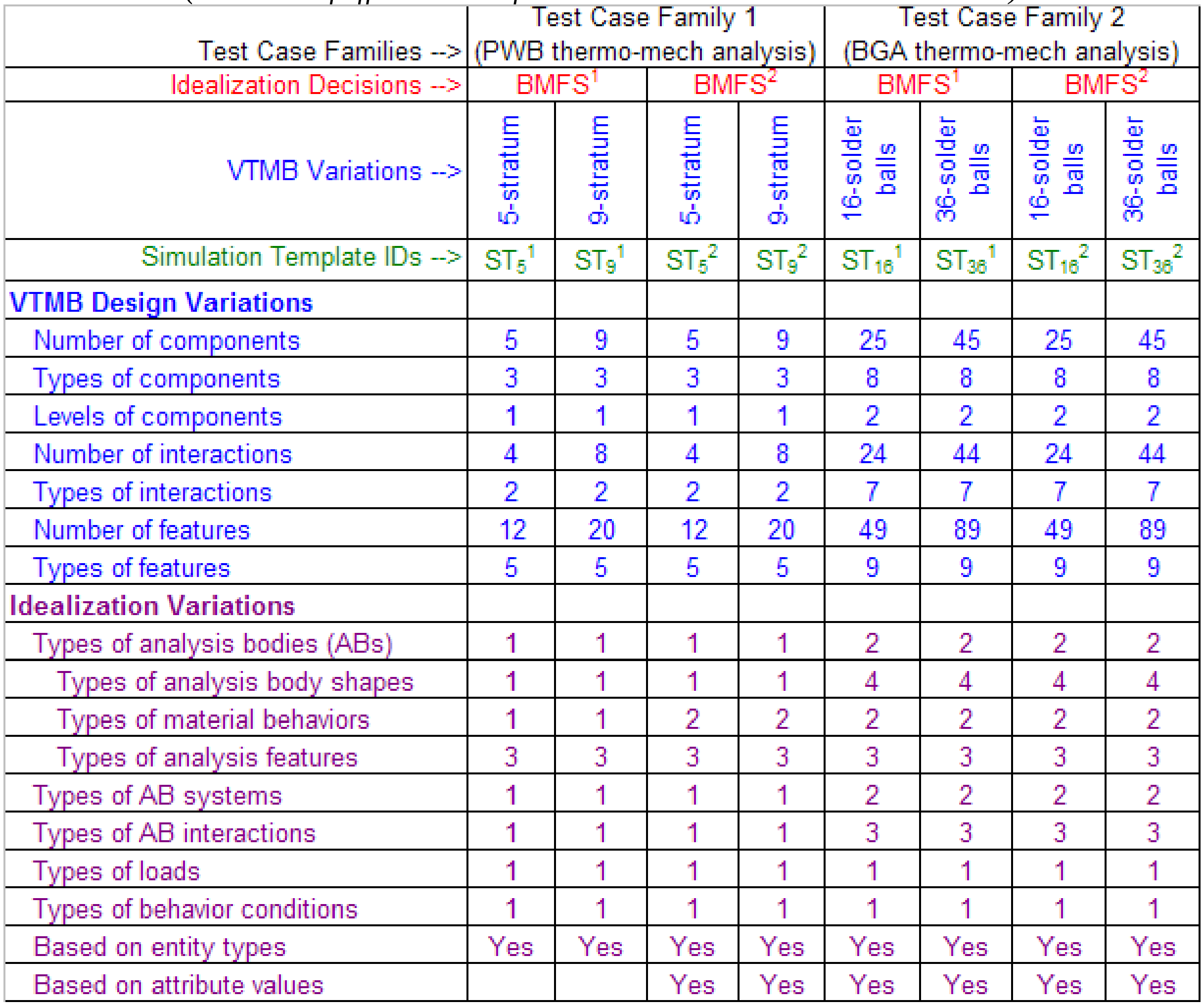


Figure 9.45: Measures of effectiveness of analysis problem formulation

In section 2.4, three measures of effectiveness of analysis problem formulation methods were presented. As shown in Figure 9.45, these measures are: (1) VTMB variations, (2) Idealization variations, and (3) Formulation Efficiency. As described in section 2.4, the effectiveness of a method for anlysis problem formulation depends on its ability to address VTMB problems and variations in idealization decisions, and formulate simulation templates efficiently. Quantitative results for the first two measures of effectiveness (VTMB variations and Idealization variations) of the Behavior Model Formulation Method as applied to test case families 1 and 2 are presented in sections 9.5.3.1 and 9.5.3.2 respectively. Results for the third measure-of-effectiveness (Formulation efficiency) are presented in section 9.5.3.3.

Table 9.5 below summarizes the effectiveness of the Behavior Model Formulation Method—as applied to test case families TCF1 (section 9.2) and TCF2 (section 9.3)—in terms of its ability to address VTMB variations and idealization variations. For each test case family, the table shows results for the four simulation templates automatically generated for combinations of two Behavior Model Formulation Specifications (BMFS^1 and BMFS^2) and two VTMB design alternatives. Eight columns corresponding to eight simulation templates created for the two test case families are shown in the table. The rows in the table show results for two measures of effectiveness of Behavior Model Formulation Method. The first set of rows corresponds to VTMB design variations, and the second set of rows corresponds to idealization variations.

Table 9.5: VTMB design variations and Idealization variations results for TCF1 and TCF2 (Measures of effectiveness of the Behavior Model Formulation Method)



Test Case Families -->	Test Case Family 1 (PWB thermo-mech analysis)				Test Case Family 2 (BGA thermo-mech analysis)			
Idealization Decisions -->	BMFS^1		BMFS^2		BMFS^1		BMFS^2	
VTMB Variations -->	5-stratum	9-stratum	5-stratum	9-stratum	16-solder balls	36-solder balls	16-solder balls	36-solder balls
Simulation Template IDs -->	ST_5^1	ST_9^1	ST_5^2	ST_9^2	ST_{16}^1	ST_{36}^1	ST_{16}^2	ST_{36}^2
VTMB Design Variations								
Number of components	5	9	5	9	25	45	25	45
Types of components	3	3	3	3	8	8	8	8
Levels of components	1	1	1	1	2	2	2	2
Number of interactions	4	8	4	8	24	44	24	44
Types of interactions	2	2	2	2	7	7	7	7
Number of features	12	20	12	20	49	89	49	89
Types of features	5	5	5	5	9	9	9	9
Idealization Variations								
Types of analysis bodies (ABs)	1	1	1	1	2	2	2	2
Types of analysis body shapes	1	1	1	1	4	4	4	4
Types of material behaviors	1	1	2	2	2	2	2	2
Types of analysis features	3	3	3	3	3	3	3	3
Types of AB systems	1	1	1	1	2	2	2	2
Types of AB interactions	1	1	1	1	3	3	3	3
Types of loads	1	1	1	1	1	1	1	1
Types of behavior conditions	1	1	1	1	1	1	1	1
Based on entity types	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Based on attribute values			Yes	Yes	Yes	Yes	Yes	Yes

Results presented in this table are described below.

9.5.3.1 VTMB Design Variations

The first set of rows in Table 9.5 measure VTMB variations of the design alternatives in each of the test case families. The variations are measured in terms of the key factors that are used for defining the assembly system topology of artifacts. As described in section 2.3, assembly system topology is characterized using number and types of components in an assembly, interactions among components in an assembly, and the features participating in these interactions. These six aspects are used for characterizing the VTMB variations for the design alternatives in the two test case families (TCF1 and TCF2). For TCF1 in which simulation templates are created for thermo-mechanical analysis of printed wiring boards (PWBs), there are two families of PWB design alternatives—one family of design alternatives for PWBs with 5 stratums and one family of design alternatives for PWBs with 9 stratums. For TCF2 in which simulation templates are created for thermo-mechanical analysis of BGA chip packages, there are two families of BGA design alternatives—one family of design alternatives for BGAs with 16 solder balls and one family of design alternatives for BGAs with 36 solder balls. The six aspects used for measuring VTMB variations in these design alternatives and the quantitative values for each design alternative are as described below.

- *Number of components,* as the name implies, corresponds to the number of components (analyzable artifacts) in the artifact assembly. For TCF1, the PWB stratums are the components. Thus, PWB design alternatives with 5 stratums have 5 components, and PWB design alternatives with 9 stratums have 9 components. For TCF2, the components in the BGA assembly consists of (1) chip mold, (2) chip, (3) die attach, (4) substrate, (5) stratums in the substrate, (6) solder balls. Thus, BGA design alternatives with 16 solder balls have 25 components—1 of each of the first four component types, 5 stratums in the substrate, and 16 solder balls. Similarly, BGA design alternatives with 36 solder balls have 45 components.

- *Types of components* correspond to the number of components with distinct functions. PWB design alternatives in TCF1 have three types of stratums—conductive, dielectric, and soldermask. Similarly BGA design alternatives in TCF2 have 8 types of components—chip mold, chip, die attach, substrate, 3 types of stratums in the substrate, and solder ball.

- *Levels of components* imply if the components are leaf-level components in the assembly or composed of multiple levels of sub-assemblies. PWB design alternatives in TCF1 have only one assembly level—PWB is an assembly composed of stratums. BGA design alternatives in TCF2 have two assembly levels—BGA assembly consists of a substrate that is composed of stratums.

- *Number of interactions* corresponds to the number of interactions among components in the assembly. For PWB design alternatives in TCF1, there are 4 interactions among stratums for 5-stratum PWBs and 8 interactions among stratums for 9-stratum PWBs. For BGA design alternatives in TCF2, the number of interactions are counted in terms of interactions between (1) mold and chip, (2) mold and substrate, (3) chip and die attach, (4) die attach and substrate, (5) stratums in the substrate, and (6) solder balls and substrate. Both 16-solder ball and 36-solder ball BGA design alternatives have one interaction of each of the first four types, 4 interactions between the stratum substrates, and one interaction between each of the solder balls and the substrate. Thus, the two sets of design alternatives have 24 and 44 interactions respectively.

- *Types of interactions* are counted based on the types of components participating in the interactions. For PWB design alternatives in TCF1, there 2 types of interactions—one between soldermask stratums and conductive stratums, and one between conductive stratums and dielectric stratums. For BGA design alternatives in TCF2, there are 7 types of interactions—2 types for the substrate and 5 other types as described above in *Number of interactions*.

- *Number of features* corresponds to the number of analyzable features defined on components. For PWB design alternatives in TCF1, (1) two features are defined for each stratum corresponding to its surfaces, (2) a feature is defined corresponding to the volume of the PWB, and (3) a feature is defined corresponding to the mid-point of the bottom surface of the last stratum in the stackup. Thus, PWB design alternatives with 5 and 9 stratums have 12 and 20 features respectively. For BGA design alternatives in TCF2, (1) one feature is defined for the mold, (2) two features are defined for the die attach, chip, each solder ball, and each stratum in the substrate, (3) one feature is defined corresponding to the volume of the BGA assembly, and (4) one feature is defined corresponding to the mid-point of the bottom surface of the last stratum in the substrate stackup. Thus, BGA design alternatives with 16 and 36 solder balls have 49 and 89 features respectively.

- *Types of features* are characterized based on the shape of the feature, function of the feature, type of artifact for which the feature is defined, or a combination of these. In the table, the types of features are characterized based on their shape and the type of artifact for which the feature is defined. For PWB design alternatives in TCF1, there are 5 types of features—3 types corresponding to the surfaces of 3 types of stratums, 1 type corresponding to the PWB volume, and 1 type corresponding to the mid-point of the bottom stratum. For BGA design alternatives in TCF2, there are 9 types of features—7 types corresponding to each of the seven types of leaf-level[27] components, and 2 types corresponding BGA volume and mid-point of the bottom surface of the last stratum in the substrate stackup.

In summary, the results from TCF1 and TCF2 demonstrate the Behavior Model Formulation Method can be used for formulating simulation templates for large set of design variations, especially VTMB-type variations.

9.5.3.2 Idealization Variations

The second set of rows in Table 9.5 measure types of idealizations used for formulating simulation templates in both test case families TCF1 and TCF2. The idealization variations are measured in terms of the number of specializations of each type of ABB used in formulating simulation templates. The table shows eight[28] types of ABBs used for measuring the variations in the idealizations. Two additional criteria are

[27] No features are defined for the substrate (as a whole) for BGA alternatives in TCF2

[28] All structural behavior parameters are to be computed for the test case families. Hence, behavior ABB does not contribute to the variations.

used to denote if idealization decisions were specified in terms of types of design objects (components, features, and interaction), or also using the properties of these objects.

Note that the rationale for defining analysis building blocks is that a relatively small set of ABBs can be used for formulating a large class of analysis problems. *Hence, an entire class of analysis problems, such as thermo-mechanical analyses of PWBs, can be formulated using a few specializations of each type of ABB*. The type of ABB corresponds to the type of decision taken by analysts.

- *Types of analysis body ABBs*: One specialization of analysis body ABB (planar shell analysis body ABB) is used for simulation templates created in TCF1, and two specializations of analysis body ABB—planar shell analysis body ABB and generic solid analysis body ABB—are used for simulation templates created in TCF2.

- *Types of shape ABBs*: One specialization of shape—planar shell shape—is used for simulation templates created in TCF1 and four specializations of shape—corresponding to the shape of mold, chip, die attach or substrate stratums, and solder ball—are used for simulation templates created in TCF2.

- *Types of material behavior ABBs*: Except for simulation templates created using BMFS[1] in TCF1, all simulation templates created in TCF1 and TCF2 use two specializations of material behavior ABB, corresponding to linear elastic isotropic temperature-independent and linear elastic orthotropic temperature-independent material behaviors.

- *Types of analysis feature ABBs*: In the simulation templates created in TCF1, 3 specializations of analysis feature ABBs are used—point feature, planar surface feature, and volume feature ABBs. In addition to these three analysis features, simulation templates created in TCF2 also used a generic surface analysis feature for representing the non-planar surfaces, such as the bottom surface of the mold.

- *Types of analysis body systems*: For simulation templates created in TCF1, one type of analysis body system (multi-shell analysis body system) is used, and for simulation templates created in TCF2, two types of analysis body systems are used—multi-shell system for the BGA substrate used in a solid-shell system.

- *Types of analysis body interaction ABBs*: For simulation templates created in TCF1, 1 type of analysis body interaction ABB is used (shell-shell tie interaction ABB), and for simulation templates created in TCF2, 3 types of analysis body interaction ABBs are corresponding to tie interactions between two solids, solid and shell, and two shells.

- *Types of load ABBs and behavior condition ABBs:* For simulation templates created in TCF1 and TCF2 one type of load ABB (temperature load) and one type of behavior condition ABB (point displacement fixed condition ABB)

Though the number of specializations of each ABB type demonstrated for simulation templates in TCF1 and TCF2 is low, the process is similar for using other specializations defined in the ABB Library or those that can be created based on the ABB Meta-Model. Graph patterns and transformation rules defined in the Artifact Model Transformation Library for composition Stages 1-4 are defined in terms of the different ABB types. Thus, all specializations of each ABB type can use the same set of patterns and rules for composition. If relationships particular to a specialized ABB need to be created in these composition stages, existing patterns and rules for that ABB type can be extended or new patterns and transformations rules may be created.

In addition to the types of ABB used for representing the idealization decisions, the Behavior Model Formulation Method also allows analysts to specify conditions for idealization decisions. Conditions can be specified based on the types of design objects, such as those that check for the existence of a specific type of component, or feature, or interaction in the design assembly. Conditions can also be specified based on the properties of design objects or other properties derived from these properties, such as those that check for attribute values of specific types of components, features, or interactions. As an example, in TCF1, simulation templates created for BMFS[2] check the value of the *function* attribute of stratums to use isotropic versus orthotropic material behavior.

In summary, the results from TCF1 and TCF2 demonstrate the Behavior Model Formulation Method can be used for formulating simulation templates for variations in idealization decisions taken by analysts.

9.5.3.3 Formulation Efficiency

In this section, quantitative results for Formulation Efficiency (third measure-of-effectiveness) of the Behavior Model Formulation Method (BMFM) in creating simulation templates for test case families TCF1 and TCF2 are presented. Table 9.6 below consists of two sets of rows. The first set of rows present results for the formulation efficiency of KCM's BMFM (referred in the table as KCM for brevity). The second set of rows show the number of entities in the source model (given) and the number of entities automatically generated by the BMFM in formulating simulation templates.

The formulation efficiency of the BMFM is characterized in terms of the percentage reduction in the time taken to formulate simulation templates using the BMFM versus current methods. The table shows how the cost of formulating simulation templates using the BMFM is computed. Here, cost is measured in terms of time (assuming a constant cost/time factor). The cost of formulating simulation templates using the BMFM consists of two parts: (1) Fixed cost, and (2) Marginal cost.

The fixed cost is an upfront cost to create VTMB design meta-model (D3 model), create ABBs, and specialize pre-defined patterns and transformation rules (if needed). KCM provides the CPM2_xKCM model that can be directly used as a VTMB design meta-model for a particular artifact. For TCF1, it took 5 hours to define a D3 model (PDMM and PAMM in section 6.2); and for TCF2, it took 7 hours to define a D3 model Note that a VTMB design meta-model is used for representing design alternatives with different assembly system topologies, and not specific to a particular type of analysis. The time for creating D3 models (as shown in the table) is based on the assumption that the D3 models did not exist previously (worst case scenario).

Table 9.6: Formulation Efficiency results for TCF1 and TCF2
(Measure-of-effectiveness of the Behavior Model Formulation Method)

Test Case Families >>	TCF1				TCF2			
Idealization Decisions >>	BMFS^1		BMFS^2		BMFS^1		BMFS^2	
VTMB Variations >>	5-stratum	9-stratum	5-stratum	9-stratum	16-solder ball	36-solder ball	16-solder ball	36-solder ball
Simulation Template IDs >>	ST_5^1	ST_9^1	ST_5^2	ST_9^2	ST_{16}^1	ST_{36}^1	ST_{16}^2	ST_{36}^2
Formulation Efficiency								
cost stated below in terms of time taken								
Total Cost (in terms of time taken) using KCM								
Fixed Cost								
Create VTMB design meta-model (D3)	5h				7h			
Create library primitives (ABBs)	$10h^{**}$				$18h^{**}$			
Specialize/Extend patterns and transformation rules	2h				2h			
Marginal Cost								
Define conceptual specifications (minutes)	$10m^b$		2m***		$30m^b$		5m***	
Automatically generate simulation template	< 5s	< 5s	< 5s	< 5s	< 5s	< 15s	< 5s	< 15s
*Total Cost per template using **KCM** = Fixed Cost / Number of templates + Marginal cost*								
Total Cost per template (for 20 templates)	0.93h	0.93h	0.95h	0.95h	1.6h	1.6h	1.64h	1.64h
Total Cost per template (for 40 templates)	0.51h	0.51h	0.53h	0.53h	0.93h	0.93h	0.97h	0.97h
Total Cost per template (for 80 templates)	0.30h	0.30h	0.31h	0.31h	0.59h	0.59h	0.63h	0.63h
Total Cost (in terms of time taken) using Current Methods	$5h^b$	$3h^{***}$	$2h^{***}$	$5h^{***}$	$15h^b$	$5h^{***}$	$5h^{***}$	$10h^{***}$
Reduction in time (KCM versus Current Methods)								
% Reduction in time for 20 templates	81%	88%	86%	91%	89%	92%	92%	93%
% Reduction in time for 40 templates	90%	94%	92%	95%	94%	95%	95%	96%
% Reduction in time for 80 templates	94%	96%	96%	97%	96%	97%	97%	97%

**[1 hr / ABB x 10-18 ABBs]=10-18 hrs; ***additional time with respect to the base time; **$x^{b:}$** base time for TCF

Number of given and generated entities								
Given entities (FTMB Analyzable Artifact Model Structure)								
Number of analyzable artifacts (AAs)	5	9	5	9	25	45	25	45
Aux entities (shapes, material behaviors,...)	10	18	10	18	50	90	50	90
Number of interactions (AAI)	4	8	4	8	24	44	24	44
Number of analyzable features (AFs)	12	20	12	20	49	89	49	89
Aux entities (shapes,...)	12	20	12	20	49	89	49	89
Number of given entities*	**43**	**75**	**43**	**75**	**197**	**357**	**197**	**357**
Automatically generated entities (FTMB Artifact Behavior Model Structure)								
Number of analysis bodies (ABs)	5	9	5	9	25	45	25	45
Aux entities (shape, material behavior)	10	18	10	18	50	90	50	90
Number of AB - AA relations	5	9	5	9	25	45	25	45
Aux entities (shape idz, material behavior idz)	10	18	10	18	50	90	50	90
Number of analysis body interactions (ABI)	4	8	4	8	24	44	24	44
Number of AAI - ABI relations	4	8	4	8	24	44	24	44
Number of analysis features (ANFs)	12	20	12	20	49	89	49	89
Number of AF - ANF relations	12	20	12	20	49	89	49	89
Aux entities (shape idz)	12	20	12	20	49	89	49	89
Number of generated entities*	**74**	**130**	**74**	**130**	**345**	**625**	**345**	**625**
Number of entities in a simulation template*	**117**	**205**	**117**	**205**	**542**	**982**	**542**	**982**

***** excluding attribute relations and auxiliary entities

Similarly, 10 ABBs were used for TCF1 and 18 ABBs were used for TCF2. Assuming that these ABBs did not exist in the library, it would typically take 1 hour to create an ABB model as an instance of KCM's ABB Meta-Model. Also assuming that pre-defined transformation rules and patterns may need to be extended for new D3 models defined as specializations of CPM2_xKCM, the table shows an additional 2 hours for such extensions. Note that these three component costs of the total fixed cost are expended once upfront. The required time (as shown in the table) is also based on the assumption that ABBs and meta-models required for the two test case families were completely different (which was certainly not the case). This is also a worst-case scenario.

The marginal cost is the additional cost to formulate each simulation template beyond the fixed cost. The marginal cost for formulating simulation templates using the BMFM consists of two components: (1) cost to specify idealization decisions (conceptual specifications), and (2) cost to automatically generate simulation templates. As with the fixed cost, the marginal cost is also stated in the table in terms of the time. For TCF1, the time required for defining conceptual specifications BMFS^1 was around 10 minutes, and the time required to modify BMFS^1 to create BMFS^2 was around 2 minutes. Similarly, for TCF2, the time required for defining conceptual specifications (BMFS^1) was around 30 minutes, and the time required to modify BMFS^1 to create BMFS^2 was around 5 minutes. The time taken to automatically generate simulation templates was of the order of seconds (15 seconds for the the 36-solder ball BGA design in TCF2).

The table shows the total cost of formulating each simulation template as a sum of the fixed cost per template and the marginal cost. The fixed cost per template is computed by dividing the total fixed cost with the number of simulation templates (as an estimate) for which the meta-models and ABBs can be reused. The total cost of formulating each simulation template is computed based on the fixed cost distributed over 20, 40, or 80 simulation templates. Note that the estimated numbers of simulation templates over which the fixed cost is distributed are realistic. As an example, 8 simulation templates are created for two test case families TCF1 and TCF2 for only two VTMB variations and two idealization variations.

In the BMFM, the conceptual specifications are defined once for all VTMB variations. However, this is not the case for existing methods where variations in the number or configuration of components and interactions require significant increases in the amount of time spent in formulating simulation templates. As an example, the time required for formulating the simulation template shown in Figure 9.40 and Figure 9.41 (for 5-stratum PWBs based on BMFS^1) is around 5 hours—based on personal experiences by the author. Table 9.6 also shows the time required for formulating simulation templates manually and modifying them for VTMB variations and idealization variations. As an example, it would take ~5 hours to formulate ST_5^1 (simulation templates for 5-stratum PWB based on BMFS^1) and an additional 3 hours to add more relationships for 9-stratum PWB (based on the same BMFS). With changes in the BMFS, the time required to manually re-wire simulation templates can be significant too. For example, it took 2 additional hours to modify ST_5^1 for BMFS^2, thereby resulting in ST_5^2. In addition to time, manual "re-wiring" of simulation templates is more error-prone and may require significant debugging effort.

Based on the time required to formulate simulation templates using the KCM and using current methods, the percentage reduction in time (using the KCM versus current methods) is presented in Table 9.6 for each simulation template. The percentage reduction is presented for all the three scenarios—fixed cost in formulating simulation templates using the KCM is distributed over 20, 40, and 80 simulation templates. *Overall, the results show 90% or greater (on average) reduction in the time required for formulating simulation templates using the KCM versus current methods.* The results clearly support the higher formulation efficiency of the KCM when compared to current methods.

Note that the fixed costs depend on the meta-models, ABBs, and extensions to pre-defined transformation rules and patterns for formulating simulation templates. With a lower fixed cost, the breakeven point for formulating simulation templates using the KCM may seem to be achievable with less usage. However, this is countered by the lost opportunity to formulate a larger variety of simulation templates. Meta-models and ABBs that may be used for representing only few types of artifact design variations and for formulating simulation templates for specific analyses may lower the fixed cost but add to the lost opportunity to formulate a large set of simulation templates.

The contribution of different aspects of the KCM in increasing the set of simulation templates that may be formulated using the Behavior Model Formulation Method, thereby lowering the cost and effort to do so is described below:

- Meta-Models: KCM provides meta-models for representing different aspects of simulation templates. CPM2_xKCM—extension of CPM2 (Fenves 2004)—is a generic meta-model for abstract representation of design alternatives in different application areas. Chapter 6 illustrates how artifact-specific design meta-models can be defined as specializations of CPM2_xKCM. These artifact-specific meta-models can be used for representing VTMB design alternatives of the artifact. The Core Behavior Model (Chapter 7) provides an abstract meta-model for representing physics-based behavior models for VTMB problems, including idealization relationships between design models and behavior models. The CBM is abstract and extensible. It depends on the analysis building blocks (ABBs) to represent domain-theoretic analysis knowledge.

- ABB Library: KCM provides an initial library of ABBs that can be used as-is. Nine different types (categories) of ABBs are defined in the ABB Meta-Model. The library contains specializations of each ABB type. Additional specializations of ABBs can be easily defined. Creation of simulation templates for a new class of problems requires creation of new specializations of existing types of ABBs. Creation of new ABBs is one of the few aspects of the KCM that is requires effort. However, this effort is minimal and can enable formulation of simulation templates for a large class of analysis problems as below.
  - A large class of analysis problems can be addressed by each new specialization of an ABB type.
  - The number of specializations of each ABB type for a given physics-based domain (such as structural analysis or thermal analysis) is limited.
  - If a new ABB can be associated with concepts in the solver tools, such as an element type in an FEA solver, then analysts do not need to represent all domain theoretic mathematical relationships when defining ABBs, thus saving time and effort. For

example, the definition of a shell analysis body ABB does not require the representation of domain theoretic equations for shell behavior since FEA tools (for example) have elements to represent shells.

- Artifact Model Transformation Library: KCM provides a library of reusable graph patterns and transformation rules based on the meta-models. For the design model stack, these rules and patterns are defined at both Level 1 and Level 3. For the behavior model stack, these rules and patterns are defined at Level 1, for each ABB type, and in some cases for specializations of ABB types. Since Level 1 meta-models do not change from one class of problems to another, the graph patterns and transformation rules defined for them can be reused for all types of simulation templates formulated using the Behavior Model Formulation Method. The Level 3 meta-model in the design model stack is for representing VTMB design alternatives for an entire family of artifacts, such as printed wiring boards. This governs the applicability of graph patterns defined for Level 3 meta-models in the design model stack.

- Conceptual specifications to control computable specifications: One of the key factors that contribute to the efficiency of formulating simulation templates using KCM's Behavior Model Formulation Method is the ability to change idealizations with relative ease. As demonstrated for test case families TCF1 and TCF2, the conceptual specifications are defined in terms of the ABBs, including the conditions for using specific ABBs. This provides a higher level of semantic handle on defining idealizations as opposed to writing procedural code (such as the computable specifications).

- Use of graph transformation-based approach to compose simulation templates: The use of the graph transformation-based approach to model composition provides a modular and extensible to automatically formulate simulation templates. Graph patterns, transformation rules, and transformation process together provide a three-tier framework (Figure 8.22, section 8.2.3) for formulating simulation templates. Graph patterns provide a declarative and highly efficient representation for describing conditions and constraints, as well as searching model elements. Due to their non-causal representation, a single pattern can be used for achieving multiple use causes depending upon variables that are bound or unbound during pattern calls (section 8.2.3). In addition, graph transformation rules enable the representation atomic units of model transformations that can reused across for formulating simulation templates. Graph transformation rules provide a declarative representation of a transformation step in terms of the source and target model graphs and not in terms of the process of creating a target model graph. This approach is more intuitive to modelers and analysts who want to define new specializations of ABBs or extend the KCM meta-models.

Table 9.6 also presents a summary of the number of entities in the source model and the number of entities automatically generated by the BMFM when formulating simulation templates (target models) for TCF1 and TCF2. The numbers provide an estimate of the number of entities automatically created in formulating information-rich simulation templates. As an example, for simulation template ST_{36}^{2} in TCF2, 625 entities were created and the total number of entities in the simulation template is ~1000. Manual creation and modification of simulation templates with large number of entities is certainly not feasible. In this light, KCM's Behavior Model Formulation Method provides a much superior approach to formulating simulation templates.

9.6 Summary

Two families of test cases are presented in this chapter to demonstrate the capability of the Behavior Model Formulation Method in handling VTMB-type design variations and idealization variations when automatically composing simulation templates. Automated composition of eight simulation templates using the Behavior Model Formulation Method—as realized in the VIATRA graph transformation framework—is demonstrated for both test case families combined. The illustrations demonstrate the extent and the depth of model elements automatically created for each simulation template. In addition, the execution of simulation templates for generating behavior model instances that can be solved in FEA tools is also demonstrated. The execution of simulation templates both in design verification and synthesis scenarios demonstrates the value of a single simulation template in addressing routine analysis problems.

In section 9.5, a detailed validation of the secondary and primary research hypotheses is presented based on the simulation templates automatically created in both test case families. The effectiveness of the Behavior Model Formulation Method in formulating simulation templates is established using the results summarized in Table 9.5. In addition, a discussion on other components of the KCM that strongly contribute to increasing both the efficiency and effectiveness of formulating simulation templates using this approach is presented.

Chapter 10 : RESEARCH CONTRIBUTIONS AND FUTURE WORK

In this chapter, a summary of research contributions and recommended future work are presented in sections 10.1 and 10.2 respectively.

10.1 Research Contributions

Figure 10.1 below shows the state-of-the-art in formulating and executing simulation templates before the development of the Knowledge Composition Methodology. Simulation templates were formulated manually / semi-automatically and modified manually for VTMB problems and for changes in idealization decisions taken by analysts. This made the usage of simulation templates ineffective and costly for multi-disciplinary design optimization problems and for evaluation of system performance in general. However, the execution of simulation templates has benefited from advancements in commercial off-the-shelf object solvers, math solvers, and solution method-specific solvers (such as FEA tools). These solvers have been used successfully to execute simulation templates—solve for the unknown (target) variables from the known (input) variables.

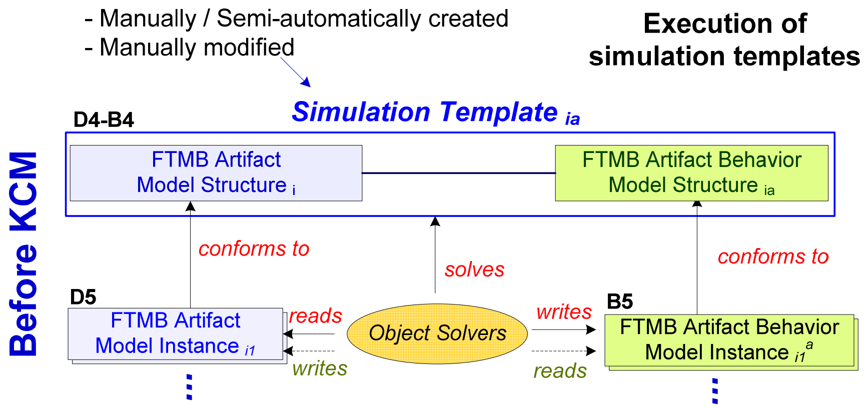

- Manually / Semi-automatically created
- Manually modified
Execution of simulation templates
Simulation Template ia
D4-B4
FTMB Artifact Model Structure i
FTMB Artifact Behavior Model Structure ia
Before KCM
conforms to
solves
conforms to
D5
FTMB Artifact Model Instance i1
reads
writes
Object Solvers
writes
reads
B5
FTMB Artifact Behavior Model Instance i1 a

Figure 10.1: Lack of effective methods to formulate VTMB-related simulation templates before KCM

The Knowledge Composition Methodology (KCM) addresses two critical research gaps in effectively formulating simulation templates—formalizing the knowledge necessary for formulating simulation templates, and providing the Behavior Model Formulation Method to automatically formulate simulation templates for VTMB problems and idealization variations. Figure 10.2 below illustrates the “enhanced” state-of-the-art in formulating and executing advanced simulation templates with the Knowledge Composition Methodology.

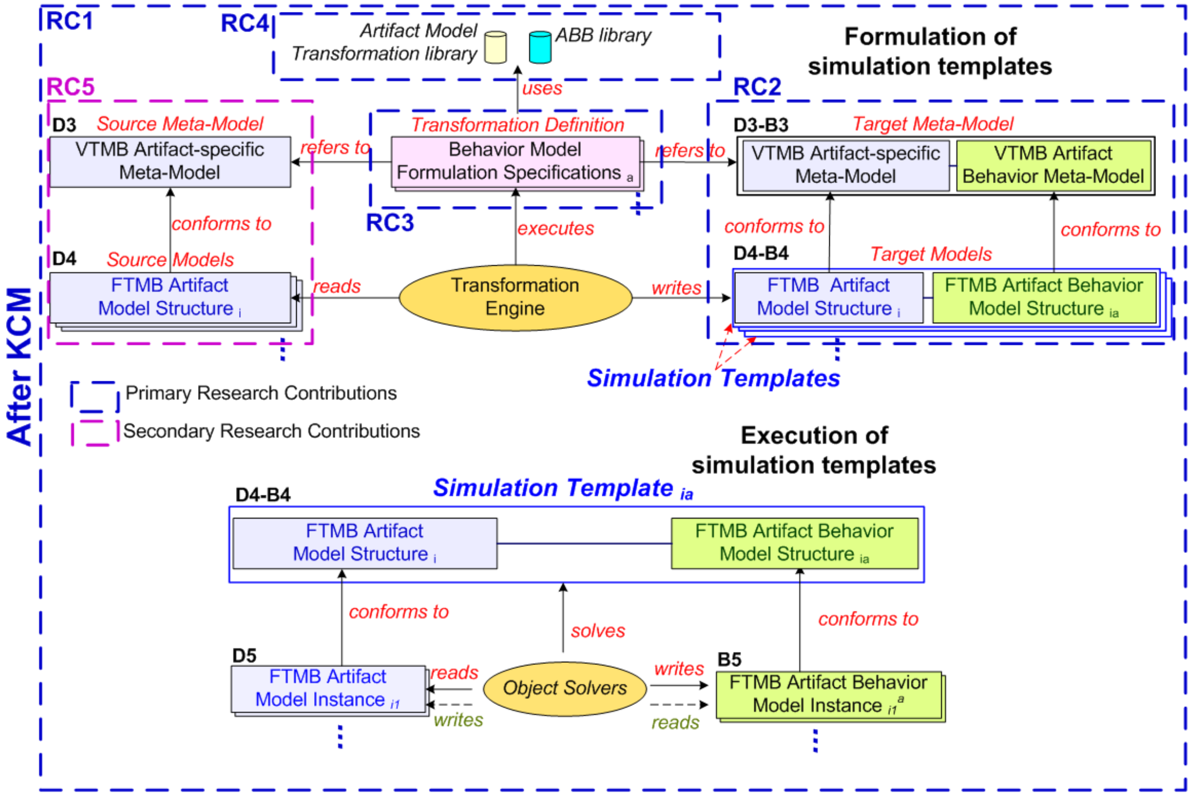


Figure 10.2: KCM enables effective formulation of advanced simulation templates

The specific research contributions (RCs) are summarized below.

Research Contribution 1 (RC1)

The Knowledge Composition Methodology developed in this research provides a mechanism to formulate advanced simulation templates in an effective manner. Simulation templates formulated by the KCM are executable. With the capability to (i) automatically formulate simulation templates for VTMB problems and variations in idealizations and (ii) execute simulation templates, KCM makes the use of simulation template more effective for multi-disciplinary design optimization problems and for evaluating system performance in general. In addition to handling VTMB problems and idealization variations, test results show significant increase in formulation efficiency using KCM versus current methods—90% or greater (on average) reduction in the time required for formulating simulation templates using the KCM versus current methods.

KCM's Behavior Model Formulation Method plays the central role in formulating simulation templates. Founded on graph transformations, the BMFM enables automated composition of simulation templates from reusable building blocks.

This dissertation also defines the concept of Assembly System Topology (AST) and a special type of graph construct and corresponding visualization diagram—an Assembly System Topology diagram—to help characterize VTMB problems and visualize and communicate changes in AST. In addition to formulating simulation templates, KCM provides a fundamental graph transformation-based approach to model formulation for variable topology problems in general, such as from logical/functional system design models to physical system design models (Friedenthal 2006).

Research Contribution 2 (RC2)

KCM provides meta-models and an approach for representing simulation templates. The Core Behavior Model developed in this research is a meta-model for representing artifact behavior models, and fine-grained relationships between behavior models and design models. KCM provides five different abstractions for representing behavior models and simulation templates, depending upon the scope of the artifacts and type of analysis. The ABB Meta-Model developed in this research is a meta-model for representing the building blocks of behavior model structure. Though focused on physics-based behavior of artifacts, the ABB Meta-Model provides generic constructs—four types of knowledge represented in building blocks—that would be used for defining building blocks for other types of behaviors, such as state-based behavior.

Research Contribution 3 (RC3)

KCM's Behavior Model Formulation Specifications (BMFS) provides a mechanism for capturing and representing idealization decisions taken by analysts. These decisions serve as specifications for simulation templates automatically formulated by the BMFM. The specifications, aka Behavior Model Formulation Specifications, are defined at two levels of abstractions—conceptual specifications and computable specifications. The conceptual specifications represent the intent of the idealization decisions and are defined by analysts. The computable specifications, derived from the conceptual specifications, represent the graph transformation process for composing simulation templates. Differentiating conceptual specifications from computable specifications enables analysts to focus on the idealization intent (conceptual specifications) and not on the actual computer code for the transformation process (computable specifications). Apart from representing idealization knowledge as conceptual specifications, this approach makes it easier for analysts to change idealization decisions and automatically re-formulate simulation templates without worrying about updating computer scripts for formulating simulation templates.

Research Contribution 4 (RC4)

KCM also provides graph transformation-based algorithms formalized as reusable graph patterns and graph transformation rules for automatically composing simulation templates from building blocks. These patterns and rules are defined in terms of the KCM meta-models (CPM2_xKCM, CBM, and ABB Meta-Model) and hence are applicable for all specializations of these meta-models. In essence, patterns and rules together provide something similar to an application programming interface (API) for the KCM. Scripts to formulate simulation templates—formalized as graph transformation process—use these pre-defined patterns and rules. In addition, KCM also provides a library of ABBs—building blocks of behavior models and hence simulation templates. KCM's Artifact Model Transformation Library includes all graph patterns and transformation rules, including transformation rules defined specifically for each type of ABB.

Research Contribution 5 (RC5)

KCM extends the Core Product Model (CPM2) to define CPM2_xKCM—a meta-model for representing VTMB artifact design alternatives. KCM provides five abstraction levels of design models to characterize the set of design alternatives represented by each model, and to distinguish models used for defining the formulation of simulation templates versus models used in simulation templates. Abstractions D3, D4, and D5 of design models are of specific importance. Instead of formulating design models for an artifact at two levels (a meta-model and instances), the KCM provides three different abstraction levels (D3, D4, and D5) that serve the following purposes: (1) D3 model used represents all variable topology alternatives of an artifact, and is used for defining specifications for composing simulation templates; (2) D4 model is the source model for formulating simulation templates, and represents a set of design alternatives with equivalent assembly system topologies; and (3) D5 model represents a specific artifact as an instance of D4 model, and is used for creating behavior model instances using a simulation template.

10.2 Recommended Future Work

The following applications and extensions of this research are recommended for the future. These recommendations are divided in two categories: (a) Conceptual extensions—theory-related extensions of the KCM, and (b) Implementation extensions—software development-oriented extensions of the KCM (or KCM Framework).

Conceptual extensions

1. Application of KCM's model transformation approach to variable topology problems in system engineering design and analysis, such as designing the following types of systems: manufacturing systems, real time embedded systems, energy distribution systems, and software systems.
2. Application of the concept of assembly system topology (defined in this research) and graph transformation-based techniques for composing simulation templates to systems with hardware, software, and human components.
3. Addition of new types of ABBs to represent concepts of state-based behavior, such as time and events, activities, and decision nodes. While state machine representation in UML (and SysML) and UML profiles such as MARTE provide a standards-based representation of these concepts, the composition of simulation templates for state-based behavior requires that these concepts be wrapped as ABBs. In addition, hybrid simulation templates composed of physics-based ABBs and state-based ABBs can be used for co-simulation.
4. Representation of dynamic simulation templates to model problems where the assembly system topology of design alternatives change during the solution process. This can be achieved by defining conditions for existence of relationships in a

simulation template. Depending upon the computed values, the relationships may be "disabled" temporarily. As an example, when the shear stresses between two layers in a PCB increases beyond the peel strength, it leads to delamination of layers. Delamination changes the assembly system topology of a PCB and hence interaction relationships between analysis bodies representing delaminated layers would need to be "disabled".

5. Application of KCM's model transformation approach (based on graph transformation) variable topology problems where transformations are performed to generate one aspect of a design model from another aspect, such as from logical design view to physical design view. Implementation of OMG's Model Driven Architecture to systems engineering involves transformations from Platform Independent Models (PIMs) to Platform Specific Models (PSMs) (Friedenthal 2006). Graph transformation-based approach to VTMB problems can provide a foundation for intra-disciplinary transformations.
6. Development of solver managers for open standards-based simulation templates. Such solver managers can solve simulation templates by delegating relationships to a "cloud of solvers" without worrying about the transformations between solver-independent and solver-specific models. This allows the automated execution of different types of relationships—procedural code to math-based constraints—in simulation templates. In addition, depending upon the nature of the relationships, simulation templates (or parts of it) can be executed in multiple directions.
7. Investigation and development of better metrics to characterize model formulation efficiency.

Implementation extensions

1. Application of the KCM's Behavior Model Formulation Method to automatically compose simulation templates for analysis problems in different disciplines, such as thermal analysis, dynamics and vibration analysis, and fluid dynamics.
2. Extension of Behavior Model Formulation Method's graph transformation-based approach to compose simulation templates from simulation templates. As an example for test case family TCF2, simulation templates for thermo-mechanical behavior of

BGA could be composed from existing simulation templates for thermo-mechanical behavior of substrates.

3. Representation and use of decision nodes in simulation templates. Decisions nodes can be represented by extending SysML constraint blocks. When used in simulation templates, decision nodes can be used for verifying if computed values of behavior parameters satisfy requirements.
4. Simulation templates, as composed by the KCM in this dissertation, consist of solution method- and solver-independent formulations of behavior models. The rationale for this was to enable analysts to use multiple solution methods and solvers for the same analysis problems. KCM's model composition approach can be used to formulate solution method-specific and solver-specific behavior models (such as FEA models in ABAQUS) that are associated with the solution- and solver-independent behavior models. Examples of FEA scripts automatically formulated from solution method- and solver-independent behavior models are shown in (Peak, Burkhart et al. 2007). Solution method and solver specifications (such as FEA mesh specifications) would be provided by analysts and will be included in the Behavior Model Formulation Specifications (BMFS). Conceptual specifications in BMFS may include conditions that are checked post-solution, such as mesh refinements based on the results. This use case corresponds to the research in adaptive idealizations by Shephard et al. (Shephard, Beall et al. 2004). Changes in the topology of simulation templates based on solution results would be handled in a similar manner as for dynamic simulation templates described in item 4 (Conceptual extensions) above.

Chapter 11 : CLOSURE

The Knowledge Composition Methodology (KCM) for effective formulation of analysis problems is presented in this dissertation. The representation of analysis problems as simulation templates enhances the reuse of analysis knowledge in formulating behavior models for a large set of design alternatives. However, simulation templates are typically brittle to variations in assembly system topology and idealization decisions taken by analysts. This makes them ineffective for analyzing the performance of design alternatives and for using them in design optimization problems. To characterize the types of changes that require manual updates and "re-wiring" of simulation templates, the concept of assembly system topology has been defined in Chapter 2 of this dissertation. Based on the concept of assembly system topology, this dissertation defines a special class of problems, namely Variable Topology Multi-Body (VTMB) problems where the assembly system topology of design alternatives varies. VTMB problems are defined and illustrated in Chapter 2. In this context, the Knowledge Composition Methodology answers the following primary research question: *How can we improve the effectiveness of the analysis problem formulation process for VTMB problems?* Specifically, KCM addresses the following two key research gaps in existing methods for formulating analysis problems: (a) lack of formalization of the knowledge used by analysts in formulating simulation templates, (b) inability to leverage this knowledge to define model composition methods for formulating simulation templates.

The Knowledge Composition Methodology is presented in details in Part 2 of this dissertation (Chapters 5-9). Based on the research questions and hypotheses presented in Chapter 4, the functional and design specifications of KCM are presented in Chapter 5. The KCM Framework is a computational embodiment of the KCM. It provides a testbed for KCM models and methods. The use cases and components of the KCM Framework are also presented in Chapter 5.

The key functional components of the KCM for formulating simulation templates were presented as follows:

- CPM2_xKCM is an extension of the Core Product Model (Fenves 2004) for the Knowledge Composition Methodology. CPM2_xKCM provides a meta-model for representing VTMB design alternatives. Based on CPM2_xKCM, five levels of abstractions of design models are described with examples in Chapter 6.
- CBM (Core Behavior Model) provides a meta-model for representing behavior models of VTMB artifacts. Based on the CBM, five different levels of abstractions of behavior models are presented in this Chapter 7. Behavior models in the KCM consist of two core components: (a) ABB System—artifact-independent model composed of ABB models, and (b) Context—model that associates an ABB System to artifact design models.
- ABB Meta-Model provides a meta-model for representing analysis building blocks (ABBs). ABBs are units of analysis knowledge that can be reused for formulating a large class of behavior models. Nine different classes of ABBs are defined based on the ABB Meta-Model. Examples of ABBs in each class are also presented. The ABB Meta-Model and ABBs are presented in Chapter 7. Some classes of ABBs defined in this version of the KCM are primarily targeted for physics-based behavior models. For other types of behavior models, such as state-based behavior models, new classes of ABBs can be defined based on the ABB Meta-Model in a similar manner. In contrast to representations of domain theoretic knowledge in existing methods, ABBs in the KCM also embody the model transformations associated with using them in a behavior model.
- Behavior Model Formulation Method (BMFM) is a model transformation approach for automatically composing behavior model structures from ABBs, based on the idealization decisions taken by analysts. The BMFM is presented in details in Chapter 8 of this dissertation. The idealization decisions are represented as selections of ABBs for idealizing VTMB design alternatives and representing the environmental conditions (such as loads and behavior conditions) in which the behavior of design alternatives is to be computed. In addition to specifying ABBs, the conditions for using one ABB versus the other based on properties of design alternatives can also be

represented. The model transformation approach in the Behavior Model Formulation Method is founded on graph transformations. Graph transformations provide a formal approach for model transformations since entity-relationship type of models can be structurally abstracted as graphs. The model transformation approach is four-tiered—graph patterns, graph transformation rules, computable specifications to explicitly schedule the execution of transformation rules, and conceptual specifications to embody the idealization decisions taken by analysts.

The test applications of KCM meta-models and methods, and validation of research hypotheses are presented in Chapter 9. The Behavior Model Formulation Method (implemented in the VIATRA graph transformation framework) is used for automatically generating simulation templates for thermo-mechanical analyses of two families of VTMB design alternatives—multi-stratum printed wiring boards, and multi-component ball grid array chip packages. The simulation templates generated for each test case family are illustrated in details in Chapter 9. Table 9.5 and Table 9.6 summarize results of the three measures of effectives of Behavior Model Formulation Method for the test case families. In addition to handling VTMB variations and idealization variations, the results clearly show a 90% or more (on average) reduction in the time taken to formulate simulation templates using the KCM versus current methods. With the increase in the number of components and interactions, the improvements in formulation efficiency are significant when using KCM's BMFM versus current methods. In contrast to existing methods where variations in idealization decisions may require several hours to update and "re-wire" simulation templates, the time required using BMFM is of the order of minutes (less than a minute for minor variations).

There are two key directions for deploying and extending the current capabilities of the KCM. The first direction concerns the ability to formulate a larger variety of simulation templates for a larger variety of design families; and the second direction concerns the ability to use KCM approach for formulating models for variable topology problems in general.

The application of KCM for analyzing artifact behavior depends on the existence of ABB models to represent domain theoretic concepts used in these analyses. These ABB models can be created as specializations of existing ABB types. Some of the ABB types defined in this dissertation are especially relevant for physics-based behavior. For analyzing other types of artifact behaviors, such as state-based behavior, additional types of ABBs and their specialization need to be defined based on the ABB Meta-Model.

For applying KCM methods for artifact families in different application areas, such as automobile, electronics, and aircrafts, the CPM2_xKCM meta-model can be leveraged to define application-specific meta-models. STEP (ISO 10303) application protocols provide an extensive set of design concepts for some of these application areas. The VTMB artifact models for representing multi-stratum PWBs in this dissertation leverages concepts defined in the STEP AP210 standard for electronics artifacts. In addition, standards such as OMG MARTE (MARTE 2008) provide constructs for representing design and analysis information for real-time embedded systems—composed of both software and hardware components—whose functions are primarily defined in terms of state-based behavior.

Overall, KCM's design meta-model (CPM2_xKCM) can be specialized to define VTMB artifact meta-models for artifact families in different application areas by leveraging concepts defined in standards adopted in that application area. KCM's behavior meta-model (CBM) and ABB Meta-Model can be specialized to define ABBs (and behavior models) for other different types of behaviors.

The second direction to deploy and extend the capabilities of KCM concerns a unique contribution of the KCM—a formal model transformation approach for formulating models for variable topology problems. The graph transformation-based approach can be used for formulating different types of artifact (or system) models where variable topology poses a significant challenge in automatically formulating and adapting models to changes in specifications provided by model authors. Examples of this are plenty in today's system engineering processes, such as creating physical system design models from logical system design models (and vice versa) based on the specifications provided by designers. For instance in this case, a designer specifies the type of physical component to be used for realizing each type of logical component (unit). With variations in number, type, or configuration of logical components, or the specifications provided by designers, KCM's model transformation approach can be used for automatically formulating physical design models.

The Knowledge Composition Methodology achieves its primary objective to make analysis problem formulation a more effective process as compared to the methods and tools representative of the current state-of-the-art. It successfully achieves this objective and in doing so opens a new application area for its model transformation approach as applied to variable topology problems. As opposed to spending costly resources on interoperability of design and analysis models, it is envisioned that the Knowledge Composition Methodology shall provide the foundation to bridge the significant gap between system definition and analysis tools. As a result, the Knowledge Composition Methodology will provide system designers, analysts, and other stakeholders a greater opportunity to focus on the function and the quality of systems.

APPENDICES

Appendix 1 : Description of Basic Concepts

Brief descriptions of the commonly used terms and concepts are presented in this appendix. The intent of this appendix is to describe these terms and concepts in the sense that they are used in this dissertation.

Data are symbols which represent information for processing purposes, based on implicit or explicit interpretation rules. In general, data lacks semantics. Even if the interpretation rules are explicit, they are informally documented (Schenck and Wilson 1994; Giarratano and Riley 1998).

Information is data with formal and explicit semantics. Information can be communicated between two or more partners. Semantics is a key aspect of information because the partners need to have a unique and unambiguous understanding of every piece of information.

Knowledge extends beyond the notion of information by also including relationships between pieces of information. Knowledge is also known as value-added information for the purpose of decision making. Knowledge may be represented in different ways, such as rules, semantic nets, schema, and logic symbols. The collective knowledge pertaining to a given universe-of-discourse may be formalized in different ways, such as taxonomies, thesauri, and ontologies.

A *Model* is a computable approximation of a “thing” for an intended purpose. A model is a surrogate for the actual thing itself and enables us to answer questions about it. The fidelity to which a model approximates a “thing” limits the types of questions that may be answered about that thing. A model that is computable may be interpreted or solved using computer-based methods. In a more generic sense, a model may imply both - a physical model or a computable model, but this research specifically focuses on the latter. The specific thing approximated by a model may be a physical object such as a car or a ship; a process, such as manufacturing or quality control; collection of physical objects or processes; specific characteristic(s) of them; or even a model itself.

There are two key aspects of a model, namely semantics and syntax. Semantics is concerned with the meaning of the thing that a model represents. Syntax is the computer-interpretable form in which the model is formalized.

Per the definition above, in this dissertation the term Model also implies Information Model (Schenck and Wilson 1994) or Knowledge Model.

A *Meta-Model* consists of constructs and rules that are needed to build models in a universe of discourse. A meta-model is also a model and it can have any number of instance models (or instances for brevity). In essence, a meta-model is a "model" of the universe of discourse. Figure A1 illustrates the conceptual relationship between a model, a meta-model, and model instance using SysML (SysML 2007) notation. The core entity is a Model. The terms meta-model and instance denote the relationship between two models such that one describes the constructs and rules necessary to create the other. A model always has a meta-model and a meta-model may have any number of model instances. A model cannot be a meta-model (or instance) of self. For example, a web page is internally represented as an information model written in HTML which confirms to a meta-model defined by World Wide Web Consortium (W3C), specifically W3C HTML DTD (W3C 1999).

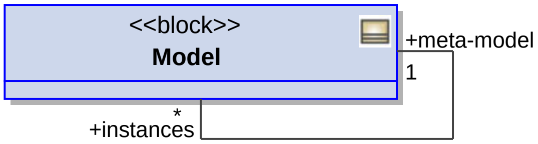


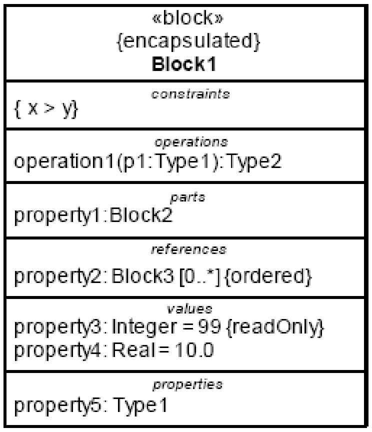
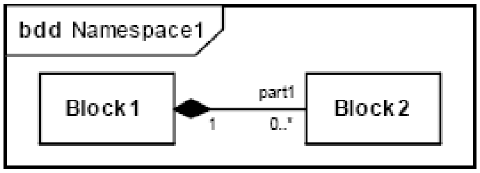
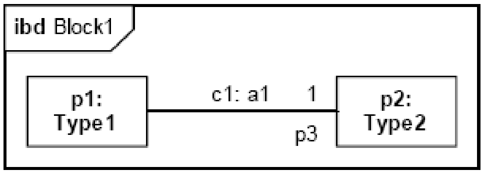
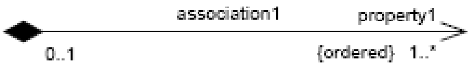
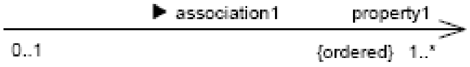
Figure A1: Conceptual relationship between meta-model and instance (using SysML notation)

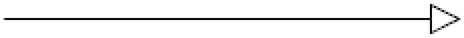
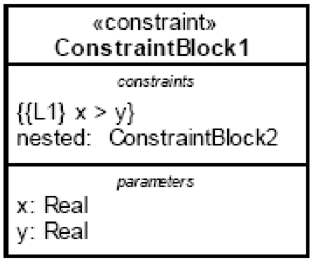
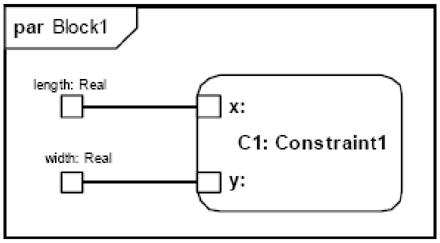
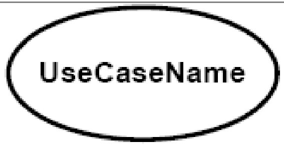
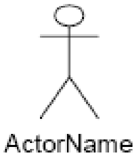
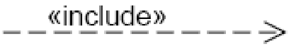
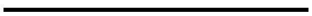
In the context of this dissertation, the terms *meta-model*, *model schema*, and *model structure* imply the same and are used interchangeably. Unless otherwise stated, the term *model* implies *model instance*.

An *Ontology* defines a set of representational primitives to model a universe of discourse. These representational primitives are classes (or sets), attributes (or properties), and relationships (relations between classes) (Gruber 1995; Gruber 2007). As an example, STEP AP210 (ISO 10303-210 2001) is an ontology for describing the design of electro-mechanical products. An ontology is concerned with defining the “semantics” to communicate about a universe of discourse, and not necessarily concerned with organizing and implementing information models of the universe of discourse across one or more databases. In the context of this dissertation, the term Ontology is used interchangeably with meta-model or model structure to denote the semantics of the universe of discourse being represented. An ontology (or meta-model or model structure) is described using a representation language, also known as a modeling language.

Appendix 2 : Systems Modeling Language (SysML) Notation

In this appendix, visual notations of OMG's System Modeling Language (SysML) used in this dissertation are presented. The text and pictures shown in the table below are abstracted from standard definitions of elements in the SysML standard specifications (SysML 2007).

Block A Block is a modular unit that describes the structure of a system or element. It may include both structural and behavioral features, such as properties and operations, that represent the state of the system and behavior that the system may exhibit. ▪ Block properties typed by blocks using part associations are known as *part properties*. ▪ Block properties typed by blocks using reference associations are known as *reference properties*. ▪ Block properties typed by primitive values (such as integer and string types) are known as *value properties*. ▪ Block properties typed by constraint blocks are known as *constraint properties*. The difference between part properties and reference properties is that block instances associated with a parent block instance as part properties are owned by the parent block.	 «block» {encapsulated} Block1 constraints { x > y} operations operation1(p1:Type1):Type2 parts property1:Block2 references property2: Block3 [0..*] {ordered} values property3: Integer = 99 {readOnly} property4: Real = 10.0 properties property5: Type1
Block Definition Diagram (BDD) A Block Definition Diagram is a view of the system model, and it shows the properties of blocks and the relationships between blocks using part associations, reference associations, and generalizations.	
Internal Block Diagram (IDB) An Internal Block Diagram shows the internal structure of a block. It shows the properties of a block and the connections between these properties.	
Part Association Part Associations are used for relating a parent block and a child block. The black diamond connects to the parent block. The other end connects to the child block. A part association means that the parent block has a property of type of the child block. When a model instantiated, a parent block instance owns the child block instance(s).	
Reference Association In contrast to part associations, when blocks related by a reference association are instantiated, the referring block instance does not own the referred block instance.	

Generalization Generalization is used for representing generalization relationship between concepts represented by blocks. The head of the arrow connects to the parent block and the tail of the arrow connects to a child block. A generalization relationship implies that the child block represents a concept that is a specialization of the concept represented by the parent block.	
Constraint Block Constraint blocks are used for representing reusable mathematical relationships, including domain concepts such as the definition of Newton's Second Law (F=m*a, or F=m*dv/dt). Constraint blocks primarily consist of constraint parameters and constraint specifications that define the mathematical relationships between constraint parameters. A constraint block may also contain other constraint blocks.	 «constraint» ConstraintBlock1 constraints {{L1} x > y} nested: ConstraintBlock2 parameters x: Real y: Real
Parametric Diagram A Parametric Diagram includes usages of constraint blocks to constraint the properties of a block. Constraint blocks used in the context of a block (as constraint properties) are denoted as rectangles with rounded corners.	 par Block1 length: Real width: Real x: C1: Constraint1 y:
Use Case Use case of a system represents the functionality of the system that is achieved when actors interact with the system.	
Actor Actors are users of a system	
Include Include relationship is defined between a base use case and the included use case. This relationship denotes that the included use case is performed as part of realizing the base use case.	
Association (Communication Path) Actors are associated with use cases via a communication path (association). The communication path represents the interaction between an actor and a system when the specific use cases are being realized.	

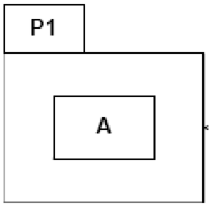
Package A Package defines a namespace for model elements, and may contain other packages.	

Table A2: Summary of SysML modeling elements used in this dissertation

Appendix 3 : KCM's Generic Properties Meta-Model

Figure A3 illustrates KCM's Generic Properties Meta-Model. The constructs defined in this meta-model are used in other KCM meta-models and models. Specifically, this meta-model defines specializations of the CoreProperty entity defined in CPM2 (and included in CPM2_xKCM Meta-Model). In CPM2_xKCM, CoreProperty is the basic abstract block used for representing properties of an artifact, such as shape and material. In the Generic Properties Meta-Model, the CoreProperty is specialized to define CoreBehaviorProperty as the base block for representing a basic set of concepts used for defining the behavior of artifacts. The concepts shown in this version of the Generic Properties Meta-Model are targeted for the test applications and models described in this dissertation—mostly physics-based behavior models with emphasis on thermal and mechanical analysis. The Generic Properties Meta-Model is intended to be extensible as new types of ABBs and analysis concepts are added to the KCM.

Two primary types of specializations to the CoreProperty concept are developed in the Generic Properties Meta-Model. The first type specialization concerns the specializations to the concept of Shape (renamed from Geometry in CPM2 to Shape in CPM2_xKCM). Shape is the parent entity for defining the geometric shape of all abstractions of artifacts and features. It is also used as the Shape ABB in the definition of analysis bodies and analysis body systems. In general, KCM shall leverage STEP Part 42 to extend the representation of geometric shapes. However, for demonstrating the test applications of KCM, some basic specializations of Shape are developed here. As shown in Figure A3, one dimensional (point), two dimensional (such as lines and arcs), and three dimensional shapes (Sphere and Cuboid) are defined as specializations of the Shape block. The Shape_Representation_2D and Shape_Representation_3D blocks are parent blocks for the representation of two dimensional and three dimensional shapes respectively.

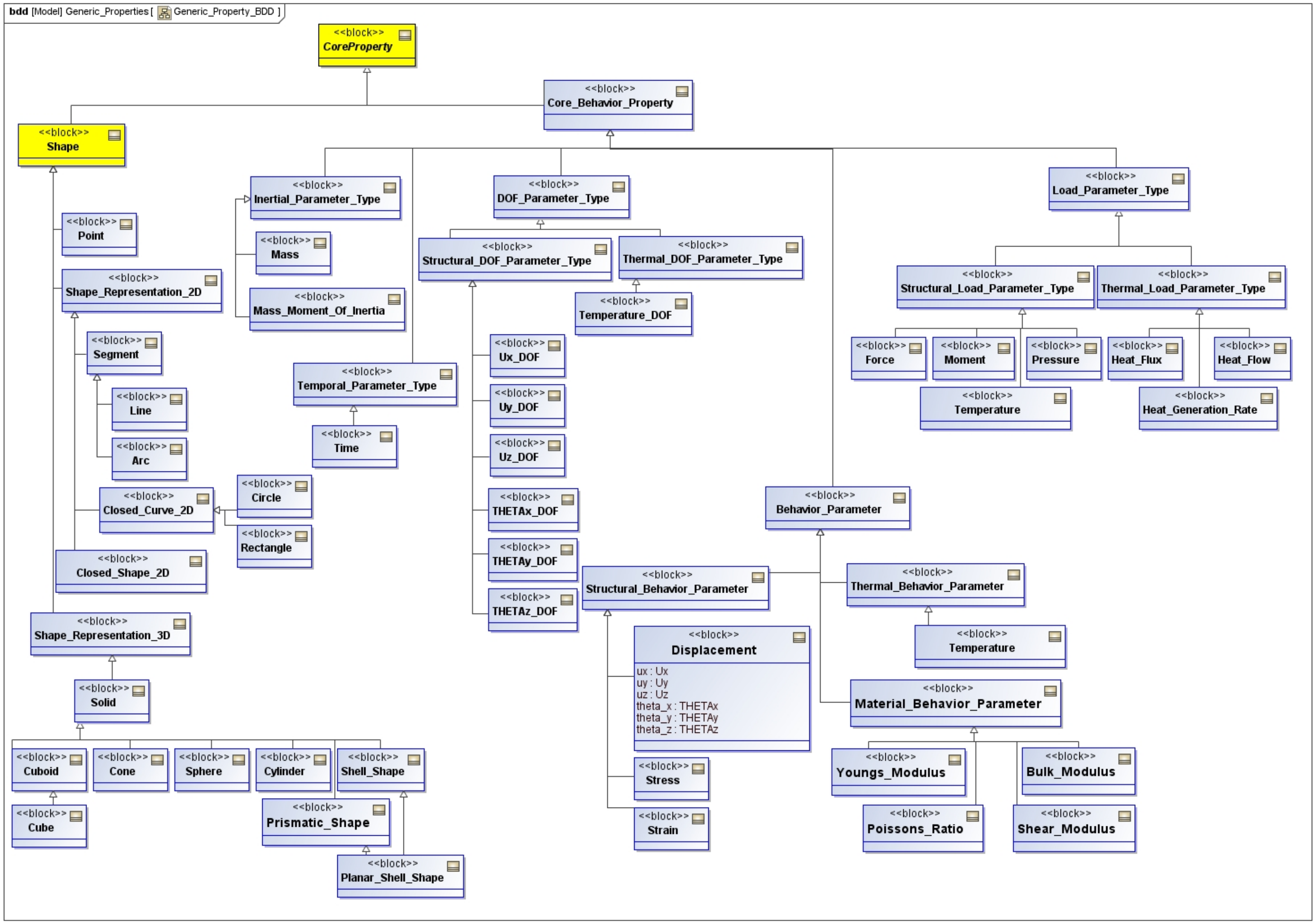


Figure A3: Generic Properties Meta-Model

The second type of specialization to the CoreProperty block is the CoreBehaviorProperty block as the parent block for representing parameters used in defining ABBs. The parameters represented in the Generic Properties Meta-Model represent the following aspects of the concepts represented by ABBs:

- definition of the dimensionality and units for representing the concept
- definition of the type of quantity used for representing the concept, such as scalars, vectors, or tensors
- definition of symbols used for denoting the concepts, such as F for force

In general, the representation of parameters is equivalent to the representation of specialized data types with symbolic notation.

The CoreBehaviorProperty block is specialized into five blocks—each representing a type of parameter—as defined below. Note that these types and the specializations within each type are based on the parameters required for demonstrating the KCM using specific test cases. Additional types of parameters and their specializations must be defined to make this meta-model useful for representing ABBs in general.

- Interial_Parameter_Type block is used for representing the inertial parameters of an artifact, such as mass and moment of inertia. These parameters are shown as specializations of the Inertial_Parameter_Type block.
- Temporal_Parameter_Type block is used for representing time and related temporal parameters that are useful in representing the dynamic behavior of artifacts. KCM shall leverage other standards such as OMG MARTE (MARTE 2008) that extensively define these temporal concepts.
- DOF_Parameter_Type block is used for representing degrees-of-freedom (DOFs) parameters associated with different types of behaviors of an artifact. The DOF_Parameter_Type block can be specialized for representing DOFs for a specific type of behavior. For example, Structural_DOF_Parameter_Type block and Thermal_DOF_Parameter_Type block represent the DOF parameters for structural and thermal behavior respectively.
- Behavior_Parameter block is used for representing behavior parameters for different analysis disciplines. The Behavior_Parameter block is specialized for each type of

analysis discipline. For example, the Behavior_Parameter block is specialized as Structural_Behavior_Parameter and Thermal_Behavior_Parameter block for representing behavior parameters for structural and thermal behavior of artifacts respectively. Additionally, the Material_Behavior_Parameter block represents the parameters used for defining the behavior of materials (constituting artifacts).

- Load_Parameter_Type block is the base block for representing parameters used for characterizing loads. It is specialized into Structural_Load_Parameter_Type and Thermal_Load_Parameter_Type block for representing structural and thermal load parameters respectively. Parameters used for representing force, moment, and pressure are examples of structural load parameters; and parameters used for representing heat generation rate and heat flux are examples of thermal load parameters.

Note that parameters defined in the Generic Properties Meta-Model are not the representation of the concepts themselves but only the definition of parameters used for denoting those concepts. For example, the force parameter is not the definition of force. The Force ABB (type of Load ABB) is used for representing the concept of force.

REFERENCES

Addanki, S., Cremonini, R. and Penberthy, S.J. (1991). "Graphs of models." *Artificial Intelligence* **51**(1-3, Special issue: Qualitative reasoning about physical systems II): 145-177.

Andries, M., Engels, G., Habel, A., Hoffmann, B., Kreowski, H.-J., Kuske, S., Plump, D., Schürr, A. and Taentzer, G. (1999). "Graph transformation for specification and programming." *Science of Computer Programming* **34**(1): 1-54.

Arabshahi, S., Barton, D.C. and Shaw, N.K. (1991). "Towards integrated design and analysis." *Finite Elements in Analysis and Design* **9**(4): 271-291.

Arabshahi, S., Barton, D.C. and Shaw, N.K. (1993). "Steps towards CAD-FEA integration." *Engineering with Computers* **9**(1): 17-26.

Armstrong, C.G., D.J. Robinson, R.M. McKeag, T.S. Li, S.J. Bridgett, R.J. Donaghy, C.A. McGleenan (1995). *Medials for Meshing and More*. 4th International Meshing Roundtable, Sandia National Laboratories, Albuquerque, NM October 1995.

Atkinson, C. and Kuhne, T. (2001). *The Essence of Multilevel Metamodeling*. 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools Toronto, Canada. October 1-5, 2001, Springer-Verlag, London, UK.

Bajaj, M. (2006). "A Composable Knowledge Methodology for Efficient Analysis Problem Formulation in Simulation-based Design," PhD Proposal thesis, Mechanical Engineering, Georgia Institute of Technology, Atlanta

Bajaj, M., Peak, R., Zwemer, D., Thurman, T., Klein, L., Liutkus, G., Brady, K., Messina, J. and Dickerson, M. (2006). *Automating Thermo-Mechanical Warpage Estimation of PCBs/PCAs Using a Design-Analysis Integration Framework*. Mentor U2U, San Jose, CA, USA May 3-5, 2006.

Belaziz, M., Bouras, A. and Brun, J.M. (2000). "Morphological analysis for product design ". *Computer-Aided Design* **32**(5-6): 377-388.

Calvanese, D., Lenzerini, M. and Nardi, D. (1998). Description Logics for Conceptual Data Modeling. *Logics for Databases and Information Systems*. J. Chomicki and G. Saake, Springer. *436***:** 229-264.

Chen, P.P. (1976). "The Entity-Relationship Model: Toward a Unified View of Data." *ACM Transactions on Database Systems* **1**(1): 9-36.

Chen, Y., Lee, J. and Eskandarian, A. (2006). *Meshless Methods in Solid Mechanics,* Springer.

Czarnecki, K. and Helsen, S. (2006). "Feature-based survey of model transformation approaches." *IBM Systems Journal* **45**(3): 621-645.

"SIMULIA (Abaqus FEA) Version 6.8"(2006), Dassault Systemes. Retrieved June 9, 2008, from http://www.simulia.com/products/abaqus_fea.html.

de Kleer, J. (1992). Qualitative Physics. *Encyclopedia of Artificial Intelligence*, John Wiley & Sons.

de Kleer, J. and Brown, J.S. (1984). "A Qualitative Physics Based on Confluences." *Artificial Intelligence* **24**(1-3): 7-83.

"DFXpert", DFXpert (SFM Technology Inc.). Retrieved April 13, 2008, from http://sfmtech.com/products.htm.

Diaz-Calderon, A., Paredis, C.J.J. and Khosla, P.K. (2000). *Organization and selection of reconfigurable models*. Proceedings of WSC 2000, Winter Simulation Conference, Orlando, FL, USA 10-13 Dec., IEEE; Piscataway, NJ, USA.

Donaghy, R.J., W. McCune, S.J. Bridgett, C.G. Armstrong, D.J. Robinson, R.M. McKeag (1996). *Dimensional Reduction of Analysis Models*. 5th International Meshing Roundtable, Sandia National Laboratories, Albuquerque, NM.

Ehrig, H., Engels, G., Kreowski, H.-J. and Rozenberg, G. (1999). *Handbook of Graph Grammars and Computing by Graph Transformation: Applications, Languages and Tools (Volume 2)*, World Scientific Publishing Company.

Engels, G. and Heckel, R. (2000). "Graph Transformation and Visual Modeling Techniques." *Bulletin of the European Association for Theoretical Computer Science (EATCS)* **71**.

"iSIGHT"(2007), Engineous Software. Retrieved October 19, 2008, from http://www.engineous.com/iSIGHT.cfm.

Falkenhainer, B. and Forbus, K.D. (1991). "Compositional modeling: finding the right model for the job." *Artificial Intelligence* **51**: 95-143.

Fenves, S.J. (2004). *CPM 2: A Revised Core Product Model for Representing Design Information (NISTIR 7185),* National Institute of Standards and Technology, NISTIR 7185, Fenves, S.J., Choi, Y., Gurumoorthy, B., Mocko, G. and Sriram, R.D. (2003). *Master Product Model for the Support of Tighter Integration of Spatial and Functional Design*, National Institute of Standards and Technology, NISTIR 7004,

Finn, D.P. (1993). "A Physical Modeling Assistant For The Preliminary Stages Of Finite Element Analysis." *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **7**(4): 275-286.

Fishwick, P.A. (1995). *Simulation model design*. Winter Simulation Conference, Arlington, Virginia, USA December 3-6, 1995.

Friedenthal, S.A. (2006). *MDA For Systems Engineering Concepts*. Systems Engineering Domain Special Interest Group (SE DSIG) Meeting, St. Louis, MO, USA April 25, 2006.

Gamma, E., Johnson, R., Helm, R. and Vlissides, J.M. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. 1, Addison-Wesley.

Gere, J.M. and Timoshenko, S.P. (1997). *Mechanics of Materials*, PWS Publishing Company.

Gero, J.S. (1990). "Design Prototypes: a knowledge representation schema for design." *AI Magazine* **11**(4): 26-36.

Giarratano, J.C. and Riley, G.D. (1998). *Expert Systems: Principles and Programming*. 3rd, Brooks/Cole.

Gordon, S. (2001). *An Analyst's View: STEP-Enabled CAD-CAE Integration*. NASA's STEP for Aerospace Workshop, JPL, Pasadena, CA Jan 17, 2001.

Gross, J.L. and Yellen, J. (2003). *Handbook of Graph Theory*. Edition 1, CRC Press LLC.

Grosse, I.R., Milton-Benoit, J.M. and Wileden, J.C. (2005). "Ontologies for supporting engineering analysis models." *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM)* **19**(1): 1-18.

"Ontology"(2007), Gruber, T.   Retrieved Mar 23, 2008, 2008, from http://tomgruber.org/writing/ontology-definition-2007.htm.

Gruber, T.R. (1992). *Model Formulation as a Problem Solving Task: Computer-assisted Engineering Modeling*, Knowledge Systems Laboratory, Computer Science Department, Stanford University, KSL 92-57, http://www-ksl.stanford.edu/KSL_Abstracts/KSL-92-57.html

Gruber, T.R. (1995). "Toward principles for the design of ontologies used for knowledge sharing." *International Journal of Human-Computer Studies* **43**(5-6): 907-928.

Hoffman, C.M. and Joan-Arinyo, R. (1998). "CAD and the product master model." *Computer-Aided Design* **30**(11): 905-918.

"AP209 ARM Overview"(2001), Hunten, K.   Retrieved Jun 20, 2006, from http://pdesinc.aticorp.org/graphics/aps/ap209_tutorial.ppt.

"IDA-STEP v4.0"(2008), IDA-STEP (LKSoft).   Retrieved April 13, 2008, from http://www.ida-step.net/.

"STEP Part 11, Description method: The EXPRESS language reference manual "(2001), ISO 10303-11.   Retrieved Jun 20, 2006, from http://www.tc184-sc4.org/SC4_Open/SC4 Legacy Products (2001-08)/STEP_(10303)/1-99/documentation.cfm.

"STEP Part 42, edition 2, Integrated generic resource: Geometric and topological representation"(2000), ISO 10303-42.   Retrieved April 12, 2008, from http://www.tc184-sc4.org/SC4_Open/SC4 Legacy Products (2001-08)/STEP_(10303)/1-99/documentation.cfm.

"STEP Part 104, Integrated application resource: Finite element analysis "(2000), ISO 10303-104.   Retrieved Jun 20, 2006, from http://www.tc184-sc4.org/SC4_Open/SC4_Work_Products_Documents/STEP_(10303)/100-199/.

"STEP Part 203, Application protocol: Configuration controlled 3D designs of mechanical parts and assemblies "(2000), ISO 10303-203.   Retrieved Jun 20, 2006, from http://www.tc184-sc4.org/SC4_Open/SC4 Legacy Products (2001-08)/STEP_(10303)/200-299/documentation.cfm.

"STEP Part 209, Application protocol: Composite and metallic structural analysis and related design"(2001), ISO 10303-209.   Retrieved Jun 20, 2006, from http://www.tc184-sc4.org/SC4_Open/SC4 Legacy Products (2001-08)/STEP_(10303)/200-299/documentation.cfm.

"STEP Part 210, Application protocol: Electronic assembly, interconnect, and packaging design"(2001), ISO 10303-210.   Retrieved Jun 20, 2006, from http://www.tc184-sc4.org/SC4_Open/SC4 Legacy Products (2001-08)/STEP_(10303)/200-299/documentation.cfm.

"STEP Part 214, Application protocol: Core data for automotive mechanical design processes"(2003), ISO 10303-214.   Retrieved April 12, 2008, from http://www.tc184-sc4.org/SC4_Open/SC4 Legacy Products (2001-08)/STEP_(10303)/200-299/documentation.cfm.

"STEP Part 215, Application protocol: Ship arrangement"(2001), ISO 10303-215. Retrieved April 12, 2008, from http://www.tc184-sc4.org/SC4_Open/SC4 Legacy Products (2001-08)/STEP_(10303)/200-299/documentation.cfm.

"STEP Part 216, Application protocol: Ship moulded forms"(2000), ISO 10303-216. Retrieved April 12, 2008, from http://www.tc184-sc4.org/SC4_Open/SC4 Legacy Products (2001-08)/STEP_(10303)/200-299/documentation.cfm.

"STEP Part 218, Application protocol: Ship structures"(2000), ISO 10303-218. Retrieved April 12, 2008, from http://www.tc184-sc4.org/SC4_Open/SC4 Legacy Products (2001-08)/STEP_(10303)/200-299/documentation.cfm.

Konigs, A. (2005). *Model Transformation with Triple Graph Grammars*
Model Transformations in Practice Workshop - MoDELS 2005 Conference October 3, 2005.

Krauthammer, T. and Ventsel, E. (2001). *Thin Plates & Shells: Theory, Analysis, & Applications,* New York, Marcel Dekker, Inc.

Law, A.M. and Kelton, W.D. (2000). *Simulation Modeling and Analysis*. 3rd edition, McGraw-Hill.

Li, X. (1991). Quality time-What's so bad about rule-based programming? *IEEE Software*. *8***:** 103,105.

Ling, R., Steinberg, L. and Jaluria, Y. (1993). "MSG: A Computer System for Automated Modeling of Heat Transfer ". *Journal of Artificial Intelligence in Engineering Design, Analysis and Manufacturing* **7** (4): 287-300. .

MARTE (2008). *OMG Modeling and Analysis of Real-time and Embedded systems (MARTE) -- UML Profile for MARTE (Beta 2),* http://www.omgmarte.org/

Mortenson, M.E. (1997). *Geometric Modeling*. 2nd edition, John Wiley & Sons.

Mullins, S. and Rinderle, J.R. (1991). "Grammatical Approaches to Engineering Design, Part 1: An Introduction and Commentary." *Research in Engineering Design* **2**(33): 121-135.

NSF (2006). *Simulation-Based Engineering Science,* National Science Foundation, http://www.nsf.gov/pubs/reports/sbes_final_report.pdf

"Design Simulation Environment", NX CAE (Siemens PLM). Retrieved November 15, 2008, from http://www.plm.automation.siemens.com/en_us/products/nx/simulation/index.shtml.

Pahl, G. and Beitz, W. (1996). *Engineering design: A Systematic Approach*. 2nd, London, U.K., Springer-Verlag.

Paredis, C.J.J., Diaz-Calderon, A., Sinha, R. and Khosla, P.K. (2001). "Composable Models for Simulation-Based Design." *Engineering with Computers* **17**(2): 112-128.

"PCB Layer Stack Editor"(2008), PCB Layer Stack Editor (LKSoft). Retrieved April 21, 2008, from http://www.ida-step.net/components/editors/pcb-layer-stack.

Peak, R., Fulton, R., Chandrasekhar, A., Cimtalay, S., Hale, M.A., Koo, D., Ma, L., Scholand, A.J., Tamburini, D.R. and Wilson, M.W. (1999). *Design-Analysis Associativity Technology for PSI, Phase I Report: Pilot Demonstration of STEP-based Stress Templates*, Georgia Institute of Technology, http://www.eislab.gatech.edu/pubs/reports/boeing-psi-1998/

Peak, R., Wilson, M., Kim, I., Udoyen, N., Bajaj, M., Mocko, G., Liutkus, G., Klein, L. and Dickerson, M. (2002). *Creating Gap-Filling Applications Using STEP Express, XML, and SVG-based Smart Figures - An Avionics Example*. NASA-ESA Workshop on Aerospace Product Data Exchange, ESA/ESTEC, Noordwijk (ZH), The Netherlands April 9-12, 2002.

Peak, R.S. (1993). "Product Model-Based Analytical Models (PBAMs): A New Representation of Engineering Analysis Models," PhD thesis, Mechanical Engineering, The Georgia Institute of Technology, Atlanta, GA, USA

Peak, R.S. (2003). *Characterizing Fine-Grained Associativity Gaps: A Preliminary Study of CAD-CAE Model Interoperability*. ASME International DETC / CIE, Chicago Sep 2-6, 2003.

Peak, R.S., Burkhart, R.M., Friedenthal, S.A., Wilson, M.W., Bajaj, M. and Kim, I. (2007). *Simulation-Based Design Using SysML Part 1: A Parametrics Primer*. The Seventeenth International Symposium of the International Council on Systems Engineering, San Diego, California, USA June 24 -28, 2007.

Peak, R.S., Burkhart, R.M., Friedenthal, S.A., Wilson, M.W., Bajaj, M. and Kim, I. (2007). *Simulation-Based Design Using SysML Part 2: Celebrating Diversity by Example*. The Seventeenth International Symposium of the International Council on Systems Engineering, San Diego, California, USA June 24 -28, 2007.

Peak, R.S. and Fulton, R.E. (1994). *A Multi-Representation Approach to CAD/CAE Integration: Research Overview*,

Peak, R.S., Fulton, R.E., Nishigaki, I. and Okamoto, N. (1998). "Integrating engineering design and analysis using a multi-representation approach." *Engineering with Computers* **14**(2): 93-114.

Peak, R.S., Paredis, C.J.J. and Tamburini, D.R. (2005). *The Composable Object (COB) Knowledge Representation: Enabling Advanced Collaborative Engineering Environments (CEEs), COB Requirements & Objectives (v1.0)*, The Georgia Institute of

Technology, http://www.eislab.gatech.edu/projects/nasa-ngcobs/COB_Requirements_v1.0.pdf

Peak, R.S., Scholand, A.J., Tamburini, D.R. and Fulton, R.E. (1999). "Towards the Routinization of Engineering Analysis to Support Product Design." *Invited Paper for Special Issue: Advanced Product Data Management Supporting Product Life-Cycle Activities, International Journal of Computer Applications in Technology* **12**(1): 1-15.

"ModelCenter"(2007), Phoenix Integration. Retrieved June 9, 2008, from http://www.phoenix-int.com/products/modelcenter.php.

Pratt, M.J. (1995). *Virtual Prototypes and Product Models in Mechanical Engineering*, National Institute of Standards and Technology, NISTIR 5650, www.citeseer.ist.psu.edu/article/pratt95virtual.html

"RacerPro"(1997), RacerPro. Retrieved April 12, 2008, from http://www.sts.tu-harburg.de/~r.f.moeller/racer/.

Rachuri, S., Han, Y.-H., Foufou, S., Feng, S.C., Roy, U., Wang, F., Sriram, R.D. and Lyons, K.W. (2006). "A Model for Capturing Product Assembly Information." *Journal of Computing and Information Science in Engineering* **6**(1): 11-21.

Reddy, J.N. (1993). *An Introduction to the Finite Element Method*. 2nd Edition, USA, McGraw-Hill Book Company.

Robinson, S., Nance, R.E., Paul, R.J., Pidd, M. and Taylor, S.J.E. (2004). "Simulation model reuse: definitions, benefits and obstacles." *Simulation Modelling Practice and Theory* **12**(7-8): 479–494.

Rumbaugh, J., Jacobson, I. and Booch, G. (2004). *The Unified Modeling Language Reference Manual*, Pearson Education.

Schenck, D.A. and Wilson, P.R. (1994). *Information Modeling: The EXPRESS Way*, New York, NY, Oxford University Press.

Sellgren, U. (2003). *Architecting Models of Technical Systems for Non-Routine Simulations*. International Conference on Engineering Design (ICED), Stockholm, Sweden Aug 19-21, 2003.

Shephard, M.S., Beall, M.W., O'Bara, R.M. and Webster, B.E. (2004). "Towards simulation-based design." *Finite Elements in Analysis and Design* **40**(12, Special Issue: The Fifteenth Annual Robert J. Melosh Competition): 1575-1598.

"Simulation Application Suite"(2006), Simmetrix Inc. Retrieved Jun 20, 2006, from http://simmetrix.com/products/SimulationApplicationSuite/main.html.

"Associative Interfaces for CAD Systems"(2008), Simulia ABAQUS. Retrieved April 12, 2008, 2008, from http://www.simulia.com/products/associative_interfaces.html?SolidW?im04#SolWork.

Sinha, R., Paredis, C.J.J. and Khosla, P.K. (2000). *Integration of mechanical CAD and behavioral modeling*. Proceedings 2000 IEEE/ACM International Workshop on Behavioral Modeling and Simulation, Orlando, FL, USA 19-20 Oct., IEEE Computer Society; Los Alamitos, CA, USA.

"OMG Systems Modeling Language (OMG SysML)"(2006), SysML. Retrieved Jun 20, 2006, from http://www.omgsysml.org/.

SysML (2007). *OMG Systems Modeling Language (OMG SysML) v1.0 -- OMG Available Specification*, http://www.omgsysml.org/

Tamburini, D.R. (1999). "The Analyzable Product Model Representation to Support Design-Analysis Integration," PhD thesis, Georgia Institute of Technology, Mechanical Engineering

Taylor and Francis (Inverse Problems in Science and Engineering) (2008). *Inverse Problems in Science and Engineering*

Timoshenko, S.P. and Goodier, J.N. (1970). *Theory of Elasticity*. 3rd Edition, Singapore, McGraw-Hill Book Company.

Turkiyyah, G.M. and Fenves, S.J. (1996). "Knowledge-based assistance for finite-element modeling." *IEEE Expert: Intelligent Systems and Their Applications* **11**(3): 23-32.

"OMG Unified Modeling Language (UML)"(2004), UML 2. Retrieved Feb 12, 2007, from http://www.uml.org/.

"OMG Unified Modeling Language (UML), Superstructure, V2.1.2"(2007), UML 2. Retrieved May 19, 2008, from http://www.uml.org/.

"OMG UML 2.0 Object Constraint Language"(2004), UML 2 OCL. Retrieved Feb 12, 2007, from http://www.uml.org/ http://www.omg.org/technology/documents/modeling_spec_catalog.htm#OCL.

Valiente, G. and Martinez, C. (1997). *An algorithm for graph pattern-matching*. Fourth South American Workshop on String Processing, Valparaso, Chile November 14-15, Carleton University Press.

Varro, D. and Balogh, A. (2007). "The model transformation language of the VIATRA2 framework." *Science of Computer Programming* **68**(3): 214-234.

Varro, D., Varro, G. and Pataricza, A. (2002). "Designing the automatic transformation of visual languages." *Science of Computer Programming* **44**(2): 205-227.

Ventsel, E. and Krauthammer, T. (2001). *Thin Plates and Shells - Theory, Analysis, and Applications,* New York, Marcel Dekker, Inc.

VIATRA (2007). VIATRA2 sub-project and specifications, Eclipse Generative Model Transformer project (GMT).

"W3C HTML Specifications"(1999), W3C. Retrieved Mar 23, 2008, 2008, from http://www.w3.org/TR/REC-html40/.

Wilson, M. (2000). "The Constrained Object Representation for Engineering Analysis Integration. ," MS thesis, Mechanical Engineering, The Georgia Institute of Technology, Atlanta, GA, USA

Wilson, M., Peak, R.S. and Fulton, R.E. (2001). *Enhancing Engineering Design and Analysis Interoperability - Part 1: Constrained Objects*. First MIT Conference Computational Fluid and Structural Mechanics (CFSM), Cambridge, Massachusetts, USA Jun 12-15, 2001.

"Mathematica"(2008), Wolfram Mathematica. Retrieved April 1, 2008, from http://www.wolfram.com/products/mathematica/index.html.

"XaiTools"(1999), XaiTools (Georgia Tech). Retrieved April 21, 2008, 2008, from http://www.eislab.gatech.edu/tools/xaitools/.

Yip, K.M.-k. (1993). *Model Simplification by Asymptotic Order of Magnitude Reasoning*. 11th National Conference on Artificial Intelligence, Washington, DC, USA July 11-15, 1993, The AAAI Press.

Zeng, S. (2004). "Knowledge-based FEA Modeling Method for Highly Coupled Variable Topology Multi-body Problems," PhD thesis, Mechanical Engineering, Georgia Institute of Technology, Atlanta

Zeng, S., Peak, R.S., Xiao, A. and Sitaraman, S. (2008). "ZAP: a knowledge-based FEA modeling method for highly coupled variable topology multi-body problems." *Engineering with Computers* **24**(4): 359-381.

"System Designer - Design entry tools and utilities", Zuken CR-5000 PSpice & HSpice. Retrieved April 13, 2008, from http://www.zuken.com/cr-5000/system_designer.asp.

Zwemer, D., Bajaj, M., Peak, R., Thurman, T., Brady, K., McCarron, S., Spradling, A., Dickerson, M., Klein, L., Liutkus, G. and Messina, J. (2004). *PWB Warpage Analysis and Verification Using an AP210 Standards-based Engineering Framework and Shadow Moiré*. IEEE EuroSimE, Brussels, Belgium May 10-12, 2004.