Citation: RS Peak, RM Burkhart, SA Friedenthal, MW Wilson, M Bajaj, I Kim (2007) Simulation-Based Design Using SysML—Part 2: Celebrating Diversity by Example. INCOSE Intl. Symposium, San Diego.

Web Release Version: May 4, 2007. Includes minor corrections beyond the original publication.

Simulation-Based Design Using SysML Part 2: Celebrating Diversity by Example

Russell S. Peak[1,†], Roger M. Burkhart[2], Sanford A. Friedenthal[3], Miyako W. Wilson[1], Manas Bajaj[1], Injoong Kim[1]

[1] Georgia Institute of Technology
http://www.pslm.gatech.edu/
http://eislab.gatech.edu/

[2] Deere & Company
http://www.johndeere.com/
[3] Lockheed Martin Corp.
http://www.lockheedmartin.com/

Abstract. These two companion papers present foundational principles of parametrics in OMG SysML™ and their application to simulation-based design. Parametrics capabilities have been included in SysML to support integrating engineering analysis with system requirements, behavior, and structure models. This Part 2 paper walks through SysML models for a benchmark tutorial on analysis templates utilizing an airframe system component called a flap linkage. This example highlights how engineering analysis models, such as stress models, are captured in SysML, and then executed by external tools including math solvers and finite element analysis solvers.

We summarize the multi-representation architecture (MRA) method and how its simulation knowledge patterns support computing environments having a diversity of analysis fidelities, physical behaviors, solution methods, and CAD/CAE tools. SysML and composable object (COB) techniques described in Part 1 together provide the MRA with graphical modeling languages, executable parametrics, and reusable, modular, multi-directional capabilities.

We also demonstrate additional SysML modeling concepts, including packages, building block libraries, and requirements-verification-simulation interrelationships. Results indicate that SysML offers significant promise as a unifying language for a variety of models—from top-level system models to discipline-specific leaf-level models.

Keywords. Simulation-based design (SBD), engineering design and analysis, simulation template, CAD-CAE interoperability, finite element analysis (FEA), multi-representation architecture (MRA), SysML parametrics, composable object (COB), multi-fidelity, multi-directional.

1 Background

Part 1 [Peak *et al.* 2007] is prerequisite reading that provides a basic tutorial of OMG SysML parametrics and its foundation on composable objects (COBs). This Part 2 paper presents a more detailed example to show how SysML [OMG, 2007a] and COBs support engineering analysis templates and simulation-based design (SBD). In this context the terms simulation and analysis are interchangeable and refer to evaluating physical behaviors such as stress and temperature; however, the techniques presented are not necessarily limited to physics-based engineering analysis and are believed to be useful for other SBD domains.

1.1 Motivation

While computer processing power continues to advance, Wilson [2000] identifies the need for a physical behavior modeling representation that supports the following characteristics in a unified manner:

- Has tailoring for design-analysis integration, including support for multi-fidelity idealizations, product-specific analysis templates, and CAD-CAE tool interoperability.
- Supports product information-driven analysis—i.e., supports plugging in detailed design objects and idealizing them into a diversity of analysis models.
- Has computer-processable lexical forms along with human-friendly graphical and lexical forms.
- Represents relations in a non-causal manner—i.e., enables multi-directional combinations of model inputs/outputs.
- Captures engineering knowledge in a modular reusable form.

In Section 1.2 we overview a conceptual architecture that addresses these challenges and achieves advanced CAD-CAE interoperability in diversity-rich environments. Interoperability can be informally defined as the ability for

† Corresponding author: *Russell.Peak@gatech.edu* See the [GIT, 2007c] website for potential updates to this material.

tools and models to communicate and share information in a seamless computer-based manner. First we provide further motivation for SysML parametrics (beyond that in Part 1) in the context of engineering analysis in particular.

Motivation for SysML Parametrics—Part 2. The parametrics approach in SysML captures constraints among performance, physical, and other quality-related properties of the system and its environment. Such constraints are specified as equations among the properties. Although SysML itself is not intended to directly execute these constraints, the constraints and associated constrained properties can be passed to other engineering analysis tools to perform such computation. This approach ensures that engineering analysis computations are performed on the same system design/architectural model. A simple example can be illustrated by considering an automobile system design that may require a series of engineering analyses. The properties related to each element of the system (e.g., body, chassis, engine, transmission, transaxle, brakes, steering, etc) are each constrained by different sets of equations that are used to analyze vehicle performance (i.e. acceleration), handling, vibration, noise, safety, fuel economy, and so on. Keeping the analysis in synch with the different alternative system architectural models can clearly become a major challenge.

In addition, the engineering analysis models that are used may vary substantially in fidelity as the system development proceeds through its life cycle from early abstract analysis models to more refined analysis models later in the life cycle. This provides further motivation for the parametrics approach to provide a unifying mechanism to synchronize the system design and engineering analysis models. In fact, parametrics can be used in systematic ways throughout the life cycle to evolve the analysis, by starting with high-level analysis of the measures of effectiveness and evolving the analysis models to progressively include more detailed properties of the systems (e.g., measures of performance/technical performance measures) and ultimately system components. In this way, parametrics can be used by different members of integrated product teams to identity and agree on critical system properties that need to be analyzed and tracked throughout the life cycle.

1.2 The Multi-Representation Architecture (MRA)

The *multi-representation architecture* (MRA) (Figure 1) is the conceptual foundation of an X-analysis integration (XAI)[1] methodology based on knowledge patterns that naturally exist in engineering analysis processes. It is particularly aimed at design-analysis integration in CAD/CAE environments with high diversity (e.g., diversities of parts/systems, analysis disciplines, analysis idealization fidelities, design tools, and analysis tools) and where explicit design-analysis associativity is important (e.g., for automation, knowledge capture, and auditing). In this context, analysis means simulating physical behaviors in a part or system (e.g., determining the stress in a circuit board solder joint).

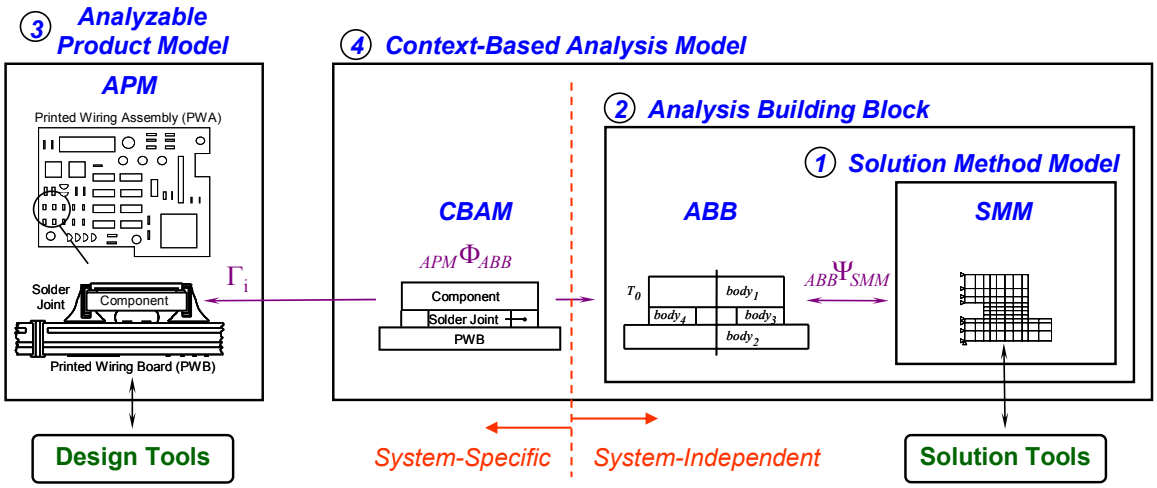


Figure 1: The multi-representation architecture (MRA) for modeling & simulation patterns. [Peak *et al.* 1998]

The MRA contains intermediate representations as stepping stones to achieve the flexibility and modularity dictated by complex fields like simulation-based design. Employing an extended object-oriented approach, these intermediate representations are natural groupings of concepts that occur between traditional design and analysis

[1] X = Models throughout the product lifecycle including design (DAI), manufacturing (MAI), and sustainment (SAI).

models. Originally the MRA was designed to capture reusable analysis knowledge at the preliminary and detailed design stages as shown in examples below. Applications to other system lifecycle stages are currently being explored, including conceptual design and feasibility studies [GIT, 2007a].

The MRA (Figure 1) includes the following conceptual patterns:

- *Analyzable product models* (APMs): Represent knowledge-based design models augmented with analysis-oriented overlays. Include multi-fidelity idealizations, Γ_i, and multi-source design information coordination (including interfacing with diverse CAD tools and design-oriented descriptive resources).
- *Context-based analysis models* (CBAMs): Represent product-specific analysis modules/templates. Capture idealization decisions inside CAD-CAE associativity relations, ${}_{APM}\Phi_{ABB}$, that connect APMs and ABBs.
- *Analysis building blocks* (ABBs): Represent product-independent analytical concepts as semantically rich, tool-independent objects that are reusable and modular. Generate SMMs via transformations, ${}_{ABB}\Psi_{SMM}$, based on solution technique-specific considerations such as symmetry and mesh density.
- *Solution method models* (SMMs): Represent solution method-specific models. Support white box reuse of existing tools (e.g., FEA tools and in-house codes). Automatic interactions occur through native command lines and/or application procedural interfaces (APIs) based on web standards like SOAP.

Note that the patterns on the left-half of the MRA—APMs and CBAMs—are dependent on the particular product or system domain of interest (e.g., circuit boards or space systems). However, the structure of these patterns and their abstract constituents are product-independent. The right-half patterns—ABBs and SMMs—are all generally product-independent and typically can be used in constructing many types of CBAMs. The reader is referred to [Peak *et al.* 1998, 1999, 2000, 2002, 2003] for further information on the MRA and other examples. The requirements and objectives document for next-generation COBs [GIT, 2007a] describes work underway to generalize these MRA patterns for the modeling and simulation of arbitrary systems-of-systems (SoS).

All the above patterns are represented as COBs. Like any COB, as described in Part 1, they can thus be a) implemented using SysML and b) executed using COB-based constraint management algorithms that leverage 3rd party solvers or in-house codes.

2 Flap Linkage Tutorial Example

In this section we present a benchmark tutorial for CAD-CAE interoperability and simulation template knowledge representation. We do so within an MRA context using a notional part family called flap linkages (Figure 2). This part family provides components for mechanism subsystems that manipulate flap control surfaces on aircraft. We developed this basic example to exercise multiple capabilities relevant to engineering design and analysis (many of which are relevant to broader simulation and knowledge representation domains), including:

- Diversity of design information sources, analysis behaviors, analysis fidelities, solution methods, and solution tools.
- Modular, reusable analytical building blocks and fine-grained inter-model associativity.

Along the way we will also cover several additional COB and SysML modeling concepts.

Figure 3 shows a panorama of the flap linkage tutorial and associated MRA-based simulation template concepts. Traditional CAD tools (left side) are used to define the manufacturable description of this product. On the right are traditional CAE tools that solve discretized and symbolic mathematical problems. In between are the four main types of MRA objects highlighted above: solution method models (SMMs), analysis building blocks (ABBs), analyzable product models (APMs), and context-based analysis models (CBAMs). These stepping stones help connect diverse tools and models in a flexible and modular manner.

Figure 4 shows a SysML package structure that implements these MRA concepts at both the generic and flap linkage-specific levels (e.g, compare with Figure 1). SysML packages can be thought of as logical groupings of related concepts similar to Java packages and STEP EXPRESS schemas. Here the common package contains product-independent concepts, while the flapLinkageApm and flapLinkageCbams packages are specific to flap linkage products. The «import» label indicates that a package leverages other packages as building blocks. The rest of this section overviews each package based on its context within the four main MRA patterns.

2.1 Flap Linkage Analyzable Product Model (APM)

APMs [Tamburini, 1999] help coordinate and merge design-oriented details coming from possibly many design tools and libraries. Corresponding to Figure 2, the lower-middle portion of Figure 3 shows a flap linkage APM constraint schematic that has features such as sleeves, shaft, cross-section, and ribs. The blue design-oriented relations show how design-oriented attributes like sleeve width and shaft width are related parametrically.

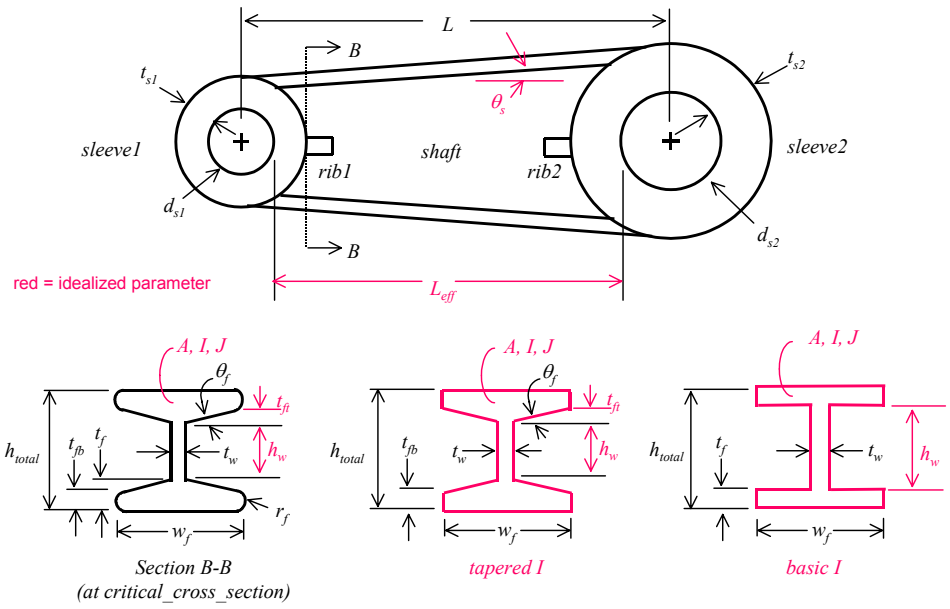


Figure 2: Flap linkage design model[2]—parametric shape features. [GIT, 2001]

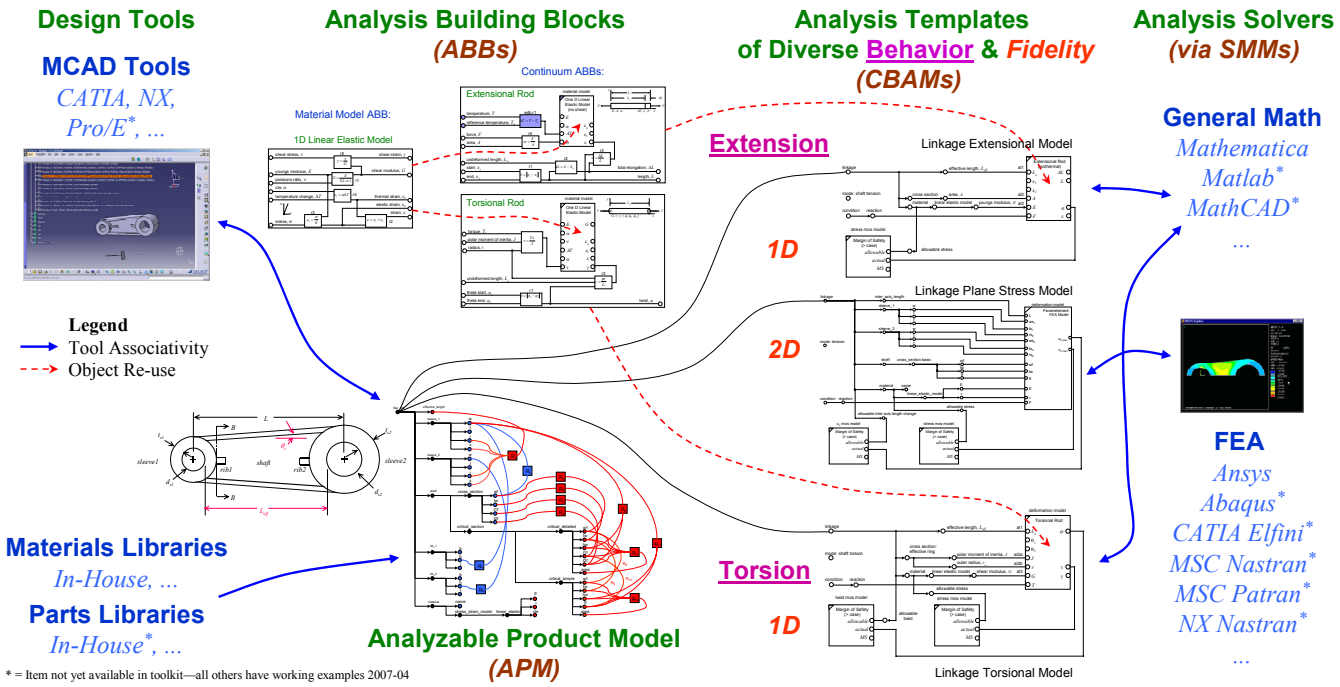


Figure 3: COB/MRA-based panorama for high diversity CAD-CAE interoperability—flap linkage tutorial.

[2] More specifically, the black aspects of this design model are from a manufacturable product model (MPM), while the black and red aspects combined are from an analyzable product model (APM) [Tamburini, 1999].

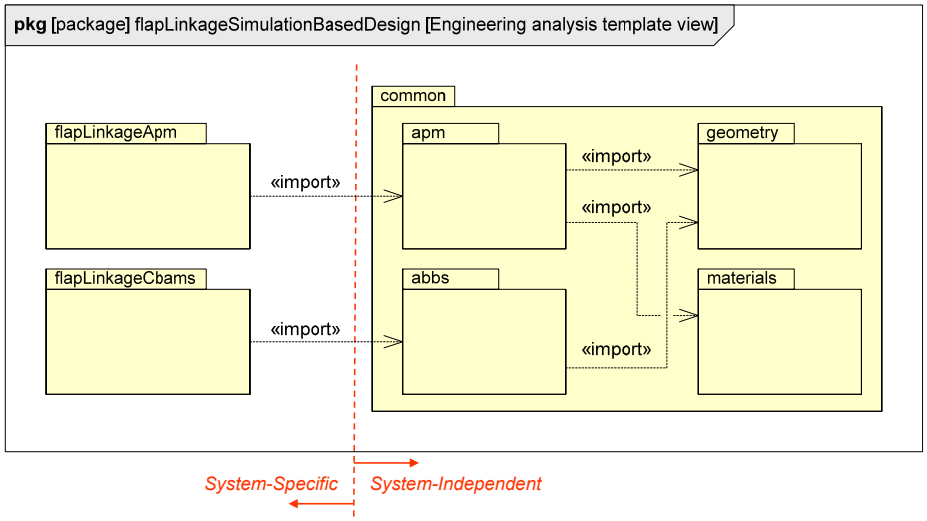


Figure 4: SysML package diagram for MRA-based engineering analysis and supporting resources.

Figure 5 provides SysML[3] block definition diagrams for the flap linkage APM at two different levels of detail. The first gives a basic overview of the blocks involved and their interrelations. The second shows more detail including package groupings and value properties (e.g., the width property of block sleeve). Comparing these figures underscores that SysML diagrams show subsets of the total model. This is good from a human comprehension point of view—otherwise things tend to get too cluttered and the value of a graphical language is diminished.

Here we want to treat materials as a shared resource (e.g., coming from a shared library) so that possibly may designs can utilize the same material instances. In SysML such situations are represented using *reference property* . In this example the block named PhysicalPart (and thus FlapLinkage by inheritance) is implemented as a reference property named material (indicated by a plain arrowed line) instead of the more commonly used SysML part[4] property (indicated by a arrowed line that has a black diamond on the opposite end). In contrast to reference values, part property values are utilized only within their defining context. For example, any rib instance that is part of a FlapLinkage instance is valid for use only within that FlapLinkage instance.

Beyond design attributes, APMs add idealizations (red in Figure 2 and Figure 3) that may be used by multiple analysis models. For example, the 1D torsion and extensional CBAMs in Figure 3 both use an idealization attribute called effectiveLength, L_{eff}. This attribute is the distance between the edges of the sleeves; it is a geometric idealization of the main material region that joins together the sleeves. While such an attribute is useful from an analysis point of view, it would not likely be shown as a dimension in the CAD model used to manufacture this part. Yet it is related to such attributes, so the APM provides a place to capture and utilize these parametric relationships.

The parametric diagram in Figure 6 aids authoring and visualizing such knowledge. Table 1 summarizes the included product design relations (pr) and idealization relations (pir) and also shows their substitution forms—i.e., what the constraint effectively behaves like in terms of the bound properties. For example, the constraint property named pir1 has a local constraint relation {Leff = L - (rhs1 + rhs2)} that effectively implements the idealization relation just mentioned, which is given by Eqn. 1.

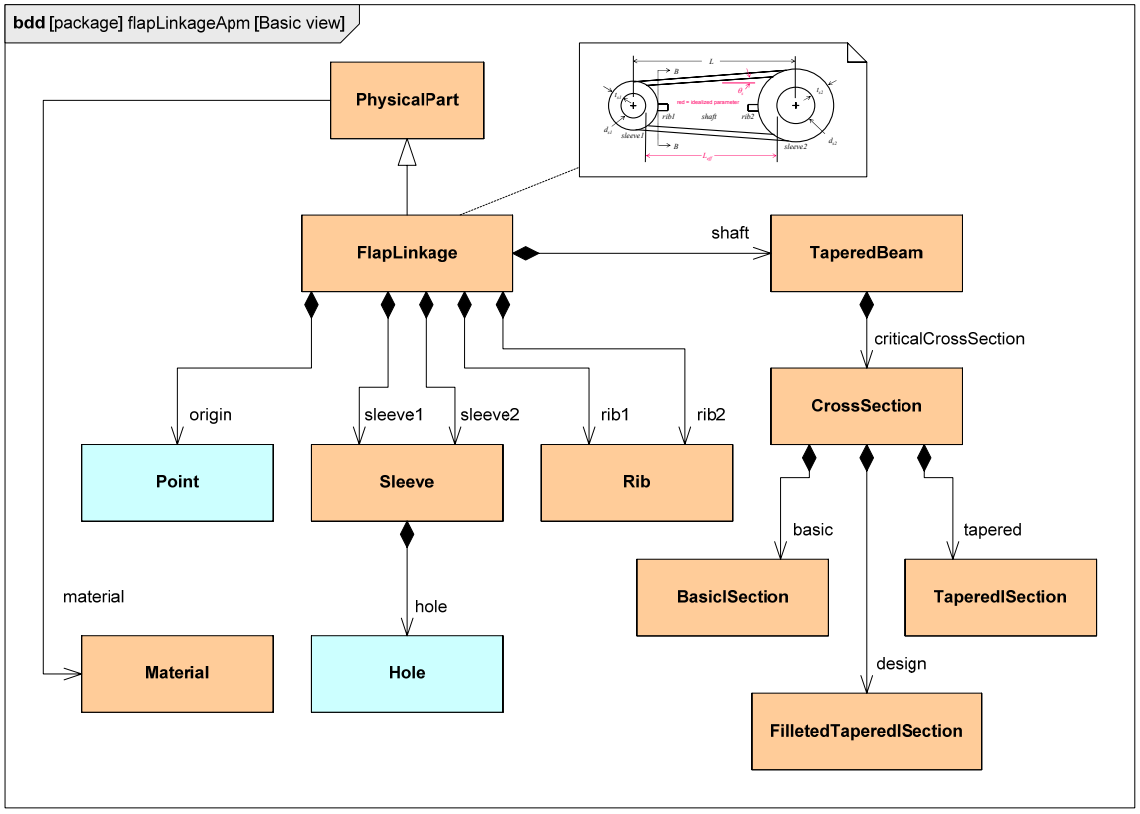
pir1: effectiveLength = interAxisLength - (sleeve1.hole.crossSection.radius + sleeve2.hole.crossSection.radius) Equation 1

With this diagram one can visually trace how these four attributes are connected via relation pir1, and how effectiveLength is also related via pir3 and pir4 to requirements-oriented properties for allowable deformation.

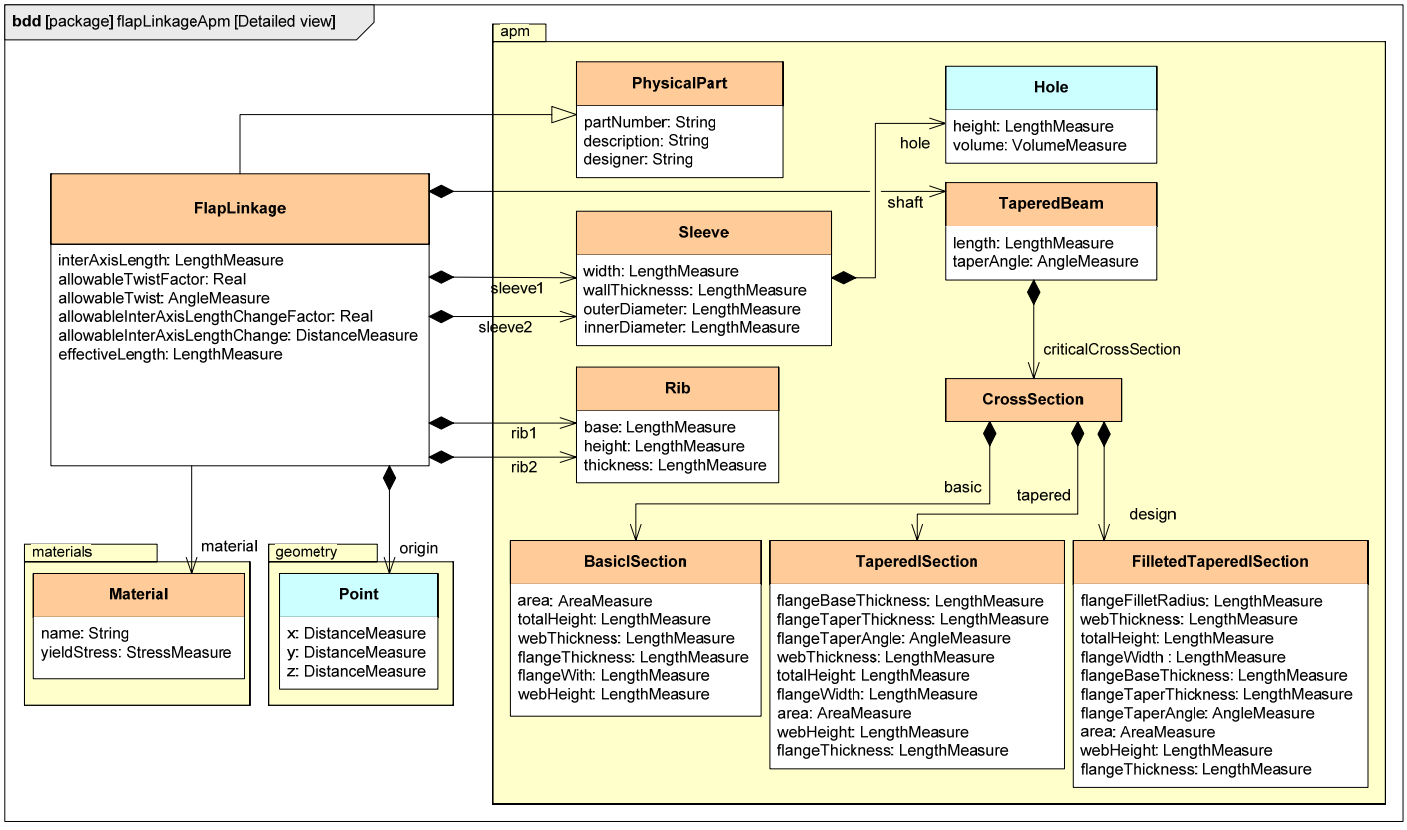
Figure 7 is an implementation of the flap linkage part family in a parametric mechanical CAD tool. While the same type of relations are present in such tool models, visual knowledge schematics are typically not well supported. Thus, without SysML parametric diagrams like Figure 6, it can be difficult for users to trace relations and affected attributes even in a basic model like this flap linkage.

[3] Unless otherwise noted, the SysML diagrams and description in these papers (Parts 1 and 2) conform to the Final Adopted Specification plus changes proposed by the SysML Finalization Task Force released February 23, 2007 [OMG, 2007b].

[4] The block named PhysicalPart should not be confused with the SysML part concept. The former is in this APM model to represent the traditional physical meaning of the word, while the latter is a modeling concept as discussed in [Peak *et al.* 2007].



(a) Basic bdd.



(b) Detailed bdd.

Figure 5: Flap linkage APM—SysML block definition diagrams (bdd).

Table 1: Parametric design and idealization relations in FlapLinkage.

Name	Constraint	Constraint substitution form
pr1:	{ys1 = y0}	{sleeve1.origin.y = origin.y}
pr2:	{ys2 = ys1 + L}	{sleeve2.origin.y = sleeve1.origin.y + interAxisLength}
pr3:	{hr1 = (ws1 - wtd) / 2}	{rib1.height = (sleeve1.width - shaft.criticalCrossSection.design.webThickness)/2}
pr4:	{hr2 = (ws2 - wtd) / 2}	{rib2.height = (sleeve2.width - shaft.criticalCrossSection.design.webThickness)/2}
pr5:	{tr1 = wtd}	{rib1.thickness = shaft.criticalCrossSection.design.webThickness }
pr6:	{tr2 = wtd}	{rib2.thickness = shaft.criticalCrossSection.design.webThickness }
pir1:	{Leff = L - (rhs1 + rhs2)}	{effectiveLength = interAxisLength - (sleeve1.hole.crossSection.radius + sleeve2.hole.crossSection.radius)}
pir2:	{htotd = ods1}	{shaft.shaft.criticalCrossSection.design.totalHeight = sleeve1.outerDiameter}
pir3:	{tha = thaf * Leff}	{allowableTwist = allowableTwistFactor * effectiveLength}
pir4:	{dLa = dLaf * Leff}	{allowableInterAxisLengthChange = allowableInterAxisLengthChangeFactor * effectiveLength}

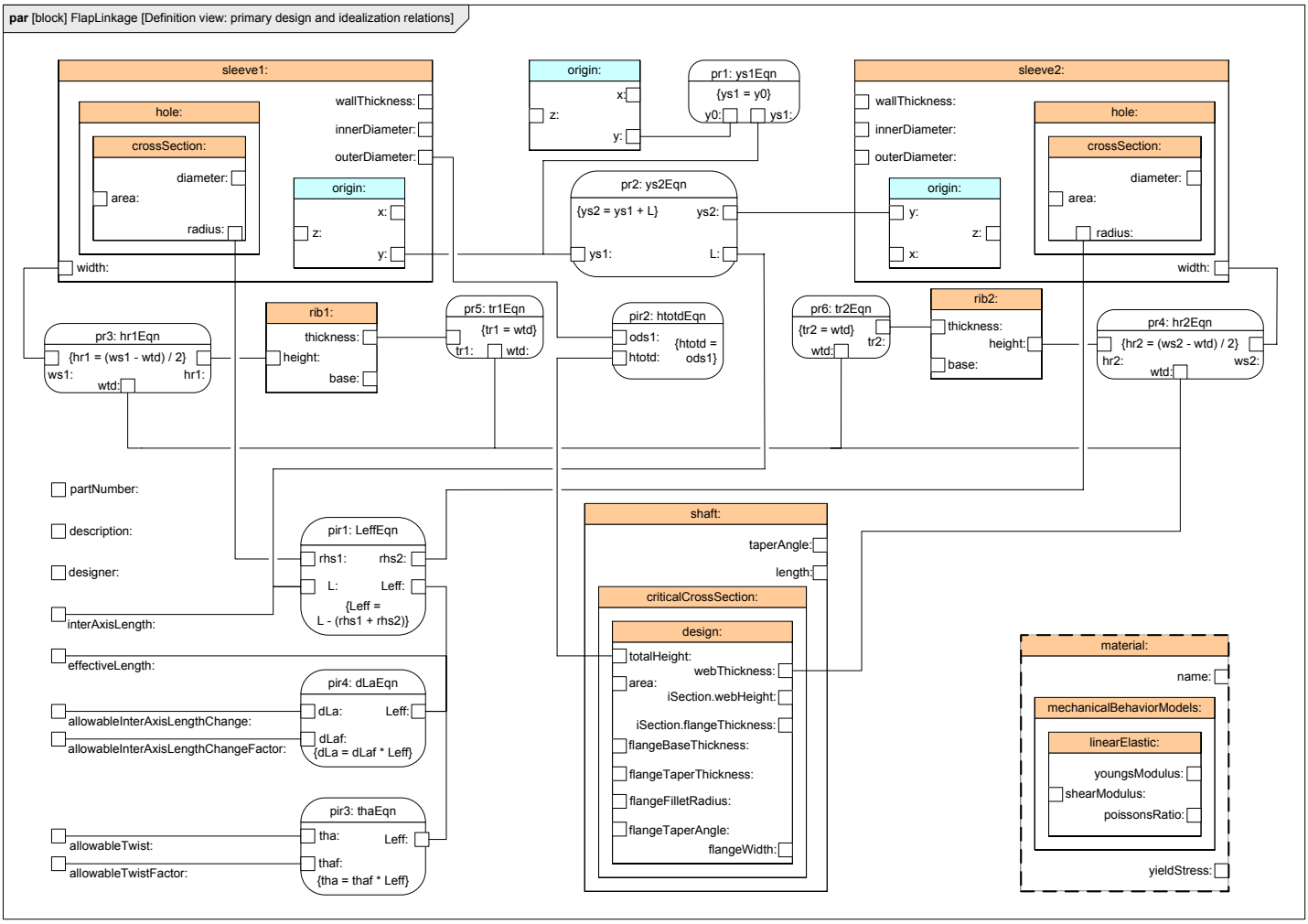


Figure 6: Flap linkage APM—a SysML parametric diagram[5] (par).

[5] By convention GIT shows internal properties of a part or reference property as small boxes flush with their outer context (e.g., height: is flush with the left in its rib1: box). This uses less space and leverages the analogy of parametric diagrams being like electrical schematics. To better distinguish them visually from constraint parameters, others prefer them not be flush. Both styles are allowable per the SysML specification.

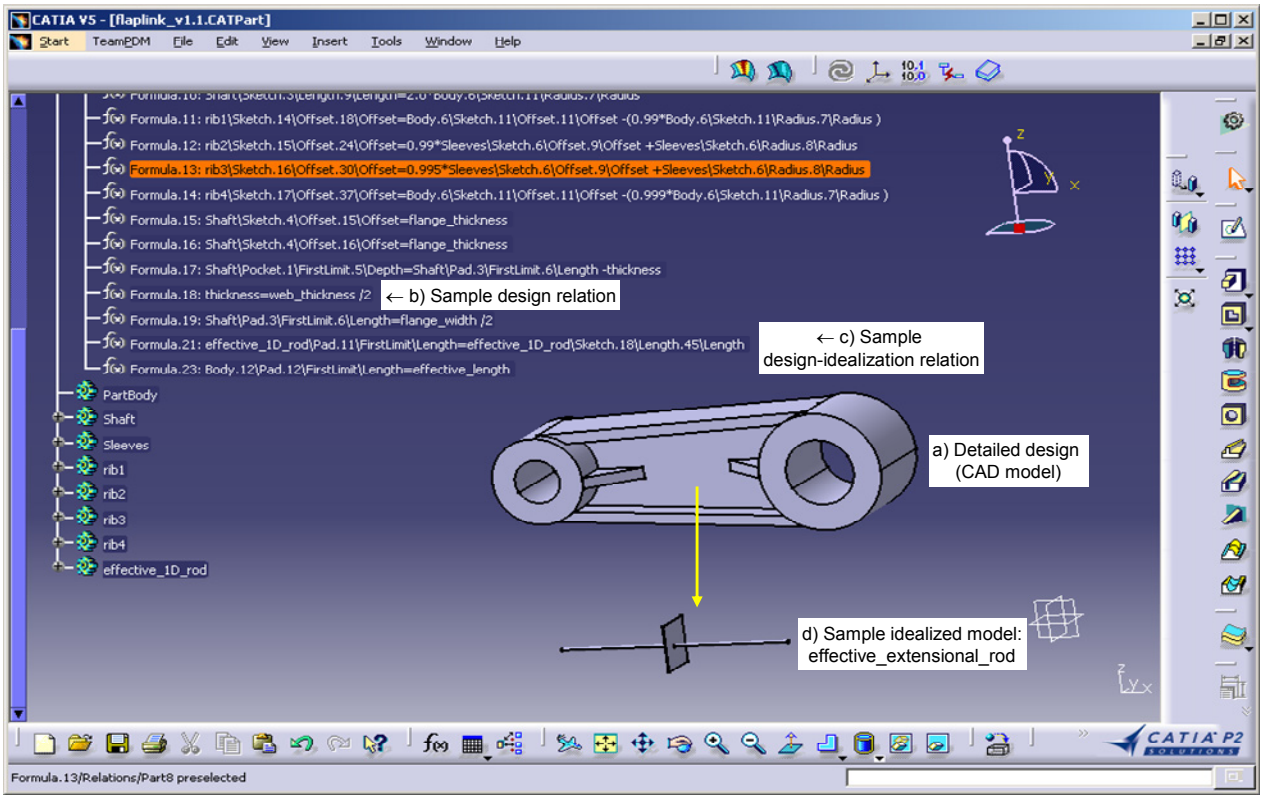


Figure 7: Flap linkage APM instance XYZ-510—CAD model implementation in CATIA v5.

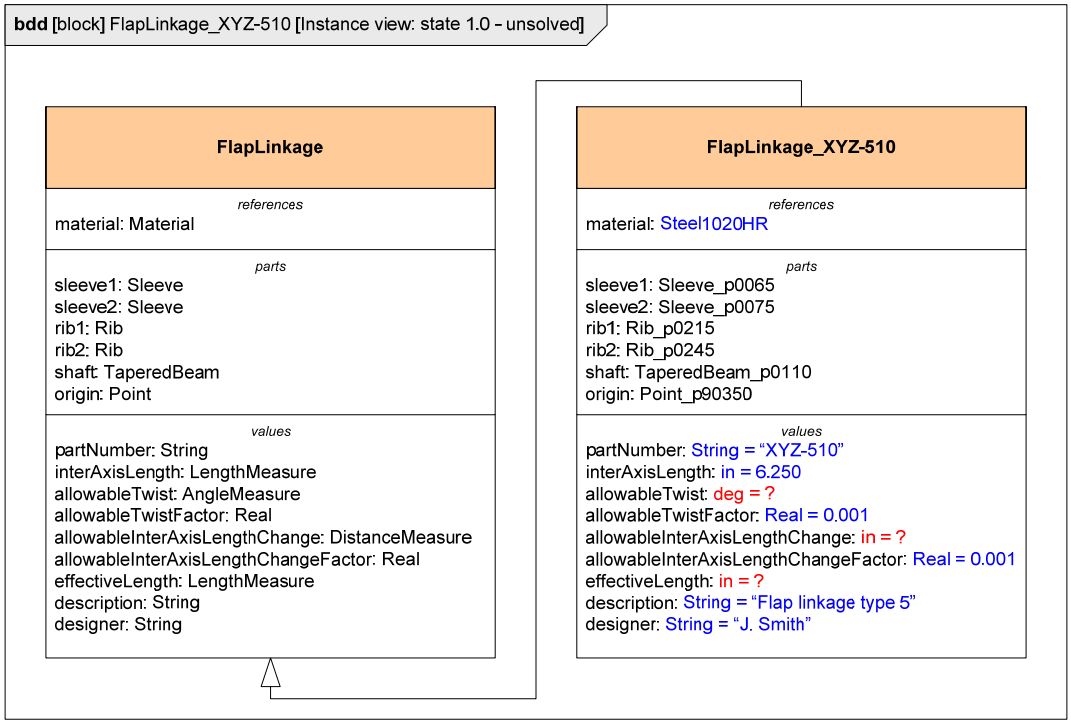


Figure 8: Flap linkage APM instance XYZ-510 with design inputs—SysML block definition diagram.

To design, analyze, and utilize parts like flap linkages, eventually one needs to work with specific values for at least some attributes. Figure 8 presents a SysML view of a flap linkage instance (part number XYZ-510) with its main design values provided as inputs (corresponding to Figure 7). In Section 2.4 we will show how to use instances like this in simulation templates.

2.2 Analysis Building Blocks (ABBs)

Next we jump to the ABB MRA pattern. ABBs represent analytical engineering concepts—irrespective of solution method and product domain—as objects that can typically be employed in many simulation template contexts. The upper-middle portion of Figure 3 contains constraint schematics for the material model ABB and continuum ABBs described in Part 1, namely OneDLinearElasticModel, ExtensionalRod, and TorsionalRod. Note the graphical depiction of multi-level modularity and reusability, as these two continuum primitives are built from the same material model primitive, and they are then utilized in two different CBAMs.

Sometimes it is convenient to create ABB systems that are tailored to specific product situations yet still built from generic ABB primitives. These types of analytical assemblies are known as *specialized analysis systems* [Peak *et al.* 1998]. Figure 9 is such a case for linkage simulation (see LinkagePlaneStressModel in Section 2.4). The SysML parametric diagram for the ABB system, (a), contains a part property that treats the FEA-based SMM template, (b), like functional relations. It turns out just about any external tool can be wrapped in a similar manner and incorporated uniformly within the SysML block structure. Zeng *et al.* [2004; 2007] and Bajaj [2006] describe progress towards generating such ABB systems (and their SMMs) on-the-fly rather than relying on pre-configured specialized parametric templates.

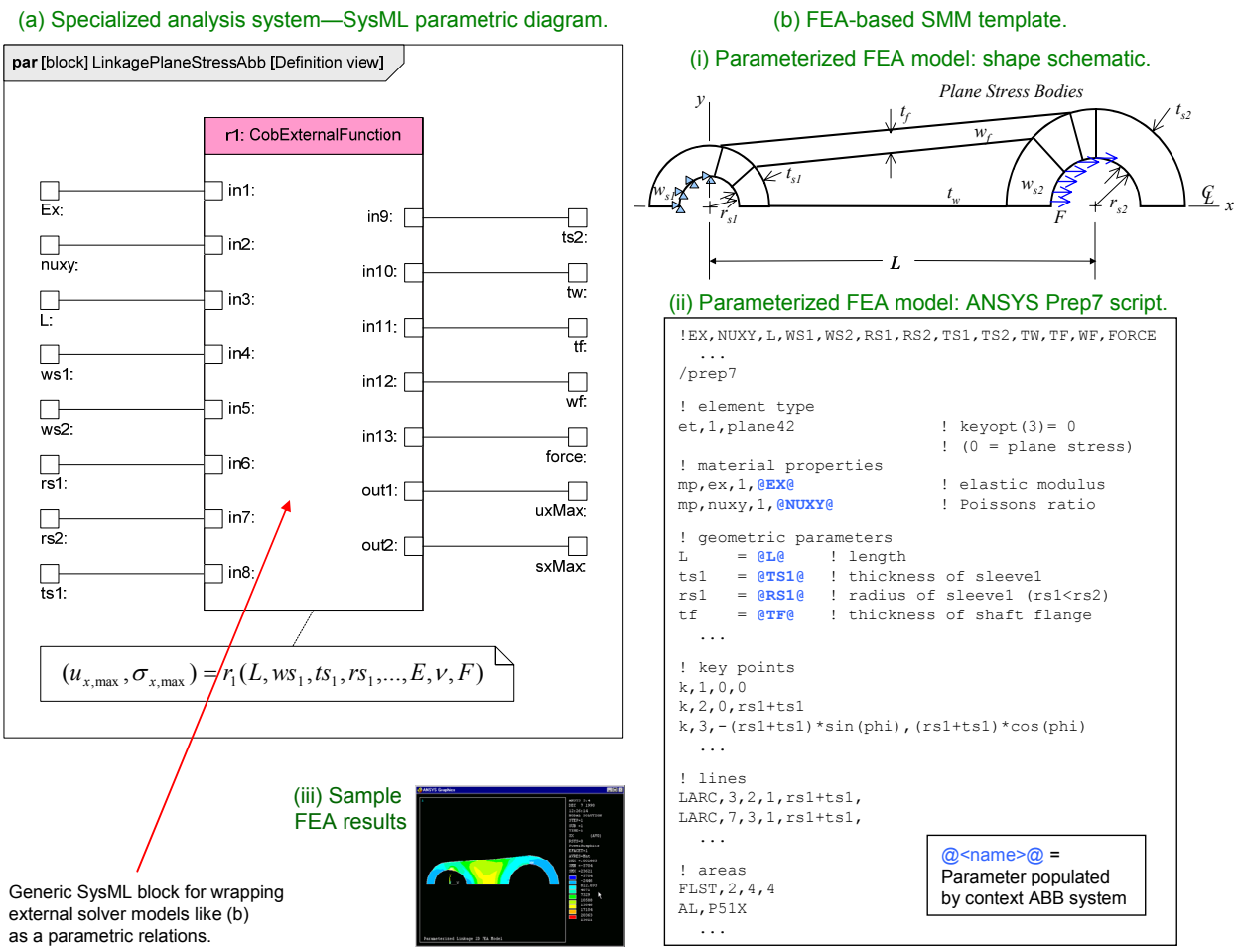


Figure 9: Example specialized ABB system with an FEA-based SMM template.

2.3 Solution Method Models (SMMs)

Whereas ABBs represent simulation concepts at the analytical level, SMMs represent them at the detailed solution method level. SMMs can be viewed as object-oriented wrappers around solution tools (such as CAE solvers) that obtain simulation results in a highly automated manner (Figure 3 far right). They support white box reuse of existing tools (e.g., finite element analysis (FEA) tools, math tools, and in-house codes) within a uniform constraint-based framework. ABBs generate SMMs based on solution technique-specific considerations such as symmetry and mesh density. SMMs capture both the inputs they send to solution tools and the outputs they retrieve (e.g., textual results and graphical results).

Figure 9(b) shows the FEA-based SMM template for the ABB system described in the previous section. Its main input, (ii), is a vendor-specific parametric script written in a vendor-specific FEA modeling language (Prep7), which has a corresponding human-sensible shape schematic, (i). Its primary outputs are summary values (as computer-sensible text) that characterize the key behaviors of interest—maximum stress and deformation in this case—as well as graphical views of results, (iii).

As another example, vendor-specific script fragments related to Eqn. 1 are given in Figure 10. COB algorithms auto-created the *Mathematica* inputs, (a), based on equations in corresponding SysML blocks (Section 2.1). *XaiTools* submitted the SMM to a *Mathematica* web service for execution as described in Part 1. It then retrieved the outputs, (b), processed them, and folded them back into their higher level COB/MRA contexts for further usage and presentation (Figure 13).

(a) Input script.

```
   ...
<<XaiTools`
output =
OpenWrite["smm_mathematica_result.txt"];

(*  ...
a22 is inter_axis_length
u20 is linkage.effective_length
x49 is sleeve1.hole.cross_section.radius
y50 is sleeve2.hole.cross_section.radius
    ... *)

solutions = Solve[ {
   ...
a22 == 6.25,
   ...
(* flap_linkage apm pir1 = Equation 1 *)
u20 == a22 - (x49 + y50),
   ...
} ];

WriteString[ output, ToString[
   CForm[ N[solutions] ] ] ];
Close[output];

Exit[];
```

(b) Output script.

```
List(List(
       ...
   Rule(u20,5.),
   Rule(x49,0.5),
   Rule(y50,0.75),
       ...
))
```

Figure 10: Sample auto-generated math solver SMM—*Mathematica* script fragments for the linkage.effectiveLength idealization (Eqn 1).

2.4 Flap Linkage Context-Based Analysis Models (CBAMs)

Finally we reach the CBAM MRA pattern. CBAMs are also known as *analysis templates*, *analysis modules*, and *simulation templates*. CBAMs explicitly capture fine-grained associativity between a design model and its possibly many analysis models (i.e., between ABBs and APMs). These associativity relations are one way to precisely represent idealization decisions and analysis intent. Figure 3 depicts three types of flap linkage CBAMs and their macro-level connections to the APM design model.

These same CBAMs are given in the Figure 11 SysML block definition diagram along with the primary ABBs they utilize. The stereotypes «cbam», «abb», and «apm» in this diagram are our user-defined specializations of the SysML «block» stereotype. These stereotypes provide further structure and semantics for these types of blocks based on their MRA context.

These analysis templates help verify whether or not linkage designs meet requirements with respect to two types

of deformation behavior: extension (stretching) and torsion (twisting). The first two templates simulate the extension behavior at two levels of fidelity as described later in this section. The italicized LinkageAnalysisModel block name indicates it is an *abstract* generalization of these three CBAMs (where "abstract" in the SysML context means it cannot have instances itself).

The left side of Figure 12 shows how the LinkageExtensionalModel analysis template would typically look in a traditional documentation-oriented view. The right side is the objectized formulation of this template annotated with its key MRA and CBAM features. Figure 13 presents the same CBAM in SysML parametric form. This CBAM captures explicit CAD-CAE associativity, i.e., how a subset of APM attributes like effectiveLength, L_{eff}, are connected[6] to precise attributes in the ExtensionalRodIsothermal ABB it is using (a specialization of the ExtensionalRod ABB defined in Part 1). Note that this same type of ABB can be used in other CBAMs for other types of products (e.g., circuit board solder joint analysis [Peak, 2000]).

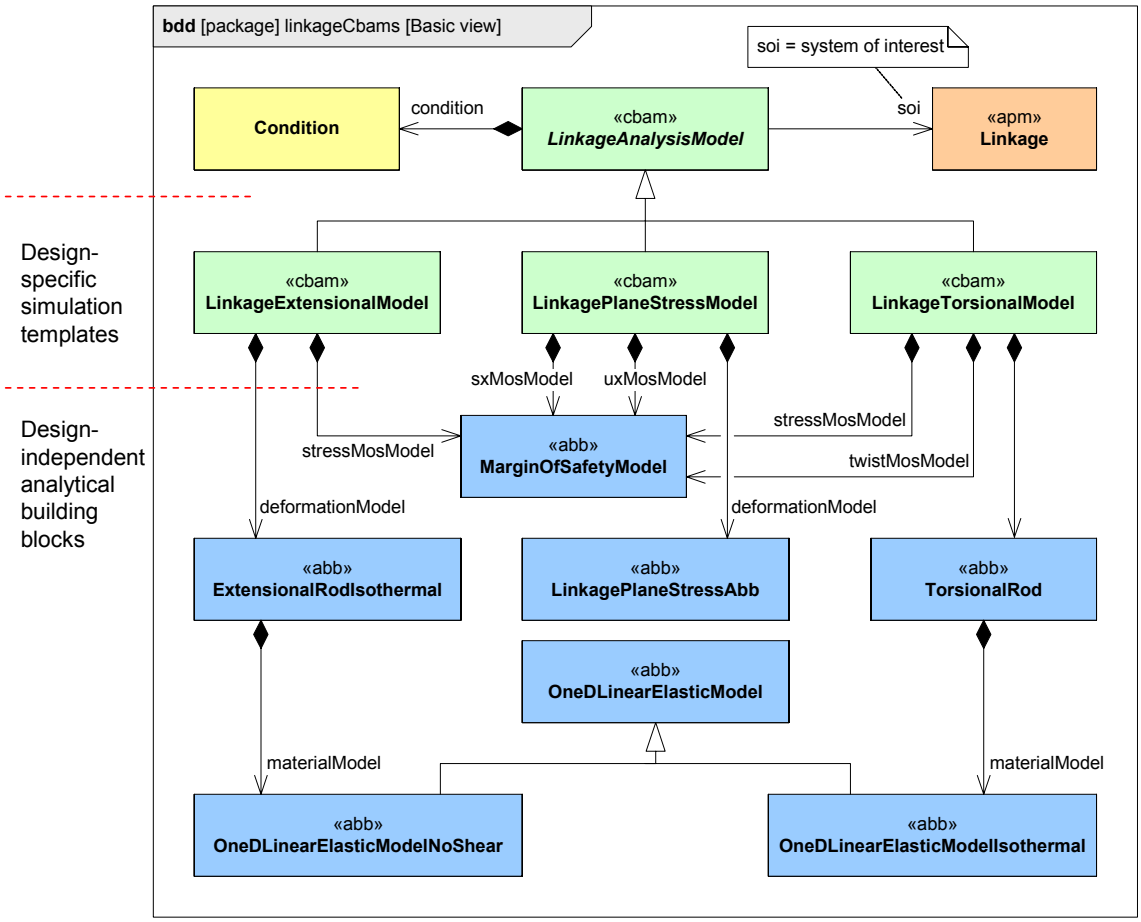


Figure 11: Linkage analysis templates & generic building blocks—basic SysML block definition diagram.

[6] As seen in Figure 13 through Figure 15, CBAM templates typically establish connections among a system of interest (soi), a simulation capability, and a margin of safety (MoS)-like capability. Per the above bdd, the soi is an APM system represented as a SysML reference property in the CBAM, and the simulation and MoS capabilities are ABBs represented as part properties.

The GIT convention is to represent equality relation connections among such items as SysML binding connectors (shown in parametric diagrams as solid line connections between their properties). A more standard approach is to represent an equality relation as a constraint property, but this would graphically take up more space on parametric diagrams—especially for CBAMs like Figure 14 with 17 equality relations. Another approach being considered is to utilize special symbols for common equations like equality to better depict them graphically.

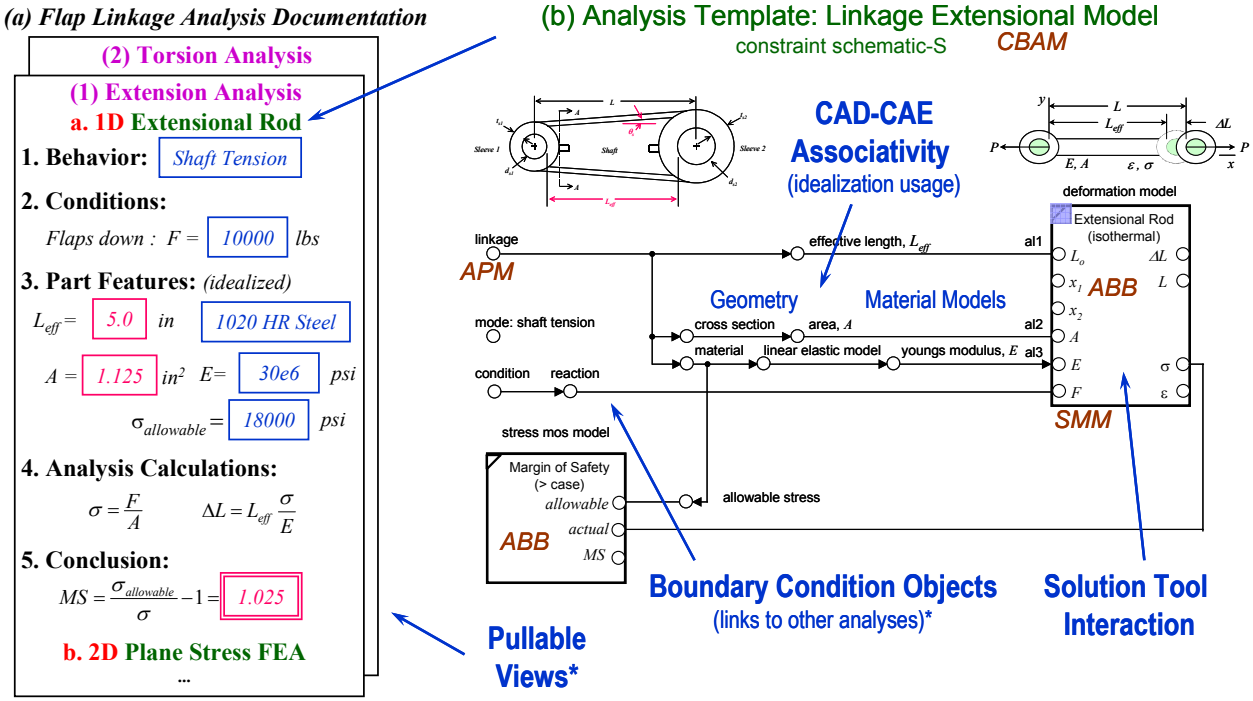
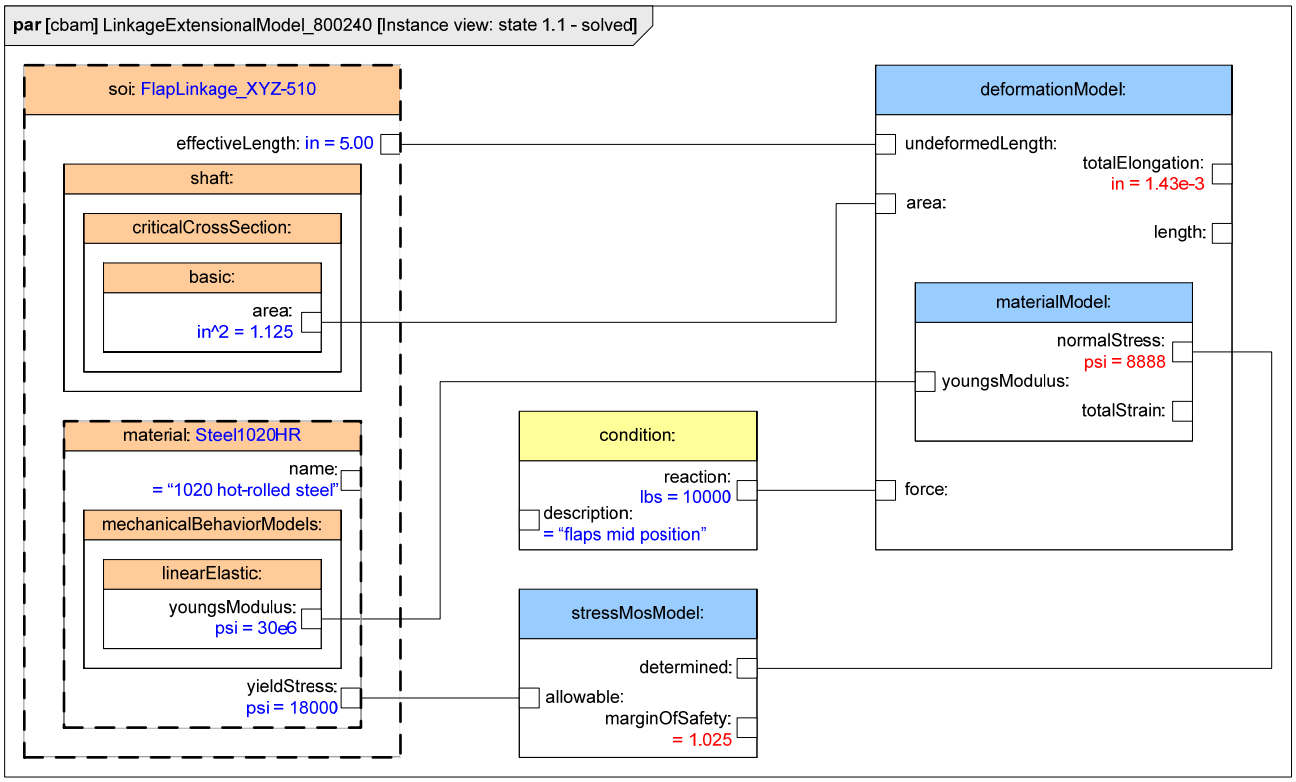


Figure 12: Traditional documentation vs. a COB-based analysis template: linkage_extensional_model.

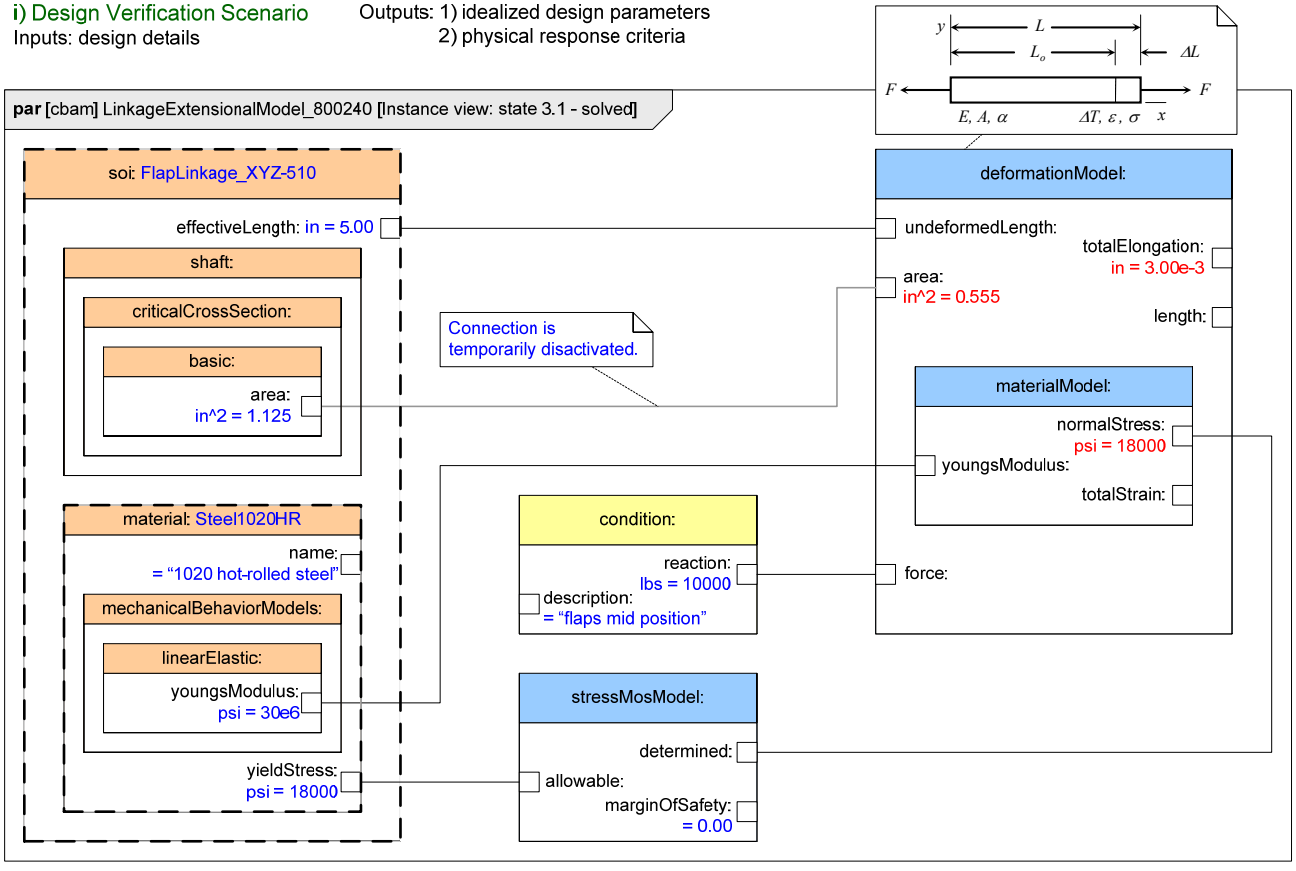
In addition to *how* the analysis model is "wired" to work, the CBAM shows *why* the analysis model exists: to confirm that the calculated stress remains less than the allowable stress (i.e., MoS ≥ 0). This template uses a MarginOfSafetyModel ABB for this purpose as seen in the lower left corners of the parametric diagrams in Figure 12 and Figure 13. Staying below allowable stress—to avoid permanent deformation or even breakage—is a typical design requirement for load-bearing parts like flap linkages. Figure 11 shows that all three types of CBAM blocks employ MarginOfSafetyModel for similar verification purposes (see also Section 2.5).

Similar to the examples in Part 1, Figure 13 also shows how such templates can be populated and executed, (b), and how a single CBAM can be run in several directions, (a), to aid both design verification and design synthesis. In solved state 1.1, the APM design details are inputs, and margin of safety (MoS) is the target output of primary interest. In solved state 3.1—the lower portion of (a)—the situation is reversed in that margin of safety is now an input and deformationModel.area is the designated output (including temporarily suspending its relation with the flap linkage instance). This capability allows one to directly compute the "optimum" design variable value in subgraph cases where systems of relations analytically support directional changes. After design details have been developed to have criticalCrossSection.basic.area achieve this target area value, the same CBAM can be used to check the design again by re-running it in the same direction as state 1.

Considering the engineering semantics of the problem, one sees that state 1 typifies a simple design verification scenario, where the "natural inputs" (physical design properties and a load) are indeed inputs and a "natural output" (a physical response to the load) is the requested output. Hence, the design is being checked to ensure it gives the desired response. As a design synthesis (sizing) scenario, state 3 reverses the situation by changing one natural output into an input and one natural input into a designated output. It effectively asks "what cross-section area (a design-oriented variable) do I need to achieve the desired margin of safety (which depends on the stress physical response)?" This COB capability to change input and output directions with the same object instance thus has important engineering utility. It is a *multi-directional* capability in that there are generally many possible input/output combinations for a given constraint graph.



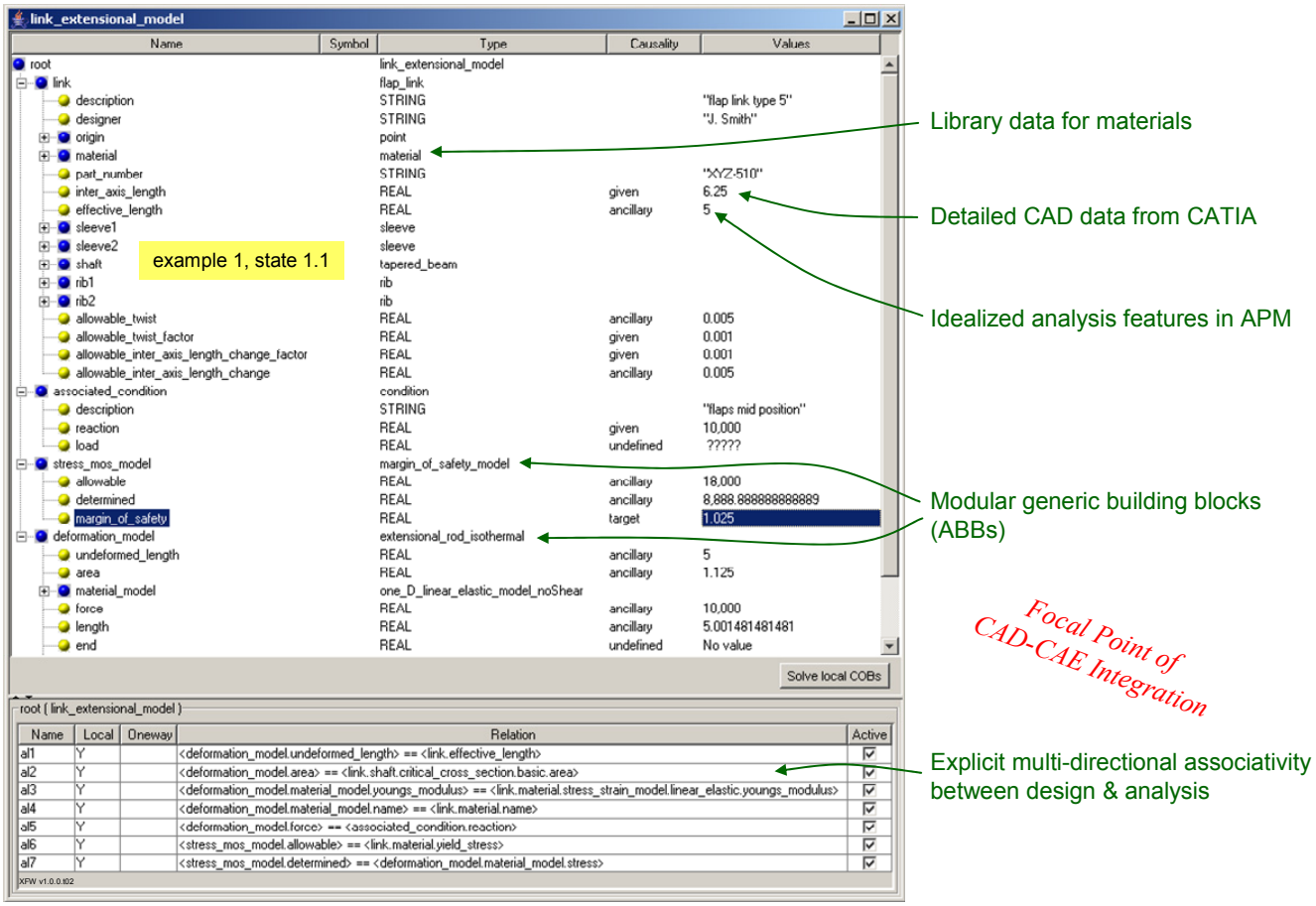
i) Design Verification Scenario
Inputs: design details
Outputs: 1) idealized design parameters
2) physical response criteria



ii) Design Synthesis Scenario
Inputs: desired physical response criteria
Outputs: 1) idealized design parameters (e.g., for sizing), and/or
2) detailed design parameters

(a) Multi-directional capabilities of a COB-based CBAM—SysML parametric diagrams.

Figure 13: Sample instance of analysis template LinkageExtensionalModel.



(b) Executable SysML parametrics model in *XaiTools* COB browser—an object-oriented spreadsheet.

Figure 13 (continued): Sample instance of analysis template LinkageExtensionalModel.

Figure 3 and Figure 11 also contain the LinkagePlaneStressModel CBAM, which simulates the same type of extension physical behavior as LinkageExtensionalModel. It utilizes the specialized FEA-based ABB/SMM system described above (Figure 9) to obtain more detailed stress and deformation answers—over a 2D field versus the 1D field in LinkageExtensionalModel. Demonstrating *multi-solver* capabilities within the same template, it still utilizes *Mathematica* to solve formula-based relations such as those in its part properties based on the MarginOfSafetyModel block. Its SysML parametric diagram (Figure 14) graphically shows that its deformationModel ABB connects with more APM geometric and material model idealizations than does the 1D CBAM. Thus, it is a higher fidelity CBAM illustrating the *multi-fidelity* capabilities of the MRA. Typically engineers use quick lower fidelity models early in the lifecycle to size the design, and more costly higher fidelity models later to check the design more accurately.

Finally, the LinkageTorsionalModel in Figure 15 illustrates the *multi-behavior* capability of the MRA (Figure 3). This CBAM simulates a different type of physical behavior (torsion) versus the previous two CBAMs (extension). Consistent with Figure 11, note that it uses the TorsionalRod ABB described in Part 1. It connects to different idealized attributes in the APM (e.g., polarMomentOfInertia), as well as to some of the same ones (e.g., effectiveLength). Via *XaiTools*, the solver tool *Mathematica* again executes the formula-based relations as another example of CAE *tool re-usage*.

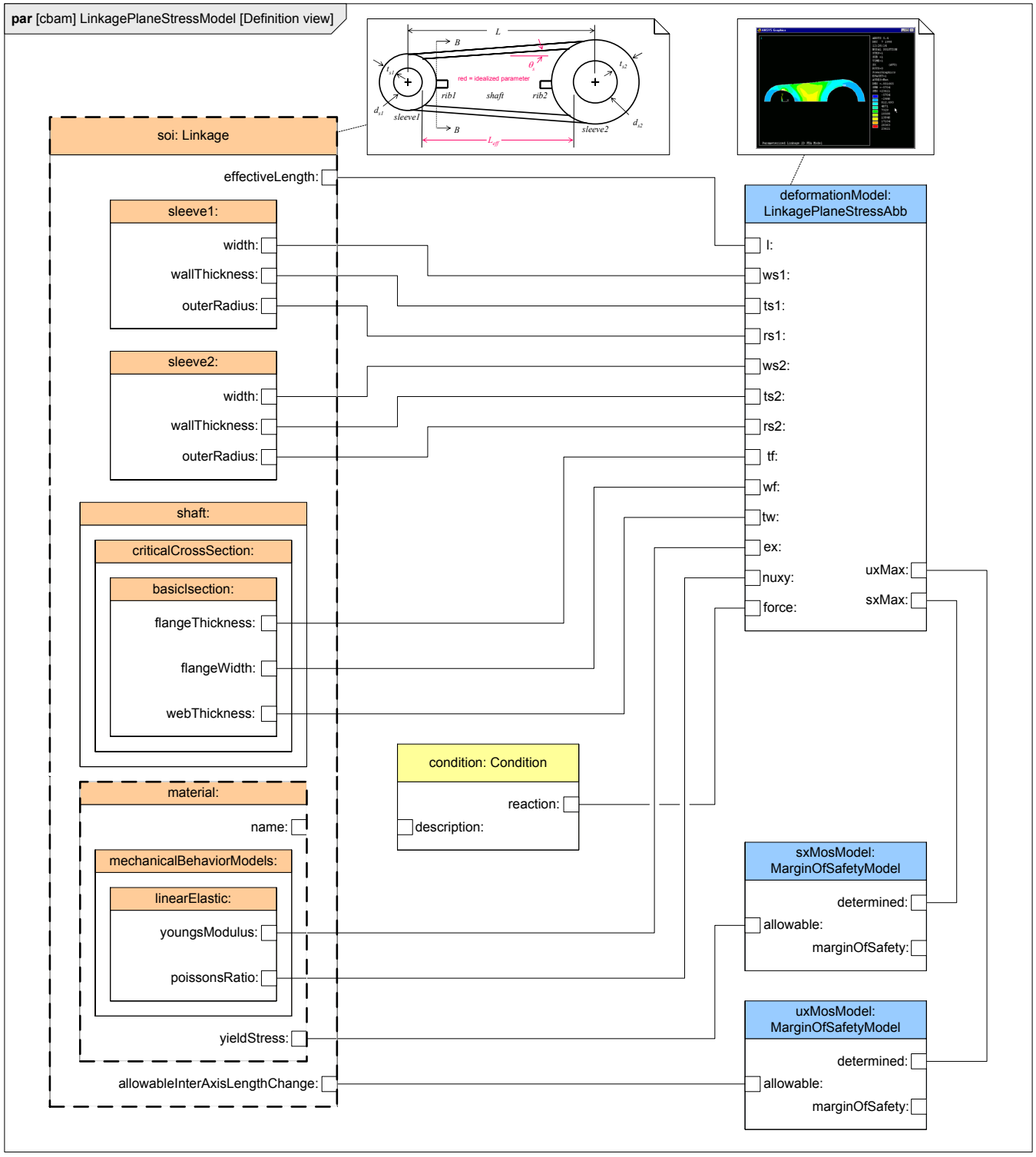


Figure 14: Analysis template LinkagePlaneStressModel—SysML parametric diagram.
(Example FEA-based model)

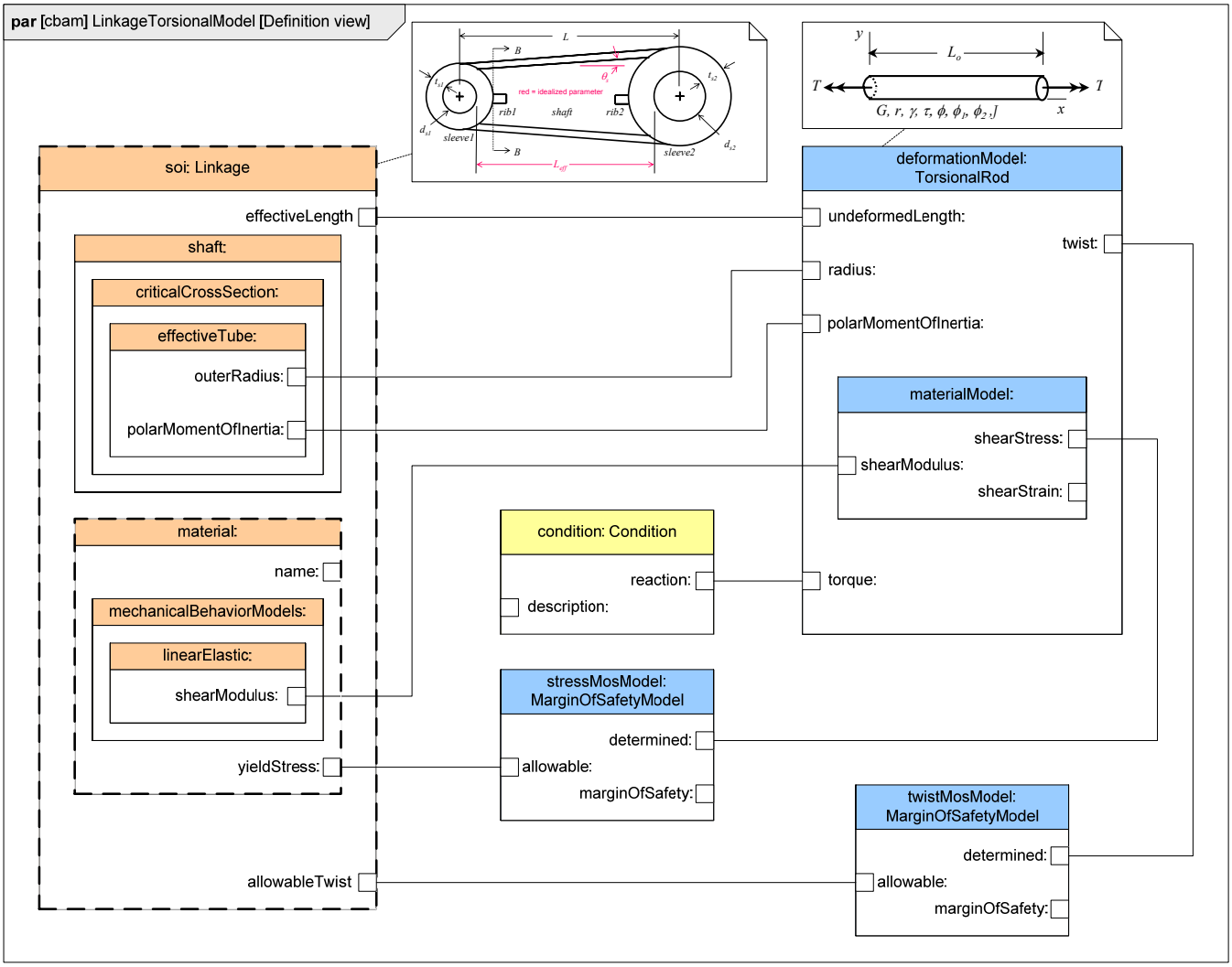


Figure 15: Analysis template LinkageTorsionalModel—SysML parametric diagram.

2.5 Requirements Testing and Verification

Several patterns relevant to simulation-based design have been identified in addition to the above MRA patterns. This section highlights their existence and embodiment in SysML and their relationship to the MRA. Figure 16 exemplifies a SysML requirements structure[7] which includes i) requirements decomposition, ii) requirements derivation, iii) requirements allocation/satisfaction to/by systems, and iv) requirements verification by test cases.

This figure shows notional requirements for the top-level system—an airframe[8]—that contains flap linkages [GIT, 2001]. The upper level requirements are broken down by aircraft flight segments to ultimately produce requirements for flap positions based on anticipated maneuvers and conditions. For example, the loads that flap linkages experience are likely to be quite different during a 2G dive versus during normal mid-position operation. The «satisfy» connections represent that FlapLinkage is intended to satisfy each of the four resulting requirements. This diagram also indicates that a «testCase» named FlapsDownTestCase is provided to «verify» that a FlapLinkage

[7] See [Friedenthal *et al.* 2006] and other publications listed at *www.omgsysml.org* for further information regarding SysML constructs for requirements, trade studies, and verification & validation.

[8] SysML also supports modeling the aircraft system block and its breakdown to reach flap linkage usage blocks, including requirements allocations to subsystems along the way. This kind of structure could also be presented on diagrams like Figure 16 to whatever degree of detail desired.

indeed satisfies the FlapsDown requirement.

Figure 17[9] [based on Tamburini, 2006] then brings it all together. It contains a SysML sequence diagram that shows how this test case employs a simulation template to verify that the system of interest satisfies the requirement. I.e., when FlapsDownTestCase is instantiated and executed, it creates a test bench object that has access to the FlapsDown requirement (Figure 16) and the FlapLinkage_XYZ-510 instance at mechanism position #3 (Figure 8). It creates a LinkagePlaneStressModel_760 simulation template instance (Figure 14), plugs in the design instance and relevant derived requirement information (a 10,000 lb load), and executes the simulation template. It then requests the primary results of interest—margins of safety for stress and deformation—and determines the verdict, i.e., whether or not the design meets the criteria (i.e., it checks if the requirement is successfully verified).

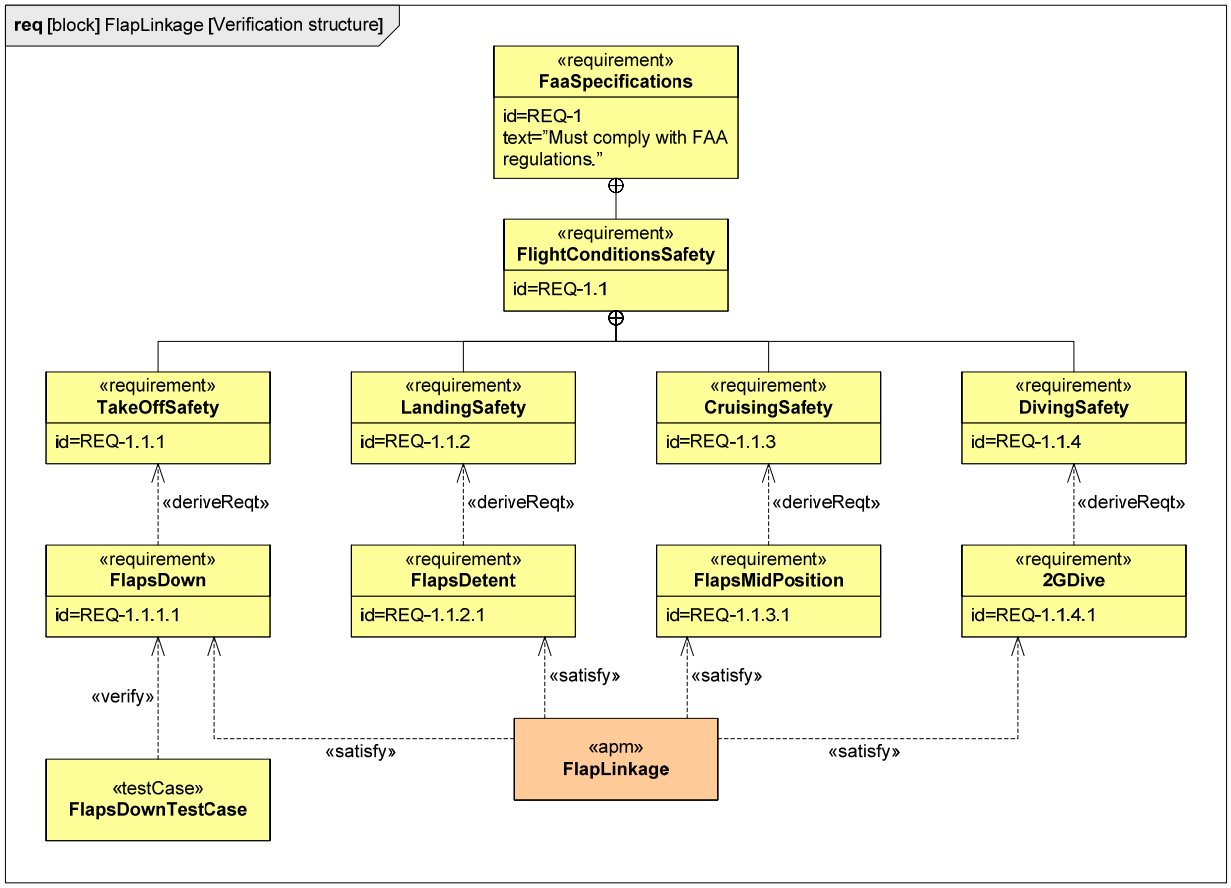


Figure 16: System context for flap linkage verification—SysML requirements diagram. [after Tamburini, 2006]

[9] This SysML-based approach effectively embodies and extends MRA analysis template use cases presented, for example, in [Peak *et al.* 1998] Figure 23 and [Peak *et al.* 1999].

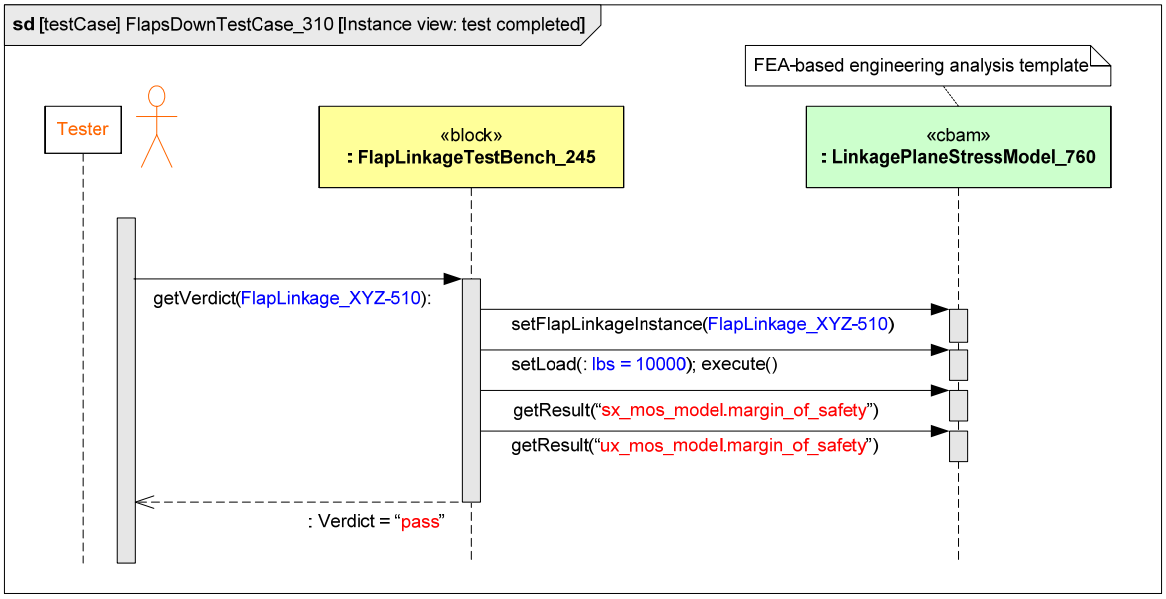


Figure 17: CBAM-based test case execution for requirements verification—SysML sequence diagram.

3 Discussion

3.1 Other Applications and Test Cases

Industrial applications of COBs and other test cases are given in [Wilson, 2000; Peak, 2000] along with structure statistics shown in Appendix A (Table A1) involving more than 260 types of COBs. These include thermomechanical analysis of printed wiring boards and assemblies (PWA/Bs), structural analysis of airframes, and thermal analysis of electronic chip packages. Benefits include decreasing overall simulation model creation time as much as 10:1 (from days to hours) [Zeng *et al.* 2006], and auto-generating complex FEA models that were not previously feasible. Table A2 summarizes the diversity of tools and techniques exercised across these test cases. Similarly, reuse of ABBs and APMs is shown in Table A3, including usage of the same ABBs in different product domains. Thus re-implementing these test cases as SysML models that support automatic COB-based parametric execution helps ensure that SysML supports such capabilities at similar degrees of depth and breadth.

Other SysML applications—developed to varying degrees of maturity—include models of space systems (e.g., the FireSat satellite [Larson and Wertz, 1999]), fluid power systems (including usage of the Modelica/Dymola simulation tool), electrical/mechanical CAD and CAE, model trains (e.g., mechatronic aspects), racing bikes, factories, and warehouses [GIT, 2007a; Burkhart, 2006; Peak, Tamburini, Paredis, McGinnis, 2006; Tamburini, 2006; Tamburini and Deren, 2006].

3.2 Other Observations

The left side of Figure 12 is a traditional documentation-oriented view of a simulation-based design problem. Shortcomings of this view are that it imposes a unidirectional sequence, it limits modularity and reusability, and it typically does not capture idealization knowledge like effective length. COB-based SysML models overcome these problems today. In the future such documentation views may be automatically derived from SysML models using technologies like XML (similar to how SysML tools can now generate requirements documents).

Additional patterns covering the full product/system lifecycle have been identified as seen, for example, in SysML itself [OMG, 2007a; Friedenthal *et al.* 2006], the core product model (CPM2) [Fenves *et al.* 2004, 2006], and the Twelve-Fold Way™ [CPDA, 2003]. They include design models and simulation models as main types of patterns and identify macro-level relationships among these patterns. The MRA complements such work by further defining these patterns and their subtypes, identifying associated intermediate patterns and concepts, and providing structures for micro-level associativity among these patterns [Peak, 2005; Collier *et al.* 2006]. Figure 12, Figure 13(b), and Figure 17 exemplify these interrelationships. Perhaps SysML coupled with similar techniques could help

detail the interrelationships among other product/system lifecycle patterns.

Today analysts and simulation specialists spend much of their time creating relatively low-level models utilizing vendor-specific formats as in Figure 9(b) and Figure 10. It may not be feasible to fully get away from doing that for the most advanced modeling situations. But perhaps with SysML and the techniques highlighted above, we may increase modeling semantics and automation for many useful problem classes.

In some cases relations contained in SysML constraint blocks cannot be inverted analytically. For example, external tool relations like those in the FEA-based specialized analysis system in Figure 9 and those based on other procedural algorithms are naturally unidirectional. In such cases COB-based SysML models can at least be used to try various inputs and attempt to achieve desired outputs (a kind of manually controlled optimization familiar to spreadsheet users). Another approach is to create an explicit optimization model as a wrapper to the relevant set of CBAMs [Cimtalay, 2000]. Ideally such optimization models would be automatically created and executed, where feasible, based on their SysML model context.

[Burkhart, 2006] overviews how SysML compares with capabilities in existing system dynamics modeling and simulation technologies like Modelica. Modelica has clear semantics based on differential algebraic equations (DAEs) and discrete states. The above flap linkage tutorial and Part 1 do not include such examples, though several applications mentioned in Section 3.1 have them to some degree. Other examples may need to be developed to more fully demonstrate how and to what degree SysML 1.0 supports such capabilities. To match resources in tools like Modelica, SysML would need libraries of pre-defined building blocks including continuous systems. These can likely be added over time (or existing libraries might be wrapped in SysML). The UML foundation of SysML provides capabilities that Modelica lacks, such as information modeling and dependency tracking across models and model elements. We believe SysML provides a more complete solution for architectural and structural description of systems, whereas tools like Modelica excel in execution of equation-based behavior—thus, they are complementary technologies.

A related question is how does SysML support dynamic equations and other time-based simulations such as spring-mass-damper systems with time-varying loads. SysML parametrics can also handle such cases that are not static in nature. The spec states that time can be modeled as a property that other properties may be dependent on. A time reference can be established, e.g., by a global clock which produces continuous or discrete time value property. This global clock time property can be bound to parameters of the constraint equations.

4 Summary and Conclusions

This Part 2 paper describes SysML models for a simulation template tutorial. It demonstrates how SysML supports a form of simulation-based design (SBD) including executable parametrics based on composable object (COB) technology. It overviews concepts from the multi-representation architecture (MRA) for simulation templates that enables advanced CAD-CAE interoperability. Employing an object-oriented approach, the MRA defines natural partitions of engineering knowledge that occur between traditional design and analysis models.

This benchmark tutorial—for a flap linkage parts family—demonstrates how SysML and the MRA together support capabilities important to engineering design and analysis. Many of these capabilities are relevant to broader simulation and knowledge representation domains, including:

- Diversity of design information sources, analysis behaviors, analysis fidelities, solution methods, and solution tools.
- Modular, reusable analytical building blocks and fine-grained inter-model associativity.

This paper also introduces additional SysML and COB modeling concepts beyond Part 1, including packages, building block libraries, and requirements-verification-simulation interrelationships. Experiences to date reinforce our belief that SysML holds great promise as a unifying language for a diversity of models—from top-level system models to discipline-specific leaf-level models.

Acknowledgements & Author Biographies

See Part 1.

References[10]

Bajaj (2006) A Composable Knowledge Methodology for Efficient Analysis Problem Formulation in Simulation-based Design. PhD Dissertation Proposal, Woodruff School of Mechanical Engineering. Georgia Tech, Atlanta.

Burkhart RM (2006) Modeling System Structure and Dynamics with SysML Blocks. Frontiers in Design & Simulation Workshop[11], GIT PSLM Center, Atlanta.

Cimtalay S (2000) *The Enhanced Optimization Model Representation: An Object-Oriented Approach to Optimization*, Doctoral Dissertation, Georgia Tech, Atlanta.

Collier WB, Peak RS, Tamburini DT, Paredis CJJ, McGinnis LM (Jan. 2006) Teleconference communications.

CPDA (2003) Introduction to the Twelve-Fold Way™: The Latest Evolution of Shared Product Structure. Collaborative Product Development Associates, Stamford CT.

Fenves SJ (2004) CPM 2: A Revised Core Product Model for Representing Design Information. NISTIR 7185. National Institute of Standards and Technology, Gaithersburg MD.

Friedenthal S, Moore A, Steiner F (2006) OMG Systems Modeling Language (OMG SysML) Tutorial. INCOSE Intl. Symposium, Orlando.

GIT (2001) Design-Analysis Associativity Technology for the Boeing Product Simulation Integration (PSI) Structures Project. Project Web Page, Georgia Tech, *http://eislab.gatech.edu/projects/boeing-psi/*.

GIT (2007a) The Composable Object (COB) Knowledge Representation: Enabling Advanced Collaborative Engineering Environments (CEEs), Project Web Page, Georgia Tech, *http://eislab.gatech.edu/projects/nasa-ngcobs/*.

GIT (2007b) Capturing Design Process Information and Rationale to Support Knowledge-Based Design-Analysis Integration, Project Web Page, Georgia Tech, *http://eislab.gatech.edu/projects/nist-dai/*.

GIT (2007c) SysML Focus Area, Product & Systems Lifecycle Management Center, Georgia Tech, *http://www.pslm.gatech.edu/topics/sysml/*.

Larson WJ and Wertz JR ed. (1999) Space Mission Analysis and Design (SMAD), 3rd Ed. Microcosm, Inc.

OMG (2007a) OMG Systems Modeling Language (OMG SysML™) website: *http://www.omgsysml.org/*.

OMG (2007b) OMG Systems Modeling Language (OMG SysML™) Specification, *http://www.omg.org/*
SysML 1.0 Proposed Available Specification (PAS), OMG document [ptc/2007-02-03] dated 2007-03-23.

Peak RS, Fulton RE, Nishigaki I, Okamoto N (1998) Integrating Engineering Design and Analysis Using a Multi-Representation Approach. *Engineering w. Computers*, 14 (2) 93-114.

Peak RS, Scholand AJ, Tamburini DR, Fulton RE (1999) Towards the Ubiquitization of Engineering Analysis to Support Product Design. Invited Paper for Special Issue: Advanced Product Data Mgt. Supporting Product Life-Cycle Activities, *Intl. J. Comp. Applications Technology* 12(1): 1-15.

Peak RS (2000) X-Analysis Integration (XAI) Technology. EIS Lab Report EL002-2000A, Georgia Tech, Atlanta.

Peak RS (2002) Techniques & Tools for Product-Specific Analysis Templates Towards Enhanced CAD-CAE Interoperability for Simulation-Based Design & Related Topics. Intl. Conf. Electronic Packaging. JIEP/ IMAPS Japan, IEEE CPMT. Tokyo.

Peak RS (2003) Characterizing Fine-Grained Associativity Gaps: A Preliminary Study of CAD-E Model Interoperability. ASME DETC Paper No. CIE-48232, Chicago.

Peak RS (2005) Achieving Fine-Grained CAE-CAE Associativity via Analyzable Product Model (APM)-based Idealizations. Invited Presentation, Developing a Design/Simulation Framework: CPDA Design & Simulation Council Workshop, Atlanta. *http://eislab.gatech.edu/pubs/seminars-etc/2005-cpda-dsfw-peak/*

Peak RS, Tamburini DR, Paredis CJJ, McGinnis LF (2006) GIT SysML Work Update. Presentation to OMG SE DSIG, Tampa FL. *http://eislab.gatech.edu/pubs/seminars-etc/2006-02-omg-se-dsig-peak/*

Peak RS, Burkhart RM, Friedenthal SA, Wilson MW, Bajaj M, Kim I (2007) Simulation-Based Design Using SysML—Part 1: A Parametrics Primer. INCOSE Intl. Symposium, San Diego.

Tamburini DR (1999) *The Analyzable Product Model Representation to Support Design-Analysis Integration*, Doctoral Dissertation, Georgia Tech, Atlanta.

Tamburini DR (2006) Defining Executable Design & Simulation Models using SysML. Frontiers in Design & Simulation Workshop, GIT PSLM Center, Atlanta.

Tamburini DR, Deren G (2006) Mechatronics Interoperability Project for Systems (MIPS). PLM World Users Conference, Long Beach CA.

Wilson MW (2000) *The Constrained Object Representation for Engineering Analysis Integration*. Masters Thesis, Georgia Tech, Atlanta.

Zeng S (2004) Knowledge-Based FEA Modeling for Highly Coupled Variable Topology Multi-Body Problems. PhD Dissertation, Georgia Tech, Atlanta.

Zeng S, Peak RS, Xiao A, Sitaraman SK (2007—to appear) A Knowledge-Based FEA Modeling Method for Highly Coupled Variable Topology Multi-body Problems. *Engineering with Computers*.

[10] Some references are available at *http://www.pslm.gatech.edu/* and *http://eislab.gatech.edu/*.
[11] *http://www.pslm.gatech.edu/events/frontiers2006/*

Appendix A

Table A1: Statistics[12] for analysis template test cases. [Wilson, 2000]

Test Cases **Libraries Used** **No. of Entities, Attributes, Relations**

Structure (COS)				COB Libraries Used	Entities	Attributes		Relations			
						Total	Aggregate	Total	Oneway	Aggregate Operation	Aggregate Instance
general(lib)	geometry.cos				4	11		3			
	abbs.cos				108	68		30			
	apm.cos			lib\geometry.cos	12	34		22			
	materials.cos				3	9		1			
product specific	flaplink	apm.cos		lib\apm.cos lib\materials.cos	1	11		10			
		cbams.cos		lib\abbs.cos apm.cos	5	25		36	2		
	pwa/b	apm.cos			77	152	8	19		9	
		cbams.cos		lib\abbs.cos apm.cos	5	21		23	3		5
	airplane	lib	abbs.cos	lib\apm.cos	24	39		12	3		
			cbams.cos	lib\geometry.cos lib\apm.cos airplane\lib\abbs.cos	2	7		16			
			fastener.cos		3	7					
			materials.cos		1	38					
		bikeframe	apm.cos	lib\geometry.cos lib\apm.cos airplane\lib\materials.cos airplane\lib\fastener.cos	4	23		20			
			cbams.cos	airplane\lib\cbams.cos airplane\bikeframe\apm.cos	2			20			
	electrical chip package (cp	lib	pwb_board.cos		13	21	2	5			
		bga (ball grid array)	apm.cos	lib\geometry.cos cp\lib\pwb_board.cos	53	177	6	103		3	22
			cbams.cos	lib\abbs.cos cp\bga\apm.cos	1	12	4	19			15
		qfp(quad flat pack)	apm.cos	lib\geometry.cos cp\lib\pwb_board.cos	25	76	1	18			2
			cbams.cos	lib\abbs.cos cp\qft\apm.cos	1	12	4	19			15
Totals					344	753	25	376	8	12	59

[12] Currently not all COBs in abbs.cos are fully developed. Many exist as placeholders for future work. Approximately one-fifth of the COBs are fully usable, thus a more accurate COB entity count in abb.cos would be ~25 vs. the 108 shown. This change gives the 260 total types of COB entities identified in the text above.

Table A2: Diversity demonstrated in test cases. [after Wilson, 2000]

	Test Case Analysis Templates						
Target Characteristics	Flap Link CBAMs			PWA/B CBAMs		Aerospace CBAMs	Electrical Chip Package CBAMs
Diversity Dimensions							
Product Domain	airframe			printed circuit board (PWA/B)		airframe	chip package
CAD Tools	CATIA (MCAD)			Mentor Graphics (ECAD) *XaiTools PWA/B*		CATIA (MCAD)	*XaiTools Chip Package (XCP)*
Discipline	structural			thermo-mechanical		structural	thermal
Behavior	deformation (extension)		deformation (torsion)	deformation (warpage)		lug & fitting ultimate shear, bending shear	temperature
Fidelity	extensional rod (1D, linear)	plane stress body (2D, linear)	torsional rod (1D, linear)	thermal bending (1D, linear)	plane strain body (2D, linear)	1.5D	thermal body (3D, linear)
Solution Method (and Tools)	formula-based (Mathematica)	FEA (Ansys, Patran, Abaqus), formula-based (Mathematica)	formula-based (Mathematica)	formula-based (Mathematica)	FEA (Ansys, Cadas), formula-based (Mathematica)	formula-based (Mathematica)	FEA (Ansys), formula-based (Mathematica); custom cob-based mesh algorithm
Directionality	multi	oneway (partially multi)	multi	multi	oneway (partially multi)	oneway (partially multi)	oneway (partially multi)
COB Usage Characteristics							
Product Design Info Usage	detailed design (COI via CATIA interface)			detailed design (STEP AP210 -Part 21 via Mentor Graphics interface)		detailed design (COI via CATIA interface)	preliminary design (COI via XCP design tool)
Automation	fully automated			fully automated		fully automated	fully automated

Table A3: Example reuse of modular building blocks. [Wilson, 2000]

Structure (COS)	Where used
1D Linear Elastic Model (ABB)	Extensional Rod ABB Torsional Rod ABB
Margin of Safety ABB	1D Linkage Extensional Flaplink CBAM for stress 1D Torsional Extensional Flaplink CBAM for stress 1D Torsional Extensional Flaplink CBAM for twist 2D Plane Stress flaplink CBAM for stress 2D linkage extensional flaplink CBAM for deformation 1D PWB Thermal Bending for warpage 2D PWBThermal Bending for warpage 1.5D Lug CBAM for stress
Flaplink APM	Linkage Extensional CBAM Linkage Plane Stress CBAM Linkage Torsional CBAM
BikeFrame APM	Lug Axial/Oblique; Ultimate/Shear CBAM Fitting Bending/Shear CBAM
PWA/B APM	Thermal Bending CBAM 6 Layer Plain Strain CBAM N Layer Plain Strain CBAM
EBGA ChipPackage APM	EBGA Thermal Resistance CBAM
PBGA ChipPackage APM	PBGA Thermal Resistance CBAM Thermal Stress CBAM
QFP ChipPackage APM	Thermal Resistance CBMA