

**Interfacing Geometric Design Models  
to Analyzable Product Models  
with Multifidelity and Mismatched Analysis Geometry**

A Thesis  
Presented to  
The Academic Faculty

by

Ashok Chandrasekhar

In Partial Fulfillment  
of the Requirements for the Degree  
Master of Science in Mechanical Engineering

Georgia Institute of Technology  
December 1999

*Reference:* A. Chandrasekhar (1999) *Interfacing Geometric Design Models to Analyzable Product Models with Multifidelity and Mismatched Analysis Geometry*. Masters Thesis, Georgia Institute of Technology, Atlanta.

*Revision:* This document is a February 29, 2000 revision for web delivery at <http://eislabs.gatech.edu/>. It may include minor updates and formatting changes versus the original document. Corrections to known pdf creation caveats are planned for future revisions.

**Interfacing Geometric Design Models  
to Analyzable Product Models  
with Multifidelity and Mismatched Analysis Geometry**

Approved:

---

Robert E. Fulton, Chairman

---

Russell S. Peak, Co-Chairman

---

Charles M. Eastman

---

David W. Rosen

Date Approved: \_\_\_\_\_

## ***Dedication***

*To my dearest father and mother,*

*M.R. Chandrasekhar and Mahalakshmi Chandrasekhar*

## ACKNOWLEDGEMENTS

During the course of my study, I have had the support of several people. I would like to express my heart-felt gratitude to them for the same.

- My advisor, mentor, and guide, Dr. Robert E. Fulton, for sharing his invaluable technical experiences with me and for his guidance in my research work. He has always put me on meaningful and industry-related projects during my thesis study. His guidance and confidence in my abilities have been an amazing source of encouragement and strength. I have learnt much more than just doing good research from him. My heart-felt thanks and gratitude will always be towards him for making my graduate study an enjoyable and fruitful experience.
- My Co-advisor, Dr. Russell S. Peak, for his tutelage in my research work. Most importantly, his exemplary meticulousness and suggestions for improvement have helped me contribute more meaningfully towards my research work. Working with him has been a wonderful learning experience from which I will always benefit, and I am ever grateful to him for imparting his technical knowledge and much more.
- Dr. Rosen and Dr. Eastman for being a part of my graduate committee. I do feel privileged to have them as a part of my graduate committee. I'd like to thank them immensely for their valuable time and suggestions. Their respective CAD courses have helped me extensively in my research work. Dr. Eastman and I have had several meaningful discussions related to CAD operations and I thank him for the same.
- Dr. Mark Hale for enabling me to use his Tk/tcl based Interpretive CATGEO load module, which considerably simplified my work. The many discussions that I have had with him have aided me a lot in my thesis study.

- Martin Prather, Boeing, for his help in acquiring material to understand typical analysis procedures in the aerospace industry. I'd also like to thank Andre Blot (Dassault Systems), Maxime Albi (IBM) and Brent Graham (Boeing) for their help in understanding the capabilities of CATIA for aerospace design components.
- Tord Dennis, College of Engineering, Georgia Institute of Technology, for his input on the capabilities of IDEAS and Pro/Engineer (<http://www.cad.gatech.edu>).
- The members of the Engineering Information Systems Lab - Dennis Ma, Diego Tamburini, Donald Koo, Sai Zeng, Selçuk Cimentalay, Tal Cohen, Chien Hsiung, Haruko Peak, M.C. Ramesh, Andy Scholand, Miyako Wilson and Xiaoling He for their feedback during the progress of my thesis work. I'd like to specially thank Dennis Ma for his help with the Flap Link CAD models. I would also like to thank Donna Rogers for her continuous help with numerous administrative issues during my study.
- My beloved father and mother for their love and affection, and for being the guiding light in my life.
- My dear brothers, M.C. Ramesh, M.C. Karthik, sister-in-law, Sujatha Ramesh and little sister, Anita Chandrasekhar for their love and support.
- My friends at Georgia Tech, Anurag Gupta, Karthik Nagapudi, Lakshmi Rajagopal, Lekha Bhargavi, Madhuri Ganta, Prameela Susarla, Priyanka Agarwal, Rajesh Chanpura, Rajiv Dunne, Reena Agarwal, Tejas Sukhadia and Vidya Krishnan for all their time, support and delicious food.
- My wife, Reema, for her unconditional love and affection.

# TABLE OF CONTENTS

<b>LIST OF TABLES.....</b>	<b>IX</b>
<b>LIST OF FIGURES.....</b>	<b>X</b>
<b>SUMMARY.....</b>	<b>XII</b>
 <b>CHAPTER I.....</b>	 <b>1</b>
<b>INTRODUCTION .....</b>	<b>1</b>
MOTIVATION FOR THE STUDY .....	1
 <b>CHAPTER II.....</b>	 <b>6</b>
<b>BACKGROUND INFORMATION .....</b>	<b>6</b>
2.1 COMPUTER AIDED DESIGN (CAD).....	6
2.1.1 <i>Types of geometric modelers.....</i>	9
2.1.2 <i>Interfacing capabilities of CAD systems.....</i>	12
2.2 ANALYSIS METHODS.....	13
2.2.1 <i>The finite element method (FEM) .....</i>	14
2.3 USE OF GEOMETRY IN ANALYSIS MODELS .....	15
2.3.1 <i>Creation of geometric models for engineering analysis .....</i>	16
2.3.2 <i>Common geometric attributes needed for analysis idealizations.....</i>	23
 <b>CHAPTER III.....</b>	 <b>27</b>
<b>RELEVANT RESEARCH .....</b>	<b>27</b>
3.1 THE ANALYZABLE PRODUCT MODEL (APM) .....	27
3.2 THE MULTI-REPRESENTATION ARCHITECTURE (MRA).....	32
3.3 XAITOOLS: ANALYSIS INTEGRATION TOOLKIT.....	35
3.4 STANDARDS FOR EXCHANGING GEOMETRIC INFORMATION BETWEEN CAE SYSTEMS .....	37
3.4.1 <i>Introduction .....</i>	37
3.4.2 <i>CAD-FEA integration with STEP AP209 technology.....</i>	39
3.5 PROBLEMS/GAPS THAT NEED TO BE ADDRESSED .....	44

<b>CHAPTER IV .....</b>	<b>45</b>
<b>FOCUS OF THIS STUDY .....</b>	<b>45</b>
4.1 THESIS OBJECTIVES .....	45
<b>CHAPTER V.....</b>	<b>450</b>
<b>TECHNIQUE EMPLOYED .....</b>	<b>50</b>
5.1 GEOMETRIC INTERFACE TECHNIQUE .....	50
5.2 TAGGING OF GEOMETRIC ENTITIES IN CAD MODELS.....	53
5.3 GENERAL FLOWCHART FOR A CUSTOMIZED CAD API ADAPTER .....	55
5.4 CAPABILITIES SUPPORTED BY CAD SYSTEMS .....	59
5.5 APPROACHES TO EXTRACTING TAGGED GEOMETRIC INFORMATION .....	61
5.5.1 <i>Approach 1 : Tagging wireframe entities and CSG primitives</i> .....	61
5.5.2 <i>Approach 2 : Tagging dimension entities</i> .....	63
5.5.3 <i>Approach 3 : Tagging parameter entities</i> .....	66
5.6 CAD SYSTEM SUPPORT FOR TAG EXTRACTION APPROACHES .....	69
<b>CHAPTER VI.....</b>	<b>71</b>
<b>TEST CASES .....</b>	<b>71</b>
6.1 INTRODUCTION.....	71
6.1.1 <i>Back plate</i> .....	71
6.1.2 <i>Flap link</i> .....	72
6.1.3 <i>Bike frame</i> .....	73
6.1.4 <i>Implementation of the Geometric Interface Technique</i> .....	76
6.2 TEST CASES FOR TAGGING GEOMETRIC ENTITIES.....	78
6.2.1 <i>Back plate (partial tagging)</i> .....	78
6.2.2 <i>Back plate (complete tagging)</i> .....	81
6.2.3 <i>Discussion on the approach &amp; its implementation</i> .....	83
6.3 TEST CASES FOR TAGGING DIMENSION ENTITIES.....	84
6.3.1 <i>Back plate</i> .....	84
6.3.2 <i>Flap link</i> .....	88
6.3.3 <i>Bike frame</i> .....	92
6.4 TEST CASES FOR TAGGING PARAMETERS .....	95
6.4.1 <i>Back plate</i> .....	96
6.4.2 <i>Flap link</i> .....	99
6.5 GEOMETRIC INFORMATION USED IN ANALYSES .....	102
6.5.1 <i>Flap link analyses</i> .....	102
6.5.2 <i>Bike frame inboard beam analysis</i> .....	113
6.6 GEOMETRIC ATTRIBUTES RETRIEVED FOR TEST CASES .....	118
6.7 COMPARISON OF APPROACHES USED IN CATIA TEST CASES .....	121
6.8 APPROACHES SATISFYING THE THESIS OBJECTIVES .....	123

<b>CHAPTER VI.....</b>	<b>125</b>
<b>CONCLUDING REMARKS .....</b>	<b>125</b>
7.1 CONCLUSIONS AND SUMMARY OF CONTRIBUTIONS .....	125
7.2 CONTRIBUTIONS.....	128
7.3 RECOMMENDATIONS .....	130
 APPENDIX A.....	 <b>132</b>
ANALYZABLE PRODUCT MODEL SCHEMAS AND MATERIAL MODELS .....	132
APPENDIX B.....	<b>146</b>
REQUEST FILES FOR THE TEST CASES .....	146
APPENDIX C.....	<b>151</b>
RESPONSE FILES FOR TEST CASES.....	151
APPENDIX D.....	<b>160</b>
SOLVED APM FILES FOR TEST CASES .....	160
APPENDIX E.....	<b>163</b>
XAiTOOLS CATIA ADAPTER - USER’S MANUAL .....	163
 REFERENCES .....	 <b>171</b>



# LIST OF TABLES

TABLE 1: FUNCTIONS NEEDED TO EXTRACT ATTRIBUTES OF DIFFERENT TYPES OF CAD ENTITIES .....	57
TABLE 2: CAPABILITIES SUPPORTED BY COMMON CAD SYSTEMS.....	59
TABLE 3: GEOMETRIC EXTRACTION APPROACHES SUPPORTED BY CAD SYSTEMS .....	69
TABLE 4: GEOMETRIC ATTRIBUTES, DIMENSIONS AND PARAMETERS THAT WERE RETRIEVED FROM THE TEST CASES .....	120
TABLE 5: COMPARISON OF THE DIFFERENT APPROACHES USED IN CATIA TEST CASES.....	122
TABLE 6: TABLE COMPARING THE THREE APPROACHES AGAINST THE OBJECTIVES OF THIS THESIS .....	124

# LIST OF FIGURES

FIGURE 1: AEROSPACE CAD MODEL WITH MULTI-FIDELITY ANALYSIS MODELS .....	2
FIGURE 2: AEROSPACE ANALYSIS MODEL REQUIRING IDEALIZED PARTIAL CAD MODEL (PART FEATURE LEVEL) .....	3
FIGURE 3: MISMATCH BETWEEN DESIGN MODEL GEOMETRY AND ITS ANALYSIS MODEL GEOMETRIES .....	4
FIGURE 4: WIREFRAME REPRESENTATION OF GEOMETRY .....	7
FIGURE 5: SURFACE REPRESENTATION OF GEOMETRY .....	7
FIGURE 6: CSG REPRESENTATION OF GEOMETRY .....	8
FIGURE 7: BOUNDARY REPRESENTATION OF GEOMETRY .....	8
FIGURE 8: A RECTANGLE WIREFRAME REPRESENTATION IN A CAD MODEL .....	10
FIGURE 9: TWO-DIMENSIONAL DETAIL REMOVAL .....	18
FIGURE 10: SIMPLIFIED MODELS FOR ANALYSIS .....	19
FIGURE 11 : MULTIFIDELITY AND MULTI-DISCIPLINE IDEALIZATIONS .....	21
FIGURE 12: VARIOUS DIMENSIONAL REDUCTIONS .....	22
FIGURE 13: ANALYZABLE PRODUCT MODEL TECHNIQUE .....	28
FIGURE 14: FLAP LINK TEST CASE APM DEFINITION FILE .....	30
FIGURE 15: THE MULTI-REPRESENTATION ARCHITECTURE FOR DESIGN ANALYSIS INTEGRATION .....	33
FIGURE 16: XAITOOLS ARCHITECTURE FOR AN AEROSPACE-ORIENTED ENVIRONMENT .....	35
FIGURE 17: SCOPE OF AP209 (HUNTEN 1997) .....	40
FIGURE 18: GEOMETRIC INTERFACE TECHNIQUE FOR INTEGRATING ANALYZABLE PRODUCT MODELS WITH CAD GEOMETRIC MODELS .....	50
FIGURE 19: A PORTION OF THE APM REQUEST MODEL IN COB INSTANCE (COI) FORMAT .....	52
FIGURE 20: RESPONSE FILE GENERATED BY THE API ADAPTER .....	53
FIGURE 21: TAGGING OF GEOMETRY IN A CAD MODEL (BACK PLATE) .....	54
FIGURE 22: SIMPLIFIED FLOWCHART FOR A CUSTOMIZED ADAPTER (BLOCK 3 OF FIGURE 18) .....	56
FIGURE 23: TAGGED GEOMETRIC ENTITIES OF THE BACK PLATE .....	62
FIGURE 24: TAGGED DIMENSION ENTITIES IN THE BIKE FRAME CAD MODEL .....	64
FIGURE 25: A PARAMETERIZED CAD MODEL WITH ALL ITS PARAMETERS .....	67
FIGURE 26: BACK PLATE MODEL .....	72
FIGURE 27: FLAP LINK DESIGN MODEL .....	73
FIGURE 28: INBOARD BEAM OF THE WING FLAP SUPPORT ASSEMBLY .....	74
FIGURE 29: BULKHEAD ATTACHMENT POINT ON INBOARD BEAM LEG1 .....	75
FIGURE 30: DIAGONAL BRACE ATTACH POINT .....	75
FIGURE 31: GEOMETRIC INTERFACE TECHNIQUE FOR OBTAINING GEOMETRIC DATA .....	76
FIGURE 32: TAGGING OF SOME GEOMETRIC ENTITIES OF THE BACK PLATE .....	78
FIGURE 33: PORTION OF THE REQUEST AND RESPONSE ‘COI’ FILES OF THE PARTIALLY TAGGED BACK PLATE .....	80
FIGURE 34: TAGGING OF THE GEOMETRIC ENTITIES OF THE BACK PLATE .....	81
FIGURE 35: PORTION OF THE REQUEST AND RESPONSE COI FILES OF THE BACK PLATE (COMPLETE TAGGING) .....	82
FIGURE 36: LABELED DIMENSION ENTITIES IN DRAFT VIEWS OF THE BACK PLATE .....	85
FIGURE 37: PORTION OF THE REQUEST AND RESPONSE COI FILES OF THE BACK PLATE .....	86
FIGURE 38: PORTION OF THE SOLVED APM .COI FILE FOR THE BACK PLATE .....	87
FIGURE 39: TAGGED DIMENSION ENTITIES IN DRAFT VIEWS OF THE FLAP LINK .....	88
FIGURE 40: TAGGED CRITICAL CROSS SECTION OF THE FLAP LINK .....	89
FIGURE 41: A PORTION OF THE REQUEST AND RESPONSE ‘COI’ FILES FOR THE DIMENSION BASED TAGGING OF THE FLAP LINK MODEL .....	90
FIGURE 42: PORTION OF THE SOLVED APM .COI FILE FOR THE FLAP LINK .....	91
FIGURE 43: TAGGED DIMENSION ENTITIES FOR TWO BIKE FRAME FEATURES .....	93
FIGURE 44: PORTION OF THE REQUEST AND RESPONSE ‘COI’ FILES FOR DIMENSION BASED TAGGING BIKE FRAME BULKHEAD ATTACH POINT .....	94

FIGURE 45: PARAMETERS OF THE BACK PLATE DESIGN MODEL .....	96
FIGURE 46: PORTION OF THE REQUEST ‘COI’ AND RESPONSE FILES OF THE BACK PLATE DESIGN USING THE PARAMETRIC APPROACH.....	98
FIGURE 47: IMPORTED FILE OF THE BACK PLATE DESIGN MODEL (CATIA IMPORT FORMAT) .....	98
FIGURE 48: FLAP LINK TAPER ANGLE DEFINED AS A MEASURED PARAMETER.....	99
FIGURE 49: PORTION OF THE REQUEST AND RESPONSE ‘COI’ FILES OF THE FLAP LINK DESIGN USING THE PARAMETRIC APPROACH.....	101
FIGURE 50: IMPORTED FILE OF THE FLAP LINK IN CATIA FORMAT .....	101
FIGURE 51: FLEXIBLE DESIGN-ANALYSIS INTEGRATION USING MRA COBs .....	103
FIGURE 52: PRODUCT ATTRIBUTES AND IDEALIZED ATTRIBUTES OF THE FLAP LINK EXTENSION ANALYSIS (TAMBURINI 1999) .....	104
FIGURE 53: REPRESENTING A FLAP LINK ANALYSIS AS A CBAM: LINKAGE EXTENSIONAL MODEL.....	105
FIGURE 54: RESULTS OF FORMULA BASED FLAP LINK EXTENSION ANALYSIS.....	106
FIGURE 55: COB LEXICAL FORM FOR LINKAGE EXTENSIONAL MODEL CBAM .....	107
FIGURE 56: CBAM USAGE OF APM-BASED IDEALIZATIONS .....	107
FIGURE 57: HIGHER FIDELITY FLAP LINK CBAM: LINKAGE PLANE STRESS MODEL .....	108
FIGURE 58: PREPROCESSING FILE (PREP7) SENT TO ANSYS (PARTIAL) .....	109
FIGURE 59: SOLVED FINITE ELEMENT MODEL OF THE FLAP LINK ( ANSYS ).....	110
FIGURE 60: FLAP LINK TORSIONAL CBAM .....	111
FIGURE 61: RESULTS FOR THE FLAP LINK TORSION ANALYSIS .....	112
FIGURE 62: CAD AND ANALYSIS ATTRIBUTES FOR THE BULKHEAD FITTING ANALYSIS.....	114
FIGURE 63: TYPICAL DESIGN MANUAL DESCRIPTION OF GENERAL FITTING ANALYSES WITHOUT DESIGN ASSOCIATIVITY.....	115
FIGURE 64: TYPICAL CURRENT PRACTICE WITHOUT EXPLICIT DESIGN ASSOCIATIVITY .....	116
FIGURE 65: BIKE FRAME BULKHEAD FITTING ANALYSIS: IMPLEMENTATION AS A CBAM (CONSTRAINT SCHEMATIC INSTANCE VIEW) .....	117
FIGURE 66: COB-BASED BULK HEAD FITTING ANALYSIS RESULTS WITH CAD ASSOCIATIVITY .....	118
FIGURE 67: BLOCKS THAT CONSTITUTE THE DESIGN AND ANALYSIS INTEGRATION SCENARIO .....	128
FIGURE 68 : METHODOLOGY FOR OBTAINING ATTRIBUTES OF GEOMETRIC ENTITIES .....	165

## SUMMARY

CAD design models are typically analyzed across various disciplines such as structural analysis, thermal analysis and vibration analysis. Further, for a given design model, each analysis discipline may require multiple analysis models. Thus, while every mechanical engineering component typically has one associated CAD model, it can have many associated analysis models. A key step in creating analysis models is to abstract the geometry of the structure that is to be analyzed. Most often, the geometry of the CAD design model is complicated and needs to be simplified and/or idealized for each analysis discipline. Much of this analysis model geometry is often common to, and/or derivable from its CAD model. In cases where there is a high mismatch between CAD geometry and analysis model geometry, the present state of engineering analysis typically requires the analyst to re-create this common and related analysis geometry from scratch in the analysis system. Thus, the associativity between the design model and its analysis models is not explicitly captured.

This study has developed a technique that enables the analyst to selectively choose and extract the attributes of desired geometric entities from CAD models, for the purpose of creating its analysis models. The capabilities of different CAD systems, namely, IDEAS, CATIA and Pro/Engineer were studied, and the technique was generalized for typical modern CAD systems. The technique was implemented with the CATIA CAD system and tested with several mechanical and aerospace components.

Results show that this technique enables explicit design-analysis associativity and facilitates the engineering analyst to create different analysis model geometries with varying degrees of idealization from the same CAD model.

# CHAPTER I

## INTRODUCTION

### 1.1 Motivation for the study

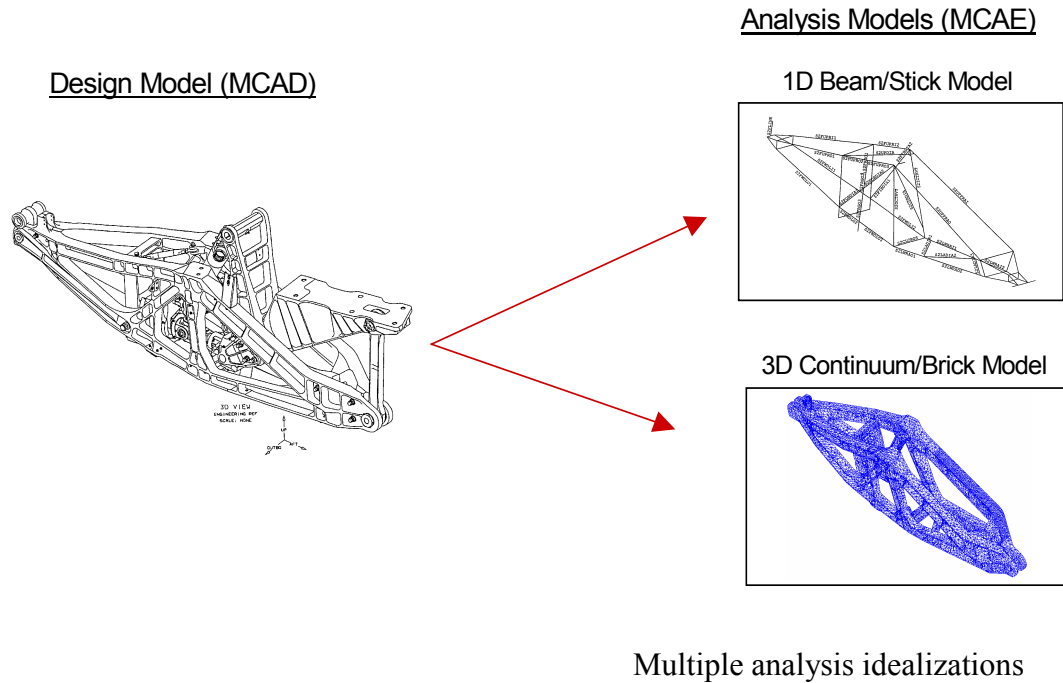
Engineering design models are typically simulated and checked for safety against various types of loads and conditions. The simulation serves to confirm long before the product goes into service that the design would perform adequately and satisfy design requirements (Arabshahi, Barton et al. 1993). This simulation that predicts the physical behavior of an engineering component is commonly termed ‘engineering analysis’.

The analysis solution method used may be of different types, including finite element analysis (FEA) and formula based analysis. Further, the discipline of analysis may be structural, thermal, vibration etc. Design models are usually analyzed across various analysis disciplines and analysis types. For a given design model, once the type and discipline of analysis is selected, many analysis models with varying levels of simplification may be defined.

Figure 1 shows a CAD model of a typical aerospace assembly on the left, while two structural analysis models with varying levels of simplification are shown to the right. At the upper right hand corner is a model composed of one-dimensional beams while to the lower right hand corner is a model composed of three-dimensional finite element bricks. Such simplifications are commonly termed ‘idealizations’.

Thus, every mechanical engineering component typically has one main CAD model associated with it; however, the component may have many analysis models associated with it. Most often, the geometry of the design is complicated and needs to be idealized for the purpose of analysis as indicated in Figure 1.

Sometimes, the analysis may have to be carried out on just a portion of the whole design model. Figure 2 shows the CAD model of an aerospace assembly on the left, while it shows an analysis model that requires idealized geometry from a portion of the CAD model on the right.

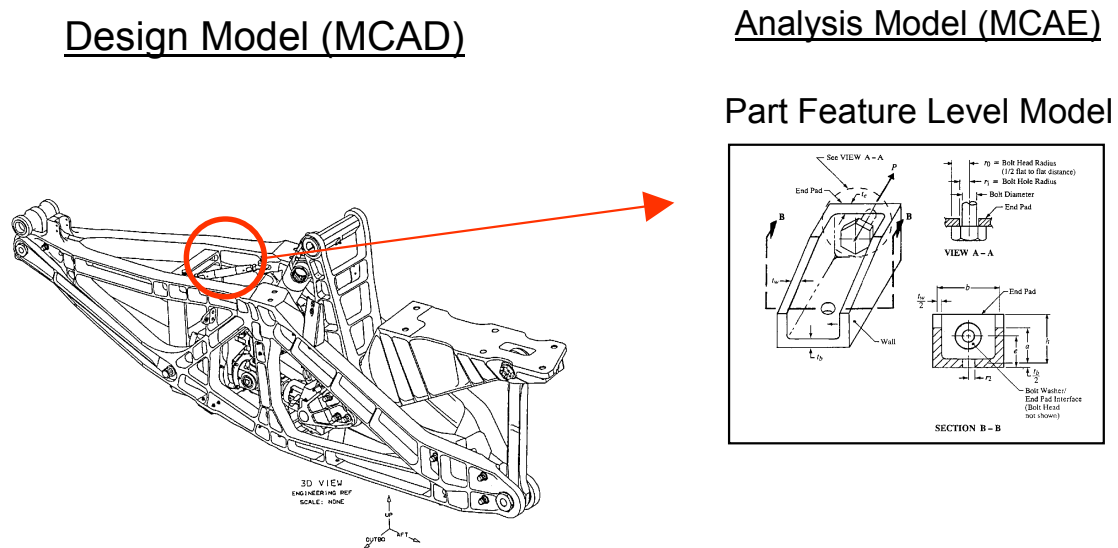


**Figure 1: Aerospace CAD model with multi-fidelity analysis models**

Thus, we see that for the same engineering component, the geometric information in the CAD design model may be very different from the geometric information in its analysis models. Much of the information that is needed to create the analysis models of a component is often common to, and/or derivable from its CAD model. However, in such cases of high design-analysis geometry mismatch, the present state of engineering analysis typically requires the analyst to re-create this common and related analysis geometry from scratch in an analysis system. The primary reason for this duplication of effort lies in the difficulties associated with transforming the full detailed design geometry held in a CAD modeler into the abstracted and simplified form of that geometry

required for analysis (Arabshahi, Barton et al. 1993). An ANSYS press release in states the following: “It is no secret that CAD models are driving more of today's product development processes. However, with the growing number of design tools on the market, the interoperability gap with downstream applications, such as finite element analysis, is a very real problem. As a result, CAD models are being recreated at unprecedented levels” (Hemmelgarn 1999). Also, Hale argues that the geometry model is the truth model (a representation of reality) used by analyses and it is important not only to simplify the transformation process of design geometry to analysis model geometry, but to also maintain consistency of interactions and to synchronize them with the design model (Hale, Craig et al. 1999).

There is presently no general way by which an analyst can selectively choose and extract the specific geometric entities that he/she needs from a CAD model in order to create its many diverse analysis models. This is particularly true in cases where there is a large mismatch in the geometry between the two types of models, as shown in Figure 3.



**Figure 2: Aerospace analysis model requiring idealized partial CAD model (part feature level)**





This study attempts to develop a methodology which enables the analyst to selectively choose and extract desired geometric entities from a CAD system for the purpose of creating geometric analysis models. Identifying the type of information that is typically needed for creating analysis geometric models from a CAD model forms an important aspect of this study. Mechanical engineering components have been the primary focus of this study. This study has focused on providing flexibility in creating different geometric analysis models with varying degrees of simplification from the same CAD model. It has also addressed enabling the analyst to selectively analyze just a portion of the whole design model.

The capabilities of different CAD systems have been studied, and a general and simple methodology has been developed in order to address the above problems and to bridge the gap between design geometry and analysis geometry.

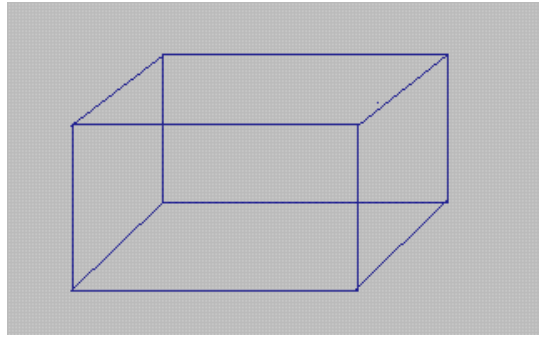
## CHAPTER II

### BACKGROUND INFORMATION

#### 2.1 Computer Aided Design (CAD)

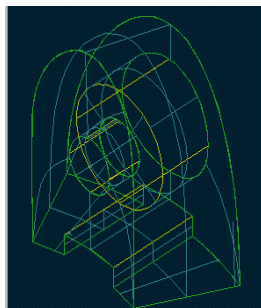
When we construct a CAD model of an engineering part, we create a substitute – a representation of that part. The component may already exist physically or it may be the design of some as yet non-existent engineering component. An effective model is usually easier to test and analyze than the actual object, and it responds, within limits, in the same way as the actual object. Creating an effective model means to give shape or form to it. CAD systems use the principles of geometric modeling to create a precise mathematical description of the shape of a real object (Mortensen 1997). Today's modeling systems can prepare one or more types of the following model representations: wireframe, surface or solid models.

A wireframe model may be two-dimensional or three-dimensional and consists of geometric data in the form of point locations and curves. A wireframe model cannot provide any surface or volumetric data for any subset of the model. Hence, if a point is not located on, or associated with a curve or a point, it has no relevance to the wireframe model and vice versa. Although the wireframe model is not a complete representation of an object, it serves as a viable function for mechanical drawings. A finite element mesh consisting of beam elements can be derived directly from a wireframe model (Finnigan, Kela et al. 1989). Figure 4 shows a simple wireframe representation of an object (Hoimyr 1996).



***Figure 4: Wireframe representation of geometry***

A surface model extends the coverage of wireframe models to include surfaces bounded by curves. These models can be rendered by an image synthesis device, but automatically determining the internal locus of points that constitute a physically realizable model is in general a difficult task. A surface model may be used to generate a finite element mesh comprising of plate and shell elements (Finnigan, Kela et al. 1989). Figure 5 shows a surface representation of an object (Hoimyr 1996).

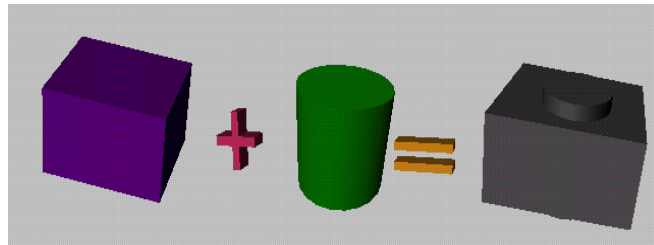


***Figure 5: Surface representation of geometry***

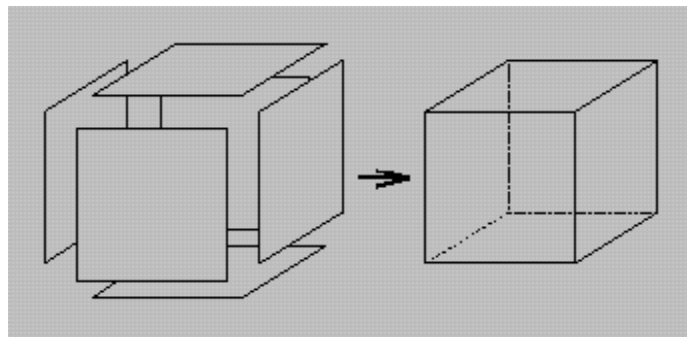
A solid model contains sufficient information about the geometry to determine the internal volume and composition of a model. It is possible to determine the interior region of a model, and therefore its mass properties can be computed in this

representation. A solid model may be used to generate a finite element mesh comprising three-dimensional solid elements (Finnigan, Kela et al. 1989). Two approaches are commonly used in solid modeling. The first is to use constructive solid geometry (CSG) where the model is described by a set of geometric primitives and boolean operators (union, intersection, difference) defining the action that the modeler will take with each primitive. The model is stored in a binary tree structure with leaves as primitives and boolean operators as nodes of a tree (Finnigan, Kela et al. 1989). Figure 6 shows the CSG solid representation (Hoimyr 1996).

The second approach is to use a boundary representation (B-rep) where the model is defined in terms of its evaluated boundary (Finnigan, Kela et al. 1989). Figure 7 shows a B-rep solid model (Hoimyr 1996).



***Figure 6: CSG representation of geometry***



***Figure 7: Boundary representation of geometry***

### 2.1.1 Types of geometric modelers

In order to utilize and take advantage of the geometric information that resides in a CAD system for the purpose of analysis, it is important that we understand the types of geometric modelers that are used by different CAD systems.

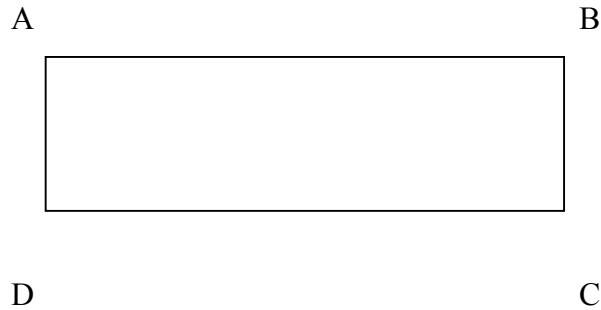
Geometric modeling in CAD systems, in a broad sense, is a two step process, namely, creation of geometry and modification of geometry. There are essentially two types of geometric modelers that are in use today, namely, explicit modelers and constraint-based or parametric modelers.

#### 2.1.1.1 Explicit modelers

An explicit modeler is one in which existing geometry generally has to be deleted before any modifications are made on it. This is because an explicit modeler does not keep a track of the order in which geometry was created (i.e. the history of construction of the geometry is unknown). For example, in Figure 8, reducing line AB by 50mm would require moving line AD or line BC to a new location of AB. Therefore, the relations are not explicit and have to be maintained by the user. Ex. point 'B' on lines AB and BC have to be co-incident (DASSAULT 1997).

#### 2.1.1.2 Parametric/Constraint based modelers

When a modification is required in a constraint based/parametric modeler, all that the user has to do is to change the desired constraint. Any affected geometry, because its history and relations are known, will be updated to reflect how it changes, based on the modifications of the first element's constraint.



***Figure 8: A rectangle wireframe representation in a CAD model***

For example, in Figure 8, reducing the length of line AB by 50mm in an explicit modeler would require moving line AD or line BC to a new location of AB. Whereas, in a constraint-based modeler, the lines would automatically be trimmed.

A constraint-based modeler performs operations based on parametric relationships. These relationships can be in two forms: a valuated relationship or a topological relationship. These parametric relationships are based on constraints that have been applied to geometric elements. For example, a parallelism condition between planes is a parametric relationship. Some of the important terminology that is used in parametric modeling is explained below<sup>1</sup>.

#### **a) Valuated relationships**

These relationships have numerical values attached to them (known as parameters) and may be defined by the user of the system. This type of relationship may be divided into:

- i ) An internal parameter (Ex. radius, diameter)
- ii ) An offset

iii ) An angle

## **b) Topological relationships**

These relationships refer to those that are between geometric entities. These relationships include tangency, parallelism, re-limitation etc.

## **c) Parameters**

A parameter is attached to every valuated relationship and is a numerical value that quantifies the relationship it is assigned to. A parameter can be assigned to more than one valuated relationship. There are three types of parameters:

- i ) INPUT parameters (they can be changed by the user)
- ii ) EVALUATED parameters (they are the output of algebraic equations among other input and evaluated parameters)
- iii ) MEASURED parameters (parameters that are read from the model as outputs, without giving an explicit algebraic relation with other parameters)

In general, modifications in a CAD model are much quicker in the case of parametric modelers than in explicit modelers (DASSAULT 1997).

---

<sup>1</sup> These concepts are described as implemented in the CATIA system. The ISO TC184 SC4 parametrics (ISO TC184/SC4/WG5 N243, 1995) describe these and other concepts in a system-independent manner.

## 2.1.2 Interfacing capabilities of CAD systems

There are two main techniques that are used to interface with CAD systems, namely, batch interfaces and application programming interfaces. These interfaces are explained below.

### 1) Batch interface

Batch interfaces in CAD systems are used to obtain design information via files that are controlled by operating system commands. These interfaces are convenient, as they do not require any programming; however, they tend to generate files that contain enormous amounts of information. For example, when neutral files such as STEP AP203 and IGES (Section 3.4) are generated through batch interfaces, all geometric information related to the design would be contained in them. Although these interfaces may be very useful while obtaining all or most information from a CAD design model, they are typically not used to obtain very selective information from CAD systems. For example, if a radius parameter alone is required from a design model, it is atypical to use batch interfaces; instead, application programming interfaces (API) are typically used.

### 2) Application programming interface (API)

The application programming interface (API) of a CAD system is a set of subroutines that are used programmatically to add, modify or read geometric data. The API provides a large library of functions that enable an external application to access the database and applications of the system in a controlled and safe manner (Parametric Technology 1997). In an API, the exchange of information typically occurs in the memory. The subroutines may also be used for developing programs with a list of commands or to create interactive applications (IBM 1991). It gives users the ability to add functionality to CAD systems by writing code in a supported programming language and integrating



the resulting application into the CAD system in a seamless way. Commonly used programming languages for such APIs include C, C++ and Tk/tcl.

It is possible to use the API to automate extraction of geometric information from a CAD model. Compared to the batch interface, the API offers finer grain control, i.e. selective geometric information can be obtained in a quicker and more interactive manner. For example, it is possible to write a simple API adapter program to extract all radii from a design model whose magnitudes are greater than five inches.

Most, if not all of the operations that can be performed through the graphical user interface (GUI) of a CAD system can typically be performed through the API subroutines as well. However, the subroutines may be used to automate repetitive tasks and create new commands, thereby increasing the functionality of the CAD system (Autodesk 1998).

## 2.2 Analysis Methods

According to Gero, analysis is defined as the means by which the behavior of a design structure can be predicted (Gero 1990). According to Chandrasekharan, analysis plays a critical role in the verification of a design (Chandrasekharan 1990). Analysis is considered to consist of three inter-related stages, namely, modeling, simulation and evaluation. Modeling involves reasoning about a design structure (or physical system) with the aim of abstracting an analysis model. This model provides the basis of the simulation phase, the mechanism by which qualitative or quantitative results that describe the physical behavior of the physical system are obtained. The results are based on analytical relations that model physical behavior based on assumptions and idealizations. In some cases, it is practical to solve the governing formula-based relations either manually, or by using compiler tools. If the analysis model is extremely complicated to be solved by simple formula-based analysis tools, other discretization approaches such as finite difference and finite element analysis are adopted. Finally, evaluation is the process

by which these results are verified with respect to the analysis model and validated with respect to the physical system (Finn 1993). This section and the next section (Sections 2.2 and 2.3) overview aspects relevant to formula-based analysis and finite element analysis (FEA), as they are two of the more commonly used approaches.

### 2.2.1 The finite element method (FEM)

The finite element method is a numerical procedure for analyzing structures and continua. Usually, the problem addressed is too complicated to be solved satisfactorily by classical analytical methods. The problem may concern stress analysis, heat conduction, or any of several other areas. In general, the finite element method models a structure as an assemblage of small parts or elements (Cook, Malkus et al. 1989).

Finite element analysis (FEA) typically involves the following steps:

- a) Construct the idealized geometry of the structure that is to be analyzed. The structure may either be a precise representation of the object or a simplified representation for the purpose of analysis.
- b) Divide the structure into finite elements.
- c) Apply the known loads: nodal forces and/or moments in stress analysis, nodal heat fluxes in heat transfer.
- d) Specify how the structure is supported, i.e. set displacements and temperatures to known values.
- e) The computer can now be used to solve for results fields like stresses and strains in the structure or continuum.

## 2.3 Use of geometry in analysis models

This section discusses the use of geometry in analysis and the difference, in general, between CAD design model geometry and that used in analysis models.

Hale argues that the CAD geometric model is the underlying truth model for product representation in engineering design. All other product information is derived from the truth model, including bill of materials, production planning and constraint analysis (Hale, Craig et al. 1999). In the case of FEA, finite element meshes are discretized geometric representations of the design object. Thus, the geometric design model forms the basis of its possibly many finite element meshes and other analysis models.

The design geometry may influence one or more of the following aspects regarding FEA models:

- a) The feasibility of alternative finite element meshes (i.e. the feasibility of the analyses themselves).
- b) The types of meshing algorithms that may be used for FEA.
- c) The types of analysis elements that may be selected.
- d) The resulting mesh density
- e) The quality of the mesh

Automobile design geometry is typically simplified before analysis is carried out, i.e. the geometry of the analysis model lacks the details of the complete automobile. Since the geometry of the analysis model is the basis for the finite element mesh, the validity and character of the geometry have a direct impact on the meshing process. In addition, boundary conditions, element types and material properties all have an effect on the modeling process. All these factors need to be taken into consideration before building the analysis model geometry (Benzley, Merkley et al. 1995). Thus, in effect, the analysis model geometry not only forms a critical aspect of analysis, but it is also the foundation on which the analysis model is built.

Finite element models generally must support a combination of solid, shell and beam elements within a single model. In some cases, mixed finite element topologies may be required. For example, in an impeller, the hub may need to be meshed with solid elements and the vanes with shell elements (Finnigan, Kela et al. 1989).

In formula-based analysis, analysis model geometry is often even more idealized than FEA models; it often consists of parameters that characterize the key shape aspects, and results are typically not fields that vary with spatial coordinates, as they do in FEA.

## 2.3.1 Creation of geometric models for engineering analysis

### 2.3.1.1 Geometric idealizations

The first step in the process of creating an analysis is the generation of an analysis-oriented geometric model, as stated earlier in Section 2.1.1. The majority of analysis models employ a geometric representation that is simplified compared to the design CAD model. Some of the common practices in the transformation of a geometric CAD model to an analysis model, such as an FEA model, are described below.

#### 2.3.1.1.1 *Detail removal*

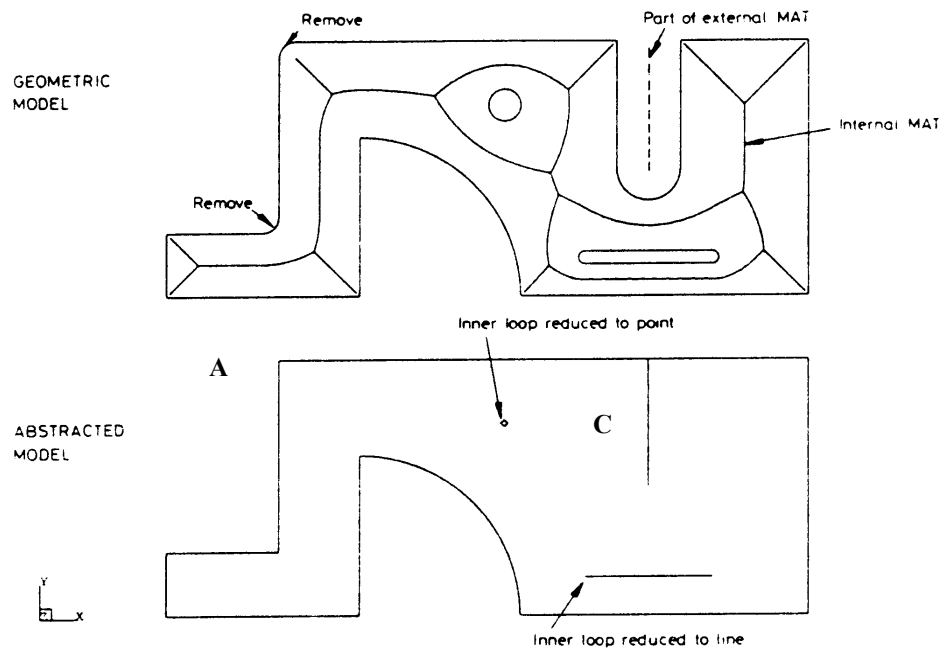
Most analysis problems contain complexities that render numerical simulation difficult and contain redundancies that are unnecessary to analyze. Thus, in practice, certain complexities are simplified for more efficient computation, and redundancies may be ignored without loss to the integrity of the physical system (Finn 1993). Some of the cases in which geometric details may be removed have been listed below.

#### *2.3.1.1.2 Ignoring concavity of edges*

Two-dimensional edges may be removed at a concavity of a design, and it is most appropriate to do so if the radii of the inscribed discs touching the edge are large in comparison with the length of the edge. Face edges may be re-grown to cover the gap left by deleted edges, or approximated by a smooth edge (Armstrong 1994). Figure 9 shows a model with some of its geometric details removed, for analysis purposes. The 2D edge has been removed at a concavity ('A' in the figure) and face edges were re-grown to cover the gap left by deleted edges.

#### *2.3.1.1.3 Deleting internal loops*

An abstraction often used in practice is to remove complete internal loops or holes in the face topology. Figure 9 shows an internal hole and a slot that have been abstracted to a point and a line respectively, thereby substantially reducing the number of finite elements required to represent the geometry (Armstrong 1994). The degenerate inner loop would form a crack in the material ('C' in the figure). The crack may also be ignored by merging the nodes that are adjacent to the two sides of the crack, depending on the probability of failure due to fracture of the component (Armstrong 1994).



**Figure 9: Two-dimensional detail removal**

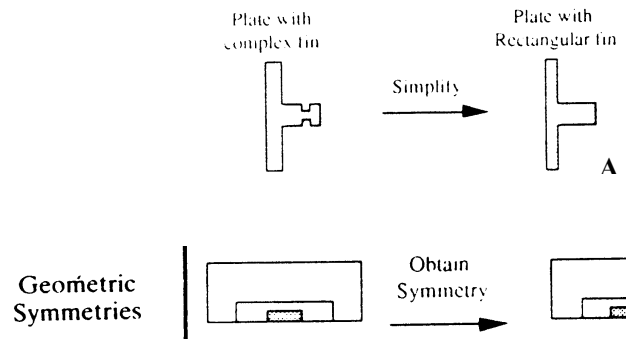
#### **2.3.1.1.4 Achieving Geometric symmetry**

Taking advantage of geometric symmetry in a CAD model reduces the cost of computation in analysis and is a common practice in cases where loads and other boundary conditions are also symmetric. Some CAD models are idealized symmetrically for analysis, even though the actual model may not be perfectly symmetric. Figure 10 shows a geometric model that takes advantage of its symmetry for analysis (Finn 1993).

#### **2.3.1.1.5 Ignoring trivial features**

Certain features may be completely removed on analysis models. For example, Figure 10 shows the complete removal of details of complex fin details for thermal analysis ('A'

in the figure) (Finn 1993). Feature removal is also commonly practiced in order to achieve spatial symmetry of analysis models.



**Figure 10: Simplified models for analysis**

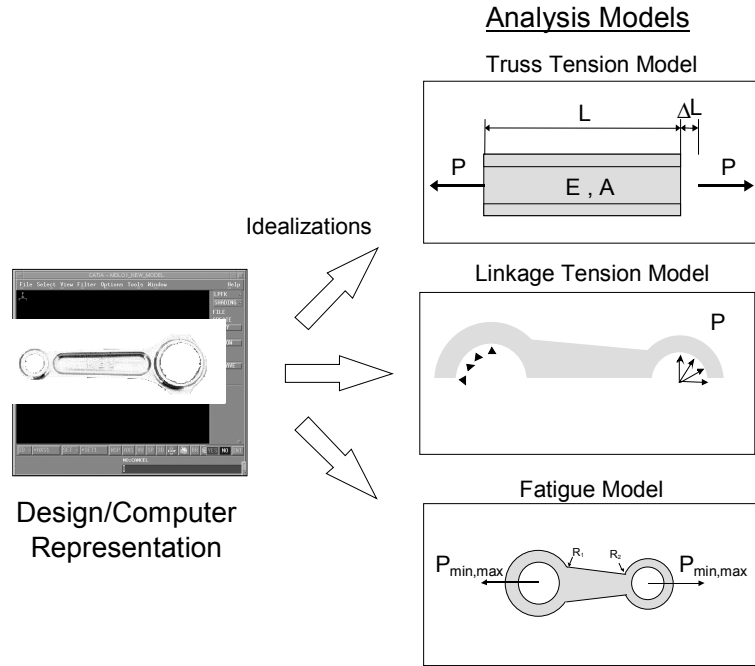
### 2.3.1.2 Analysis idealizations

In engineering terms, to ‘*idealize*’ is to construct an abstracted model of the real system that will admit some form of mathematical analysis (Shigley and Mischke 1989). Idealizations are applied to design information because most problems contain complexities that render numerical simulation difficult or impossible to analyze. In addition, it is usually neither feasible nor desirable to analyze in detail all aspects of a product because of its inherent complexities. Thus, in practice, certain complexities can be idealized in order to make numerical computation more efficient and/or possible (Finn, Grimson et al. 1992). Some of the common analysis idealizations that affect the geometry of an engineering component are explained below, although other types of analysis idealizations also exist.

#### *2.3.1.2.1 Multiple analysis idealizations*

Several analyses such as thermal, structural and vibration analyses may have to be carried to ensure the safety of a given design. Typically, for each of these disciplines of analyses, a separate finite element model must be defined, as different geometric features are relevant to different types of analysis. For example, the flap link model in Figure 11 shows a single CAD model to its left while to its right are tension and fatigue analysis models (Tamburini 1999). These analysis models have all been generated from the original CAD model, either by using the exact dimension values, or by using idealized dimension values. An ‘idealized attribute’ has been explained in Section 2.3.2. Thus, a given design model can have many analysis models associated to it, each model being an ‘idealization’ or an ‘abstraction’ of the original design model.

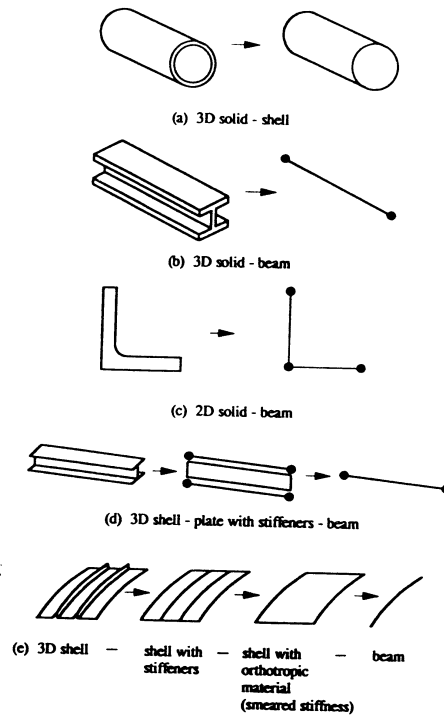




***Figure 11 : Multifidelity and multi-discipline idealizations***

#### 2.3.1.2.2 Dimensional reduction

Dimensional reduction of a geometric design model involves reducing the degree of spatial analysis. This may involve reducing a three-dimensional model to a two-dimensional model or a one-dimensional geometric/analysis model. In Figure 12, three-dimensional solid models have been idealized to two-dimensional or one-dimensional analysis models, i.e. solid models have been transformed to beam and shell finite element analysis models (Armstrong 1994). If a three dimensional solid model is idealized as a plane stress problem, the analysis model geometry is dimensionally reduced and plane stress elements may be used.



**Figure 12: Various dimensional reductions**

### 2.3.1.2.3 Analysis of partial geometry

Often, only a portion of the design model or a part of the whole engineering assembly requires analysis. In such cases, the whole model geometry is not needed and only the relevant geometric information is used for analysis. Figure 2 shows a case where an analysis is performed on a portion of the bike frame. Also, in building design, for example floors are commonly analyzed separately from the frame. A gravity load analysis might be needed on the floor, while gravity and wind analysis might be needed on the frame of the building.

**Concluding remarks:** The full detailed geometric representation of an engineering component as it exists in a CAD system is thus often inappropriate for analysis purposes.

Formula based analysis models must support geometric idealizations, especially idealized attributes. Finite element models sometimes need to support a combination of solid, shell and beam elements within a single model, a case of mixed finite element topologies (Finnigan, Kela et al. 1989). These factors complicated associating a CAD model with its possibly many analysis models.

### 2.3.2 Common geometric attributes needed for analysis idealizations

From Section 2.3.1 it is clear that finite element models and formula based analysis models require attributes from the design model. The attributes needed may be explicit attributes of the product model (design model) or idealized attributes. This study has identified the following types of geometric attributes needed for analyses:

#### 1) Geometric entity/primitive attribute

The attributes of primitive wireframe geometry, namely those of points, lines and circles are of common interest to the analyst. The length of a line, the radius of a circle and coordinates of points in space are typical attributes of this type of geometric representation. Some examples are as follows:

- a) Length of a line
- b) Co-ordinates of the end points that define a line
- c) Radii of circles
- d) Co-ordinates of a point in space

CSG primitives such as cubes, cones, spheres and cylinders are commonly used to represent the shape of an engineering part. It is essential to get the attributes of these geometric entities for analysis, and the method that would be developed should support the same. Examples of these attributes are:

- a) Inner and outer radius of a hollow cylinder

- b) Inner and outer radius of a hollow sphere
- c) Length, breadth and height of a cuboid

## **2) Inter-entity and inter-assembly attributes**

Inter-entity attributes are those attributes that are defined as a constraint between two geometric entities/primitives on the same design part. Examples of inter-entity attributes are:

- a) Distance between two points, between two lines and between a point and a line.
- b) Perpendicular distance between parallel tangents of circles
- c) Angle between two lines or between two planes

An inter-entity attribute may be obtained from two-dimensional draft views. The distance between any two points, between any two lines or between a point and a line on a draft view, are often values of interest for analysis. The ability to extract such values provides tremendous flexibility to the analyst, as they may often not be properties of any specific single geometric entity. For instance, it is easier to obtain the length of a line than to obtain the distance between a point and a line, as this latter distance would not be an attribute of either the line or the point.

Inter-assembly attributes are those attributes that are defined as a constraint between geometric entities/primitives of two different assemblies. An example of inter-assembly attributes is:

- a) Angle between the axes of two CSG cylinders that occur in two different assemblies

## **3) Idealized attributes**

Idealized attributes are fictitious in nature. In other words, these attributes are “made up” by the analyst, based on his or her experience and they usually cannot be directly

physically measured on the product, since they do not actually exist in an explicit physical form. Idealized attributes are defined by mathematical relations containing other design attributes. Examples of idealized attributes are critical area of a plate, effective length of a link and lumped coefficient of thermal expansion of a multi-layer PWB (Tamburini 1999). While transforming a design model to an analysis model or while doing formula based analysis, idealized attributes are often required. That means the analysis may need some attributes that are related to one or more physical attributes that exist in the CAD model. For example, in Figure 27, the effective length (  $L_{eff}$  ) shown is an idealized attribute and is computed using the following relation:

$$L_{eff} = K * ( L - ( ( d_{s1} + d_{s2} ) / 2 ) )$$

Note that ‘L’, ‘ $d_{s1}$ ’ and ‘ $d_{s2}$ ’ are physically measurable attributes in the design model and ‘K’ is an empirical span factor that is typically based on physical tests. Note that these may be termed as non-spatial attributes in cases where they are not physically present or measurable on the design model. It is important to note that idealized attributes are often not attributes of any one single geometric entity and may be derived from relations. However, it is sometimes possible to obtain these idealized values from two-dimensional draft views, in cases where the idealized attribute may be the distance between two points, between two lines or between a point and a line. Since these distances can be dimensioned, the value of the dimensions can be accessed from the CAD system. Effective geometric attributes are commonly employed in formula-based analysis. Sometimes, effective areas of cross-sections may be needed for analysis. As an example, while abstracting a three-dimensional design model to a finite element beam model as shown in Figure 1, the effective area of each beam section is needed.

#### **4) Attributes from sectional views & other compound geometric operations**

Analyses typically require attributes from sectional views such as area of a cross-section. In order to obtain these attributes, it is necessary for the designer to first specify a sectional view and then extract attributes of the same.

#### **5) Redundant attribute**

In parametric modeling, an over-constrained model that has additional parameters defined in it can be created. For example, in the case of the back plate in Figure 26, the parameters 'length1', 'length2' and 'length3' are sufficient to compute the parameter 'length4'. However, 'length4' may also be defined as a parameter. In this case, since 'length1', 'length2' and 'length3' are known, 'length4' is said to be a redundant attribute. A parameter that over-constrains an existing parametric model is referred to as a redundant attribute. Often, analyses utilize these redundant attributes directly either for generalization, or to avoid repetition of cumbersome relations from which they are derived.

#### **6) Mass and volume properties**

The mass, moment of inertia and volume are properties that are commonly used in analysis.

The techniques developed in this study focus on extracting the above-mentioned attributes for analysis, or in some cases, extracting attributes from which the above are derived.

## CHAPTER III

### RELEVANT RESEARCH

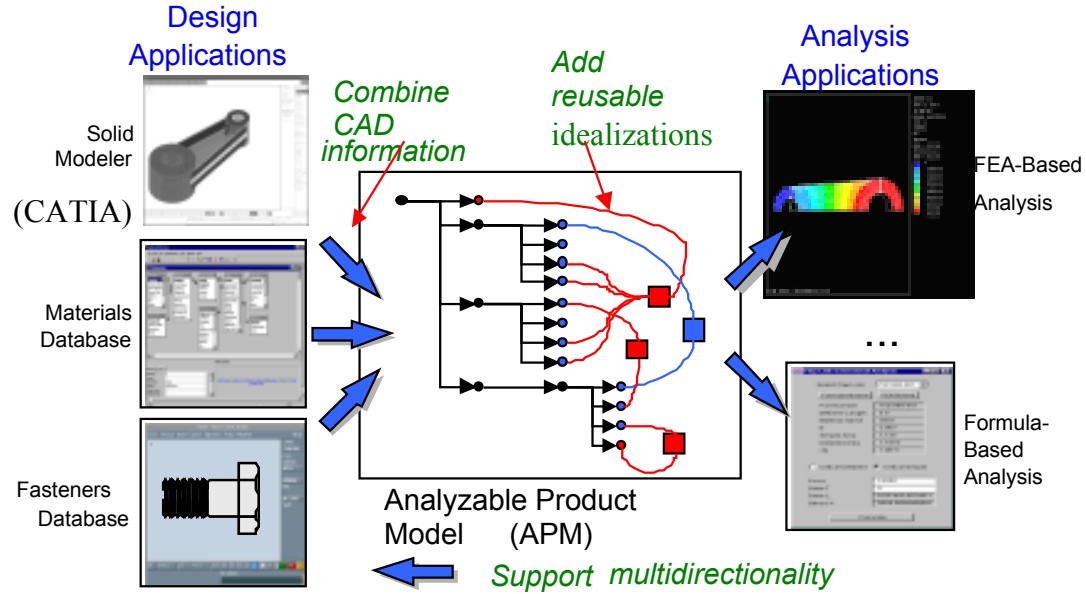
#### 3.1 The analyzable product model (APM)

The analyzable product model representation for design-analysis integration was developed by Tamburini (Tamburini 1999). The design-analysis scenario of this example is illustrated in Figure 13 and involves three design applications (a solid modeler, a materials database manager and a fasteners database manager). Each design application creates information about a different aspect of the product. For instance, the solid modeler creates geometric information and the materials database manager creates a database of the detailed properties of materials available for the fabrication of the flap link. This information is stored in separate design repositories (labeled “Geometric Data”, “Material Data” and “Fasteners Data”).

The design information is used, as shown on the right side of Figure 13, in order to drive two analysis applications, namely, FEA and formula based analysis. Both analysis applications are used to estimate the change in length and the axial stress of the flap link due to an applied extensional force. The two analyses differ in their *solution methods* and *degree of fidelity*, i.e. one is 1D formula-based and the other is 2D finite-element based.

As shown in Figure 13, an APM (“Flap Link APM”) is located between the design and the analysis applications, providing a single integrated source of analysis-oriented product information. *Both* analysis applications and *both* design applications read and write information from and to this single source. It mostly contains data that is used by analysis models which is a subset of all data generated by the design tools; and more

importantly, it supports idealizations of the design data that can be shared by multiple analysis models.



**Figure 13: Analyzable product model technique**

This APM is defined using a special modeling language developed by Tamburini [Tamburini, 1999] for this work and is called the APM Structure Definition Language (APM-S). With this language, developers define the source sets, domains, attributes, relations and source set links that make up the structural definition of the APM. The APM Definition is stored in the APM Definition File and the flap link APM shown in detail in Figure 14. This representation shows the different domains defined in the APM (such as flap\_link, sleeve, beam, etc.), their attributes (such as effective\_length, sleeve\_1, material, etc.) and some of the design and idealization relations among them (“pir1”, “pir2”, “pir12” and “pr2”).



Continuing with the example of Figure 13, the information from the individual design repositories is loaded into the APM. In this test case, for example, there is a Source Data Wrapper to read STEP data (STEP Wrapper) and another to read APM-I data (APM-I Wrapper). These wrapping objects read the design data, perform the necessary conversions, and pass it to the APM in a neutral form understood by the APM.

<pre> APM flap_link;  SOURCE_SET flap_link_geometric_model ROOT_DOMAIN flap_link;  DOMAIN flap_link;   ESSENTIAL part_number : STRING;   IDEALIZED effective_length : REAL;   sleeve_1 : sleeve;   sleeve_2 : sleeve;   shaft : beam;   rib_1 : rib;   rib_2 : rib;   ESSENTIAL material : STRING; PRODUCT_RELATIONS   pr1 : "&lt;rib_1.length&gt; == &lt;sleeve_1.width&gt;/2 - &lt;shaft.tw&gt;/2";   pr2 : "&lt;rib_2.length&gt; == &lt;sleeve_2.width&gt;/2 - &lt;shaft.tw&gt;/2"; PRODUCT_IDEALIZATION_RELATIONS   pir1 : "&lt;effective_length&gt; == &lt;sleeve_2.center.x&gt; - &lt;sleeve_1.center.x&gt; - &lt;sleeve_1.radius&gt; -     &lt;sleeve_2.radius&gt;";   pir2 : "&lt;shaft.wf&gt; == &lt;sleeve_1.width&gt;";   pir3 : "&lt;shaft.hw&gt; == 2*( &lt;sleeve_1.radius&gt; + &lt;sleeve_1.thickness&gt; - &lt;shaft.tf&gt; )";   pir4 : "&lt;shaft.length&gt; == &lt;effective_length&gt; - &lt;sleeve_1.thickness&gt; - &lt;sleeve_2.thickness&gt;"; END_DOMAIN;  DOMAIN sleeve;   ESSENTIAL width : REAL;   ESSENTIAL thickness : REAL;   ESSENTIAL radius : REAL;   center : coordinates; END_DOMAIN;  DOMAIN coordinates;   ESSENTIAL x : REAL;   ESSENTIAL y : REAL; END_DOMAIN;  DOMAIN beam;   critical_cross_section : MULTI_LEVEL cross_section;   length : REAL;   ESSENTIAL tf : REAL;   ESSENTIAL tw : REAL;   ESSENTIAL t2f : REAL;   ESSENTIAL wf : REAL;   ESSENTIAL hw : REAL; PRODUCT_IDEALIZATION_RELATIONS   pir5 : "&lt;critical_cross_section.detailed.tf&gt; == &lt;tf&gt;";   pir6 : "&lt;critical_cross_section.detailed.tw&gt; == &lt;tw&gt;";   pir7 : "&lt;critical_cross_section.detailed.t2f&gt; == &lt;t2f&gt;";   pir8 : "&lt;critical_cross_section.detailed.wf&gt; == &lt;wf&gt;";   pir9 : "&lt;critical_cross_section.detailed.hw&gt; == &lt;hw&gt;"; END_DOMAIN; </pre>	<pre> MULTI_LEVEL_DOMAIN cross_section;   detailed : detailed_l_section;   simple : simple_l_section; PRODUCT_IDEALIZATION_RELATIONS   pir10 : "&lt;detailed.wf&gt; == &lt;simple.wf&gt;";   pir11 : "&lt;detailed.hw&gt; == &lt;simple.hw&gt;";   pir12 : "&lt;detailed.tf&gt; == &lt;simple.tf&gt;";   pir13 : "&lt;detailed.tw&gt; == &lt;simple.tw&gt;"; END_MULTI_LEVEL_DOMAIN;  DOMAIN simple_l_section SUBTYPE_OF l_section; PRODUCT_IDEALIZATION_RELATIONS   pir14 : "&lt;area&gt; == 2*&lt;wf&gt;*&lt;tf&gt; + &lt;tw&gt;*&lt;hw&gt;"; END_DOMAIN;  DOMAIN detailed_l_section SUBTYPE_OF l_section;   IDEALIZED t1f : REAL;   IDEALIZED t2f : REAL; PRODUCT_IDEALIZATION_RELATIONS   pir15 : "&lt;area&gt; == &lt;wf&gt;*( &lt;t1f&gt; + &lt;t2f&gt; ) + &lt;tw&gt;*( &lt;t2f&gt; - &lt;t1f&gt; ) +     &lt;tw&gt;*&lt;hw&gt;";   pir16 : "&lt;t1f&gt; == &lt;tf&gt;"; END_DOMAIN;  DOMAIN l_section;   IDEALIZED wf : REAL;   IDEALIZED tf : REAL;   IDEALIZED tw : REAL;   IDEALIZED hw : REAL;   IDEALIZED area : REAL; END_DOMAIN;  DOMAIN rib;   ESSENTIAL base : REAL;   ESSENTIAL height : REAL;   length : REAL; END_DOMAIN;  END_SOURCE_SET; </pre>
<pre> SOURCE_SET flap_link_material_properties ROOT_DOMAIN material;  DOMAIN material;   ESSENTIAL name : STRING;   stress_strain_model : MULTI_LEVEL material_levels; END_DOMAIN;  MULTI_LEVEL_DOMAIN material_levels;   temperature_independent_linear_elastic : linear_elastic_model;   temperature_dependent_linear_elastic : temperature_dependent_linear_elastic_model; END_MULTI_LEVEL_DOMAIN;  DOMAIN linear_elastic_model;   IDEALIZED youngs_modulus : REAL;   IDEALIZED poissons_ratio : REAL;   IDEALIZED cte : REAL; END_DOMAIN;  DOMAIN temperature_dependent_linear_elastic_model;   IDEALIZED transition_temperature : REAL; END_DOMAIN;  END_SOURCE_SET;  LINK_DEFINITIONS   flap_link_geometric_model.flap_link.material == flap_link_material_properties.material.name; END_LINK_DEFINITIONS;  END_APM; </pre>	

**Figure 14: Flap link test case APM definition file**

As the data stored in the design repositories is loaded into the APM, corresponding APM instances are created. These instances are grouped in the APM by their source of origin (that is, instances coming from the same design repository are grouped together in the same source set). The next step is to *link* these instances, according to linking rules defined in the APM Definition. The result of this linking operation is a single, unified set of instances.

Once the design instances are loaded and linked, analysis applications may access them through a specific set of access functions collectively known as in the APM Protocol. Values for derived or idealized attributes (attributes not created by the design applications but needed for analysis) are computed in the APM as they are requested by the analysis applications. For example, the formula-based analysis application of the flap link example requires the value of an idealized attribute called ‘effective length’. The APM Definition File specifies the mathematical relation needed to calculate its value given the coordinates of the centers of the two sleeves of the flap link.

The APM sends the relations and the values needed to calculate the effective length to an external constraint solver (Wolfram Research’s Mathematica (Wolfram 1996) in this example). The constraint solver solves the system of equations and returns the value of the effective length back to the APM.

Some analysis applications will require more APM information than others, depending on their degree of fidelity and the analysis models on which they are based. For example, the FEA-based analysis requires more detailed information about the flap link than the simpler, less accurate formula-based analysis.

This example also illustrates two important features of the APM regarding analysis idealizations. The first is the APM’s support for *multi-fidelity idealizations*.

Secondly, note the *reusability* of the idealizations defined in the APM. As also shown in Figure 13, both analyses use a ‘simple’ idealized version of the critical cross section of the shaft.

As indicated in Figure 13, the sequence of events just described in this example could also take place in the reverse order, i.e. multi-directional flow of information is possible. For example, the process could start with the tensional analysis determining a target value for the effective length of the flap link. Then, the same idealization relation used before (“pir1”) would be run this time “in reverse” to calculate (or *synthesize*) the coordinates of one of the sleeves given the effective length and the coordinates of the other sleeve as inputs. Once the value for this design attribute is obtained, it could be stored back to the original design repository and read by the solid modeler.

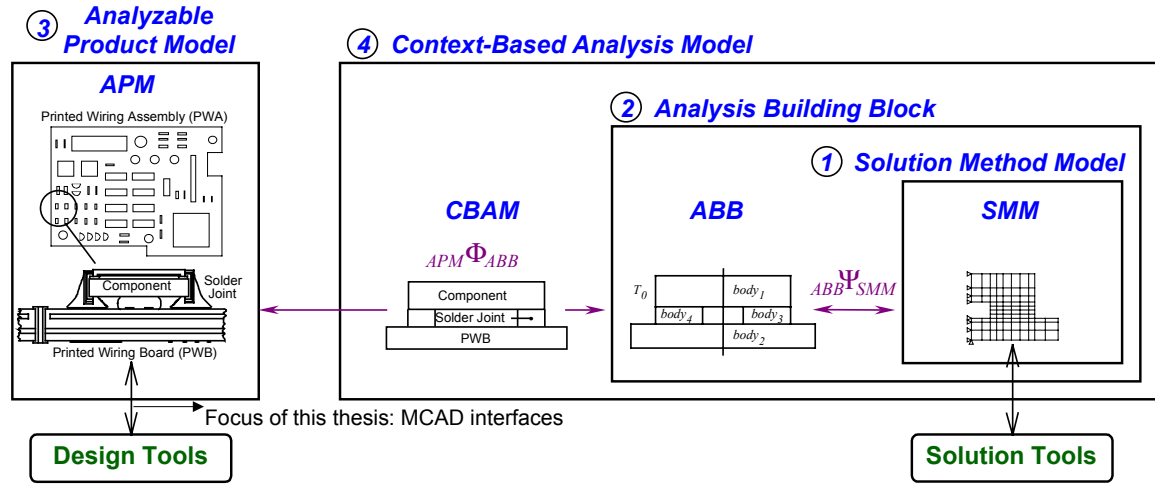
This thesis study has developed APM-compatible geometric interfaces with CAD systems in order to populate the design geometry portions of APMs and thus facilitate integration between design and analysis.

### 3.2 The multi-representation architecture (MRA)

Peak (Peak 1993; Peak and Fulton 1993a; Peak and Fulton 1993b; Peak, Fulton et al. 1998; Peak, Fulton et al. 1999) developed the *multi-representation architecture* (MRA, Figure 15), a design-analysis integration strategy that views CAD/CAE integration as an information-intensive mapping between design models and analysis models. Peak argues that the gap between design and analysis models is too large for a single general integration bridge, and therefore divides the MRA into four information representations that act as stepping stones between the design and analysis tool extremes. These four information representations are: solution method models (SMMs), analysis building blocks (ABBs), analyzable product models (APMs), and context model-based analysis models (CBAMs).

On the right extreme of the MRA (Figure 15) are *solution method models* (SMMs) representing analysis models in relatively low-level, solution-specific form. SMMs combine solution tool inputs, outputs and control into a single information entity (an object) to facilitate automated solution tools access and results retrieval. SMMs are object-oriented wrappers around solution tools (e.g., FEA systems) that utilize an agent-

based framework to obtain analysis results in a highly automated manner. *Analysis building blocks* (ABBs) represent engineering analysis concepts in a manner that is largely independent of product application and solution method. ABBs obtain results by generating SMMs through transformations (labeled  $_{ABB}\Psi_{SMM}$ ) that are based on solution method considerations. *Analyzable Product Models* (APMs, on the left extreme) represent detailed, design-oriented product information (Tamburini 1999).

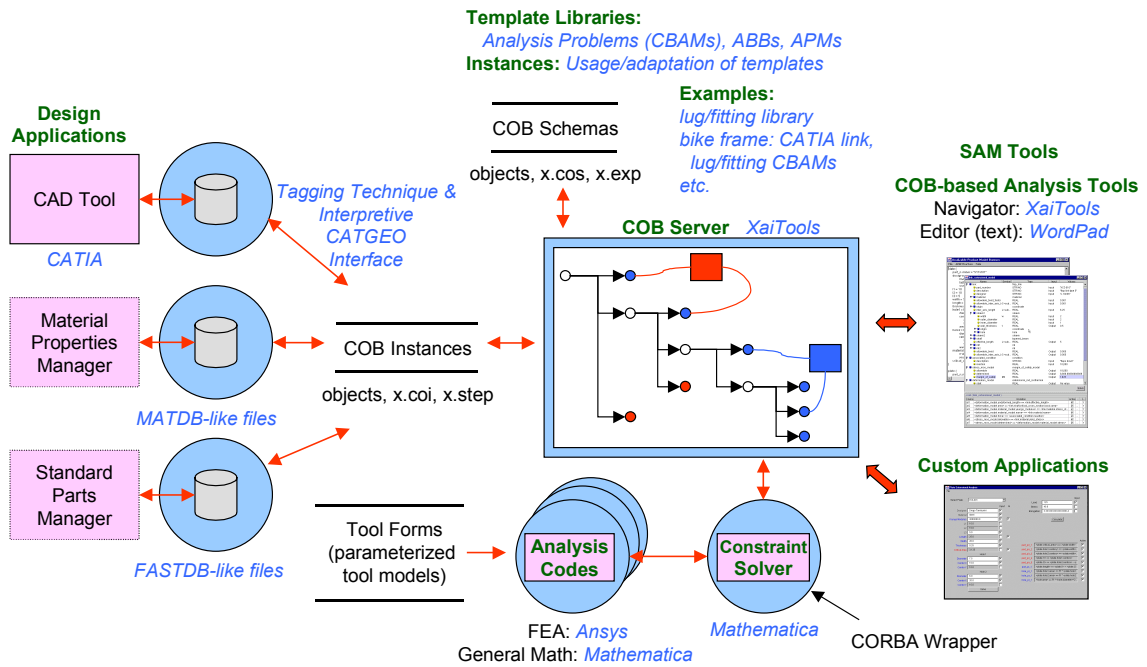


**Figure 15: The multi-representation architecture for design analysis integration**

An APM is a subset of a full product model(PM), the master description of a product which supplies information to other life cycle tasks, including engineering analysis and manufacturing. To enable usage by potentially many analysis applications, PMs in the MRA go beyond their traditional role and aid creation of APMs which support idealizations relating detailed, design-oriented attributes with simplified, analysis-oriented attributes. Finally, *context-based analysis models* (CBAMs) contain linkages (labeled  $_{APM}\chi_{ABB}$ ) that represent design-analysis associativity between APMs and ABBs. These associativity linkages indicate the usage of idealizations for a particular analysis application. CBAMs have been used to create catalogs of ready-to-use analysis modules for electronic packaging applications such as solder joint deformation and

fatigue, PWB warpage, plated-through holes (Peak 1993; Peak and Fulton 1993a; Peak and Fulton 1993b; Peak, Scholand et al. 1996; Peak, Fulton et al. 1999) and aerospace applications like lug and fitting analysis (Peak, Fulton et al. 1999).

### 3.3 XaiTools: Analysis integration toolkit



**Figure 16: XaiTools Architecture for an Aerospace-Oriented Environment**

*XaiTools* is a Java-based toolkit developed in the EIS Lab of Georgia Tech for X-analysis integration that is a reference implementation of MRA concepts (Engineering Information Systems Lab 1999). Earlier work by Peak, Fulton et al. showed the Smalltalk-based first generation toolkit, *DaiTools*, in action in electronic packaging environments (Peak, Fulton et al. 1997). Recent work has migrated and extended these product-data driven analysis capabilities into the Java-based *XaiTools* toolkit.

Demonstrating architecture applicability across product domains, a *XaiTools* architecture for aerospace-oriented environments is summarized in Figure 16 (Peak, Fulton et al. 1999). It has the following characteristics:

- 1) Integration with representative analysis tools:
  - a) *FEA tools*: ANSYS
  - b) *Symbolic solver/general math tool*: Mathematica
  - c) *Other solution tools*: Via black box wrapping approach
- 2) Integration with representative design tools:
  - d) *Geometric modeling tool*: CATIA
  - e) *Materials database*: MATDB-like format
  - f) *Fasteners database*: FASTDB-like format
  - g) *Other design tools*: via native constrained object(COB) instance format or STEP Part 21
- 3) COB-based analysis template libraries with various forms<sup>2</sup>
- 4) COB editing and navigation/browsing tools
- 5) Usage of *Mathematica* as the main CORBA-wrapped constraint solver

Tools of other types and vendors can be added in a similar manner (Peak, Fulton et al. 1997; Peak, Fulton et al. 1998). This thesis focuses on a technique for integrating geometric modeling tools like CATIA.

---

<sup>2</sup> *XaiTools* currently supports constrained object (cob) schemas (cos) and constrained object instances (coi) which are generalizations of the APM definition and instance languages Wilson, Miyako (1999). It also supports reading/writing STEP Part 21 and STEP EXPRESS files, respectively, and writing HTML formatted versions. Graphical editing & interaction tools for constraint schematics are planned.



## 3.4 Standards for exchanging geometric information between CAE systems

### 3.4.1 Introduction

Neutral standards such as STEP (Standard for The Exchange of Product Model Data) and IGES (Initial Graphics Exchange Specification) are currently used for information interchange between CAD/CAE systems. The information interchange may occur between homogeneous CAE systems or heterogeneous CAE systems. For example, in the case of STEP AP203 and IGES file formats, exchange of information is between two CAD systems and is termed ‘homogeneous exchange’; however, in the case of STEP AP209, exchange of information is between CAD systems and finite element analysis (FEA) systems and is termed ‘heterogeneous exchange’ (Peak, Scholand et al. 1999). Peak states that the multifidelity aspect of CAD-CAE integration makes it a particularly challenging case of heterogeneous exchange that is not generally addressed by the individual STEP standards for CAD geometry (E.g. AP203) and analysis (E.g. AP209).

IGES was the first specification for CAD data exchange published in 1980 as a U.S.A. National Bureau of Standards (NBS) report. IGES version 1.0 was accepted as an ANSI standard in 1981. This standard supports CAD geometry and some finite element modeling. IGES had several drawbacks and it was necessary to develop another standard in order to overcome the limitations of the standard.

The limitations of the IGES standard are being addressed by the ISO 10303 STEP standard series. The goal of STEP is providing a complete, unambiguous, computer-interpretable definition of the physical and functional characteristics of a product throughout its life cycle. STEP is a neutral standard with a series of ‘application protocols’ (AP) being created by a team of international experts from disciplines such as

aerospace, automotive, shipping, process plants, CAD/CAE/CAM, academia, and government (PDES 1997).

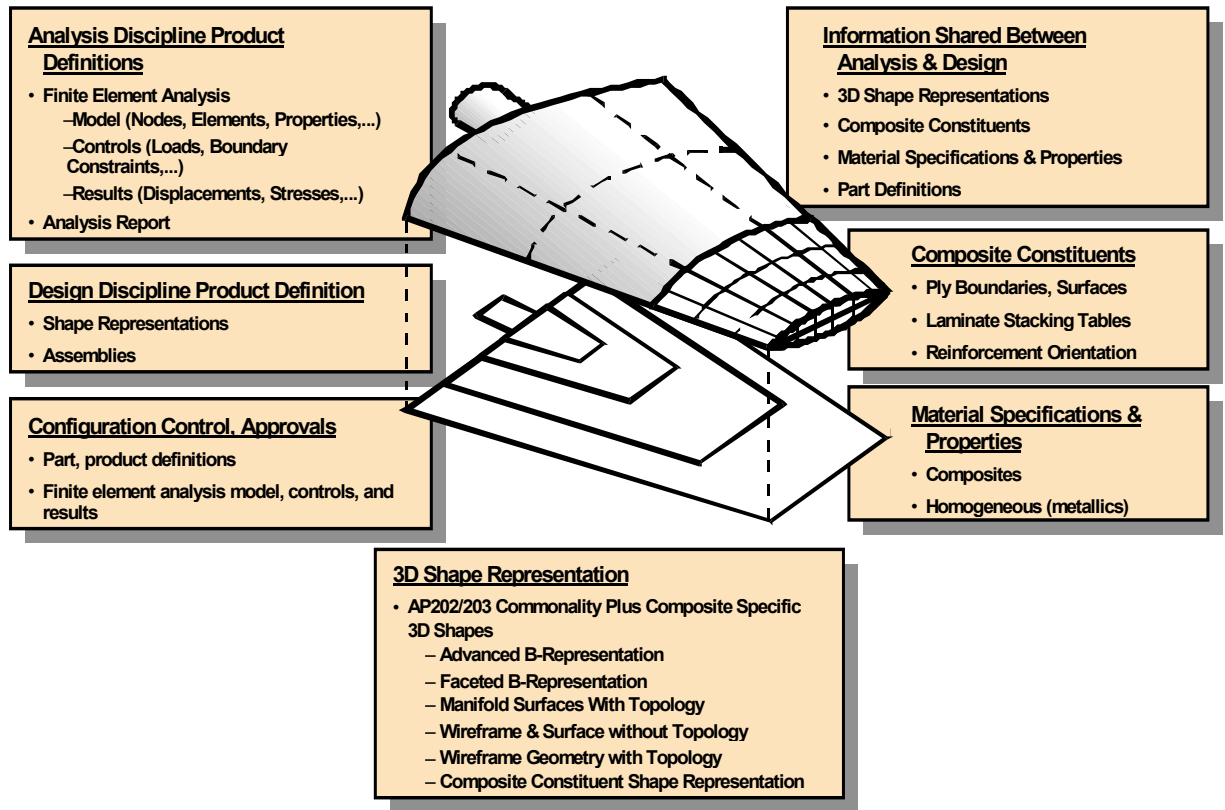
### 3.4.2 CAD-FEA integration with STEP AP209 technology

The design/structural analysis integration problem is typified by the requirement to share geometric shape and analysis information in an iterative environment. The ISO 10303-209 STEP Application Protocol (AP), Composite and Metallic Structural Analysis and Related Design has been developed to address this approach to the design and structural analysis problem (Hunten 1997).

#### 3.4.2.1 Scope of AP209

The scope of AP209 is illustrated in Figure 17. A central theme of partitioning of information within AP209 is that there are separate product definitions for the analysis and design disciplines. This division is primarily a constraint from the aerospace industry, however, similar requirements were noted from shipping, offshore and automotive industries. Another crucial concept is that the shape and analysis information is meant to be implemented to enable bi-directional transfer, i.e. to enable feedback of information in the iterative design/analysis environment (Hunten 1997).

The analysis discipline product definitions primarily concern finite element models, analysis controls and analysis outputs. Loads and boundary conditions may be applied to either mesh or geometry. Linear statics, modes, and frequency analysis types are supported. AP209 was designed so as to be easily extended, in order to support non-linear analyses. In fact, roughly 90% of the non-linear problem is addressed at the present time.



**Figure 17: Scope of AP209 (Hunten 1997)**

The design discipline product definition is primarily concerned with shape representation and assemblies thereof. The geometric shape representations within AP209 are entirely interoperable with those in AP203 that are currently being implemented by most CAD and CAE vendors. There is one additional shape representation unique to AP209 that is utilized to represent the shape of composite constituents such as plies and sandwich cores.

Material specifications and properties are represented for both composite as well as homogeneous (metallic) materials. The specifications and properties may be expressed either at the design level, or more specialized analysis specifications and properties may be represented.

An important feature of AP209 is the sharing of information between the design and analysis product definitions. The shape information is shared at the lowest level allowing locations for nodes to be the same as points defining curves, surfaces, and solids. Both disciplines may also share composite constituents, material properties, and material specifications.

The CAD-based shape design tools today often have at least some level of functionality of Finite Element Model (FEM) generation capability. The use of AP209 provides a standardized format, so that mesh information and any related geometric associativity created in the CAD tool may be shared with FEA systems. The composite shape and structural information may also be associated and shared (Hunten 1997).

#### 3.4.2.2 AP209 implementations

There have been three successful implementations of AP209, and two more are underway. The completed implementations were performed by three teams of companies: one under the auspices of the PDES, Inc. consortium, another under a contract from the US Air Force Manufacturing Technologies Directorate's PDES Application Protocol Suite for Composites (PAS-C) program, and the third under contract from the US Army Tank Command (TACOM) (Hunten 1997).

##### **a) PDES, Inc. FEA pilot**

In the case of the PDES, Inc. FEA pilot team, the exchanges centered upon the analysis of a metallic automotive engine crankshaft. A solid model of the crankshaft was transferred from ComputerVision (CV) to PATRAN where an idealized analysis shape and the derived analysis model were created. The model was subsequently analyzed in MSC/NASTRAN, and also written out in AP209 format and read into CV Stresslab where an identical analysis was performed. Thus, at the end of the cycle there were

analysis results available in PATRAN and in the AP209 visualizer, demonstrating the sharing of analysis information.

#### **b) TACOM pilot**

The TACOM pilot concentrated on the design and analysis of a composite upper hull of an armored vehicle. The participants included South Carolina Research Authority (SCRA), Lockheed Martin, and MacNeal Schwendler Corporation. The solid model of the nose CAV was transferred out of Intergraph and into PATRAN, as were the surface/wireframe representations of the ply boundaries. A finite element model of the nose was then made in PATRAN, and output to AP209 format and appended to the AP209 repository. The analysis was then performed in ABAQUS, translated back into PATRAN, and then into AP209 format and appended to the repository. The completed repository was then read back into Intergraph. In the end there were three applications able to visualize the analysis output: PATRAN, Intergraph, and the AP209 visualizer.

#### **c) PAS-C program AP209 pilot**

The PAS-C program AP209 pilot was performed with AP232 (Technical Data Packaging Core Information and Exchange) to show how the two APs cooperated in performing a configuration controlled design and analysis modification to a horizontal stabilizer skin of an airlifter. The PAS-C pilot began with a metallic horizontal stabilizer skin native CAD (Unigraphics) and AP209 files with related configuration control information being transmitted to a subcontractor via AP232. The subcontractor then took the AP209 shape of the metallic skin and used it as a basis to create a hat-stiffened composite replacement skin design. The composite design was first created using a zone composite description, and then converted to a ply description. The shape information was shared with PATRAN to create a finite element model, and the zone descriptions paired with the finite element model used to automatically create the skin elastic response

matrices. The analyses of the metallic skin (again in AP209 format) was used as a basis for loads and boundary conditions to analyze the replacements composite skin. Analysis results were viewed both in PATRAN and the AP209 visualizer.

#### 3.4.2.3 Future work in AP209

The Phase two PDES, Inc. engineering analysis (EA) pilot has been underway since October 1997. The scope of the pilot is to expand the richness of AP209 implementations completed in the Phase one FEA pilot. Current efforts underway will ensure that AP209 will function inter-operably within a suite of engineering analysis APs that address the multi-disciplinary analysis problems that are increasingly facing engineers (Hunten 1997).

### 3.5 Problems/Gaps that need to be addressed

As we have seen throughout Section 2.3, there is often a large mismatch between the geometric design model that resides in a CAD system and the analysis model geometry. There is presently no general way for exchanging geometric information between these two types of models in such cases where there is no 1:1 correspondence between them. In addition, the analysis models may require idealized dimensions, which are not easily obtained by direct translation of CAD data into an analysis system. Therefore, there is a need to enable the analyst to simplify and idealize CAD geometry for the purpose of using it in analysis models.

STEP AP209 supports FEA; however, it does not support any of the other analysis solution methods and the mismatched geometry they often require. In addition, STEP AP209 does not support bi-directional flow of geometric information very well, i.e. analysis systems are unable to seamlessly return geometric shape information back to the CAD design modeler.

In particular, there is a need to facilitate the analyst to simplify and idealize CAD geometry for the purpose of creating analysis models by enabling the analyst to select specific geometric entities from a CAD model and extract the attributes of the same, for the purpose of using them in different types of analyses (E.g. finite element analysis and formula based analysis).



## CHAPTER IV

### FOCUS OF THIS STUDY

#### 4.1 Thesis objectives

Given the needs identified in Section 3.5, the overall objective of this thesis is to devise a technique to facilitate the extraction of selective geometric information from CAD design models for the purpose of analysis. In cases where design geometry and analysis geometries do not have a 1:1 correspondence, a method by which the analyst can selectively extract the desired geometry for analysis becomes necessary. Specific objectives needed to accomplish this overall objective are identified next. An additional overall objective is for the new technique to be compatible with the APM representation with the MRA approach (Section 3.2). The compatibility is desired, as it will enable achieving many of the below stated objectives in a ready manner.

***Objective 1:*** To support a variety of design model geometric representations

This study intends to support the extraction of geometric attributes of various CAD geometric representations, including, 2D draft views, 2D wireframes, 3D wireframes and 3D solids (CSG and B-Rep solids). While the latter more advanced representations are being used more and more due to their advantages, the former representations are common in legacy data repositories and less advanced tools. The attributes of the following geometric entities should be supported:

- i ) Wireframe geometry

- ii ) Constructive Solid Geometry (CSG) primitives
- iii ) Dimension entities from two-dimensional draft views
- iv ) All parameters, in the case of a 3D parametric model

The methodology intends to support attributes of the above-mentioned geometric entities. However, the attributes of complex curves, such as, Bezier curves and Nurbs curves, are beyond the scope of this study.

**Objective 2:** To provide a means to obtain a variety of geometric attributes which support diverse analysis idealization needs

The need for idealized attributes in analyses has been explained in Section 2.3 and this study provides a means for achieving the same. This study would support diverse analysis idealization needs such as:

- a) idealized attributes
- b) multi-fidelity idealizations
- c) detail removal
- d) symmetry

Some of the attributes that would be supported are as follows:

- a) Primitive geometric entity attribute (Ex. radius of a cylinder, radius of a circle, length of a line)
- b) Inter-entity attribute (Ex. distance between the parallel tangents or centers of two circles)
- c) Geometric attributes from operation results, such as attributes of sectional views
- d) Spatially non-measurable, idealized attributes that are obtained from relations or equations. ‘Idealized attributes’ are explained in Section 2.3.2.

**Objective 3:** To enable the analysis of a portion of a design model

Often times, it is required to analyze just a portion of a design model, such as a feature of a part or a part of an assembly, as shown in Figure 2. This study should incorporate a method by which an analyst can select specific geometric information from a portion of the whole design model and obtain the attributes of the same.

**Objective 4:** To identify a method that enables bi-directional flow of information between a CAD system and its analysis models

Once the analyst has made his study of the physical behavior of the problem, he/she might suggest a few changes in the design. The designer would then have to alter his design, i.e. he would have to change the shape of the engineering component in the CAD system. Once the changes in the design have been approved, the ability to automatically feed in the geometric dimensions and to update the CAD model would be extremely useful.

**Objective 5:** To allow multiple analysis tools to use the same shared geometric data

For a given engineering component, an analyst may use many analysis methods, each of which may require a different analysis tool. In addition, each analyzer may be familiar with a particular FEA system and may want to use the same. Therefore, it is necessary to translate geometric information from a CAD system in a manner that is system independent, so as to enable any analysis system to use this information.

**Objective 6:** To provide a general technique that is applicable to most CAD systems

Each designer would be familiar with a particular CAD system and would want to use the same. Therefore, it is important to devise a strategy that would work for any or most modern CAD systems. This would mean that the different CAD systems and their capabilities would have to be studied before a common solution is adopted.

**Objective 7:** To reduce manual re-entry of geometric data for analysis; thus reducing errors and providing automation

When a FEA model is generated, the analyst first creates the geometric model for analysis. That means, the analyst has to re-type many dimension attributes of the geometry, even if this information resided in a CAD model. This is an error-prone process, and has even proved to be fatal in the past. One such instance was experienced during construction for the 1996 Olympic games, when an analyst had wrongly typed a value during the analysis of a simple structure (Qu 1997). This study hopes to alleviate this type of problem.

**Objective 8:** To support the geometry needs of analysis models for the same design models

The study has also aimed at obtaining geometric information for generating multiple analysis models for the same design model. The above mentioned concept has been explained in Section 2.3.1.2.1.

***Objective 9:*** To support multiple analysis solution methods

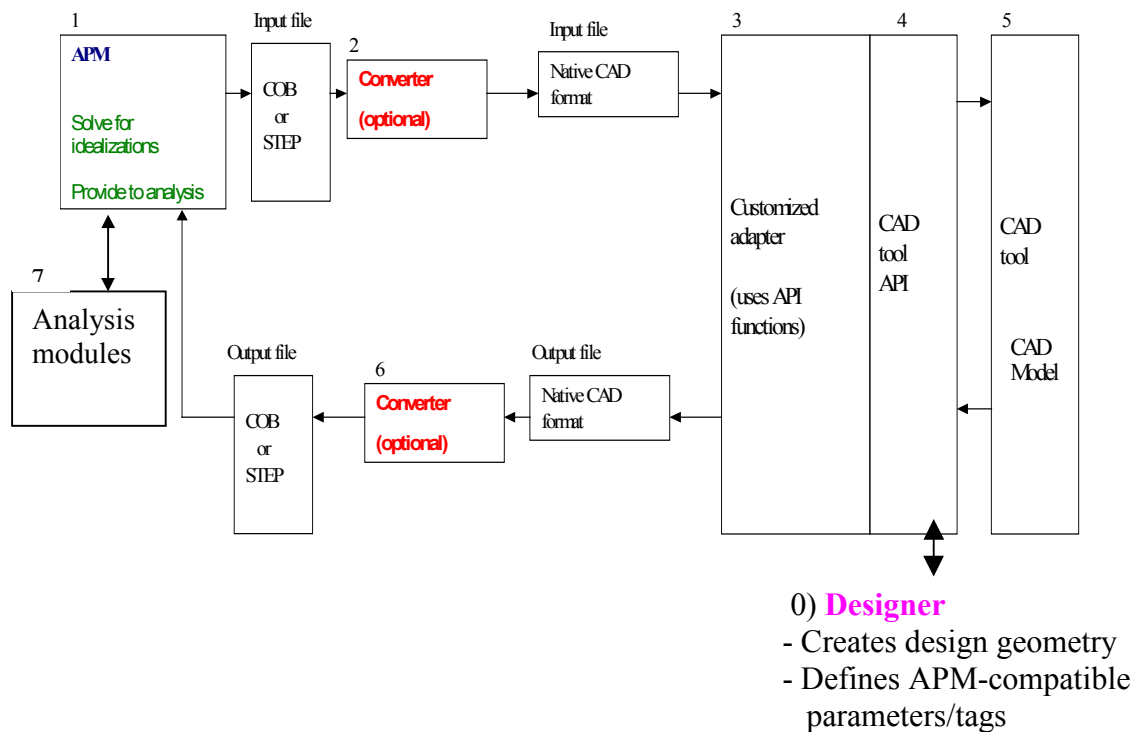
The study would also support geometric information that is needed for multiple analysis solution methods, including, formula-based analysis and finite element analysis. The above mentioned concept has been explained in Section 2.3.1.2.1.

## CHAPTER V

### TECHNIQUE EMPLOYED

#### 5.1 Geometric interface technique

Figure 18 illustrates the general methodology that has been adopted in order to extract geometric information from a CAD system, for the purpose of using this information in FEA or formula-based analysis models. The figure shows seven blocks



*Figure 18: Geometric interface technique for integrating analyzable product models with CAD geometric models*

that achieve the desired integration. This section highlights each block and later sections describe key blocks in depth.

In Figure 18, block 5 is a CAD system. The designer uses the CAD tool and constructs the design geometry. Once the geometry has been created, the analyst selects specific entities in the CAD tool and tags them with a unique identifier. These ‘tags’ have to match those that are used to define the idealized attributes in the APM (block 1), which can be used by multiple analysis models (block 7).

Block 1 contains information regarding an analyzable product model (APM). The APM representation has been described in Section 3.1. An APM contains the product design-related information required for a given set of analysis templates (CBAMs), including geometric attributes and geometric idealizations. The full APM COB schema for this part (back plate), which is depicted in Figure 16, is given in Appendix A. APMs can be implemented in a COB tool like XaiTools. For a given class of parts, the APM is defined in an APM template known as ‘COB schema’. Once the COB schema has been defined for a particular part design, the APM is capable of generating COB instances, each of which lists the attributes that are required for analysis. Rather than manually populating the design attributes of an APM, one can use an APM to generate a request model that specifies the attributes it desires from the CAD model (block 5). This request model (a.k.a. input file in Figure 19) is typically in the form of a text file as explained in Section 3.1. In this case, the file indicates that the radius of ‘circle1’ is needed. The file also requests the length of line1 and the coordinates of the starting point of the line, namely the x, y and z coordinates. This file may either be in COB instance (coi) format, as shown in Figure 19, or another format like the Standard for the Exchange of Product model data (STEP) Part 21 format. The attributes requested must correspond to the entities tagged in the CAD model (block 5 in Figure 18). The naming conventions are discussed in the Appendix E.

```

DATA;
INSTANCE_OF backplate;
.....
circle1.radius : ?;
line1.length : ?;
line1.start.x : ?;
line1.start.y : ?;
line1.start.z : ?;
END_DATA;

```

***Figure 19: A portion of the APM request model in COB instance (coi) format***

In Figure 18, ‘block 2’ is a converter which may be required to convert the input COB/STEP file into a native CAD format that can be read in by a customized adapter.

Block 3 is a customized adapter that is typically needed and is written in a language such as C, C++, Tk/tcl etc., depending on the CAD system that is being used. The adapter has to be specifically programmed to read in and identify the geometric attributes that are needed for analysis purposes from an input file. The requested attributes in the input file are extracted from the CAD model, by using and manipulating the application programming interface (API) functions of a CAD system (block 4). Once the requested attributes have been extracted, they would be written to an output file in the native CAD format. Block 5 is a CAD system and the attributes of its tagged entities are queried through an adapter which extracts the requested information from the CAD model.

Block 6 is an optional converter which may again be required to convert this output file in the native CAD format into a STEP file or a COB file. Figure 20 shows a portion of a typical output file (a.k.a. a response model or a response file) that is in the COB instance (coi) format. This is a file that is generated in response to the input file shown in Figure 19.



```

DATA;
INSTANCE_OF backplate;
.....
circle1.radius : 8.0 ;
line1.length : 60.0 ;
line1.start.x : 0.0 ;
line1.start.y : 20.0 ;
line1.start.z : 0.0 ;
END_DATA;

```

***Figure 20: Response file generated by the API adapter***

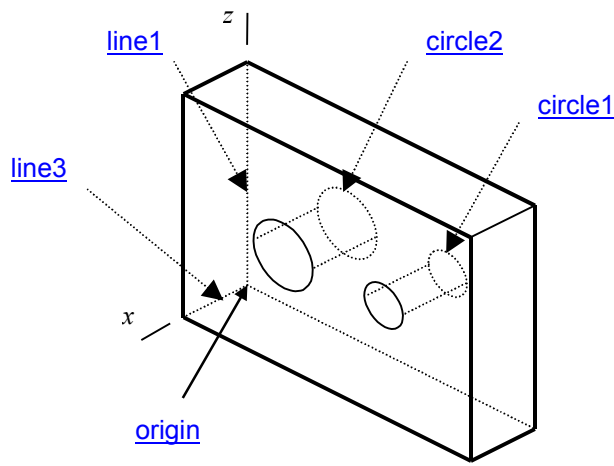
The APM, shown as ‘Block 1’, then reads in the response and is used to solve for some idealized attributes that have previously been defined by specific relations, for the purpose of using them in one or more analysis models (Block 7).

The idealized geometric information that is derived can be used in FEA or formula-based analyses. Furthermore, once the analyses have been carried out, if any changes need to be made in the design model, an input file with the changed attribute values can be fed into the CAD system through the customized adapter. The customized adapter then enables the design model to be automatically updated. Using Figure 18 as a roadmap, the following sections describe key aspects of this process.

## **5.2 Tagging of geometric entities in CAD models**

The tagging technique forms an important part of the geometric interface technique to extract geometric information from a CAD system, as explained in Section 5.1. Every geometric entity in a CAD model is automatically given a unique *tag* by the CAD system (block 5 in Figure 18). The tag may also be referred to as an *identifier* or *label*. For example, line entities may be assigned tags, such as, ‘line1’, ‘line2’, ‘line3’ and so on, in a CAD system like CATIA. Although the CAD system may exactly be able to recognize an

entity by its unique tag, it is impossible for the end-user to know the same unless he analyzes all the lines in the model. However, it is possible for the user to change the tags of geometric entities in most CAD systems. For example, the dimension entities in Figure 21 were originally identified as LN1 and LN2 by the CATIA CAD system. However, these CAD, system-defined tags were changed to 'line1' and 'line3' by the analyst, as shown in Figure 21 (block 0 in Figure 18). The circle entities were identified by 'C1' and 'C2' and the point entity was identified as 'PT1' by the CAD system. These CAD, system-defined tags were changed by the analyst to 'circle1', 'circle2' and 'origin' respectively.



***Figure 21: Tagging of geometry in a CAD model (back plate)***

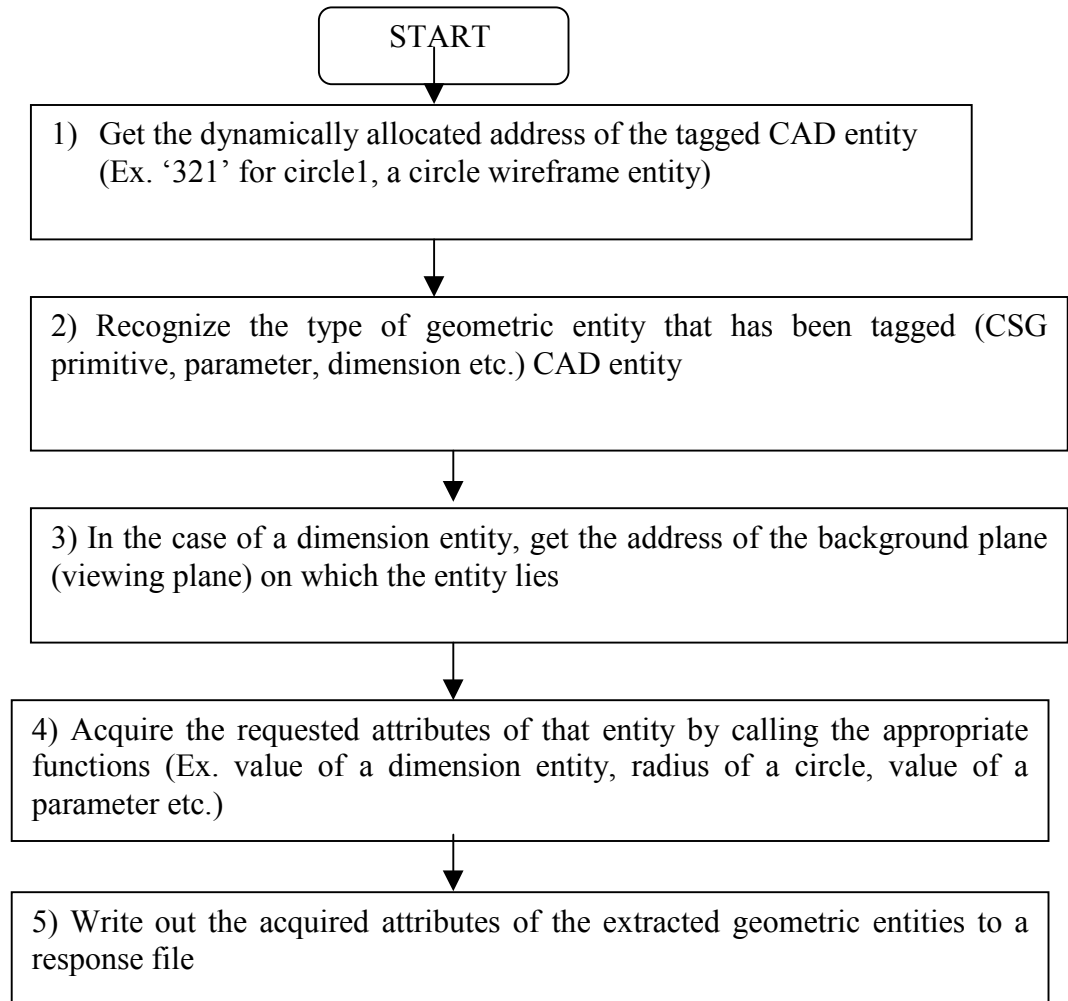
The advantage of tagging geometric entities is that the analyst is then able to extract the attribute values of the tagged entities from the CAD model, by feeding an input file with the unique tags of the desired attributes. This process is highly automated and repeatable versus today's typical manual extraction approach. In Figure 21, the attributes of the circles and the lines are of primary interest to the analyst. The coordinates of the point (origin) may also be obtained. In this approach, the analyst or the designer tags a set of entities/attributes from which needed attributes will be extracted and returned to the analyzable product model (APM).

### 5.3 General flowchart for a customized CAD API adapter

This section overviews the functionality of ‘Block 3’ that is shown in Figure 18. If a given CAD model has been tagged as described in the preceding section, the flowchart shown in Figure 22 can be used to extract geometric attributes from most CAD systems (Block 3, Figure 18). The APIs of three CAD systems, namely, CATIA, Pro/Engineer and IDEAS support the logical approach shown here.

This simplified flowchart does not include complexities that may be encountered in the actual implementation of the API XaiTools CATIA adapter (for details, please refer Section 6.1.4) However, the most important steps have been listed in the flowchart (Figure 22).

- Given:** 1) A tagged CAD model  
2) An APM request file with a list of all attributes needed for analysis (input file)



**Figure 22: Simplified flowchart for a customized adapter (Block 3 of Figure 18)**

Once the tag of the CAD entity has been read by the customized interface adapter, the CAD interface functions that are needed in order to extract attributes of geometric entities, dimension entities and parametric entities are described below.

Functions to extract attributes:

1. Get the unique identifier of the requested ‘tagged CAD entity’ from the request file (geometric, dimension or parametric).
2. Determine the type of CAD entity (Ex. wireframe, solid, dimension, parameter etc.).
3. Determine the background plane (view) on which the CAD entity exists and switch to the appropriate plane (view) for the dimension entity approach.
4. Determine the type of attribute that is being requested for the CAD entity (Ex. radius, diameter, length etc.).
5. Call the function of the appropriate CAD entity and obtain the attribute value that is being requested (from the list of attribute values that the function may return).
6. Write the attribute value to an output/response file.

Depending on the type of CAD entity being requested, the table below lists the functions that are needed to extract attributes of the same (the functions below are numbered such that they correspond to the numbers that prefix the description of the functions above).

***Table 1: Functions needed to extract attributes of different types of CAD entities***

Tagged entity	Function 1	Function 2	Function 3	Function 4	Function 5	Function 6
Geometric entity						
Dimension entity						
Parameter entity						

Additional functions needed to input new values into the CAD system (parametric approach):

1. Determine if the value of the attribute is being input into the CAD system or being extracted (output) from the CAD system
2. If the attribute is an input attribute and if change is feasible (satisfies parametric constraints), change the attribute value to the new value. If change is not feasible, return an appropriate error message.
3. Update the CAD model after the attribute value is changed.

Additional functions/capabilities needed for the user to tag entities:

1. Create the required geometric entity or entities. If dimension entities need to be tagged, the appropriate draft view(s) must be created.
2. Select the entity or entities.
3. Assign a tag to the geometric entity/entities of interest.
4. Define a parametric relation in the case of a parametric design model.
5. View the existing tags in the design model.
6. View the current parametric relations in the case of a parametric design model.

## 5.4 Capabilities supported by CAD systems

In order to generalize an approach for extracting geometric information, different CAD systems were compared and their capabilities were studied. Some important and relevant characteristics of three of the commonly used CAD systems were compared, namely Pro/Engineer, IDEAS and CATIA.

***Table 2: Capabilities supported by common CAD systems***

CAD System	Pro/E	IDEAS	CATIA
1) Geometric entity tagging a) Primitive/wireframe entity b) Dimension entity c) Parameter entity			
2) Interface approach a) API (Name of API) b) Batch interface c) Tables of parameters	(Pro/Toolkit)  (Family table)	(Open-architecture)	(CATGEO)  (Entity-attribute table)
3) Degree of parameterization a) Complete b) Local/partial			
4) Log file capability			

‘Geometric entity tagging’ is explained in Section 5.2. In the case of solid CSG primitives and wireframe entities, the solid or the wireframe entity itself is tagged (Ex.

Circle, line, cylinder etc.). In the case of dimension entities, the dimension entities are selected and tagged (Ex. Radius dimension of a circle, length dimension of a line etc.). In the case parameter entities, the respective parameter entities in a 3D parametric model are tagged (Ex. Offset parameter between two lines, radius parameter of a circle etc.). Approaches for tagging entities are discussed in Section 5.5. The first row in Table 2 indicates that all the three CAD systems support tagging of geometric entities, solid primitives, dimension entities and parametric entities. This implies that their respective tags may be changed by the user. Other systems (E.g. Zuken CR/5000) do not support tagging.

The API and batch interface approaches have been described in Section 2.1.2. The second row indicates that all three CAD systems support the application programming interface (API) and batch interface approaches. The respective API names are listed in the table. It is possible to tag different CAD entities and extract their respective attributes through the API. Some CAD systems typically allow the user to create tables of geometric entities along with their dimensions, attributes and identifiers. For example, it is possible to select a geometric entity or primitive and add it to the table. The table would contain all dimensions, tags and attributes of these selected entities which can be obtained, edited and imported. It is possible to edit the table, change the dimensions of a geometric entity and regenerate the CAD model with the changed dimensions. Some CAD systems allow the user to access tables through the API.

The third row compares parametric modeling capability of the three systems. All the three CAD tools support parametric modeling. Pro/E does not support local parameterization, as the whole model in Pro/E is parametric. Full parameterization can be difficult for complex CAD models like the bike frame model, as can be seen to the left of Figure 1.

Row four in the table compares the log file capability of the systems. This file is generated while geometric models are being constructed in CAD systems. The file contains all the system commands that were used to generate the CAD model. CATIA does not generate a log file or a history file, while Pro/E and IDEAS do generate this file.



## 5.5 Approaches to extracting tagged geometric information

In order to obtain geometric information, the two main features of CAD systems that have been used by all approaches are the application programming interface (API) and the technique for tagging geometric entities (refer Sections 2.1.2 and 5.2 respectively).

This section describes the three approaches for tagging of CAD entities. As described in later sections, the feasibility of each approach depends on:

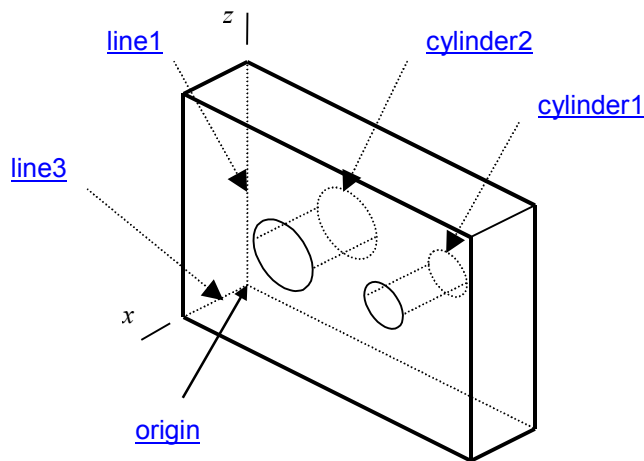
- a) The type of geometric model being dealt with
- b) The type of information needed by the analyzable product model (APM /Block 1)
- c) The capabilities of the programming interface (API) and the adapter (Blocks 3 and 4)

### 5.5.1 Approach 1 : Tagging wireframe entities and CSG primitives

The first approach for extracting geometric information from a CAD model involves tagging of geometric entities, namely wireframe and solid CSG primitives. For example, in order to get the length of a specific line in the CAD model, the analyst would have to select that line and give it a unique tag. The Figure 23 shows a CAD model with two of its lines tagged as 'line1' and 'line3'. Once this is done, the API adapter is able to retrieve the properties of the two lines such as its length and its starting and ending coordinates.

Similarly, if attributes of a CSG cylinder primitive are needed, the analyst selects the primitive, gives it a unique tag and then queries its dimensions. Figure 23 shows two CSG cylinder primitives that have been 'tagged'. While querying the geometric attributes of these primitives, the queries would have to conform with the standard naming

convention that has been adopted for the CSG primitives. One naming convention is explained in Appendix E.



**Figure 23: Tagged geometric entities of the back plate**

#### 5.5.1.1 Characteristics of the approach

- a) Attributes of entities such as points, lines and CSG primitives are supported.
- b) Geometric entities such as points, lines and circles in the 3D CAD model are selected and are tagged by the analyst. For example, in Figure 23, line1 and cylinder1 are the tags of a line entity and a CSG cylinder entity in the 3D CAD model. Although the geometric entities are tagged, their attributes are queried using a standard naming convention. For instance, if the length of 'line1' is needed, the naming convention may require that 'line1.length' be queried through a request file.

- c) It supports partial tagging, i.e. all geometric information need not be retrieved from the CAD model. Only the desired information for analysis can selectively be retrieved.
- d) All attributes that may be extracted are true length values in three-dimensional space.
- e) It does not support tagging of idealized attributes (which typically require inter-entity values).
- f) It typically supports unidirectional flow of information, i.e. it does not allow the attributes of the entities to be changed.

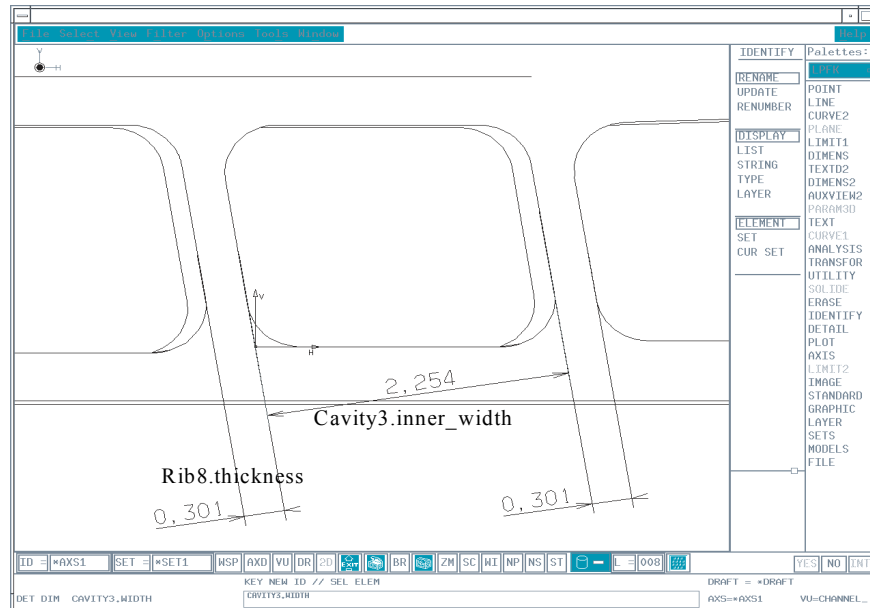
**Note:** This approach can be very tedious for complex designs and was found to be an impractical solution to the problem.

### 5.5.2 Approach 2 : Tagging dimension entities

The second approach for extracting geometric information from a CAD model involves tagging dimension entities, typically in two-dimensional draft views of the model depending on the CAD system. For example, in order to get the value of a dimension entity from a two-dimensional draft view, the analyst would have to select that particular dimension entity and give it a unique tag. It would then be possible to query the tagged dimension entity and extract the same from the CAD system. Figure 24 shows a CAD model of the bike frame part (Figure 2), with its dimension entities tagged, such as, 'cavity3.inner\_width' and 'rib8.thickness'. When the values of these dimension entities are queried through an input file, the API adapter retrieves them (2.254 and 0.301 respectively).

### 5.5.2.1 Characteristics of the approach

- a) Dimension entities are selected and tagged. Some CAD systems like CATIA may require separate draft views instead of supporting dimension entities directly on 3D models.
- b) Principal views as well as sectional views are typically supported, i.e. all attributes of tagged dimension entities from any view of a CAD model can be retrieved.
- c) All lengths in a draft view are projected lengths, i.e. in order to get the true length of a line from a draft view, the line must be parallel to the plane of projection.



**Figure 24: Tagged dimension entities in the bike frame CAD model**

- d) Dimension attributes of cross-sections can be obtained for analysis purposes. For example, it might be possible to compute the critical cross-sectional area from these dimension attributes, as is commonly needed for formula-based analysis.

Since tags may be distributed across many views, the CAD adapter may have to check every view in order to retrieve all the requested dimension values.

- e) Dimension entities are selected and tagged. Some CAD systems like CATIA may require separate draft views instead of supporting dimension entities directly on 3D models.
- f) Principal views as well as sectional views are typically supported, i.e. all attributes of tagged dimension entities from any view of a CAD model can be retrieved.
- g) All lengths in a draft view are projected lengths, i.e. in order to get the true length of a line from a draft view, the line must be parallel to the plane of projection.
- h) Dimension attributes of cross-sections can be obtained for analysis purposes. For example, it might be possible to compute the critical cross-sectional area from these dimension attributes, as is commonly needed for formula-based analysis.
- i) Since tags may be distributed across many views, the CAD adapter may have to check every view in order to retrieve all the requested dimension values.
- j) This approach is well suited for legacy CAD models, including 2D models, where the draft views and dimension entities typically already exist, and can readily be tagged.
- k) A limitation of this approach is that, some CAD systems support only one-directional flow of information, i.e. they do not permit the values of dimension entities that exist in draft views of the CAD model to be changed.

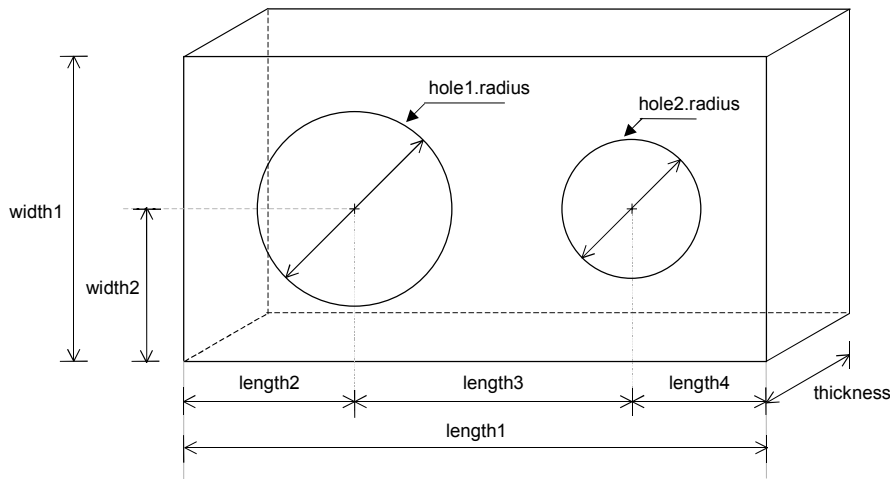
### 5.5.3 Approach 3 : Tagging parameter entities

The third approach for extracting geometric information from a CAD model involves parameterizing a CAD model in such a manner that its parameter entities can be used for its analyses. Figure 25 shows a parameterized model of a plate with two holes. The parameters that are used to define a CAD model may or may not be sufficient for analysis, as analysis models may need additional idealized or redundant parameters. However, parametric modeling in CAD systems allows the user to define parameters that are functions of the existing parameters in the CAD model. In other words, it allows idealized parameters to be defined. For example, in Figure 25, 'width2' is defined as equal to half the value of width1, i.e.  $(width1/2)$ . The CAD system automatically tags each parameter uniquely, by its label; however, most CAD systems allow the parameter tags to be changed by the designer. These parameter tags have to match the labels that are used to define the APM (Block 1). The tags in Figure 25, namely, 'length1', 'length2' etc. have to match those that are used to define the APM. Once the CAD model has been parameterized and its idealized analysis parameters have been defined in the CAD system, it is possible to query these parameter values and extract the same from the through the customized adapter ( 'Block3' in Figure 18).

Once the design has been analyzed, the analyst may recommend changing some parameters in the design model. The designer can make the required changes in the response file and feed it into the API adapter (depending on the CAD system capabilities). The design model will automatically be updated with the changes. Depending on the CAD system, the following capabilities may also be provided:

- a) The imported file can also define new parameters and relations.
- b) The parameters may also be replaced by a relation.

Depending on the CAD system, the order in which the parameters in the model get updated may vary. However, in most CAD systems, the order in which the parameters are updated is the order in which they are listed in the response file. However, the parameters with relations are updated after all other parameters with numerical values are first read in, i.e. parameters with relations are the last to get updated even if they are listed before the last parameter. Examples of the importing capability are shown in the test cases.



**Figure 25: A parameterized CAD model with all its parameters**

#### 5.5.3.1 Characteristics of the approach

- Many geometric parameter values required for analysis can be extracted, including idealized values, distance between points and distance between points and lines.
- Parameters of CAD models are selected and tagged in this approach (Ex. 'width1').
- All parameter values that are obtained are true length values in three-dimensional space.
- It may be possible to parameterize just a portion of the CAD model, depending on the CAD system and analysis requirements.

- e) The parameters typically can be imported into the CAD system and the design is automatically updated. Therefore, this is an excellent approach for design and analysis iterations. However, in some systems the role of the parameter as an input or output cannot be changed (input/output is not reversible). In these cases, one must be careful to import only the 'input' capable parameters.



## 5.6 CAD system support for tag extraction approaches

Table 2 compares the degree to which different CAD systems support the different approaches that have been explained in Section 5.5. Pro/E, IDEAS and CATIA have been compared in the table.

***Table 3: Geometric extraction approaches supported by CAD systems<sup>3</sup>***

CAD System	Pro/E (Release 18.0)	IDEAS (Master series 7.0)	CATIA (Version : 4.1.9)
1) Geometric entity tags - Wireframe geometry - Solid geometry			
2) Dimension entity tags - inter-entity attribute - inter-assembly attribute - redundant attribute			
3) Parameter entity tags  Export entity attributes Import entity attributes Reversible input/output			No
3) Other attributes  a) Volume b) Mass properties			No No

<sup>3</sup> The capabilities of IDEAS and Pro/Engineer were verified with Tord Dennis, Computer Specialist, Georgia Institute of Technology Dennis, Tord (1999) .

The table indicates that Pro/E, IDEAS and CATIA support the three approaches for tagging geometric entities and extracting their attributes from a CAD model, namely, the entity approach, dimension approach and parametric approach that were discussed in section 5.5. Further, the geometric entity approach of extracting attributes of wireframe and solid geometry are supported by all three CAD systems.

The CAD systems that support the extraction of various attributes using the dimension approach have been listed. The definitions of the various attributes in row two can be found in Section 2.3.2.

The third row in the table implies that parameter attributes may be exported and imported in all the three CAD systems. If a CAD system has a parameter ( $L_{eff}$ ) defined by the relation shown below, it is common for CAD systems to be able to compute the value of  $L_{eff}$  given the values of  $K$ ,  $d_{s1}$  and  $d_{s2}$ . However, given the same parametric relation, if a CAD system can compute the value of  $d_{s1}$ , given the values of  $L_{eff}$ ,  $K$ ,  $L$ ,  $d_{s1}$  and  $d_{s2}$ , the CAD system is said to support reversible input/output. Pro/Engineer and IDEAS support reversible input/output. However, CATIA does not support the same.

$$L_{eff} = K * ( L - ( ( d_{s1} + d_{s2} ) / 2 ) )$$

The fourth row compares the CAD systems that support extraction of mass and volume properties. For the definition of mass and volume properties used in analysis, please refer to Section 2.3.2.

## CHAPTER VI

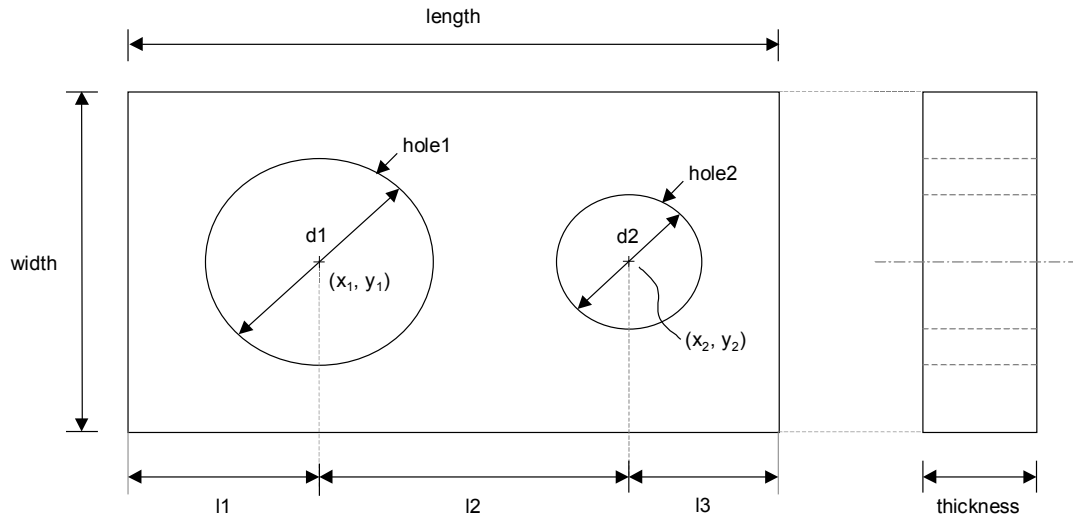
### TEST CASES

#### 6.1 Introduction

Three engineering design models were constructed in CATIA and have been used as test cases for this study, as explained here. The preceeding adapter concepts were implemented in the CATIA CATGEO API in a prototype tool called the XaiTools CATIA adapter.

##### 6.1.1 Back plate

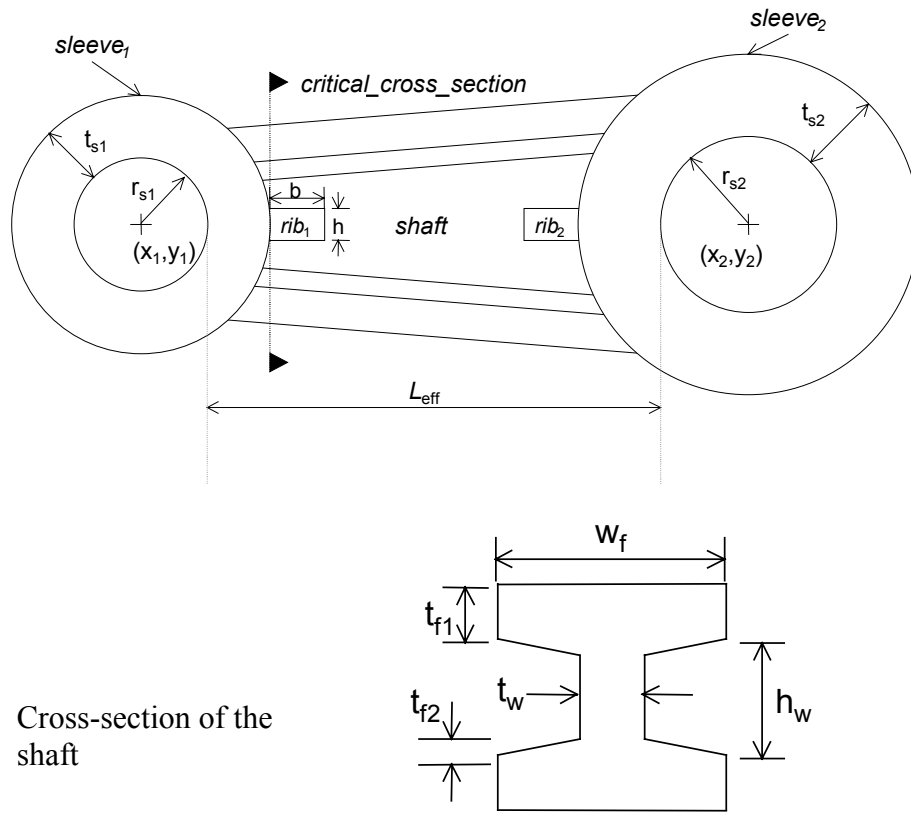
The back plate is a plate with two holes that was devised as a simple part to illustrate key aspects of the approaches. Figure 26 shows the front and side views of the model. The holes may have varying diameters, but the height of their centers must be equal to half the width of the plate. The diameter of the first hole must be greater than that of the second hole. As implemented in Figure 26, the distance 'l2' should always be greater than half the sum of 'd1' and 'd2', i.e.  $(d1+d2) / 2$  in order to be physically realizable. Also, the width has to be greater than the larger diameter (d1), and the 'length' has to be greater than the sum of the diameters, 'd1' and 'd2' (  $length > (d1+d2)$  ).



**Figure 26: Back plate model**

### 6.1.2 Flap link

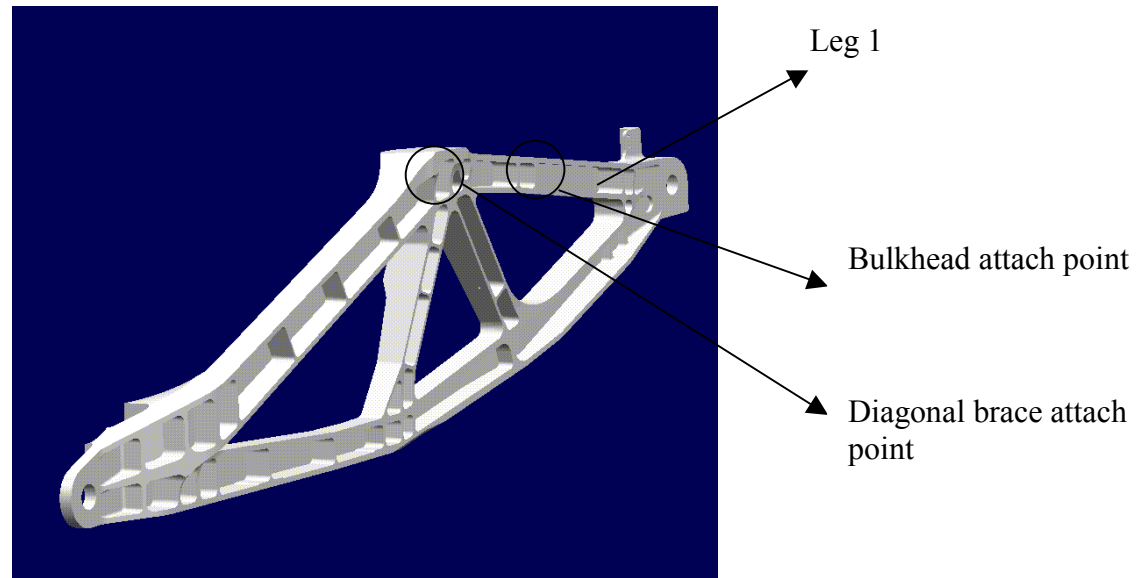
The flap link is an imaginary part that is assumed to be part of an airplane wing flap mechanism assembly. It is a simple rod that connects two parallel shafts. As shown in Figure 27, the flap link is composed of two sleeves (sleeve1 and sleeve2), a shaft and four ribs (two shown). The shaft that connects the two sleeves has an I-shaped cross section of width ( $w_f$ ), and variable height ( $h_w$ ), as shown in Figure 26. Two flap link models were used as test cases, both models being of the same shape, but of varying sizes.



**Figure 27: Flap link design model**

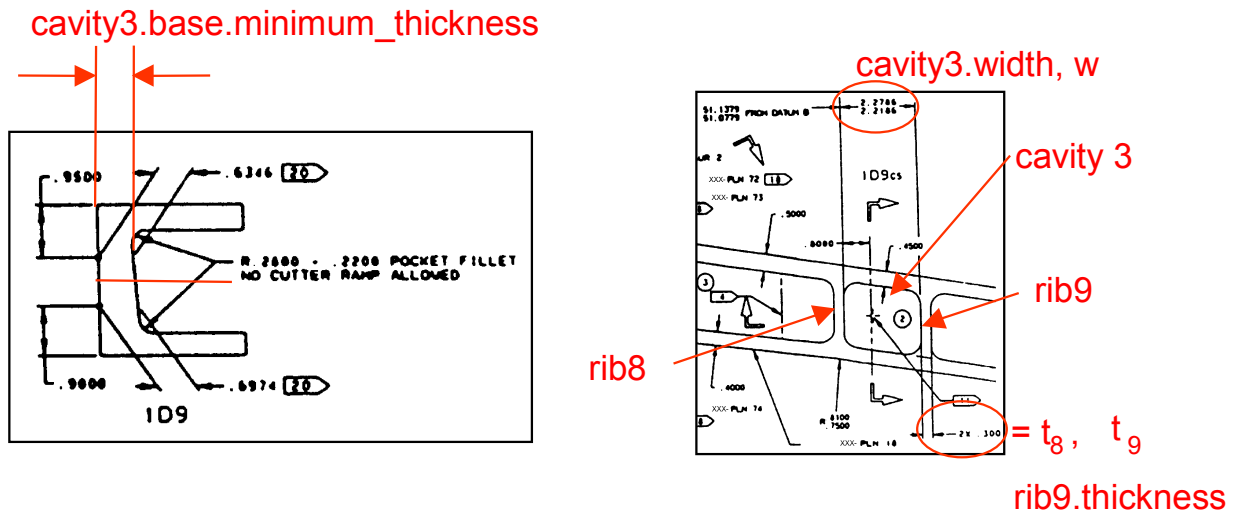
### 6.1.3 Bike frame

The wing flap support assembly is a typical aerospace system that includes two beams, known as ‘inboard beam’ and ‘outboard beam’, which are held together by bolts as shown in Figure 1. The inboard beam of this assembly has been used as a test case and is termed ‘bike frame’ because of its shape. Figure 28 shows the CAD model of the inboard beam of the wing flap support assembly.



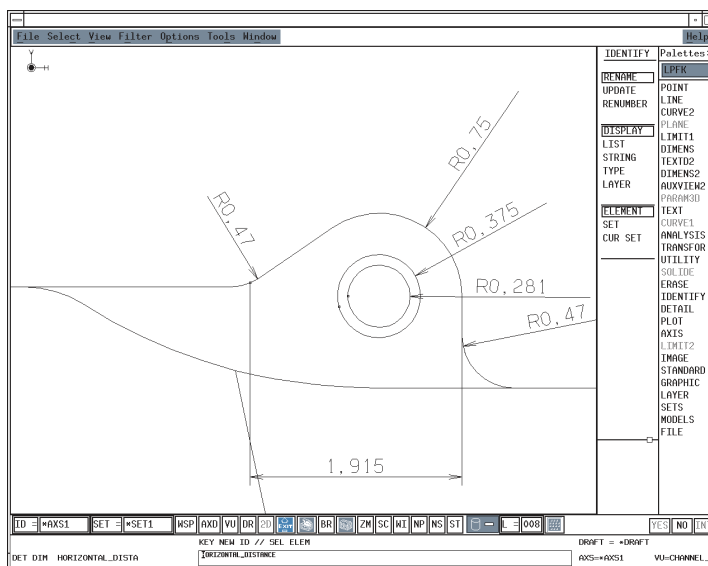
***Figure 28: Inboard beam of the wing flap support assembly***

Figure 28 shows a portion of leg1 highlighted at the bulkhead attachment point. A cavity is present at the attachment point, called ‘cavity3’. Cavity3 is confined by two ribs, namely, ‘rib8’ and ‘rib9’. The dimensions of the cavity are shown in the figure below (Tamburini 1999).



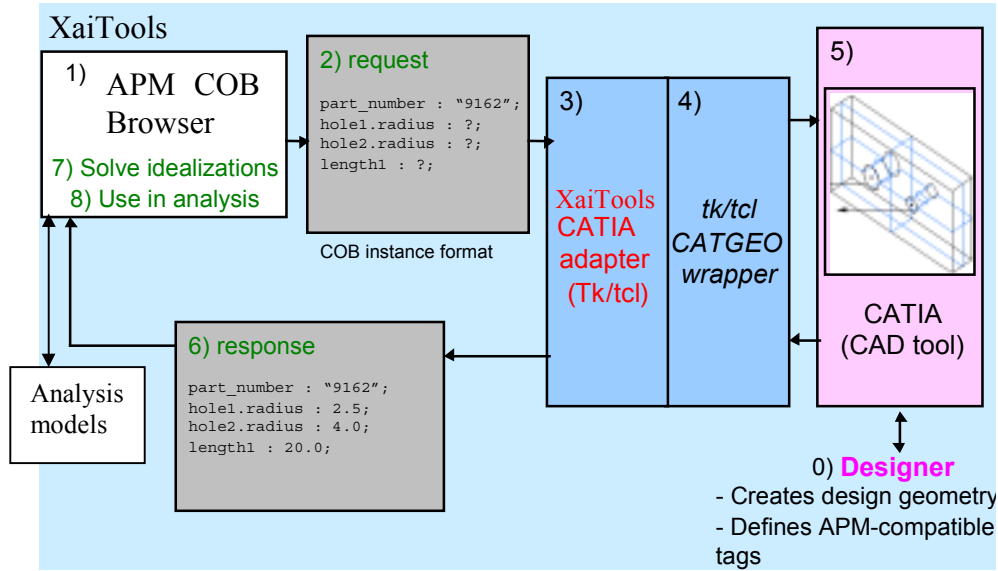
**Figure 29: Bulkhead attachment point on inboard beam Leg1**

Figure 30 shows the draft view of another feature called the diagonal brace attach point, as indicated in Figure 28.



**Figure 30: Diagonal brace attach point**

## 6.1.4 Implementation of the Geometric Interface Technique



**Figure 31: Geometric interface technique for obtaining geometric data**

Figure 31 shows APMs implemented as constrained objects (COBs) in XaiTools (Figure 18) that were adopted in order to extract geometric attributes, dimensions and parameters from CATIA design models.

Block 1 shows APMs implemented as constrained objects in XaiTools<sup>4</sup>. The APM contains the definition of the analysis model of the design. The APM is defined in a COB definition language<sup>4</sup> that has been tailored to facilitate design and analysis. The COB definition language is used to define an APM in terms of its geometry as well as its material properties. The geometric definition includes geometric entities, their attributes and the relations that define the geometry of the specific entities. APMs supports idealized relations as well. The COB browser is capable of generating a file with a list of all the geometric parameters that are needed by an APM for a variety of analyses.



Block 5 is CATIA, a commercial CAD system. The designer creates the geometric model of the design and then tags geometric entities that are specifically needed for analyses. The tagging is done in such a manner that the tags are compatible with the definition of APM, i.e. tags need to match the labels in the definition of the analysis model (Tamburini 1999).

The COB browser creates a request file (block two) with a list of the geometric parameters needed by an APM. The request file is read into the XaiTools CATIA adapter (block three). The adapter is an interface program that has been written in Tk/tcl language in order to extract geometric attributes, dimensions and parameters from CATIA as part of this study. The XaiTools CATIA adapter is capable of reading the requested attributes from the file, retrieving the respective attributes and subsequently writing them to a response file (Block 6) as shown in the figure. For example, block six indicates that the dimensions of hole1.radius, hole2.radius and length1 were retrieved by the XaiTools CATIA adapter. The adapter supports the three ways of extracting geometric information from CATIA as explained in section 5.5.

The XaiTools CATIA adapter (Block3, Figure 18) uses the interpretive CATIA load module interface developed by Hale (Hale 1998) which is a CATGEO (CATIA API) wrapper. The interpretive interface enables the use of the Tk/tcl language in place of FORTRAN for greater flexibility.

The response file is shown in block 6 and contains a list of all the parameters that have been retrieved from CATIA as well as the list of attributes that were not retrieved from it. The response file that is written out by the XaiTools CATIA adapter is fed into the APM and, if possible, it solves for the unknown attributes by using the relations that are defined in it.

The end result is an APM object with retrieved attribute dimensions and parameters from a CAD model for driving multiple analyses (Block 6).

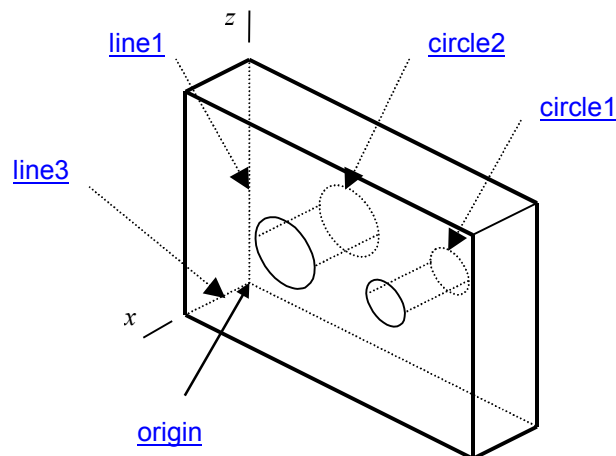
---

<sup>4</sup> Constrained objects (COBS) Wilson, Miyako (1999) are a generalized form of the original APM objects Tamburini, Diego R. (1999)..

## 6.2 Test cases for tagging geometric entities

The approach for tagging geometric entities and solid primitives and extracting their respective attributes has been explained in section 5.5.1. In essence, the geometric entity or CSG primitive is given a unique label by the analyst and its values are then queried through an input file. The unique labels that were used, the input files and the output files have been listed below. The following models have been used for this approach.

### 6.2.1 Back plate (partial tagging)



*Figure 32: Tagging of some geometric entities of the back plate*

#### 6.2.1.1 Geometric construction and tagging of the back plate (partial description)

The back plate shown in Figure 32 consists of two circles, 'circle1' and 'circle2'. It also consists of a rectangle which was constructed by using four lines and four points. The circles and the rectangle are in the same plane, and they were extruded in the positive direction of the x-axis to form a prism.

The plate was then labeled using unique identifiers, namely 'line1' and 'line3' for the two line entities and 'circle1' and 'circle2' for the two circle entities, as shown in

Figure 32. The point entity at the origin was labeled as 'origin'. It is important to note that all entities need not be tagged; only the geometric entities that are essential for analyses were tagged. Thus, a complete description of the back plate is not needed for this model. For example, the length of the rectangular prism cannot be determined in the APM because the appropriate line entity was not labeled (but this is okay assuming no analysis models need this attribute).

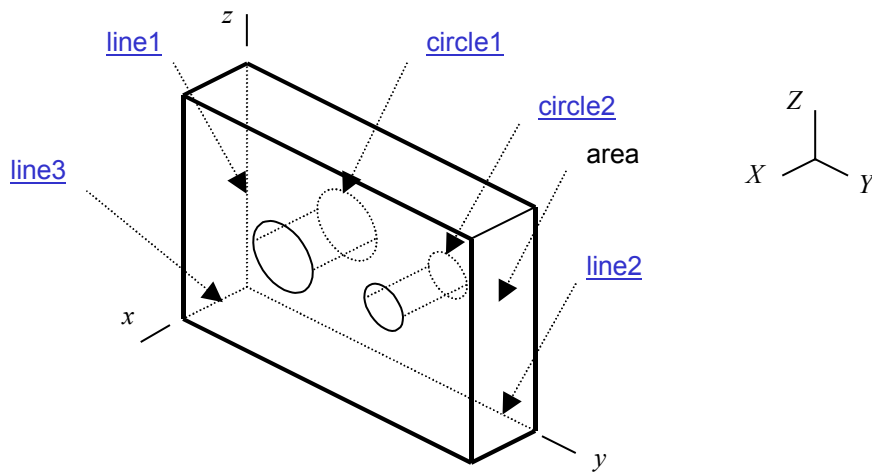
#### 6.2.1.2 Request and response files

The request file was generated from the APM by the COB browser. This file contains the list of geometric parameters that are needed for one or more analyses. The request file is shown in Figure 33. This file was then fed into the CATIA interface program (adapter). The requested attributes of geometric entities and primitives were extracted from the CAD system and they were written out to a response file, as shown below. The files in the figure are partial files; the complete request and response files are listed in Appendix B and Appendix C, respectively, along with the associated APM schema in Appendix A (in the COB 'cos' format).

Request file	Response file
DATA; INSTANCE_OF back_plate; .... circle1.radius : ? ; circle1.diameter : ? ; circle1.area : ? ; circle1.origin.x : ? ; circle1.origin.y : ? ; circle1.origin.z : ? ; line1.length : ? ; line1.start.x : ? ; line1.start.y : ? ; line1.start.z : ? ; line1.end.x : ? ; line1.end.y : ? ; line1.end.z : ? ; origin.x : ? ; origin.y : ? ; origin.z : ? ; .... END_DATA;	DATA; INSTANCE_OF back_plate; .... circle1.radius : 4.000000; circle1.diameter : 8.0; circle1.area : 50.2654; circle1.origin.x : 0.000000; circle1.origin.y : 20.000000; circle1.origin.z : 15.000000; line1.length : 30.0; line1.start.x : 0.0; line1.start.y : 0.0; line1.start.z : 0.0; line1.end.x : 0.0; line1.end.y : 0.0; line1.end.z : 30.0; origin.x : 0.000000; origin.y : 0.000000; origin.z : 0.000000; .... END_DATA;

***Figure 33: Portion of the request and response ‘coi’ files of the partially tagged back plate***

### 6.2.2 Back plate (complete tagging)



**Figure 34: Tagging of the geometric entities of the back plate**

#### 6.2.2.1 Geometric construction and tagging of the back plate (complete description)

The back plate that is shown in Figure 34 is identical to the model that is shown in Figure 32. Its construction has been explained in section 6.2.1.1. In this case the plate has been tagged using unique identifiers, namely 'line1' and 'line2' and 'line3' for the two line entities and 'circle1' and 'circle2' for the two circle entities, as shown in Figure 34. The point entity at the origin was labeled as 'origin'. It is important to note that although all the geometric entities were not tagged, a complete geometric description

of the back plate can be obtained from the labeled entities. For example, the length of the rectangular prism can be determined, since the appropriate line element, namely, 'line2' has been labeled. This information was unavailable in the partial description of the back plate in section 6.2.1.

### 6.2.2.2 Request and response files

A portion of the request file is shown in Figure 35. This file was then fed into the

Request file	Response file
DATA; INSTANCE_OF back_plate; .... circle1.radius : ? ; circle1.diameter : ? ; circle1.area : ? ; circle1.origin.x : ? ; circle1.origin.y : ? ; circle1.origin.z : ? ; line1.length : ? ; line1.start.x : ? ; line1.start.y : ? ; line1.start.z : ? ; line1.end.x : ? ; line1.end.y : ? ; line1.end.z : ? ; origin.x : ? ; origin.y : ? ; origin.z : ? ; .... END_DATA;	DATA; INSTANCE_OF back_plate; .... circle1.radius : 4.000000; circle1.diameter : 8.0; circle1.area : 50.2654; circle1.origin.x : 0.000000; circle1.origin.y : 20.00000; circle1.origin.z : 15.00000; line1.length : 30.0; line1.start.x : 0.0; line1.start.y : 0.0; line1.start.z : 0.0; line1.end.x : 0.0; line1.end.y : 0.0; line1.end.z : 30.0; origin.x : 0.00000; origin.y : 0.00000; origin.z : 0.00000; .... END_DATA;

***Figure 35: Portion of the request and response coi files of the back plate (complete tagging)***

CATIA interface program (adapter). The requested attributes of the entities and primitives were automatically extracted from the CAD system and they were written out

to a response file, as shown Figure 35. The files in the figure are partial files, the complete request and response files are listed in Appendix B and Appendix C respectively, along with the associated APM schema in Appendix A (in the COB ‘cos’ format).

### 6.2.3 Discussion on the approach & its implementation

- i) All wireframe, surface and solid entities can be extracted by using this approach. However, surfaces were not tagged in the test cases, as the XaiTools CATIA adapter has not been programmed to support surface entities.
- ii) Tagging every geometric entity is not an easy task, especially in the cases of complicated CAD models such as aircraft parts. For example, labeling a line entity in an aircraft part design can be very tedious, as there would be hundreds of line entities.
- iii) Innumerable API functions are needed to extract all CAD geometric information. Thus, this is a very programming intensive approach and a simpler approach would be preferred.
- iv) The time taken by the API adapter to extract the information was less than five seconds for all cases that were tested.
- v) This approach of obtaining geometric data is tedious and impractical, especially for complicated CAD models and analyses. A better approach is needed for achieving this purpose.

## 6.3 Test cases for tagging dimension entities

The approach for tagging dimension entities and extracting its values has been explained in section 5.5.2. In essence, the dimension entity is given a unique label by the analyst or designer and its value is then queried, using an input file. The input file is fed into the CATIA API adapter (or interface program). The output file has the extracted geometric information in it. These geometric values can be used to drive a number of analyses via the APM. The unique labels that were used, the input files and the output files have been listed below. The following models have been used for this approach:

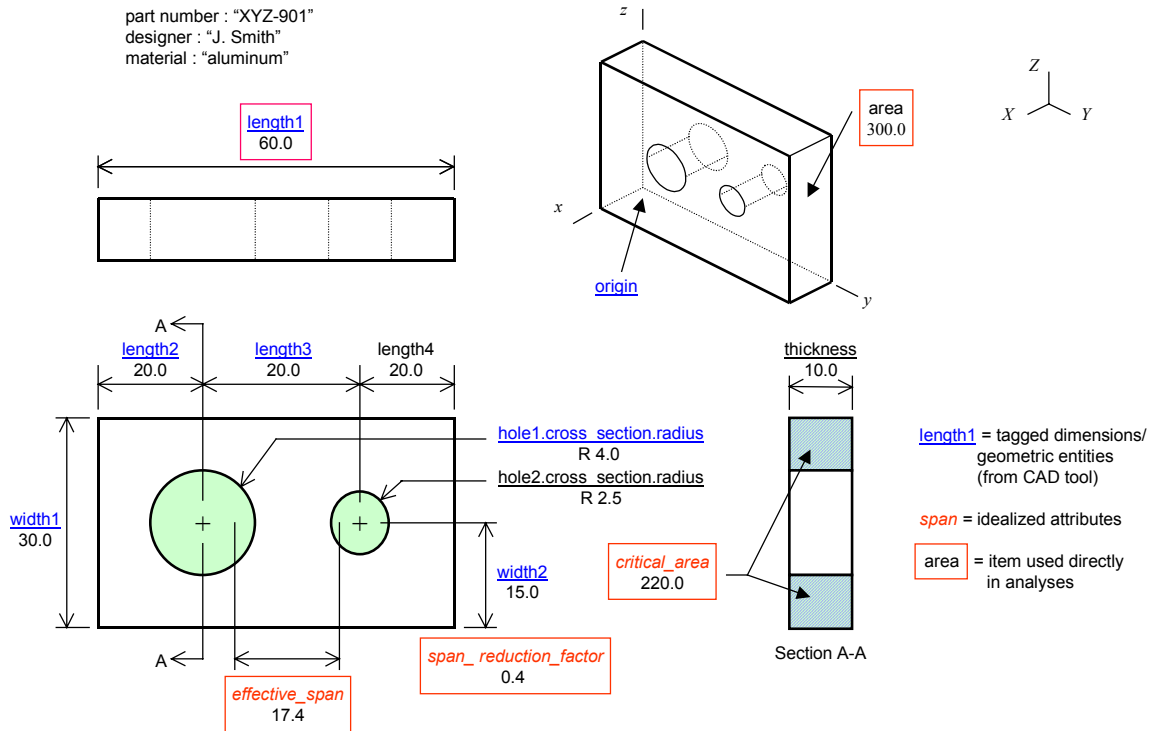
### 6.3.1 Back plate

#### 6.3.1.1 Geometric construction & tagging of the back plate

The three-dimensional CAD model of the back plate used in the geometric entity tagging approach has been used in this case. The geometric construction has been explained in section 6.2.1.1. In addition to the three-dimensional CAD model, its principal views (draft views) were created in CATIA. The dimension entities in the draft views of the CAD model have been labeled.

The dimension entities of the plate were then labeled using unique identifiers, namely 'length1', 'length2', 'length3' and so on, as shown in Figure 36. The point entity at the origin was labeled as 'origin'.





**Figure 36: Labeled dimension entities in draft views of the back plate**

### 6.3.1.2 Request and response files

The request file was generated from the APM by the COB browser. The requested values were automatically extracted from the CAD system and they were written out to a response file, as shown below. The files in the figure are partial files, the complete request and response files are listed in Appendix B and Appendix C respectively, along with the associated APM schema in Appendix A (in the COB 'cos' format).

Request file	Response file
DATA; INSTANCE_OF back_plate; .... length1 : ? ; length2 : ? ; length3 : ? ; length4 : ? ; width1 : ? ; width2 : ? ; thickness : ? ; hole1.cross_section.radius : ? ; hole2.cross_section.radius : ? ; origin.x : ? ; origin.y : ? ; origin.z : ? ; .... END_DATA;	DATA; INSTANCE_OF back_plate; .... length1 : 60.000000; length2 : 20.000000; length3 : 20.000000; length4 : ? ; width1 : 30.000000; width2 : 15.000000; thickness : 10.000000; hole1.cross_section.radius : 4.000000; hole2.cross_section.radius : 2.500000; origin.x : 0 ; origin.y : 0 ; origin.z : 0 ; .... END_DATA;

**Figure 37: Portion of the request and response coi files of the back plate**

### 6.3.1.3 Solved APM files

Once the design attributes are extracted from the CAD system and written out to a response file, these attributes may be used by the APM to calculate ‘idealized attributes’. Although idealized attributes may not be present in design applications, they may be needed for analyses and are therefore computed in the APM. For example, the analysis of the back plate requires the values of idealized attributes such as ‘effective span’ and ‘critical\_area’. The APM definition contains the mathematical relations needed to calculate these values and a constraint solver is typically used to solve the system of equations. After computing the idealized attributes, these values may then be used in analyses.

Solved APM file
<pre> DATA;  INSTANCE_OF back_plate; ..... length1 : 60.0; length2 : 20.0; length3 : 20.0; length4 : 20.0; width1 : 30.0; width2 : 15.0; thickness : 10.0; area : 300.0; effective_span : 17.4; span_reduction_factor : 0.4; critical_area : 220.0; part_number : "XYZ-901" ; designer : ? ; hole1.height : 10.0; hole1.volume : 502.6548245743669; hole1.origin.x : 0.0; hole1.origin.y : 20.0; hole1.origin.z : 15.0; hole1.cross_section.radius : 4.0; hole1.cross_section.diameter : 8.0; hole1.cross_section.area : 50.26548245743669; hole1.cross_section.origin.x : 0.0; hole1.cross_section.origin.y : 20.0; hole1.cross_section.origin.z : 15.0; ..... END_DATA; </pre>

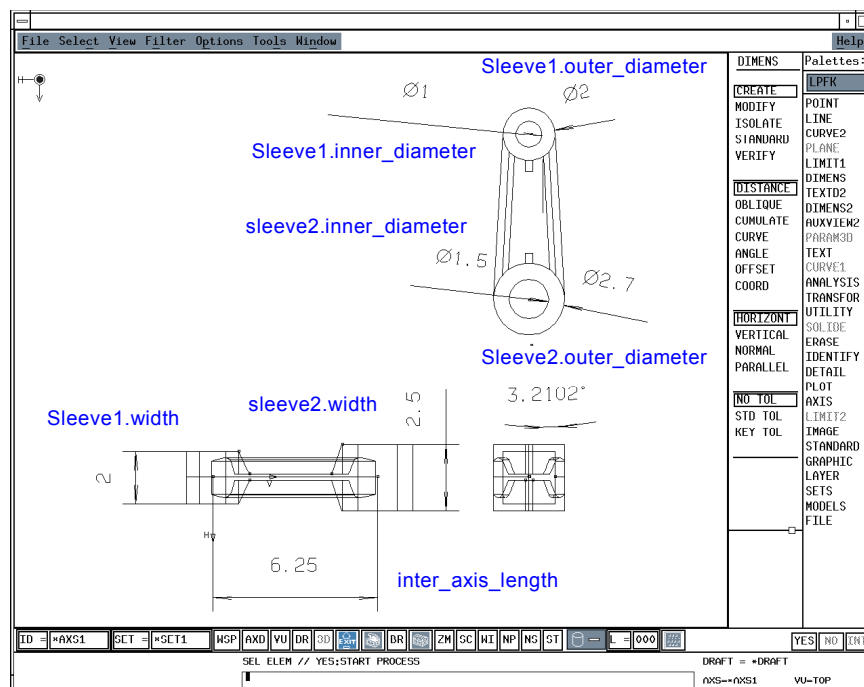
***Figure 38: Portion of the solved APM .coi file for the back plate***

## 6.3.2 Flap link

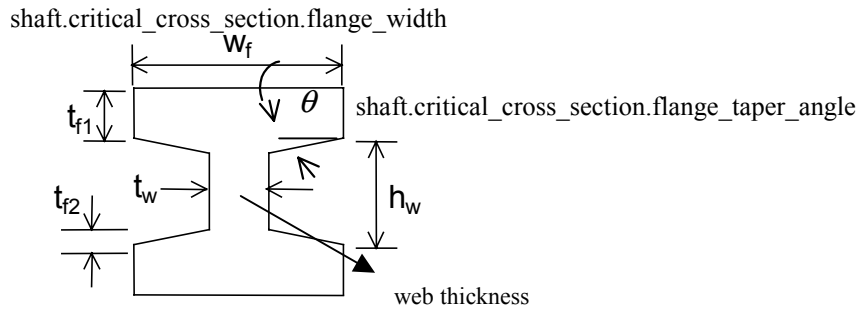
### 6.3.2.1 Geometric construction and tagging of entities of the flap link

A three-dimensional solid model of a flap link was constructed and its principal views were created. The views are shown in Figure 39. The method and order of construction of the solid model is irrelevant, as the dimension entities on the draft views are labeled for extracting dimensions from the CAD model.

The flap link was then labeled using unique identifiers, namely, 'sleeve1.width', 'sleeve2.width' etc., as shown in Figure 36. The point entity at the origin was labeled as



**Figure 39: Tagged dimension entities in draft views of the flap link**



**Figure 40: Tagged critical cross section of the flap link**

‘origin’. It is also possible to take a sectional draft view and label its geometric dimensions as shown in Figure 40. This is a useful capability, as idealized values like critical cross section dimensions are often important for the purpose of doing analysis.

### 6.3.2.2 Request and response files

The request file was generated from the APM by the COB browser, and it contains the geometric parameters that are needed for analysis. A portion of the request file is shown in Figure 41. This file was fed into the CATIA interface program (adapter). The requested values were extracted from the CAD system and they were written out to a response file, as shown below. The files in the figure are partial files, the complete request and response files are listed in Appendix B and Appendix C respectively, along with the associated APM schema in Appendix A (in the COB ‘cos’ format).

Request file	Response file
<pre> DATA; INSTANCE_OF flap_link; .... origin.x : ? ; origin.y : ? ; origin.z : ? ; inter_axis_length : ? ; sleeve1.width : ? ; sleeve1.outer_diameter : ? ; sleeve1.inner_diameter : ? ; sleeve2.width : ? ; sleeve2.outer_diameter : ? ; sleeve2.inner_diameter : ? ; shaft.taper_angle : ? ; shaft.critical_cross_section.flange_width : ?; shaft.critical_cross_section.web_thickness : ?; shaft.critical_cross_section.flange_fillet_radius : ?; .... END_DATA; </pre>	<pre> DATA; INSTANCE_OF flap_link; .... origin.x : 0.000000; origin.y : 0.000000; origin.z : 0.000000; inter_axis_length : 6.250000; sleeve1.width : 2.000000; sleeve1.outer_diameter : 2.000000; sleeve1.inner_diameter : 1.000000; sleeve2.width : 2.500000; sleeve2.outer_diameter : 2.700000; sleeve2.inner_diameter : 1.500000; shaft.taper_angle : 3.210243; shaft.critical_cross_section.flange_width : 1.5 ; shaft.critical_cross_section.web_thickness : 0.25 ; shaft.critical_cross_section.flange_fillet_radius : 0.13; .... END_DATA; </pre>

***Figure 41: A portion of the request and response ‘coi’ files for the dimension based tagging of the flap link model***

### 6.3.2.3 Solved APM files

Once the design attributes are extracted from the CAD system and written out to a response file, these attributes may then be used by the APM to calculate ‘idealized attributes’. Although idealized attributes may not be present in design applications, they may be needed for analyses and are therefore computed by the APM. For example, the analysis of the flap link requires the values of idealized attributes such as ‘effective\_length’, ‘shaft.critical\_cross\_section.tapered.web\_thickness’ and ‘shaft.critical\_cross\_section.basic.area’. The APM definition contains the mathematical relations needed to calculate these values and a constraint solver is typically used to solve the

system of equations. After computing the idealized attributes, these values can then be used in analyses. Figure 42 shows a portion of the solved APM file and the complete file is listed in Appendix E.

Solved APM file
<pre> DATA; INSTANCE_OF flap_link; ..... part_number : "XYZ-510" ; inter_axis_length : 6.25; sleeve1.width : 2.0; sleeve1.outer_diameter : 2.0; sleeve1.inner_diameter : 1.0; sleeve1.wall_thickness : 0.5; sleeve1.origin.y : 0.0; sleeve1.hole.cross_section.radius : 0.5; sleeve1.hole.cross_section.diameter : 1.0; sleeve1.hole.cross_section.area : 0.7853981633974483; sleeve1.hole.height : 2.0; sleeve1.hole.volume : 1.570796326794896; shaft.taper_angle : 3.210243; shaft.critical_cross_section.basic.web_thickness : 0.25; shaft.critical_cross_section.basic.area : 1.125; shaft.critical_cross_section.basic.web_height : 1.5; shaft.critical_cross_section.tapered.total_height : 2.0; shaft.critical_cross_section.tapered.flange_width : 1.5; shaft.critical_cross_section.tapered.flange_base_thickness : 0.25; shaft.critical_cross_section.tapered.flange_taper_thickness : 0.05; shaft.critical_cross_section.tapered.web_thickness : 0.25; effective_length : 5.0; rib1.height : 0.875; rib1.thickness : 0.25; ..... END_DATA; </pre>

**Figure 42: Portion of the solved APM .coi file for the flap link**

### 6.3.3 Bike frame

#### 6.3.3.1 Geometric construction and tagging of entities of the bike frame

A three-dimensional solid model of a bike frame was obtained. An attachment point on the bike frame model, known as the ‘bulkhead fitting’ was utilized. One of the draft views of the bulkhead fitting is shown in

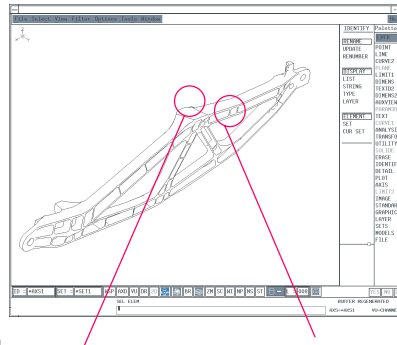
Figure 43. The method and order of construction of the solid model is irrelevant, as the dimension entities on the draft views were used for extraction of geometric data. The bulkhead fitting of the bike frame model was then labeled using unique identifiers, such as, ‘rib8.thickness’, ‘cavity3.inner\_width’ and ‘rib8.thickness’ etc., as shown in

Figure 43. As shown in

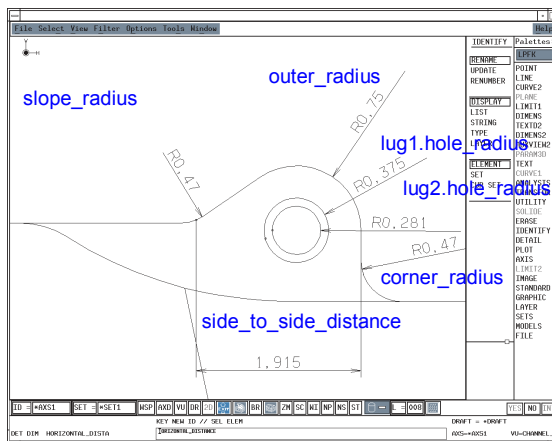
Figure 43, another part of the bike frame, known as the ‘diagonal brace lug’ was also utilized in this test case. The figure shows a draft view of the lug. Specific dimension entities on the lug were selected and labeled with unique tags, as shown in the figure.



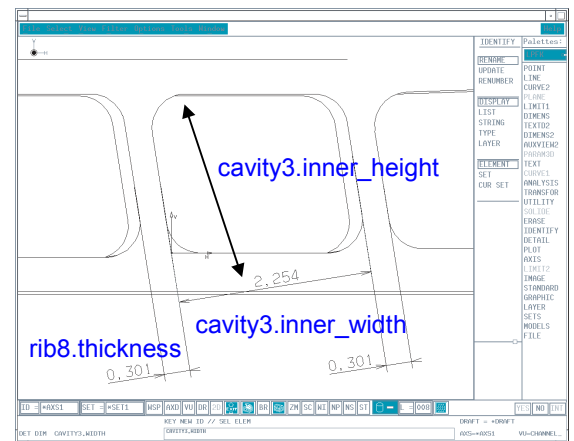
## Bike Frame CATIA CAD Model



### Diagonal Brace Lug



### Bulkhead Fitting Casing



**Figure 43: Tagged dimension entities for two bike frame features**

### 6.3.3.2 Request and response files

The request file was generated by the analyzable product model tool (APM) and it contains the geometric parameters that are needed for analysis. A portion of the request file is shown in

Figure 44. This file was fed into the XaiTools CATIA adapter. The files in the figure are partial files, the complete request and response files are listed in Appendix B and

Appendix C respectively, along with the associated APM schema in Appendix A (in the COB ‘cos’ format).

Request file	Response file
<pre> DATA;   INSTANCE_OF flap_link;   .... cavity3.inner_width : ?; cavity3.inner_breadth : ?; cavity3.inner_height : ?; cavity3.minimum_base_thickness : ?; cavity3.top_thickness : ?; cavity3.bottom_thickness : ?; cavity3.hole_diameter : ?; cavity3.base_angle : ?; cavity3.hole_bottom_edge_height : ?; rib8.thickness : ?; rib9.thickness : ?; diagonal_brace_attach_point.slope_radius : ?; diagonal_brace_attach_point.corner_radius : ?; diagonal_brace_attach_point.outer_radius : ?; diagonal_brace_attach_point.slope_angle : ?; diagonal_brace_attach_point.inter_lug_distance : ?; diagonal_brace_attach_point.lug1.hole_radius : ?; diagonal_brace_attach_point.lug1.thickness : ?; diagonal_brace_attach_point.lug2.hole_radius : ?; diagonal_brace_attach_point.lug2.thickness : ?;   .... END_DATA; </pre>	<pre> DATA;   INSTANCE_OF flap_link;   .... cavity3.inner_width : 2.248610; cavity3.inner_breadth : 2.011259; cavity3.inner_height : 1.885029; cavity3.minimum_base_thickness : 0.604138; cavity3.top_thickness : 0.450000; cavity3.bottom_thickness : 0.400000; cavity3.hole_diameter : 0.598700; cavity3.base_angle : 4.062816; cavity3.hole_bottom_edge_height : 1.031965; rib8.thickness : 0.300000; rib9.thickness : 0.300000; diagonal_brace_attach_point.slope_radius : 0.470000; diagonal_brace_attach_point.corner_radius : 0.47000; diagonal_brace_attach_point.outer_radius : 0.750000; diagonal_brace_attach_point.slope_angle : 34.771807; diagonal_brace_attach_point.inter_lug_distance : ?; diagonal_brace_attach_point.lug1.hole_radius : 0.375; diagonal_brace_attach_point.lug1.thickness : 0.35000; diagonal_brace_attach_point.lug2.hole_radius : 0.281; diagonal_brace_attach_point.lug2.thickness : 0.3500;   .... END_DATA; </pre>

**Figure 44: Portion of the request and response ‘coi’ files for dimension based tagging bike frame bulkhead attach point<sup>5</sup>**

<sup>5</sup> Note: Values in Figure 43 are slightly different than Figure 44 because the draft view in Figure 43 was originally taken at a plane non-perpendicular to the rib surfaces.

## 6.4 Test cases for tagging parameters

The parametric approach has been explained in section 5.5.3. It requires the designer to first construct a parametric model of the design. A parameter may be a value or a relation and has a unique label by which it is identified (refer section 2.1.1.2). The parameters need to be tagged such that they are compatible with the labels in the APM tool. Furthermore, a parameter can be an idealized relation, therefore idealized relations that are needed for analyses can be defined as parameters. Alternately, the idealization relations can be included explicitly in the APM.

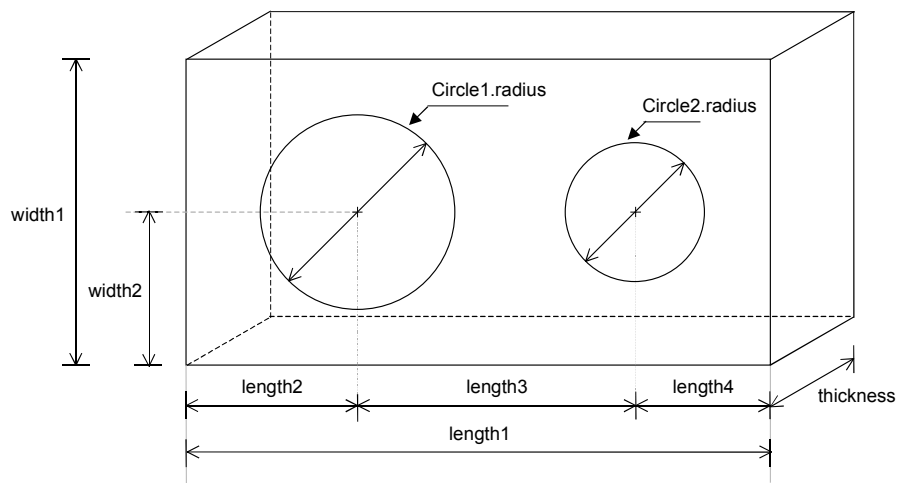
The COB browser generates an input file that contains a list of all parameters that are needed by the APM for analyses. The input file is then fed into the CATIA API adapter (interface program). The output file contains the extracted geometric information in it. These geometric values can be used to drive a number of analyses via the APM.

The unique tags that were used, the input file, the output file and the imported file have been listed below. The models that have been used for this approach are explained in section 6.1

## 6.4.1 Back plate

### 6.4.1.1 Geometric construction and labeling of the back plate

The CAD model that was used for the geometric entity tagging approach has been used in this approach. In addition, the CAD model was parameterized and labeled as shown in Figure 45.



***Figure 45: Parameters of the back plate design model***

#### 6.4.1.2 Files read into and written out of the CAD system

The request and response files were created by the APM and the API adapter respectively (Block 1 and Block 3, Figure 31) and are shown in Figure 46. Once the parameters are read from the CATIA model and used for analyses, the analyst may recommend changes to the design dimensions. For this reason, some of the parameters were changed in the response file, imported into CATIA and the design model was updated with the changed parameter values. The exporting and importing of parameters can be done using two methods:

- a) Using the CATIA GUI IMPORT/EXPORT capability
- b) Using the customized XaiTools CATIA adapter to achieve the same purpose

If the first approach is used, the files have to be imported and exported in the standard CATIA format. However, if the second approach is used, the need to be imported and exported in the APM 'coi' format. In both cases, before importing the file, the values have to manually be changed in the text files and conform to the appropriate file format. Though it appears to be feasible, the XaiTools CATIA adapter has not yet been extended to support importing new values. For further details on exporting and importing a file, please refer the user's manual in Appendix E. It is important to note that the imported file may contain changed parameters, new relations and new parameters.

Request file	Response file
DATA; INSTANCE_OF back_plate; .... length1 : ? ; length2 : ? ; length3 : ? ; length4 : ? ; width1 : ? ; width2 : ? ; thickness : ? ; hole1.cross_section.radius : ? ; hole2.cross_section.radius : ? ; .... END_DATA;	DATA; INSTANCE_OF back_plate; .... length1 : 60.000000; length2 : 20.000000; length3 : 20.000000; length4 : ? ; width1 : 30.000000; width2 : 15.000000; thickness : 10.000000; hole1.cross_section.radius : 4.000000; hole2.cross_section.radius : 2.500000; .... END_DATA;

**Figure 46: Portion of the request ‘coi’ and response files of the back plate design using the parametric approach**

Imported file
length1 = 60.000000; length2 = 20.000000; length3 = length1/5; length4 = length1-length2-length3 ; width1 = 30.000000; width2 = 15.000000; thickness = 10.000000; hole1.cross_section.radius = 9.000000; hole2.cross_section.radius = 4.500000;

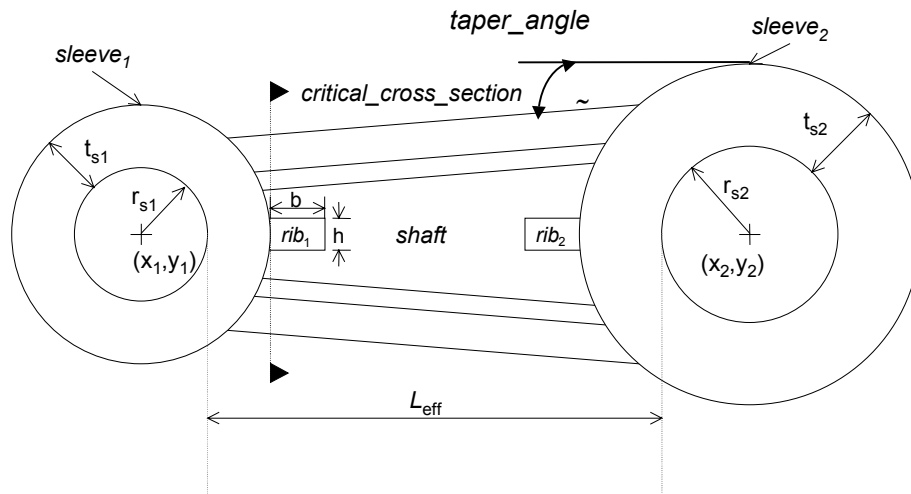
**Figure 47: Imported file of the back plate design model (CATIA import format)**

## 6.4.2 Flap link

### 6.4.2.1 Geometric construction and labeling of the Flap Link

The CAD models that were used for the dimension entity tagging approach have been used in this approach (two flap link models of varying sizes). In addition, the CAD model was parameterized and labeled as shown in Figure 39.

The ‘taper angle’ may either be defined as a measured parameter in the parametric CAD model or defined as a trigonometric relation in the analyzable product model (APM). It was found to be much simpler to define the taper angle in the design model (without any explicit mathematical relation) than in the APM, as it was less time consuming and free of human error.



**Figure 48: Flap link taper angle defined as a measured parameter**

#### 6.4.2.2 Files read into and written out of the CAD system

The request file was generated by the analyzable product model. The request file is shown in Figure 49. This file was then fed into the CATIA interface program (adapter). The requested values were extracted from the CAD system and they were written out to a response file, as shown below. It is also important to note that the taper angle was defined and extracted as a measured parameter from the CAD model, without the use of a trigonometric relation in the definition of the APM.

Once the parameters are read from the CATIA model and used for analyses, the analyst may recommend changes to the design dimensions. For this reason, some of the parameters were changed in the response file, imported into CATIA and the design model was updated with the changed parameter values. The exporting and importing of parameters can be done using two methods as in the back plate:

- c) Using the CATIA GUI IMPORT/EXPORT capability
- d) Using the customized XaiTools CATIA adapter to achieve the same purpose



Request file	Response file
<pre> INSTANCE_OF flap_link; .... allowable_twist_factor : ?; allowable_inter_axis_length_change_factor : ?; inter_axis_length : ?; sleeve1.width : ?; sleeve1.outer_diameter : ?; sleeve1.inner_diameter : ?; sleeve2.width : ?; sleeve2.outer_diameter : ?; sleeve2.inner_diameter : ?; shaft.critical_cross_section.design.flange_width : ?; .... shaft.critical_cross_section.design.web_thickness : ?; shaft.taper_angle : ?; .... END_DATA; </pre>	<pre> INSTANCE_OF flap_link; .... allowable_twist_factor : 0.001; allowable_inter_axis_length_change_factor : 0.025; inter_axis_length : 200.000000; sleeve1.width : 70.000000; sleeve1.outer_diameter : 60.000000; sleeve1.inner_diameter : 30.000000; sleeve2.width : 80.000000; sleeve2.outer_diameter : 100.000000; sleeve2.inner_diameter : 50.000000; shaft.critical_cross_section.design.flange_width : 33.0; .... shaft.critical_cross_section.design.web_thickness : 3.0; shaft.taper_angle : 5.739170; .... END_DATA; </pre>

**Figure 49: Portion of the request and response ‘coi’ files of the flap link design using the parametric approach**

Imported file
<pre> allowable_twist_factor = 0.001; inter_axis_length = 300.000000; sleeve1.width = 80.000000; sleeve2.width = 90.000000; </pre>

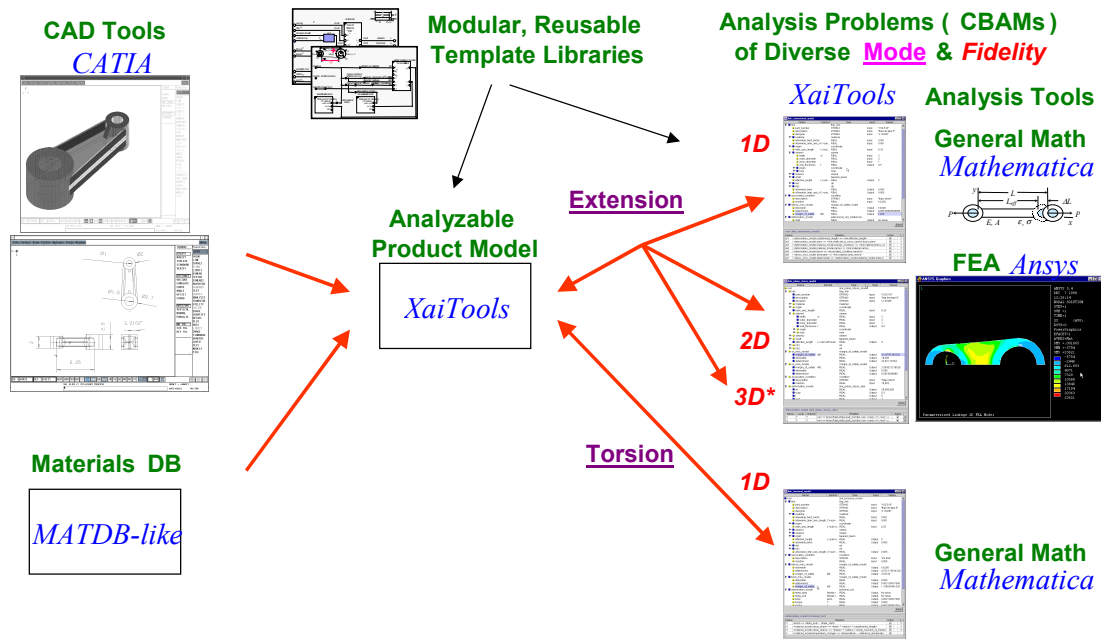
**Figure 50: Imported file of the flap link in CATIA format**

## 6.5 Geometric information used in analyses

The geometric data that was retrieved from CATIA has been used to drive analyses. Some examples are discussed below.

### 6.5.1 Flap link analyses

Using the MRA approach explained in Section 3.2, the flap link was used for extension analysis as well as for torsion analysis as shown in Figure 51 [Tamburini, 1999 #53; Wilson, 1999 #129]. The geometric data that was needed for analyses was obtained from CATIA by using the XaiTools CATIA adapter. The idealized values as well as the geometric information that was retrieved from CATIA were used in formula based analysis calculations and also in the construction of the preprocessor FEA model. Once the analysis was done, the changes recommended by the analyst were also incorporated in the design model. The values in the response file (coi format) were modified and the changed parameters were imported into the CAD system (in the standard CATIA format). The procedure for the analyses are explained below.

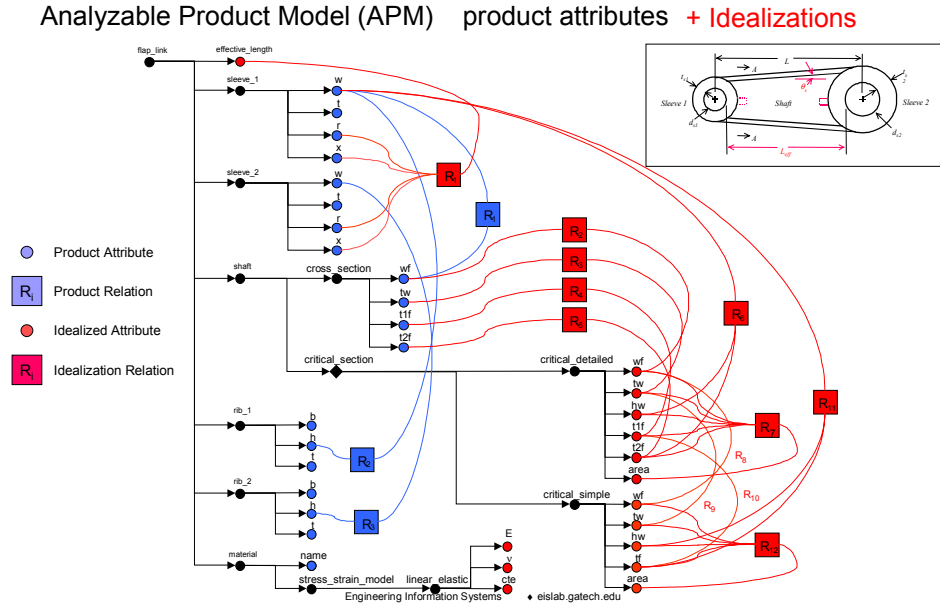


**Figure 51: Flexible Design-Analysis Integration Using MRA COBs**

#### 6.5.1.1 Flap link extension analyses

The flap link analysis model is shown in Figure 27. The purpose of the flap link extensional analysis is to compute the elongation of the flap link when it is subjected to an axial load. Two types of analyses were used for this purpose, namely, finite element analysis and formula based analysis.

Figure 52 shows the product attributes and idealized attributes that are used in analyses. The necessary product attributes were obtained from the CATIA geometric model through the adapter. The APM uses some of these parameters and computes idealized dimension values. Idealized dimension values in the case of the flap link model are the effective length and the critical section properties. The XaiTools CATIA adapter supports the definition and extraction of idealized parameters as well, but in this case, the APM computed them from the product attributes.



**Figure 52: Product attributes and idealized attributes of the flap link extension analysis (Tamburini 1999)**

In the formula based analysis, the following formula is used to calculate the elongation of a rod subjected to an axial load ‘P’ and change in temperature ‘ $\alpha T$ ’ (Gere and Timoshenko 1990):

$$\alpha L = \frac{PL}{AE} + (\alpha T) L$$

$\alpha L$  : elongation of the flap link

P : applied axial force

L : effective length of the flap link (  $L_{eff}$  )

E : Young’s modulus

$A$  : critical cross-section of the flap link ( simple or detailed )

$\sim$  : co-efficient of thermal expansion

$\alpha T$  : change in temperature

This type of model was implemented as a context based analysis model (CBAM) shown in Figure 56 which uses a portion of the APM (Block1, Figure 31). A snapshot of the COB browser shows the results of the formula based analysis in

Figure 54.

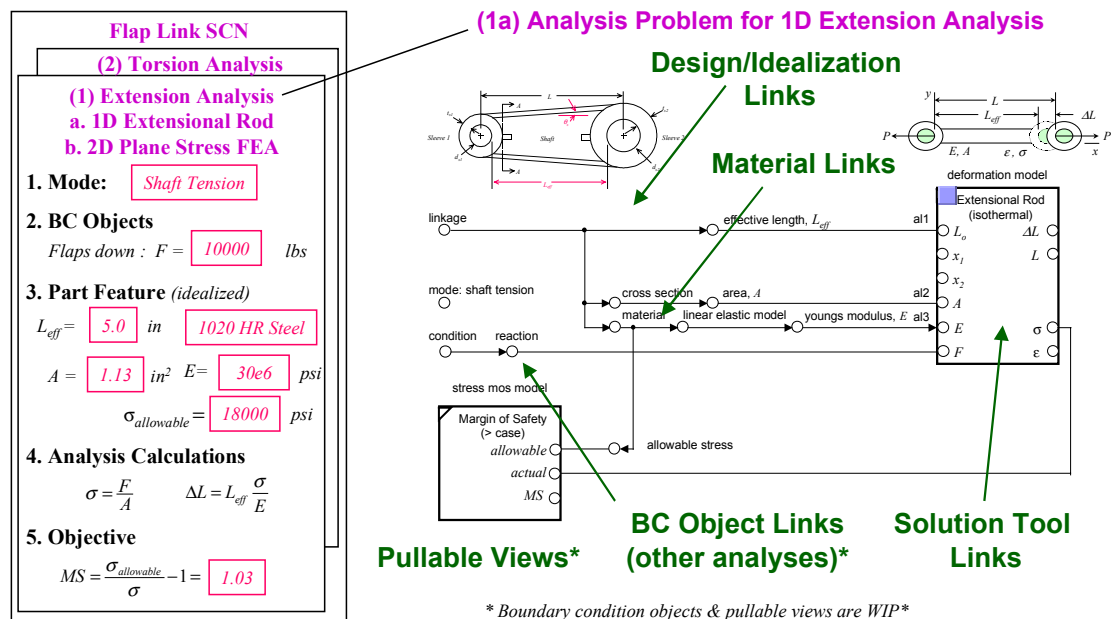


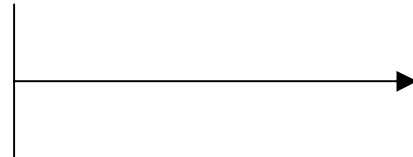
Figure 53: Representing a flap link analysis as a CBAM: Linkage Extensional Model

● reaction		REAL	Input	10,000
☐ stress_mos_model		margin_of_safety_model		
● allowable		REAL	Output	18,000
● determined		REAL	Output	8,888.88888888889
● margin_of_safety	MS	REAL	Output	1.025
☐ deformation_model		extensional_rod_isothermal		
● start		REAL	Output	No value

**Solve**

**root ( link\_extensional\_model )**

Name	Relation	Active	...	...	L...
al1	<deformation_model.undeformed_length> == <link.effective_length>	<input checked="" type="checkbox"/>	...	...	Y
al2	<deformation_model.area> == <link.shaft.critical_cross_section.basic.area>	<input checked="" type="checkbox"/>	...	...	Y
al3	<deformation_model.material_model.youngs_modulus> == <link.material.stress_st...	<input checked="" type="checkbox"/>	...	...	Y
al4	<deformation_model.material_model.name> == <link.material.name>	<input checked="" type="checkbox"/>	...	...	Y
al5	<deformation_model.force> == <associated_condition.reaction>	<input checked="" type="checkbox"/>	...	...	Y
al6	<stress_mos_model.allowable> == <link.material.yield_stress>	<input checked="" type="checkbox"/>	...	...	Y
al7	<stress_mos_model.determined> == <deformation_model.material_model.stress>	<input checked="" type="checkbox"/>	...	...	Y


 Geometric attributes  
derived from the  
XaiTools CATIA

***Figure 54: Results of formula based flap link extension analysis***

```

COB link_extensional_model SUBTYPE_OF link_analysis_model;
DESCRIPTION
  "Represents 1D formula-based extensional model.";
ANALYSIS_CONTEXT
  PART_FEATURE
    link : flap_link
  BOUNDARY_CONDITION_OBJECTS
    associated_condition : condition;
  MODE
    "tension";
  OBJECTIVES
    stress_mos_model : margin_of_safety_model;
ANALYSIS_SUBSYSTEMS */
  deformation_model : extensional_rod_isothermal;
RELATIONS
  al1 : "<deformation_model.undeformed_length> == <link.effective_length>";
  al2 : "<deformation_model.area> == <link.shaft.critical_cross_section.basic.area>";
  al3 : "<deformation_model.material_model.youngs_modulus> ==
        <link.material.stress_strain_model.linear_elastic.youngs_modulus>";

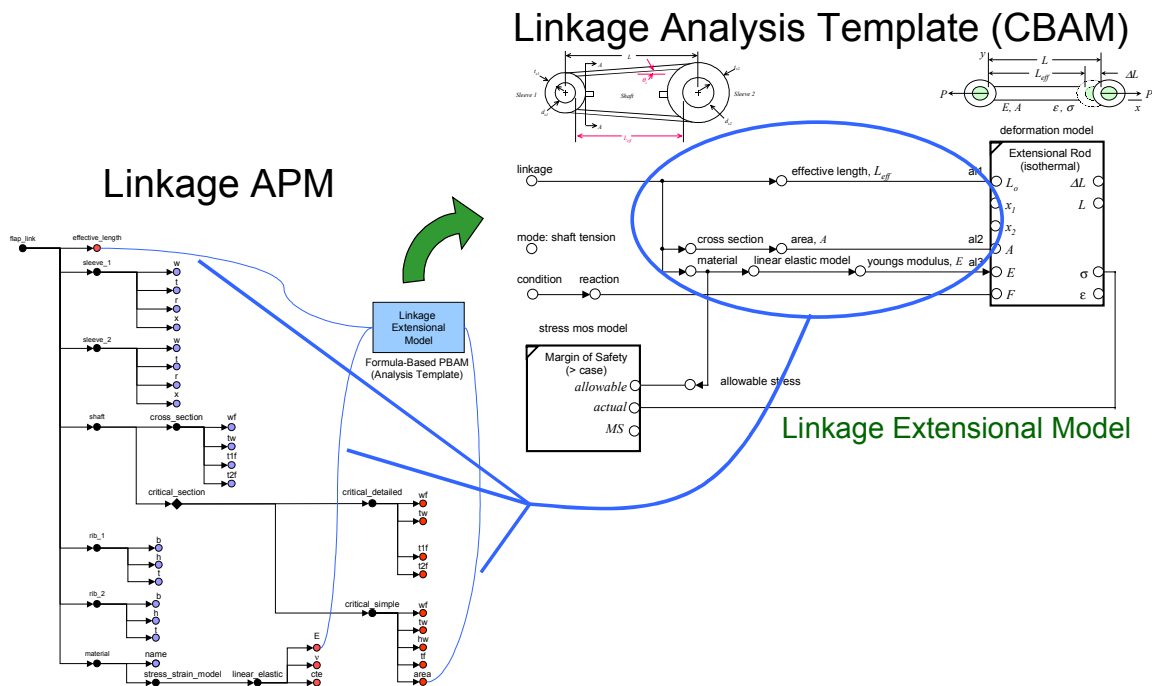
  al4 : "<deformation_model.material_model.name> == <link.material.name>";
  al5 : "<deformation_model.force> == <associated_condition.reaction>";

  al6 : "<stress_mos_model.allowable> == <link.material.yield_stress>";
  al7 : "<stress_mos_model.determined> == <deformation_model.material_model.stress>";
END_COB;

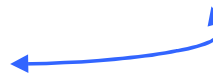
```

*Desired categorization of attributes is shown above (as manually inserted) to support pullable views.  
Categorization capabilities is a planned XaiTools extension.*

**Figure 55: COB Lexical Form for Linkage Extensional Model CBAM**

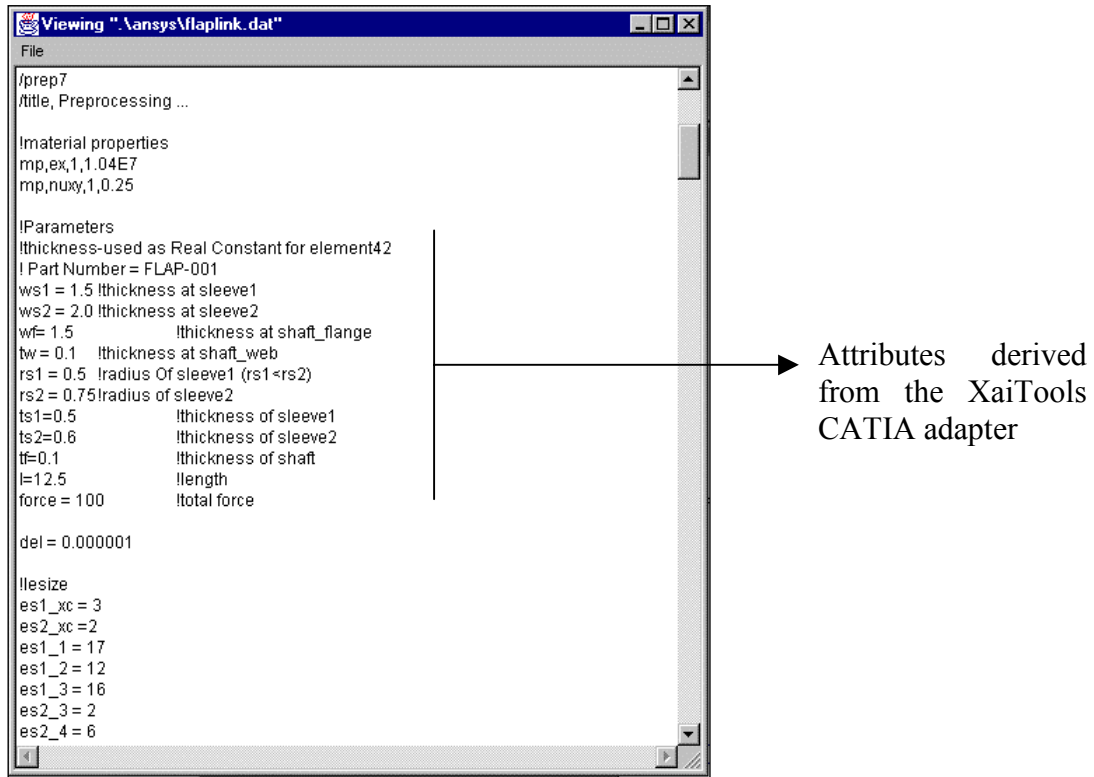


**Figure 56: CBAM Usage of APM-based Idealizations**

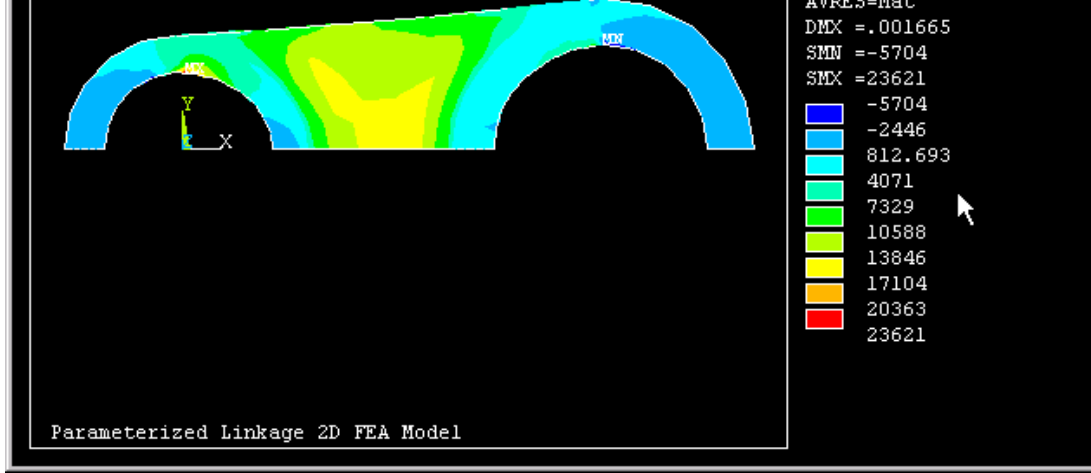


In the case of finite element analysis, the attributes obtained from the CATIA XaiTools CATIA adapter were used to create an ANSYS preprocessor file (Prep7 file). The file is shown in Figure 58. A snapshot of the solved ANSYS model is shown in Figure 59. Note that while more APM attributes are required for this CBAM (Figure 57), some are the same as those used in the lower fidelity version (Figure 52).





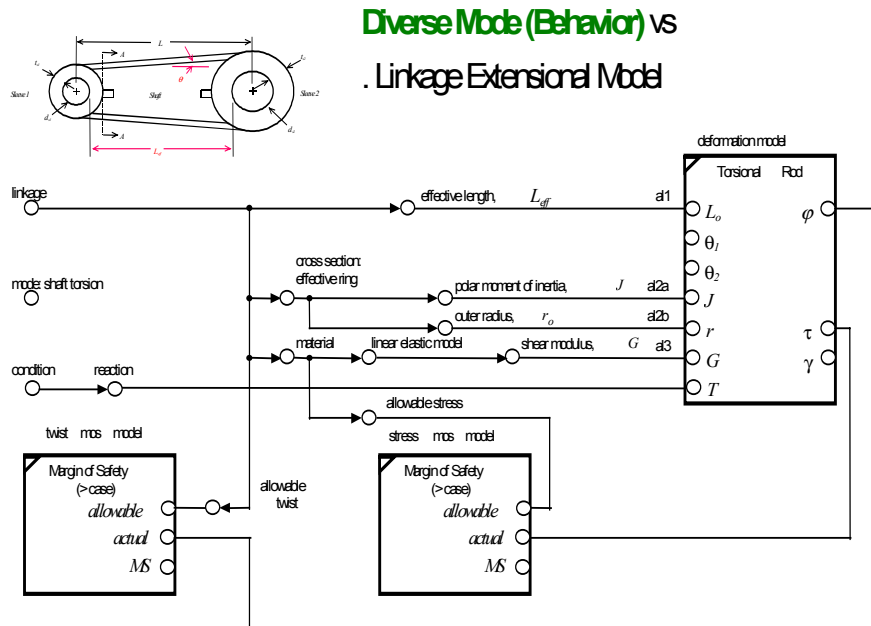
**Figure 58:** Preprocessing file (Prep7) sent to ANSYS (partial)



***Figure 59: Solved finite element model of the flap link ( ANSYS )***

#### ***6.5.1.1.1 Analysis of the flap link under torsional loading***

The flap link was also tested and analyzed under torsional loading conditions. Some of the same geometric parameters that were extracted for the flap link extension analysis were used for the torsional analysis. Figure 60 shows the CBAM of the flap link torsional model while Figure 61 shows the results of the flap link analysis under torsional loading in the COB browser.



**Figure 60: Flap link torsional CBAM**

reaction		REAL	Input	5,000
stress_mos_model		margin_of_safety_model		
allowable		REAL	Output	18,000
determined		REAL	Output	4,703.115814226..
margin_of_safety	MS	REAL	Output	2.82725
twist_mos_model		margin_of_safety_model		
allowable		REAL	Output	0.005
determined		REAL	Output	0.002139917695
margin_of_safety	MS	REAL	Output	1.336538461538
deformation_model		torsional_rod		
theta_start	&theta;<...	REAL	Output	No value
theta_end	&theta;<...	REAL	Output	No value
twist	&phi;	REAL	Output	0.002139917695
torque	T	REAL	Output	5,000
radius	r	REAL	Output	0.952380952381

Solve

deformation_model (torsional_rod)				
Name	Relation	Active	...	...
r1	<twist> == <theta_end> - <theta_start>	<input checked="" type="checkbox"/>	...	Y
r2	<material_model.shear_strain> == <twist> * <radius> / <undeformed_length>	<input checked="" type="checkbox"/>	...	Y
r3	<material_model.shear_stress> == <torque> * <radius> / <polar_moment_of_inertia>	<input checked="" type="checkbox"/>	...	Y
r1	<material_model.temperature_change> == <temperature> - <reference_temperatur...	<input checked="" type="checkbox"/>	...	...

**Figure 61: Results for the flap link torsion analysis**

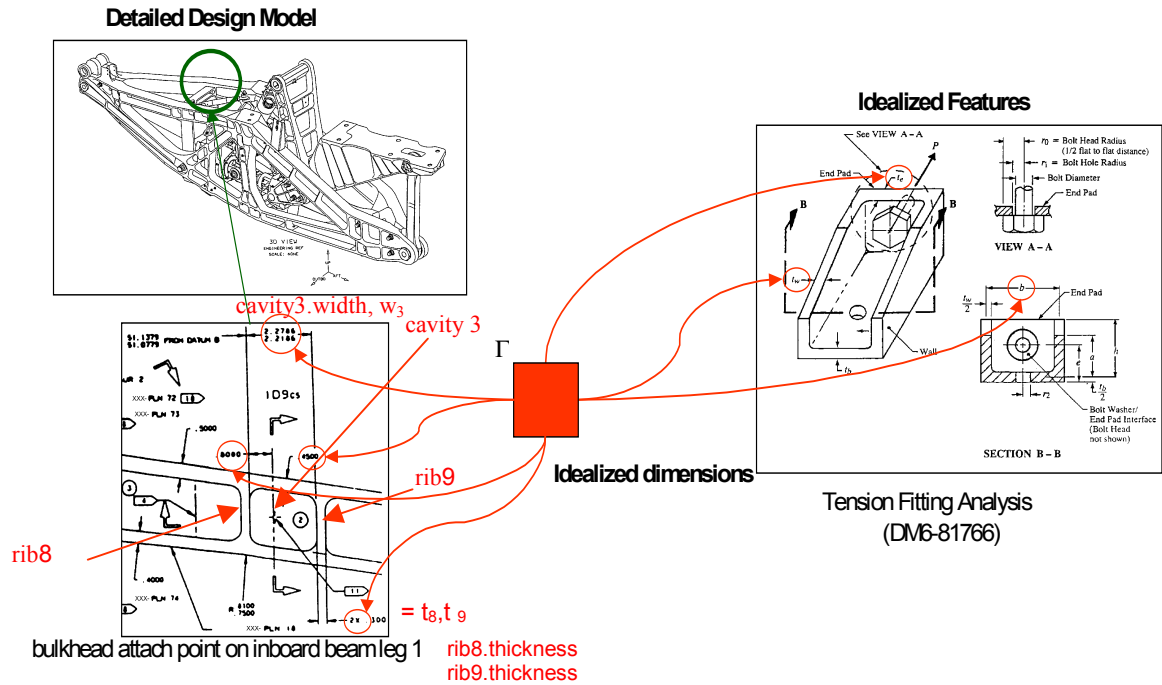
## 6.5.2 Bike frame inboard beam analysis

The detailed design model in Figure 62 shows a point of attachment of the inboard beam in the bike frame design of a typical aerospace system, and it is called ‘bulkhead attach point’.

The purpose of the analysis is to estimate the stresses and allowable loads at various critical points in the bulkhead attachment point of the inboard beam, caused by loads transmitted by the fastener that is attached to the bulkhead. In such cases, analysts typically choose standard analysis templates for generic channel fittings that are available in corporate design manuals and implemented in analysis tools.

The dimension values from the CATIA design model (CAD model) are needed to compute idealized attributes that are needed by the idealized features in the analysis. For example, as shown in Figure 62, the analysis attributes are derived by using the design attributes (from the CATIA model) in the following relations:

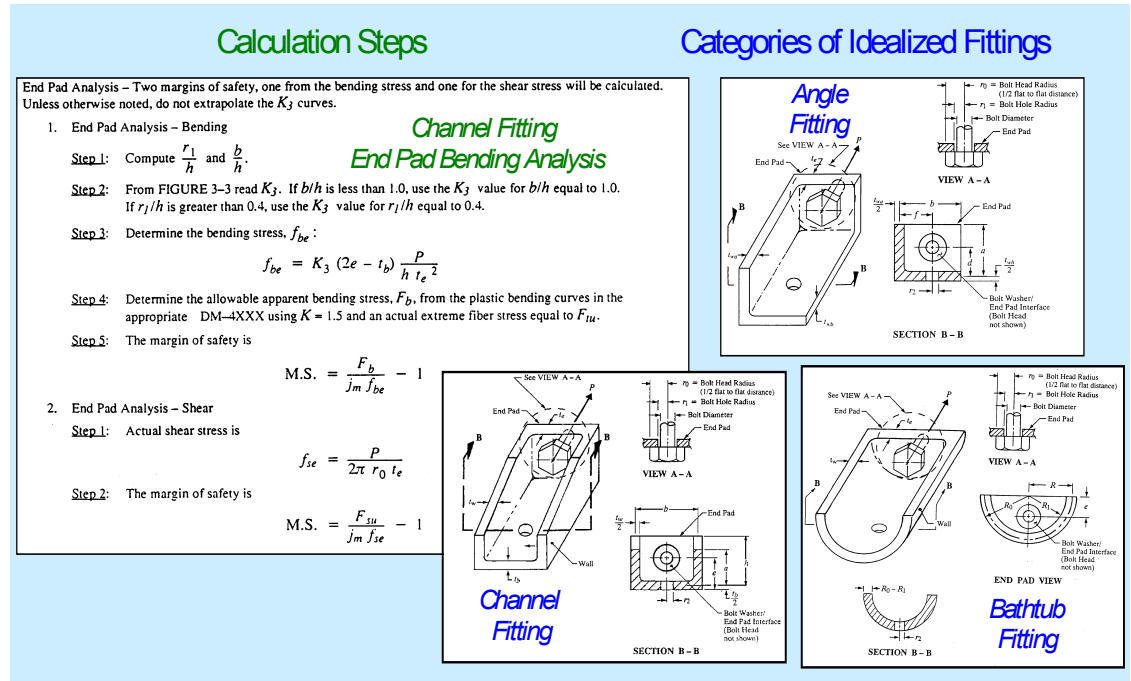
$$\begin{aligned}b &= \text{cavity3.inner\_width} + \text{rib8.thickness}/2 + \text{rib9.thickness}/2 \\t_e &= \text{cavity3.base.minimum\_thickness}\end{aligned}$$



**Figure 62: CAD and analysis attributes for the bulkhead fitting analysis**

Idealization attributes like those in Figure 62 needed for analysis models are often contained in design manuals and electronic templates compiled by companies, professional organizations or academic publications. They are normally well established, tested and known to provide accurate results (Tamburini 1999).

The idealized analysis parameters form the analysis model geometry. There is typically no explicit automated link that exists between the parameters in the CAD model and those that are used in the analysis model, as shown in Figure 63 and Figure 64.



**Figure 63: Typical design manual description of general fitting analyses without design associativity**

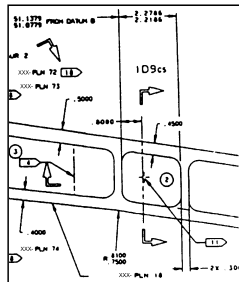
It is important to note that the parameters or dimensions that have been labeled in the CAD model are not just real numbers, but they also form an inherent part of the information content in the CAD model. Figure 65 shows a CBAM for the bike frame.

Figure 66 shows that the approach used in this study links the CAD attributes to the attributes used in analysis model geometry in this CBAM. The approach adopted enables the explicit automated link between the design and fitting analysis of the bulk head attach point.

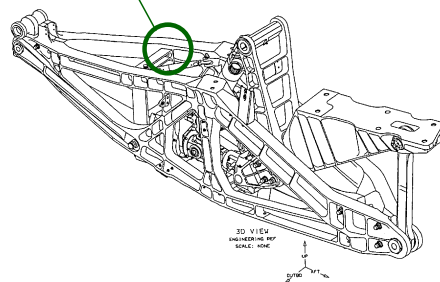
## channel fitting analysis

## CAD Model

bulkhead assembly attach point



*No explicit  
fine-grained  
CAD-CAE  
associativity*



material  
properties

idealized  
geometric  
attributes

analysis  
results

LINKAGE SUPPORT NO. 2 (INBOARD BEAM REF [1234567])  
Bulkhead Assembly Attach Point at Upper Beam Location

BATHYTUP TYPE TENSION FITTING ANALYSIS  
REF: DMR-81766, "Tension-type fittings"

**Material Properties & Geometry:**

Pu =	67000	PSI
PtUL =	65000	PSI
Fcy =	57000	PSI
FyUL =	52000	PSI
Fsu =	39000	PSI
epu =	0.067	IN/IN
epUL =	0.030	IN/IN
tw =	0.310	IN
b =	1.267	IN
c =	2.440	IN

**TENSION FITTING TYPE**  
CHANNEL FITTING

**Wall Tension Analysis:**

Anec =	1.848	IN <sup>2</sup>
Agross =	1.848	IN <sup>2</sup>

**Wall Bending Analysis:**

Kwall =	1.803	
Ftw =	132791	PSI
Mu =	60428	LB-IN
Rdw =	0.058	(Actual)

**Wall Bending & Tension Interaction:**

n =	1.29
gamma =	0.913

**\*\*\*\*\* PLASTIC BENDING ANALYSIS \*\*\*\*\***

Rcu =	0.490	(Allowable)
Rdu =	0.931	(Allowable)
Mdwall =	9.77	

**End Pad Bending Analysis:**

\*\*\*\*\* PLASTIC BENDING ANALYSIS \*\*\*\*\*

K3 =	0.591	
Fbe =	15038	PSI
Fbe =	91844	(Allowable)
Mdsep =	5.11	

**End Pad Shear Analysis:**

fse =	3620	PSI
Mdsep =	9.77	

**Allowable Loads:**

Pallow =	36395	LBS
----------	-------	-----

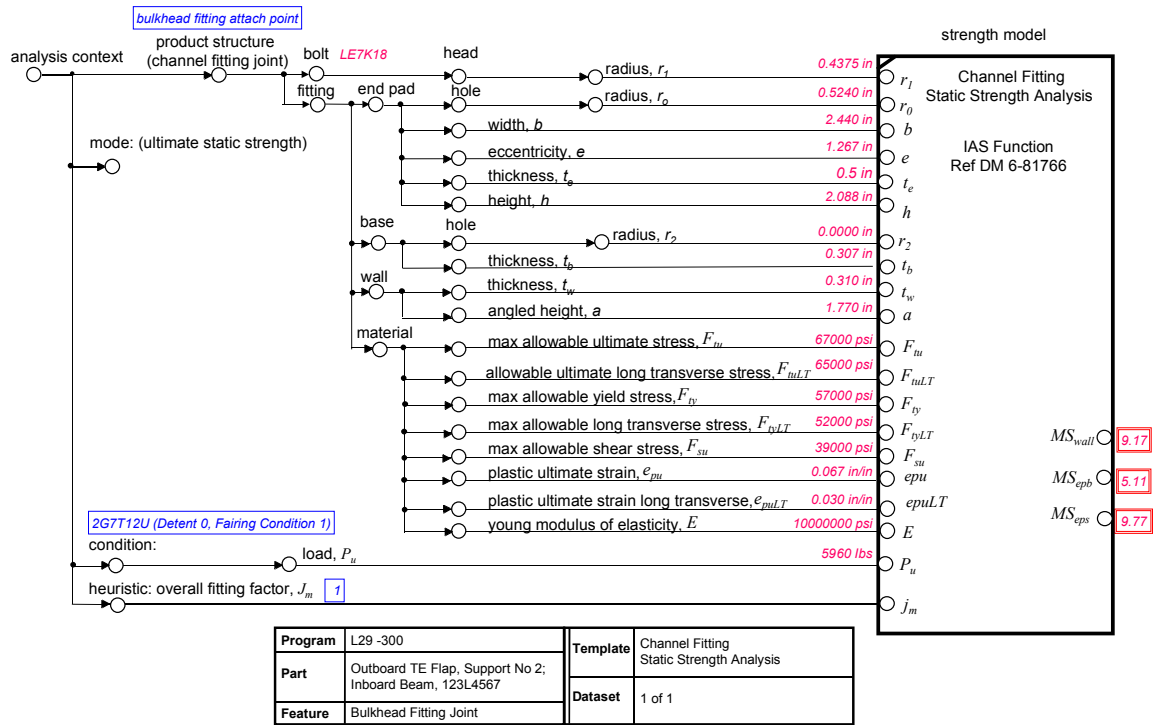
WARNING: edge distance "h - e - tb/2" should be at least twice the hole DIAMETER (2(r)) from the free edge to prevent tension failure in wall.

Fastener is L67K18 and represented as beam element number 362 in FEA model. Load considered is 2GT12U intact (Detent 0, Failing Condition 1) and is obtained from the FEA model axial beam loads.

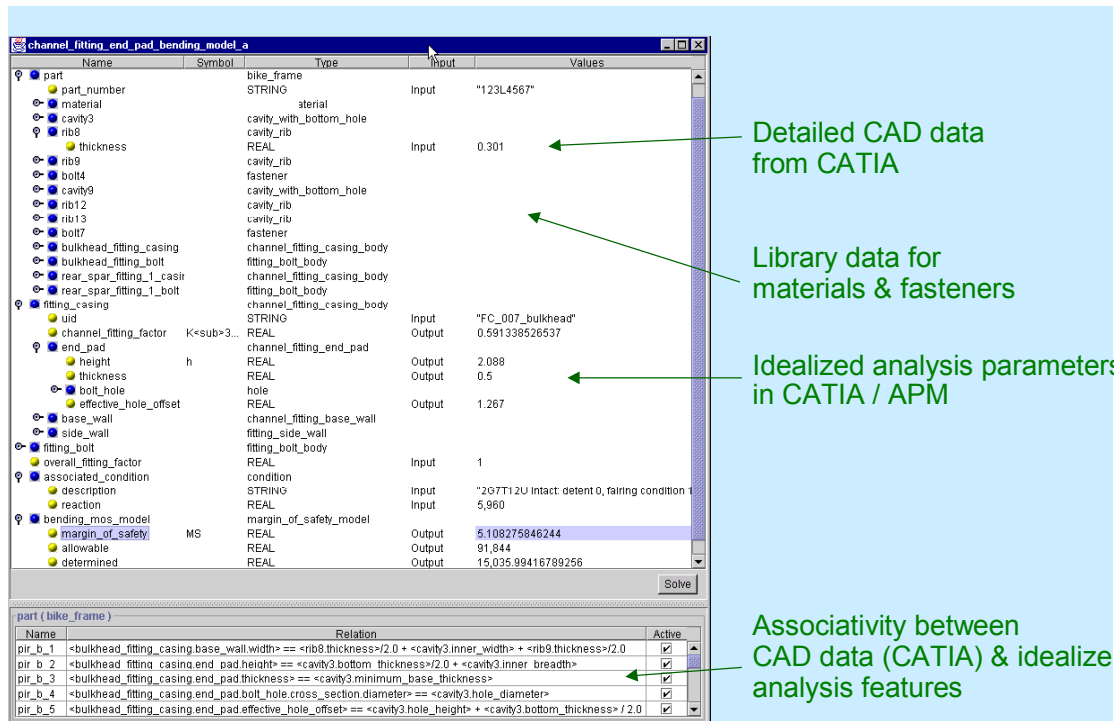
ENGR.	NAME	12/20/98	REVISED	DATE	Outboard TE Flap, Support No. 2 Bulkhead Attachment Location to [1234567] ibulkx.com ibulkx.dta ENGINEER DEVELOPED TEMPLATE	D-9 300
CHECK						
APP						
APP						
POW	a734-07-PRGD	IAS				page 206

*Figure 64: Typical current practice without explicit design associativity*





**Figure 65: Bike Frame Bulkhead Fitting Analysis: Implementation as a CBAM (Constraint Schematic Instance View)**



**Figure 66: COB-based Bulk head fitting analysis results with CAD associativity**

## 6.6 Geometric attributes retrieved for test cases

The geometric attributes of different entities and primitives that were obtained from test case design models from the XaiTools CATIA adapter have been listed in Table 4. Although the magnitude of geometric dimensions may be the same, different CAD entities were tagged in the different approaches. For example in the geometric entity tagging approach for the back plate, a circle entity was labeled in the three dimensional model. In the dimension tagging technique, the radius dimension entity on a 2D draft view was tagged and the value was obtained from the draft view and not from the three-dimensional CAD model. In the case of the parametric approach, a radius parameter was tagged in the 3D design model.

From the table, it can be observed that the co-ordinates of points can only be obtained from the geometric entity approach. The attributes of a cross-section or of any

section in a design model can only be extracted by using the dimension approach. The parametric approach is unique as it supports attributes to be exported from as well as imported into the CAD system.

Note that, while the bike frame CAD model is quite complex, the information needed for its analysis is not so complex. This complexity reduction is typical of the analysis idealization process.

*Table 4: Geometric attributes, dimensions and parameters that were retrieved from the test cases*

Design test cases	Geometric entity tagging approach	Dimension entity tagging approach	Parameter tagging approach
	(Tagged:Geometric Entities)	(Tagged:Dimension entities)	(Tagged: Parameter entities)
Back plate	Lines Circles Points CSG primitives	Lengths of Lines Radii of circles Distances between lines Distances between points Distances between points & lines Idealized attributes	Lengths of Lines Radii of Circles Distances between points Distances between lines Distances between points & lines Idealized attributes
Flap link	Points Lines Circles	Length of a Line Radius of a Circle Angle between two lines Distances between points Distances between lines Sectional views (possible)	Lengths of Lines Radii of Circles Angle between lines Distances between points Distances between lines Distances between points & lines Idealized attributes
Bike frame	Not done	Lines Radii of circles Idealized attributes Distances between lines	Not done

## 6.7 Comparison of approaches used in CATIA test cases

The three approaches that were used in the test cases in order to obtain geometric data needed for analyses have been compared with the different characteristics in Table 5. This table is used later in order to draw conclusions on the approach that may be most appropriate for engineering analysis requirements.

Row 1 compares the different types of CAD entities that are tagged in the design model. Row 2 indicates the types of attributes that may be extracted by the approaches. Row 3 discusses the ease and feasibility of tagging the different CAD entities and it was observed from the test cases that labeling the geometric entities was difficult and impractical for complex design models. Row 4 compares the ability of the approaches to extract selective attributes and not all the geometric attributes of a CAD model and it was observed that all approaches supported selective extraction of attributes. The possibility of bi-directional flow of information is discussed in Row 5; it can be observed that only the parametric approach supports this capability. Row 6 lists whether the magnitudes of the extracted attributes are true lengths or otherwise. Row 7 compares the possibility of defining a dimension as a relation and only the parametric approach allows this type of definition. Rows 8 and 9 discuss the time taken to extract attributes and the ease of programming while using the different approaches. The type of documentation of analysis attributes in the design model, i.e. the link that exists between the design and analysis attributes is compared in Row 10.

Rows 11 and 12 list the drawbacks and overall judgement of the different approaches for extracting attributes from CAD systems.

*Table 5: Comparison of the different approaches used in CATIA test cases*

<b>Characteristic</b>	<b>Geometric entity tagging approach</b>	<b>Dimension entity tagging approach</b>	<b>Parameter tagging approach</b>
1) Entities tagged	Geometric entity/primitive in 3D/2D space	Dimension entities in 2D draft views	Parameters of a parametric model in 3D/2D space
2) Attributes that may be extracted	Attributes of geometric entities/primitives	Dimensions in 2D draft views	Parameters in a 3D solid model except sectional properties
a) Idealized attributes b) Cross-section attributes	No No	Yes Yes	Yes No
3) Ease of tagging	Easy for a few basic attributes but complicated for complex models	Easiest (after creation of draft views)	Easier (but must plan the parametric approach)
4) Supports extraction of partial CAD attributes	Yes	Yes	Yes
5) Supports design change with same topology	Yes, if the model is parametric; not otherwise.	Yes, if the model is parametric; not otherwise.	Yes, always
6) Magnitude of attributes	True length values	Projected length values; true sometimes	True length values
7) Change a design dimension to a relation	No	No	Yes
8) Time taken for test cases	Five seconds	Less than five seconds	Less than five seconds
9) Ease of programming	Complex; several functions are needed	Simple	Simple
10) Documentation of analysis parameters in the design model	Structured	Structured	Structured
11) Drawbacks	Not practical for complex models	Requires the draft view creation	Cannot easily obtain cross section dimensions
12) Overall judgement	Not for complex models	OK	OK

## 6.8 Approaches satisfying the thesis objectives

Table 6 lists the objectives of this thesis that have been satisfied by the three different approaches adopted in this study. The objectives of the thesis have been discussed in Section 4.1.

Row 1 lists the different types of entities whose attributes can be obtained. Row 2 implies that idealized attributes cannot be obtained by the geometric entity approach; however, it is partially supported by the dimension approach and completely supported by the parametric approach. Row 3 indicates that partial attributes for analysis can be derived from all the three approaches, and that complete geometric data need not be extracted each time. Row 4 in the table implies that the bi-directional flow of information from and to a CAD system is supported by the parametric approach only. Row 5 compares the feasibility of using these three approaches for obtaining geometric attributes in other CAD systems. Row 6 implies that all three approaches alleviate errors that may be caused by manual entry of geometric data while creating analysis model geometry. Row 7 implies that the same geometric data may be used by several analysis tools and hence is re-usable.

This table shows that the thesis objectives have been satisfied to a large degree.

***Table 6: Table comparing the three approaches against the objectives of this thesis***

<b>Thesis objective</b>	<b>Geometric entity tagging approach</b>	<b>Dimension tagging approach</b>	<b>Parametric tagging approach</b>
1) Extraction of geometric entity information	Points, Lines, circles, CSG primitives	Dimension values from any draft view	Geometric parameters, idealized parameters and relations
2) Extraction of idealized attributes	Not usually supported	Supported in some cases with the exception of complex relations	Supported
3) Allows extracting data for a part of a design assembly	Yes	Yes	Yes
4) Bi-directional flow of information	Not supported, unless the model is a parametric model	Not supported, unless the model is a parametric model	Supported (under the model constraints)
5) Allow multiple analysis tools to use the CAD data (neutral format)	Yes	Yes	Yes
6) Approach is valid for most CAD systems	Yes	Yes	Yes
7) Alleviates errors caused by manual data entry	Yes	Yes	Yes
8) Support geometry needs for multiple analysis models	Yes	Yes	Yes
9) Support multiple analysis solution methods	Yes	Yes	Yes



## CHAPTER VII

### CONCLUDING REMARKS

#### 7.1 Conclusions and summary of contributions

The main purpose of this study has been to facilitate the integration of design and analysis by enabling the analyst to selectively extract geometric dimensions from a CAD system, especially in cases where the design geometry and its analyses geometry (i.e. do not have a 1:1 correspondence). This study has developed a technique which captures design-analysis associativity and facilitates the analyst to choose specific entities from CAD models that maybe needed for multiple analyses, and extract the geometric attributes of the same. These attributes were used to drive finite element and formula based analyses. The use of these attributes aids the analyst to create analysis model geometry for finite element analysis (FEA). This technique can benefit the analyst in one or more of the following ways, each of which has been described in detail in Section 2.3:

- a) To remove unnecessary geometric details from a CAD model
- b) To analyze just a portion of the design geometry or part of a whole design assembly
- c) To obtain idealized dimension values for analysis model geometry
- d) To generate multiple analyses models of varying fidelity and/or discipline, for a given design model
- e) To create dimensionally reduced analysis models
- f) To create FEA models that may be meshed successfully

- g) To achieve geometric symmetry for analysis models
- h) Aid the analyst to use these parameters and dimensions in formula based analysis computations.

Three approaches were identified in order to extract the geometric information from a CAD model. A technique to ‘tag’ or ‘label’ CAD geometric entities was also identified. All three approaches use this ‘tagging’ technique in order to extract geometric information. These three approaches together enable the following dimensions, parameters and attributes of CAD geometric entities to be obtained:

- a) Points
- b) Lines
- c) Circles
- d) CAD dimension entities from any draft view, including sectional views
- e) Any geometric parameter from a parametric CAD model
- f) Idealized dimension values/relations from a parametric model
- g) Geometric attributes of CSG primitives (cylinders, spheres etc.)

The approaches were compared and it was inferred that the ‘parameter tagging approach’ is most useful for engineering analyses, as this technique supports defining idealized attributes and the extraction of geometric parameters from a 3D parameterized CAD model. This approach also supports bi-directional flow of geometric information i.e. it is also possible to import a file with a list of parameters whose values need to be changed in the CAD model. The degree of bi-directionality depends on the CAD system’s ability to change input/output directions for an attribute.

The ‘dimension tagging approach’ is simple and should be used when dimensions from a cross-sectional view or dimensions from any two-dimensional draft view are needed for analyses. The technique can frequently be used to extract critical cross sectional dimensions of a design model.

The ‘geometric entity tagging approach’ is the only approach by which one can obtain all the three coordinates of a point entity in three-dimensional space coordinates and would need to be used for the same purpose. However, this approach to extract attributes of geometric entities is tedious for complicated design models.

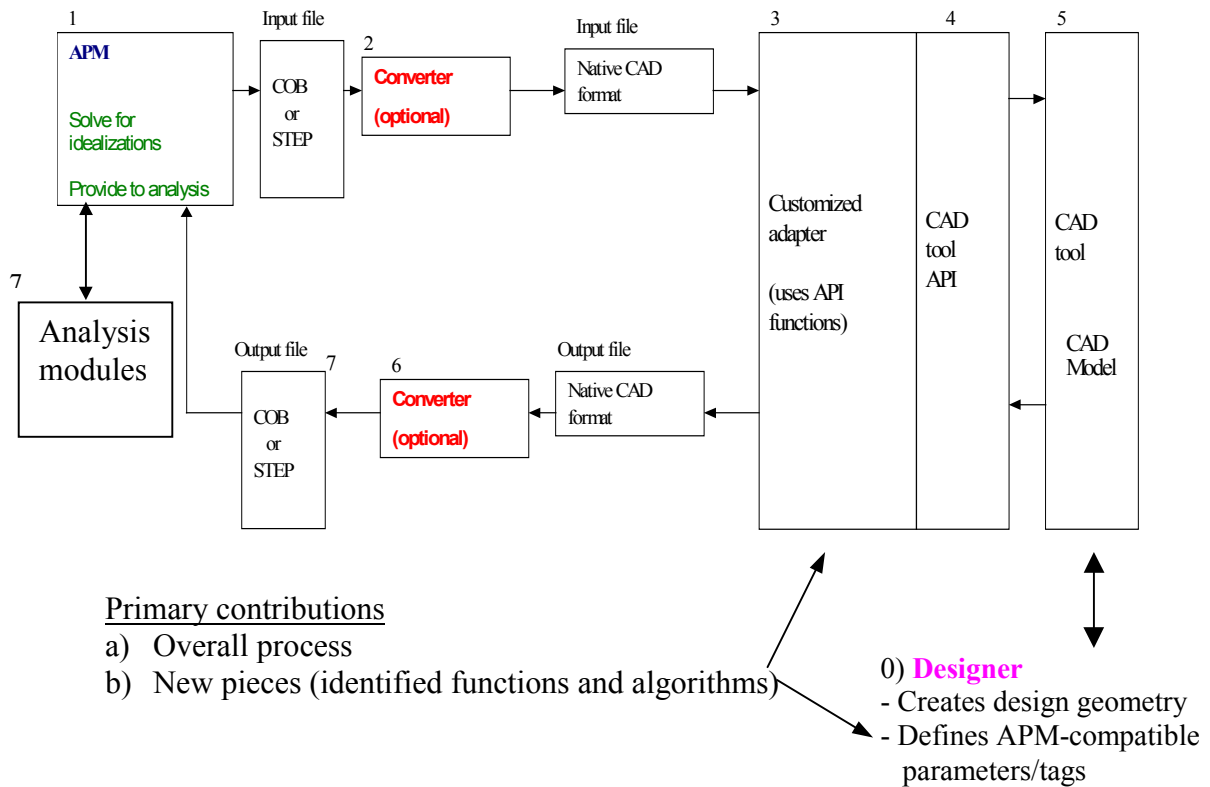
The above three approaches were implemented and tested in an interface adapter for CATIA. The adapter program was written in ‘Tk/tcl’ and ‘C’ programming languages. The interface capabilities of different CAD systems were studied. The logical flowchart that was used to implement the CATIA XaiTools CATIA adapter shown in Figure 22 is believed to be valid for most CAD systems. Thus, the approaches that were identified in order to obtain geometric information for analysis geometric models can be generalized as appropriate for most CAD systems.

Three test cases were used in this study for each of the three approaches. The extracted CAD data from the test case design models was then used to create analysis model geometry for both formula-based and FEA-based analysis computations. The CAD data was also used to create analysis model geometry for whole design models as well as partial design models.

Further, the test cases that were used in the ‘parametric approach’ were tested for bi-directional flow of geometric information, i.e. flow of data from the CAD system and to the CAD system. This implies that the XaiTools CATIA adapter allows the designer to make changes in the geometric dimensions of the CAD design model (with the same topology), as recommended by an analyst. This facilitates continuous design and analysis iterations.

The methodology that is proposed in this thesis also helps alleviate errors that may arise from manual data entry while creating analysis model geometry.

## 7.2 Contributions



**Figure 67: Blocks that constitute the design and analysis integration scenario**

The distinct contributions of this thesis are highlighted in Figure 67 and briefly described below:

- The types of geometric attributes needed for creating multi-fidelity and multi-disciplinary analysis model geometry as well as those needed for formula based analysis were identified, and are explained in Section 2.3.2.

- The overall process of Figure 67 was developed in this thesis. While some blocks pre-existed, the relevant blocks and how they work together in order to achieve thesis objectives had to be determined.
- The general algorithm and functions needed in a customized interface adapter (block 3, Figure 67) in order to extract the attributes of CAD entities were identified and developed as part of this thesis study. The algorithm and functions are outlined in Figure 22 and ‘block 3’ of Figure 67 uses them to facilitate the extraction of geometric information from a CAD model for the purpose of using it in diverse analyses. Although an example customized adapter was written in Tk/tcl and implemented in the CATIA CAD system, the concepts used in this adapter were generalized for typical CAD systems. Thus, the approach used to achieve the objectives of this thesis study may be used in most modern CAD systems, such as, Pro/Engineer, IDEAS and CATIA.

### 7.3 Recommendations

In order to accomplish similar results as this thesis study, once the CAD model is tagged with user-specified tags, the geometric attributes of the CAD model could be retrieved directly from a neutral file such as a CAD AP203 STEP file. At the beginning of this study AP203 translators were not known to support this capability. The three approaches that were used in this study may be used as they are, except that a tagged STEP AP203 file would have to be searched for the geometric entities that are needed for analysis. Instead of using a 'coi' request file for the list of all the geometric parameters that are needed for analysis, a STEP AP203 file would be used for the same purpose. A response file can be generated in a similar manner, with the list of requested attributes and their corresponding values beside them. Although using a STEP file would standardize the XaiTools CATIA adapter and make it compatible with the ISO standard, the file would have a large amounts of unnecessary data contained in it. This would mean that the time taken to extract geometry would potentially be much greater when using a STEP file. One would need to strike a balance between conforming to a neutral standard and the time taken to achieve the desired results. One would finally have to choose between the two, depending on which of the two factors is of greater importance.

The request file generated by the analyzable product model (APM) is capable of querying attributes of CAD entities. However, it could be very useful if the APM could support higher level requests based on a sequence of operations. As an example, in the case of the flap link design model, the sequence may be:

- a) Take a section defined by a 'plane1' at 'sleeve1'
- b) Get the area of the sliced section

Presently, idealized relations may be defined in the APM or in the parameterized CAD design model. It would help to investigate where best to represent and calculate idealized relations.

The current methodology may also be extended in order to support the interaction between APMs and CAD systems via standards such as CORBA.

Finally, the ability to automatically coordinate APM tags with CAD model tags would increase ease of use (versus the manual user-performed tagging that is currently required). Means to automatically morph between design geometry and idealized analysis geometries (or have them co-exist in CAD systems) would be even better.

# Appendix A

## Analyzable product model schemas and material models

### SCHEMA MODELS (.COS)

#### 1. Geometric entity tagging approach

##### a) Backplate APM (complete description)

APM backplate;

(\*  
Case: 1  
Version: 980901  
Syntax: cob v2.0

Purpose: Demonstrate simple geometric aspects of an apm including attributes that come from a cad model. See figure for definition of parameters.

Case Characteristics:

- Geometric primitive-based links to a cad model [[design features are defined by relations with geometric primitives defined in a cad model]].
- "Full" description [[All necessary & sufficient inter-relations among design features are specified. All necessary & sufficient relations between design features and geometric primitives are specified. All inter-relations among geometric primitives implied by the figure are not given here [[e.g., line1.start=line2.end, etc.]].
- Coordinates expressed wrt global coordinate system.

Copyright [[C]] 1998  
Georgia Tech Engineering Information Systems Lab  
eislabs.gatech.edu  
)

SOURCE\_SET full\_apm ROOT\_DOMAIN backplate;

DOMAIN part;  
part\_number : STRING;  
designer : STRING;  
origin : coordinate;  
END\_DOMAIN;

DOMAIN backplate SUBTYPE\_OF part;

(\* design features \*)  
length1 : REAL;  
length2 : REAL;  
length3 : REAL;  
length4 : REAL;  
width1 : REAL;  
width2 : REAL;  
thickness : REAL;  
hole1 : hole;  
hole2 : hole;  
material : material;  
area : REAL;



```

(* geometric primitives *)
circle1 : circle;
circle2 : circle;
line1 : line;
line2 : line;
line3 : line;

IDEALIZED critical_area : REAL;
(* area at hole1 cross section - assumes hole1 > hole2*)
IDEALIZED effective_span : REAL;
IDEALIZED span_reduction_factor : REAL;

PRODUCT_RELATIONS
pr1 : "<length1> == <length2> + <length3> + <length4>";
pr2 : "<width2> == <width1> / 2";
(* hole1.origin == origin + #[[0, length2, width2]] *)
pr3 : "<hole1.height> == <thickness>";
pr4 : "<hole1.origin.x> == <origin.x>";
pr5 : "<hole1.origin.y> == <origin.y + length2>";
pr6 : "<hole1.origin.z> == <origin.z + width2>";
pr7 : "<hole2.height> == <thickness>";
(* hole2.origin == hole1.origin + #[[0, length3, 0]] *)
pr8 : "<hole2.origin.x> == <hole1.origin.x>";
pr9 : "<hole2.origin.y> == <hole1.origin.y + length3>";
pr10 : "<hole2.origin.z> == <hole1.origin.z>";
pr11 : "<area> == <width1> * <thickness>";

(* relations with geometric primitives [[in cad model]] *)
prp1 : "<length1> == <line2.length>";
prp2 : "<width1> == <line1.length>";
prp3 : "<thickness> == <line3.length>";
(* hole1.cross_section == circle1 *)
prp4 : "<hole1.cross_section.radius> == circle1.radius";
prp5 : "<hole1.origin.x> == circle1.origin.x";
prp6 : "<hole1.origin.y> == circle1.origin.y";
prp7 : "<hole1.origin.z> == circle1.origin.z";
(* hole2.cross_section == circle2 *)
prp8 : "<hole2.cross_section.radius> == circle2.radius";
prp9 : "<hole2.origin.x> == circle2.origin.x";
prp10 : "<hole2.origin.y> == circle2.origin.y";
prp11 : "<hole2.origin.z> == circle2.origin.z";

PRODUCT IDEALIZATION_RELATIONS
pir1 : "<critical_area> == (<width1> - <hole1.cross_section.diameter>) * <thickness>";
pir2 : "<effective_span> == <length3> - (<hole1.cross_section.radius> + <hole2.cross_section.radius>) *
<span_reduction_factor>";
END_DOMAIN;

(* --- part features --- *)

DOMAIN hole;
origin : coordinate;
cross_section : circle;
height : REAL;
volume : REAL;
RELATIONS
r1 : "<volume> == <height> * <area>";
r2 : "<cross_section.origin.x> == origin.x";
r3 : "<cross_section.origin.y> == origin.y";
r4 : "<cross_section.origin.z> == origin.z";
END_DOMAIN;

DOMAIN material;
(* actually a material model *)
name : STRING;

```

```

    youngs_modulus : REAL;
    poissons_ratio : REAL;
    cte : REAL;
END_DOMAIN;

(* --- geometric primitives --- *)

DOMAIN coordinate;
  x : REAL;
  y : REAL;
  z : REAL;
END_DOMAIN;

DOMAIN line;
  start : coordinate;
  end : coordinate;
  length : REAL;
RELATIONS
  r1: "<length> == ((end.x - start.x)^2 + (end.y-start.y)^2 + (end.z-start.z)^2)^0.5";
END_DOMAIN;

DOMAIN circle;
  origin : coordinate;
  radius : REAL;
  diameter : REAL;
  area : REAL;
RELATIONS
  r1: "<diameter> == 2 * <radius>";
  r2: "<area> == Pi * <radius>^2";
END_DOMAIN;

END_SOURCE_SET;

END_APM;

```

## b) Backplate APM (partial description)

SCHEMA back\_plate;

/\*

Case: 2

Syntax: cob v2.1

Authors: A. Chandrasekhar, R. Peak, D. Tamburini

Purpose: Demonstrate simple geometric aspects of an apm including attributes that come from a cad model. See figure for definition of parameters.

Case Characteristics:

- Geometric primitive-based links to a cad model (same as Case 1).
- "partial" description (compared to Case 1, the only geometric entities present are those necessary for calculating area and effective\_span)
- Coordinates are expressed wrt global coordinate system & all values are represented in CATIA model units. All angles are expressed in degrees.
- Uses the same CATIA model as case1

Copyright (C) 1998  
Georgia Tech

Versions:  
981231  
- initial release

Other Possibilities:  
- catia model needs to answer designer, and material name, etc. (manually added in response)  
- separate out to use materials source set (E, etc. not added) as in case3  
- use library geometric features like flap\_link does

\*/

SOURCE\_SET partial\_apm ROOT\_COB back\_plate;

COB part;  
part\_number : STRING;  
designer : STRING;  
origin : coordinate;  
END\_COB;

COB back\_plate SUBTYPE\_OF part;

/\* design features \*/

length3 : REAL;  
width1 : REAL;  
thickness : REAL;  
hole1 : hole;  
hole2 : hole;  
material : material;  
area : REAL;

/\* geometric entities \*/

circle1 : circle;  
circle2 : circle;  
line1 : line;  
line3 : line;

IDEALIZED effective\_span : REAL;  
IDEALIZED span\_reduction\_factor : REAL;

RELATIONS

PRODUCT\_RELATIONS

pr9 : "<hole2.origin.y> == <hole1.origin.y> + <length3>";  
pr11 : "<area> == <width1> \* <thickness>";

/\* relations with geometric primitives (in cad model) \*/

prp2 : "<width1> == <line1.length>";  
prp3 : "<thickness> == <line3.length>";  
/\* hole1.cross\_section == circle1 - only needed aspects are related here\*/  
prp4 : "<hole1.cross\_section.radius> == <circle1.radius>";  
prp6 : "<hole1.origin.y> == <circle1.origin.y>";  
prp8 : "<hole2.cross\_section.radius> == <circle2.radius>";  
prp11 : "<hole2.origin.y> == <circle2.origin.y>";

PRODUCT\_IDEALIZATION\_RELATIONS

pir2 : "<effective\_span> == <length3> - (<hole1.cross\_section.radius> + <hole2.cross\_section.radius>) \* <span\_reduction\_factor>";  
END\_COB;

/\* --- part features --- \*/

COB hole;  
origin : coordinate;  
cross\_section : circle;

```

    height : REAL;
    volume : REAL;
RELATIONS
PRODUCT_RELATIONS
    r1 : "<volume> == <height> * <cross_section.area>";
    r2 : "<cross_section.origin.x> == <origin.x>";
    r3 : "<cross_section.origin.y> == <origin.y>";
    r4 : "<cross_section.origin.z> == <origin.z>";
END_COB;

COB material;
/* actually a material model */
name : STRING;
youngs_modulus : REAL;
poissons_ratio : REAL;
cte : REAL;
END_COB;

/* --- geometric primitives --- */

COB coordinate;
x : REAL;
y : REAL;
z : REAL;
END_COB;

COB line;
start : coordinate;
end : coordinate;
length : REAL;
RELATIONS
PRODUCT_RELATIONS
    r1 : "<length> == ((<end.x> - <start.x>)**2 + (<end.y> - <start.y>)**2 + (<end.z> - <start.z>)**2)**0.5";
END_COB;

COB circle;
origin : coordinate;
radius : REAL;
diameter : REAL;
area : REAL;
RELATIONS
PRODUCT_RELATIONS
    r1 : "<diameter> == 2 * <radius>";
    r2 : "<area> == PI * <radius>**2";
END_COB;

END_SOURCE_SET;

END_SCHEMA;

```

## 2. Dimension entity tagging approach

### a) Backplate APM

SCHEMA back\_plate;

```
/*  
Copyright (C) 1998  
Georgia Tech  
Engineering Information Systems Lab  
eislabs.gatech.edu
```

Case: 3

Syntax: cob v2.1

Authors: R. Peak, A. Chandrasekhar, D. Tamburini

Purpose: Demonstrate simple geometric aspects of an apm including attributes that come from a cad model. See figure for definition of parameters.

Case Characteristics:

- Dimension-based associativity with a cad model (via tagged dimensions).
- fully specified (tagged entities fully describe the model)
- Coordinates are expressed wrt global coordinate system & all values are represented in CATIA model units. All angles are expressed in degrees.
- Uses different CATIA model vs. case1 (but same shape, size, etc.) since now tags are attached to dimension entities vs. to geometric entities.

Versions:

981130  
- initial version (works w/ catia 4.1.9 tk/tcl interface)

981231  
- initial release

Other Possibilities:

- catia model needs to answer designer, and material name, etc. (manually added in response)
- separate out to use materials source set (E, etc. not added)
- use library geometric features like flap\_link does

\*/

SOURCE\_SET back\_plate ROOT\_COB back\_plate;

COB back\_plate SUBTYPE\_OF part;  
part\_number : STRING;

/\* design features \*/

length1 : REAL;  
length2 : REAL;  
length3 : REAL;  
length4 : REAL;  
width1 : REAL;  
width2 : REAL;  
thickness : REAL;  
hole1 : hole;  
hole2 : hole;  
material : STRING;  
area : REAL;

/\* Idealized relations \*/

```

IDEALIZED effective_span : REAL;
IDEALIZED span_reduction_factor : REAL;
IDEALIZED critical_area : REAL;

RELATIONS
PRODUCT_RELATIONS
/* design parameter relations */
pr9 : "<width2> == (<width1> / 2)";
pr10 : "<length1> == (<length2> + <length3> + <length4>)";
pr11 : "<area> == <width1> * <thickness>";

/* design feature relations */
pr12 : "<thickness> == <hole1.height>";
pr13 : "<thickness> == <hole2.height>";
pr14 : "<hole1.origin.x> == <origin.x>"; /* assume same as back plate's origin */
pr15 : "<hole1.origin.y> == <origin.y> + <length2>";
pr16 : "<hole1.origin.z> == <origin.z> + <width2>";
pr17 : "<hole2.origin.x> == <origin.x>"; /* assume same as back plate's origin */
pr18 : "<hole2.origin.y> == <origin.y> + <length2> + <length3>";
pr19 : "<hole2.origin.z> == <origin.z> + <width2>";

PRODUCT IDEALIZATION RELATIONS  pir1 : "<critical_area> == ( <thickness> * ( <width1> - ( 2 *
<hole1.cross_section.radius> ) ) )";
pir2 : "<effective_span> == <length3> - (<hole1.cross_section.radius> + <hole2.cross_section.radius>) *
<span_reduction_factor>";

END_COB;

USE_FROM lib/apm.cos;

END_SOURCE_SET;

USE_FROM lib/materials.cos AS_SOURCE_SETS;

LINK_DEFINITIONS
back_plate.back_plate.material == materials.material.name;
END_LINK_DEFINITIONS;

END_SCHEMA;

```

## b) Flap link APM

SCHEMA flap\_link\_apm;

/\*  
Copyright 1998 by Georgia Tech  
Engineering Information Systems Lab  
eislabs.gatech.edu

Syntax: cob v2.1

Authors: R. Peak, D. Tamburini, M. Wilson

Versions:  
pre 9809-9812  
- Initial versions

981231  
- Initial release

Other Possibilities:  
- Add aliases for inner/outer\_diameter of sleeves  
- Completely define location of feature origins, angle relations, etc  
- Do multi-level effective\_length - how as primitive attribute?

```
- Add checks (e.g., flange width <= sleeve width)
*/
```

```
SOURCE_SET flap_link_geometric_model ROOT_COB flap_link;
```

```
USE_FROM lib/apm.cos;
```

```
COB flap_link SUBTYPE_OF part;
```

```
part_number : STRING;
```

```
inter_axis_length, L<sub>a</sub> : REAL;
```

```
/* IDEALIZED */ effective_length, L<sub>eff</sub> : REAL;
```

```
sleeve1 : sleeve;
```

```
sleeve2 : sleeve;
```

```
shaft : tapered_beam;
```

```
rib1 : rib;
```

```
rib2 : rib;
```

```
allowable_twist : REAL;
```

```
allowable_twist_factor : REAL;
```

```
allowable_inter_axis_length_change_factor, C<sub>phi</sub> : REAL;
```

```
allowable_inter_axis_length_change, C<sub>Delta;La</sub> : REAL;
```

```
RELATIONS
```

```
PRODUCT_RELATIONS
```

```
pr1 : "<sleeve1.origin.y> == <origin.y>";
```

```
pr2 : "<sleeve2.origin.y> == <sleeve1.origin.y> + <inter_axis_length>";
```

```
pr3 : "<rib1.height> == (<sleeve1.width> - <shaft.critical_cross_section.design.web_thickness>)/2";
```

```
pr4 : "<rib2.height> == (<sleeve2.width> - <shaft.critical_cross_section.design.web_thickness>)/2";
```

```
pr5 : "<rib1.thickness> == <shaft.critical_cross_section.design.web_thickness>";
```

```
pr6 : "<rib2.thickness> == <shaft.critical_cross_section.design.web_thickness>";
```

```
PRODUCT IDEALIZATION RELATIONS
```

```
pir1 : "<effective_length> == <inter_axis_length> - (<sleeve1.hole.cross_section.radius> +  
<sleeve2.hole.cross_section.radius>);"
```

```
pir2 : "<shaft.critical_cross_section.design.total_height> == <sleeve1.outer_diameter>";
```

```
pir3 : "<allowable_twist> == <allowable_twist_factor> * <effective_length>";
```

```
pir4 : "<allowable_inter_axis_length_change> == <allowable_inter_axis_length_change_factor> * <effective_length>";
```

```
/* example allowables definitions - should be related to DR&O's */
```

```
END_COB;
```

```
END_SOURCE_SET;
```

```
USE_FROM lib/materials.cos AS_SOURCE_SETS;
```

```
LINK_DEFINITIONS
```

```
flap_link_geometric_model.flap_link.material == materials.material.name;
```

```
END_LINK_DEFINITIONS;
```

```
END_SCHEMA;
```

## Flap link material model (lib/materials.coi)

```
DATA;
```

```
/*
```

```
----- COB INSTANCE LIBRARY -----
```

```
*/
```

```
/*
```

```
Copyright 1998 by Georgia Tech  
Engineering Information Systems Lab  
eislabs.gatech.edu
```

```
Authors: D. Tamburini, R. Peak, M. Wilson
```

Versions:

980919

- Initial version based on XaiTools Smalltalk objects

Notes:

- All given in std English units.

Other Possibilities:

- Add yield\_stresses to all below

- Auto-generate this from materials databases

- Utilize standard material ids

\*/

```
INSTANCE_OF material;
  name : "aluminum";
  yield_stress : ?;
  stress_strain_model.linear_elastic.youngs_modulus : 1.0e7;
  stress_strain_model.linear_elastic.poissons_ratio : 0.32;
  stress_strain_model.linear_elastic.cte : 2.0e-5;
  stress_strain_model.linear_elastic.shear_modulus : ?;
END_INSTANCE;
```

```
INSTANCE_OF material;
  name : "steel";
  yield_stress : 1.8e4;
  /* half of 1020 HR steel yield */
  stress_strain_model.linear_elastic.youngs_modulus : 3.0e7;
  stress_strain_model.linear_elastic.poissons_ratio : 0.3;
  stress_strain_model.linear_elastic.cte : 1.0e-5;
  stress_strain_model.linear_elastic.shear_modulus : ?;
END_INSTANCE;
```

```
INSTANCE_OF material;
  name : "solder";
  yield_stress : ?;
  stress_strain_model.linear_elastic.youngs_modulus : 1.5e6;
  stress_strain_model.linear_elastic.poissons_ratio : 0.4;
  stress_strain_model.linear_elastic.cte : 2.1e-5;
  stress_strain_model.linear_elastic.shear_modulus : ?;
END_INSTANCE;
```

```
INSTANCE_OF material;
  name : "alumina";
  yield_stress : ?;
  stress_strain_model.linear_elastic.youngs_modulus : 3.7e7;
  stress_strain_model.linear_elastic.poissons_ratio : 0.3;
  stress_strain_model.linear_elastic.cte : 6.7e-6;
  stress_strain_model.linear_elastic.shear_modulus : ?;
END_INSTANCE;
```

```
INSTANCE_OF material;
  name : "copper";
  yield_stress : ?;
  stress_strain_model.linear_elastic.youngs_modulus : 1.7e7;
  stress_strain_model.linear_elastic.poissons_ratio : 0.35;
  stress_strain_model.linear_elastic.cte : 1.66e-5;
  stress_strain_model.linear_elastic.shear_modulus : ?;
END_INSTANCE;
```

```
INSTANCE_OF material;
  name : "FR4";
  yield_stress : ?;
  stress_strain_model.linear_elastic.youngs_modulus : 1.6e6;
  stress_strain_model.linear_elastic.poissons_ratio : 0.28;
  stress_strain_model.linear_elastic.cte : 1.5e-5;
  stress_strain_model.linear_elastic.shear_modulus : ?;
```



```

END_INSTANCE;

INSTANCE_OF material;
  name : "eutectic_solder";
  yield_stress : ?;
  stress_strain_model.linear_elastic.youngs_modulus : 1.5e6;
  stress_strain_model.linear_elastic.poissons_ratio : 0.4;
  stress_strain_model.linear_elastic.cte : 2.1e-5;
  stress_strain_model.linear_elastic.shear_modulus : ?;
END_INSTANCE;

INSTANCE_OF material;
  name : "Sn60_Pb40_solder";
  yield_stress : ?;
  stress_strain_model.linear_elastic.youngs_modulus : 1.5e6;
  stress_strain_model.linear_elastic.poissons_ratio : 0.4;
  stress_strain_model.linear_elastic.cte : 2.1e-5;
  stress_strain_model.linear_elastic.shear_modulus : ?;
END_INSTANCE;
END_DATA;

```

## c) Bike frame APM

SCHEMA bike\_frame\_apm;

```

/*
(c) 1998, 1999
Georgia Institute of Technology
CALS Technology Center
Engineering Information Systems Lab
eislabs.gatech.edu

```

Authors: D. Tamburini, R. Peak, S. Cimentalay, M. Wilson

Syntax: cob v2.1

Description:  
APM for a representative aerospace structural part (a flap support assembly inboard beam) nicknamed "bike frame"

Versions:  
990106  
- first release  
- expanded from Diego's simple\_inboard\_beam 9/98

Other Possibilities:

- rear spar features and fitting idealizations need to be checked to see if matches DESIGN\_MANUAL & CATIA documentation
- add features used in other analyses in the example DESIGN\_MANUAL
- use aggregates vs. numbered attributes? e.g. rib[1] (or feature['rib1']) vs. rib1
- define leg aggregate of associated design/analysis features?
- divide out design/geometric features from apm lib & do use\_from?
  - rename hole cob definition to hole\_feature & promote radius via alias (ie. hole.radius)
  - add design/geometric feature

Ex. make rib = geometric\_feature + semantic categorization for design?

```

*/

```

SOURCE\_SET bike\_frame\_geometric\_model ROOT\_COB bike\_frame;

```

USE_FROM lib\geometry.cos;
USE_FROM lib\apm.cos;
USE_FROM lib\abbs.cos;

```

USE\_FROM el\_aero\lib\abbs.cos;

COB bike\_frame SUBTYPE\_OF part;

part\_number : STRING;  
leg1 : leg;  
cavity3 : cavity\_with\_bottom\_hole;  
rib8 : cavity\_rib;  
rib9 : cavity\_rib;  
bolt4 : STRING;  
cavity9 : cavity\_with\_bottom\_hole;  
rib12 : cavity\_rib;  
rib13 : cavity\_rib;  
bolt7 : STRING;

/\* idealized features \*/

bulkhead\_fitting\_casing : channel\_fitting\_casing\_body;  
bulkhead\_fitting\_bolt : fitting\_bolt\_body;  
rear\_spar\_fitting\_1\_casing : channel\_fitting\_casing\_body;  
rear\_spar\_fitting\_1\_bolt : fitting\_bolt\_body;

RELATIONS

PRODUCT IDEALIZATION RELATIONS

/\* bulkhead fitting idealizations as adapted from std. library

(see el\_aero/lib/cbams.cos ) \*/

pir\_b\_1 : "<bulkhead\_fitting\_casing.base\_wall.width> == <rib8.thickness>/2.0 + <cavity3.inner\_width> + <rib9.thickness>/2.0";  
pir\_b\_2 : "<bulkhead\_fitting\_casing.end\_pad.height> == <cavity3.bottom\_thickness>/2.0 + <cavity3.inner\_breadth>";  
pir\_b\_3 : "<bulkhead\_fitting\_casing.end\_pad.thickness> == <cavity3.minimum\_base\_thickness>";  
pir\_b\_4 : "<bulkhead\_fitting\_casing.end\_pad.bolt\_hole.cross\_section.diameter> == <cavity3.hole\_diameter>";  
pir\_b\_5 : "<bulkhead\_fitting\_casing.end\_pad.effective\_hole\_offset> == <cavity3.hole\_height> + <cavity3.bottom\_thickness> / 2.0";  
pir\_b\_6 : "<bulkhead\_fitting\_casing.base\_wall.thickness> == <cavity3.bottom\_thickness>";  
pir\_b\_7 : "<bulkhead\_fitting\_casing.base\_wall.bolt\_hole.cross\_section.diameter> == 0";  
/\* should generalize so not always zero \*/  
pir\_b\_8 : "<bulkhead\_fitting\_casing.side\_wall.thickness> == ( <rib8.thickness> + <rib9.thickness> ) / 2.0";  
pir\_b\_9 : "<bulkhead\_fitting\_casing.side\_wall.effective\_height> == 0";  
/\* parameter 'a' - need to add value in coi file \*/  
pir\_b\_10 : "<bulkhead\_fitting\_bolt.head.radius> == <bolt4.head.flat\_to\_flat\_width> / 2.0";

/\* rear spar fitting #1 idealizations - currently same as bulkhead \*/

pir\_rs1\_1 : "<rear\_spar\_fitting\_1\_casing.base\_wall.width> == <rib12.thickness>/2.0 + <cavity9.inner\_width> + <rib13.thickness>/2.0";  
pir\_rs1\_2 : "<rear\_spar\_fitting\_1\_casing.end\_pad.height> == <cavity9.bottom\_thickness>/2.0 + <cavity9.inner\_breadth>";  
pir\_rs1\_3 : "<rear\_spar\_fitting\_1\_casing.end\_pad.thickness> == <cavity9.minimum\_base\_thickness>";  
pir\_rs1\_4 : "<rear\_spar\_fitting\_1\_casing.end\_pad.bolt\_hole.cross\_section.diameter> == <cavity9.hole\_diameter>";  
pir\_rs1\_5 : "<rear\_spar\_fitting\_1\_casing.end\_pad.effective\_hole\_offset> == <cavity9.hole\_height> + <cavity9.bottom\_thickness> / 2.0";  
pir\_rs1\_6 : "<rear\_spar\_fitting\_1\_casing.base\_wall.thickness> == <cavity9.bottom\_thickness>";  
pir\_rs1\_7 : "<rear\_spar\_fitting\_1\_casing.base\_wall.bolt\_hole.cross\_section.diameter> == 0";  
/\* should generalize so not always zero \*/  
pir\_rs1\_8 : "<rear\_spar\_fitting\_1\_casing.side\_wall.thickness> == ( <rib12.thickness> + <rib13.thickness> ) / 2.0";  
pir\_rs1\_9 : "<rear\_spar\_fitting\_1\_casing.side\_wall.effective\_height> == 0";  
/\* parameter 'a' - need to add value in coi file \*/  
pir\_rs1\_10 : "<rear\_spar\_fitting\_1\_bolt.head.radius> == <bolt7.head.flat\_to\_flat\_width> / 2.0";

END\_COB;

COB leg;

END\_COB;

COB cavity\_with\_bottom\_hole;

inner\_width : REAL;  
inner\_breadth : REAL;  
inner\_height : REAL;

```

    minimum_base_thickness : REAL;
    top_thickness : REAL;
    bottom_thickness : REAL;
    hole_diameter : REAL;
    hole_height : REAL;
END_COB;

COB cavity_rib;
    thickness : REAL;
END_COB;

END_SOURCE_SET;

USE_FROM el_aero\lib\materials.cos AS_SOURCE_SETS;

USE_FROM el_aero\lib\fasteners.cos AS_SOURCE_SETS;

LINK_DEFINITIONS
    bike_frame_geometric_model.bike_frame.bolt4 == el_aero_fasteners.fastener.part_number;
    bike_frame_geometric_model.bike_frame.bolt7 == el_aero_fasteners.fastener.part_number;
    bike_frame_geometric_model.bike_frame.material == el_aero_materials.el_aero_material.id;
END_LINK_DEFINITIONS;

END_SCHEMA;

```

## Bike frame material model

```

DATA;

/*
material properties

981231

- assume can extract such info from MATDB, etc.
*/

INSTANCE_OF el_aero_material;
/* from bike_frame DESIGN_MANUAL p.6 */
id : "7050-T7452" ;
ultimate_tensile_stress_long : 67000 ;
ultimate_tensile_stress_long_tran : 65000;
ultimate_tensile_stress_short_tran : 64000 ;
yield_tensile_stress_long : 57000 ;
yield_tensile_stress_long_tran : 52000 ;
yield_tensile_stress_short_tran : 50000 ;
yield_compression_stress_long : 54000 ;
yield_compression_stress_long_tran : 55000 ;
yield_compression_stress_short_tran : 54000;
ultimate_shear_stress : 39000 ;
yield_shear_stress : 31000 ;
Fbru_eD_onefive : 82000 ;
Fbru_eD_oneseven : ? ;
Fbru_eD_two : 109000 ;
elongation_long : 9.0 ;
elongation_long_tran : 4.0 ;
elongation_short_tran : 3.0 ;
youngs_modulus : 10200000 ;
youngs_modulus_comp : 10600000 ;
shear_modulus : 3900000 ;
poissons_ratio : 0.33 ;
specific_weight : 0.102;
shape_factor_ntu_long : 22.0 ;
shape_factor_ntu_long_tran : 11.0 ;

```

```

shape_factor_ntu_sort_tran : 9.8 ;
shape_factor_nti_long : 14.0 ;
shape_factor_nti_long_tran : 14.0 ;
shape_factor_nti_sort_tran : 9.3 ;
shape_factor_nci_long : 15.0 ;
shape_factor_nci_long_tran : 18.0 ;
shape_factor_nci_sort_tran : 20.0 ;
shape_factor_nsu : 16.0 ;
shape_factor_nsi : 15.0 ;
ultimate_strain_long : 0.067 ;
ultimate_strain_long_tran : 0.03 ;
ultimate_strain_short_tran : 0.022 ;
allowable_apparent_bending_stress : 91844.0;
/* allowable from 4094 graph - generalize as relation? */
END_INSTANCE;

```

```

INSTANCE_OF el_aero_material;
/* dummy instance */
id : "7040xx-T7452-MS7-214" ;
ultimate_tensile_stress_long : 70000 ;
ultimate_tensile_stress_long_tran : 65000;
ultimate_tensile_stress_short_tran : 64000 ;
yield_tensile_stress_long : 57000 ;
yield_tensile_stress_long_tran : 52000 ;
yield_tensile_stress_short_tran : 50000 ;
yield_compression_stress_long : 54000 ;
yield_compression_stress_long_tran : 55000 ;
yield_compression_stress_short_tran : 54000;
ultimate_shear_stress : 39000 ;
yield_shear_stress : 31000 ;
Fbru_eD_onefive : 82000 ;
Fbru_eD_oneseven : ? ;
Fbru_eD_two : 109000 ;
elongation_long : 9.0 ;
elongation_long_tran : 4.0 ;
elongation_short_tran : 3.0 ;
youngs_modulus : 15000000 ;
youngs_modulus_comp : 12000000 ;
shear_modulus : 3900000 ;
poissons_ratio : 0.33 ;
specific_weight : 0.102;
shape_factor_ntu_long : 22.0 ;
shape_factor_ntu_long_tran : 11.0 ;
shape_factor_ntu_sort_tran : 9.8 ;
shape_factor_nti_long : 14.0 ;
shape_factor_nti_long_tran : 14.0 ;
shape_factor_nti_sort_tran : 9.3 ;
shape_factor_nci_long : 15.0 ;
shape_factor_nci_long_tran : 18.0 ;
shape_factor_nci_sort_tran : 20.0 ;
shape_factor_nsu : 16.0 ;
shape_factor_nsi : 15.0 ;
ultimate_strain_long : 0.067 ;
ultimate_strain_long_tran : 0.03 ;
ultimate_strain_short_tran : 0.022 ;
allowable_apparent_bending_stress : 8000.0;
END_INSTANCE;

```

```

END_DATA;

```

### 3. Parametric entity tagging approach

The APM schemas for the back plate, flap link and bike frame are the same as those listed for the test cases in the dimension entity tagging approach.

## Appendix B

### Request files for the test cases

#### Instance models (.coi)

##### **a) Back Plate**

DATA;

INSTANCE\_OF backplate;

```
length1 : ? ;
length2 : ? ;
length3 : ? ;
length4 : ? ;
width1 : ? ;
width2 : ? ;
thickness : ? ;
area : ? ;
critical_area : ? ;
effective_span : ? ;
span_reduction_factor : 0.4 ;
part_number : "XYZ-901" ;
designer : "J.Smith" ;
hole1.height : ? ;
hole1.volume : ? ;
hole1.origin.x : ? ;
hole1.origin.y : ? ;
hole1.origin.z : ? ;
hole1.cross_section.radius : ? ;
hole1.cross_section.diameter : ? ;
hole1.cross_section.area : ? ;
hole1.cross_section.origin.x : ? ;
hole1.cross_section.origin.y : ? ;
hole1.cross_section.origin.z : ? ;
hole2.height : ? ;
hole2.volume : ? ;
hole2.origin.x : ? ;
hole2.origin.y : ? ;
hole2.origin.z : ? ;
hole2.cross_section.radius : ? ;
hole2.cross_section.diameter : ? ;
hole2.cross_section.area : ? ;
hole2.cross_section.origin.x : ? ;
hole2.cross_section.origin.y : ? ;
hole2.cross_section.origin.z : ? ;
material.name : "aluminium" ;
material.youngs_modulus : ? ;
material.poissons_ratio : ? ;
material.cte : ? ;
circle1.radius : ? ;
circle1.diameter : ? ;
circle1.area : ? ;
circle1.origin.x : ? ;
```

```

circle1.origin.y : ? ;
circle1.origin.z : ? ;
circle2.radius : ? ;
circle2.diameter : ? ;
circle2.area : ? ;
circle2.origin.x : ? ;
circle2.origin.y : ? ;
circle2.origin.z : ? ;
line1.length : ? ;
line1.start.x : ? ;
line1.start.y : ? ;
line1.start.z : ? ;
line1.end.x : ? ;
line1.end.y : ? ;
line1.end.z : ? ;
line2.length : ? ;
line2.start.x : ? ;
line2.start.y : ? ;
line2.start.z : ? ;
line2.end.x : ? ;
line2.end.y : ? ;
line2.end.z : ? ;
line3.length : ? ;
line3.start.x : ? ;
line3.start.y : ? ;
line3.start.z : ? ;
line3.end.x : ? ;
line3.end.y : ? ;
line3.end.z : ? ;
origin.x : ? ;
origin.y : ? ;
origin.z : ? ;

```

END\_INSTANCE;

END\_DATA;

## b) Flap link

DATA;

```

/*
flap link catia request/response file
(values are given in the desired response file version)

```

```

case 3 - dimension-based tagging
part number "XYZ-510"

```

```

981204 7:10pm
R. Peak

```

This file is a semi-automatically created request/response file.

The request version shows what the cad model should answer

- group 1 has a subset of all those needed (all but cross section)
- group 2 has all the rest needed to fully define the geometric aspects of the apm

```

121098
- added allowable_twist_factor
- added allowable_inter_axis_length_change_factor

```

```

*/

INSTANCE_OF flap_link;

/* ---- variables requested from cad model ---- */

/* - group 1 - */
part_number : "XYZ-510" ;
description : "flap link type 5" ;
designer : "J. Smith" ;
material : "steel" ;
allowable_twist_factor : 0.001;
allowable_inter_axis_length_change_factor : 0.001;
origin.x : ? ;
origin.y : ? ;
origin.z : ? ;
inter_axis_length : ? ;
sleeve1.width : ? ;
sleeve1.outer_diameter : ? ;
sleeve1.inner_diameter : ? ;
sleeve2.width : ? ;
sleeve2.outer_diameter : ? ;
sleeve2.inner_diameter : ? ;

/* - group 2 - */
shaft.critical_cross_section.design.flange_width : ? ;
shaft.critical_cross_section.design.flange_base_thickness : ? ;
shaft.critical_cross_section.design.flange_taper_angle : ? ;
shaft.critical_cross_section.design.web_thickness : ? ;
shaft.critical_cross_section.design.flange_fillet_radius : ? ;
shaft.critical_cross_section.design.flange_taper_thickness : ? ;

/* ---- some variables to be solved for after cad response received ---- */
effective_length : ? ;
sleeve1.wall_thickness : ? ;
sleeve2.wall_thickness : ? ;
shaft.critical_cross_section.design.area : ? ;
shaft.taper_angle : ? ;

END_INSTANCE;

END_DATA;

```

## c) Bike frame

```

DATA;

/*
19981231a
- currently not all dimensions are tagged in the catia model (untagged ones added manually here)
- note minor discrepancies between catia model vs. DESIGN_MANUAL

199907 WIP
- completing tagging in catia model
*/

INSTANCE_OF bike_frame;

part_number : "123L4567";
material : "7050-T7452-MS7-214";

```



```

/* items for bulkhead attach point */

cavity3.inner_width : ?;
cavity3.inner_breadth : ?;
cavity3.inner_height : ?;
cavity3.minimum_base_thickness : ?;
cavity3.top_thickness : ?;
cavity3.bottom_thickness : ?;
cavity3.hole_diameter : ?;
cavity3.base_angle : ?;
cavity3.hole_bottom_edge_height : ?;

/* remove from model due to new cross section view */

rib8.thickness : ?;
rib9.thickness : ?;

/* from DESIGN_MANUAL pg. 206 */
/*
    rib8.thickness : 0.31;
    rib9.thickness : 0.31;
*/

bolt4 : "BF-400-010";

/* items for diagonal brace attach point */
/* Dennis: add others & check model */

diagonal_brace_attach_point.side_to_side_distance : ?;
diagonal_brace_attach_point.slope_radius : ?;
diagonal_brace_attach_point.corner_radius : ?;
diagonal_brace_attach_point.outer_radius : ?;
diagonal_brace_attach_point.slope_angle : ?;
diagonal_brace_attach_point.inter_lug_distance : ?;

diagonal_brace_attach_point.lug1.hole_radius : ?;
diagonal_brace_attach_point.lug1.thickness : ?;
diagonal_brace_attach_point.lug2.hole_radius : ?;
diagonal_brace_attach_point.lug2.thickness : ?;

/* items for rear spar attach point #1 */
/* note: dummy numbers used - not tagged yet in catia model */
cavity9.inner_width : 4;
cavity9.inner_breadth : 3;
cavity9.inner_height : 2;
cavity9.minimum_base_thickness : 0.40;
cavity9.top_thickness : 0.4;
cavity9.bottom_thickness : 0.3;
cavity9.hole_diameter : 0.5;
cavity9.hole_height : 1;

/* from catia model */
rib12.thickness : 0.3;
rib13.thickness : 0.3;

bolt7 : "BF-400-030";

/* idealized features */
bulkhead_fitting_casing.uid : "FC_007_bulkhead";
bulkhead_fitting_casing.channel_fitting_factor : ?;
bulkhead_fitting_casing.end_pad.height : ?;
bulkhead_fitting_casing.end_pad.thickness : ?;
bulkhead_fitting_casing.end_pad.bolt_hole.cross_section.diameter : ?;

```

```

bulkhead_fitting_casing.end_pad.effective_hole_offset : ?;

bulkhead_fitting_casing.base_wall.width : ?;
bulkhead_fitting_casing.base_wall.thickness : ?;
bulkhead_fitting_casing.base_wall.bolt_hole.cross_section.diameter : ?;

bulkhead_fitting_casing.side_wall.effective_height : ?;
bulkhead_fitting_casing.side_wall.thickness : ?;

bulkhead_fitting_bolt.uid : "FB_007_bulkhead";
bulkhead_fitting_bolt.head.radius : ?;

rear_spar_fitting_1_casing.uid : "FC_008_rear_spar_1";
rear_spar_fitting_1_casing.channel_fitting_factor : ?;
rear_spar_fitting_1_casing.end_pad.height : ?;
rear_spar_fitting_1_casing.end_pad.thickness : ?;
rear_spar_fitting_1_casing.end_pad.bolt_hole.cross_section.diameter : ?;
rear_spar_fitting_1_casing.end_pad.effective_hole_offset : ?;

rear_spar_fitting_1_casing.base_wall.width : ?;
rear_spar_fitting_1_casing.base_wall.thickness : ?;
rear_spar_fitting_1_casing.base_wall.bolt_hole.cross_section.diameter : ?;

rear_spar_fitting_1_casing.side_wall.effective_height : ?;
rear_spar_fitting_1_casing.side_wall.thickness : ?;

rear_spar_fitting_1_bolt.uid : "FB_008_rear_spar_1";
rear_spar_fitting_1_bolt.head.radius : ?;

END_INSTANCE;

END_DATA;

```

## Appendix C

### Response files for test cases

#### 1) Geometric entity tagging approach

##### a) Back plate

##### **Response file for partial description of the back plate**

DATA;

INSTANCE\_OF backplate;

length1 : ? ;  
length2 : ? ;  
length3 : ? ;  
length4 : ? ;  
width1 : ? ;  
width2 : ? ;  
thickness : ? ;  
area : ? ;  
critical\_area : ? ;  
effective\_span : ? ;  
span\_reduction\_factor : 0.4 ;  
part\_number : "XYZ-901" ;  
designer : "J.Smith" ;  
hole1.height : ? ;  
hole1.volume : ? ;  
hole1.origin.x : ? ;  
hole1.origin.y : ? ;  
hole1.origin.z : ? ;  
hole1.cross\_section.radius : ? ;  
hole1.cross\_section.diameter : ? ;  
hole1.cross\_section.area : ? ;  
hole1.cross\_section.origin.x : ? ;  
hole1.cross\_section.origin.y : ? ;  
hole1.cross\_section.origin.z : ? ;  
hole2.height : ? ;  
hole2.volume : ? ;  
hole2.origin.x : ? ;  
hole2.origin.y : ? ;  
hole2.origin.z : ? ;  
hole2.cross\_section.radius : ? ;  
hole2.cross\_section.diameter : ? ;  
hole2.cross\_section.area : ? ;  
hole2.cross\_section.origin.x : ? ;  
hole2.cross\_section.origin.y : ? ;  
hole2.cross\_section.origin.z : ? ;

```

material.name : "aluminium" ;
material.youngs_modulus : 0.4 ;
material.poissons_ratio : 0.4 ;
material.cte : 0.4 ;
circle1.radius : 4.000000;
circle1.diameter : 8.0;
circle1.area : 50.2654;
circle1.origin.x : 0.000000;
circle1.origin.y : 20.000000;
circle1.origin.z : 15.000000;
circle2.radius : 2.500000;
circle2.diameter : 5.0;
circle2.area : 19.6349;
circle2.origin.x : 0.000000;
circle2.origin.y : 40.000000;
circle2.origin.z : 15.000000;
line1.length : 30.0;
line1.start.x : 0.0;
line1.start.y : 0.0;
line1.start.z : 0.0;
line1.end.x : 0.0;
line1.end.y : 0.0;
line1.end.z : 30.0;
line2.length : 60.0;
line2.start.x : 0.0;
line2.start.y : 60.0;
line2.start.z : 0.0;
line2.end.x : 0.0;
line2.end.y : 0.0;
line2.end.z : 0.0;
line3.length : 10.0;
line3.start.x : 0.0;
line3.start.y : 0.0;
line3.start.z : 0.0;
line3.end.x : 10.0;
line3.end.y : 0.0;
line3.end.z : 0.0;
origin.x : 0.000000;
origin.y : 0.000000;
origin.z : 0.000000;

END_INSTANCE;

END_DATA;

```

## Response file for the complete description of the back plate

```

DATA;

INSTANCE_OF back_plate;

length3 : ? ;
width1 : ? ;
thickness : ? ;
area : ? ;
effective_span : ? ;
span_reduction_factor : 0.4 ;
part_number : "XYZ-901" ;
designer : "J. Smith";
hole1.height : ? ;
hole1.volume : ? ;
hole1.origin.x : ? ;

```

```

hole1.origin.y : ? ;
hole1.origin.z : ? ;
hole1.cross_section.radius : ? ;
hole1.cross_section.diameter : ? ;
hole1.cross_section.area : ? ;
hole1.cross_section.origin.x : ? ;
hole1.cross_section.origin.y : ? ;
hole1.cross_section.origin.z : ? ;
hole2.height : ? ;
hole2.volume : ? ;
hole2.origin.x : ? ;
hole2.origin.y : ? ;
hole2.origin.z : ? ;
hole2.cross_section.radius : ? ;
hole2.cross_section.diameter : ? ;
hole2.cross_section.area : ? ;
hole2.cross_section.origin.x : ? ;
hole2.cross_section.origin.y : ? ;
hole2.cross_section.origin.z : ? ;
material.name : "aluminum" ;
circle1.radius : 4.000000;
circle1.diameter : 8.0;
circle1.area : 50.2654;
circle1.origin.x : 0.000000;
circle1.origin.y : 20.000000;
circle1.origin.z : 15.000000;
circle2.radius : 2.500000;
circle2.diameter : 5.0;
circle2.area : 19.6349;
circle2.origin.x : 0.000000;
circle2.origin.y : 40.000000;
circle2.origin.z : 15.000000;
line1.length : 30.0;
line1.start.x : 0.0;
line1.start.y : 0.0;
line1.start.z : 0.0;
line1.end.x : 0.0;
line1.end.y : 0.0;
line1.end.z : 30.0;
line3.length : 10.0;
line3.start.x : 0.0;
line3.start.y : 0.0;
line3.start.z : 0.0;
line3.end.x : 10.0;
line3.end.y : 0.0;
line3.end.z : 0.0;
origin.x : 0.000000;
origin.y : 0.000000;
origin.z : 0.000000;

END_INSTANCE;

END_DATA;

```

## 2. Dimension entity tagging approach

### a) Back plate

```
DATA;

INSTANCE_OF back_plate;

length1 : 60.000000;
length2 : 20.000000;
length3 : 20.000000;
length4 : ? ;
width1 : 30.000000;
width2 : 15.000000;
thickness : 10.000000;
material : "steel" ; /* change from ? to "steel" */
area : ? ;
effective_span : ? ;
span_reduction_factor : 0.4 ;
critical_area : ? ;
part_number : "XYZ-901" ;
designer : ? ;
hole1.height : ? ;
hole1.volume : ? ;
hole1.origin.x : ? ;
hole1.origin.y : ? ;
hole1.origin.z : ? ;
hole1.cross_section.radius : 4.000000;
hole1.cross_section.diameter : ? ;
hole1.cross_section.area : ? ;
hole1.cross_section.origin.x : ? ;
hole1.cross_section.origin.y : ? ;
hole1.cross_section.origin.z : ? ;
hole2.height : ? ;
hole2.volume : ? ;
hole2.origin.x : ? ;
hole2.origin.y : ? ;
hole2.origin.z : ? ;
hole2.cross_section.radius : 2.500000;
hole2.cross_section.diameter : ? ;
hole2.cross_section.area : ? ;
hole2.cross_section.origin.x : ? ;
hole2.cross_section.origin.y : ? ;
hole2.cross_section.origin.z : ? ;
origin.x : 0 ;          /* change from ? to 0 */
origin.y : 0 ;          /* change from ? to 0 */
origin.z : 0 ;          /* change from ? to 0 */

END_INSTANCE;

END_DATA;
```

### b) Flap link

```
DATA;

/*
flap link catia request/response file
(values are given in the desired response file version)

case 3 - dimension-based tagging
```

part number "XYZ-510"

981204 - 7:10pm R. Peak

981207 - Dennis, Ashok, Russell

- working for catia model (flaplink510g.model)
- gets response for group1 variables from catia model
- group2 variables not tagged yet, so added here manually

121098

- add allowable\_twist\_factor
  - add allowable\_inter\_axis\_length\_change\_factor
- \*/

INSTANCE\_OF flap\_link;

part\_number : "XYZ-510" ;  
description : "flap link type 5" ;  
designer : "J. Smith" ;  
material : "steel" ;  
allowable\_twist\_factor : 0.001;  
allowable\_inter\_axis\_length\_change\_factor : 0.001;

/\* ---- variables requested from cad model ---- \*/

/\* - group 1 - \*/

origin.x : 0.000000;  
origin.y : 0.000000;  
origin.z : 0.000000;  
inter\_axis\_length : 6.250000;  
sleeve1.width : 2.000000;  
sleeve1.outer\_diameter : 2.000000;  
sleeve1.inner\_diameter : 1.000000;  
sleeve2.width : 2.500000;  
sleeve2.outer\_diameter : 2.700000;  
sleeve2.inner\_diameter : 1.500000;  
shaft.taper\_angle : 3.210243;

/\* - group 2 - \*/

shaft.critical\_cross\_section.design.flange\_width : 1.5 ;  
shaft.critical\_cross\_section.design.flange\_base\_thickness : 0.25 ;  
shaft.critical\_cross\_section.design.flange\_taper\_angle : 10.0 ;  
shaft.critical\_cross\_section.design.web\_thickness : 0.25 ;  
shaft.critical\_cross\_section.design.flange\_fillet\_radius : 0.13 ;  
shaft.critical\_cross\_section.design.flange\_taper\_thickness : 0.05 ;

/\* ---- some variables to be solved for after cad response received ---- \*/

effective\_length : ? ;  
sleeve1.wall\_thickness : ? ;  
sleeve2.wall\_thickness : ? ;  
shaft.critical\_cross\_section.design.area : ? ;

END\_INSTANCE;

END\_DATA;

## c) Bike frame

DATA;

/\*

19981231a

- currently not all dimensions are tagged in the catia model (untagged ones added manually here)
- note minor discrepancies between catia model vs. DESIGN\_MANUAL

199907 WIP

- completing tagging in catia model

\*/

INSTANCE\_OF bike\_frame;

part\_number : "123L4567";

material : "7050-T7452-MS7-214";

/\* items for bulkhead attach point \*/

/\* Dennis: add others & check model \*/

cavity3.inner\_width : 2.248610;  
 cavity3.inner\_breadth : 2.011259;  
 cavity3.inner\_height : 1.885029;  
 cavity3.minimum\_base\_thickness : 0.604138;  
 cavity3.top\_thickness : 0.450000;  
 cavity3.bottom\_thickness : 0.400000;  
 cavity3.hole\_diameter : 0.598700;  
 cavity3.base\_angle : 4.062816;  
 cavity3.hole\_bottom\_edge\_height : 1.031965;

/\* remove from model due to new cross section view \*/

rib8.thickness : 0.300000;  
 rib9.thickness : 0.300000;

/\* from DESIGN\_MANUAL pg. 206 \*/

/\*

rib8.thickness : 0.31;  
 rib9.thickness : 0.31;

\*/

bolt4 : "BF-400-010";

/\* items for diagonal brace attach point \*/

/\* Dennis: add others & check model \*/

diagonal\_brace\_attach\_point.side\_to\_side\_distance : 2.089810;  
 diagonal\_brace\_attach\_point.slope\_radius : 0.470000;  
 diagonal\_brace\_attach\_point.corner\_radius : 0.470000;  
 diagonal\_brace\_attach\_point.outer\_radius : 0.750000;  
 diagonal\_brace\_attach\_point.slope\_angle : 34.771807;  
 diagonal\_brace\_attach\_point.inter\_lug\_distance : ?;

diagonal\_brace\_attach\_point.lug1.hole\_radius : 0.375175;  
 diagonal\_brace\_attach\_point.lug1.thickness : 0.350000;  
 diagonal\_brace\_attach\_point.lug2.hole\_radius : 0.281400;  
 diagonal\_brace\_attach\_point.lug2.thickness : 0.350000;

/\* items for rear spar attach point #1 \*/

/\* note: dummy numbers used - not tagged yet in catia model \*/

cavity9.inner\_width : 4;  
 cavity9.inner\_breadth : 3;  
 cavity9.inner\_height : 2;  
 cavity9.minimum\_base\_thickness : 0.40;  
 cavity9.top\_thickness : 0.4;  
 cavity9.bottom\_thickness : 0.3;  
 cavity9.hole\_diameter : 0.5;  
 cavity9.hole\_height : 1;



```

/* from catia model */
rib12.thickness : 0.3;
rib13.thickness : 0.3;

bolt7 : "BF-400-030";

/* idealized features */
bulkhead_fitting_casing.uid : "FC_007_bulkhead";
bulkhead_fitting_casing.channel_fitting_factor : ?;
bulkhead_fitting_casing.end_pad.height : ?;
bulkhead_fitting_casing.end_pad.thickness : ?;
bulkhead_fitting_casing.end_pad.bolt_hole.cross_section.diameter : ?;
bulkhead_fitting_casing.end_pad.effective_hole_offset : ?;

bulkhead_fitting_casing.base_wall.width : ?;
bulkhead_fitting_casing.base_wall.thickness : ?;
bulkhead_fitting_casing.base_wall.bolt_hole.cross_section.diameter : ?;

bulkhead_fitting_casing.side_wall.effective_height : ?;
bulkhead_fitting_casing.side_wall.thickness : ?;

bulkhead_fitting_bolt.uid : "FB_007_bulkhead";
bulkhead_fitting_bolt.head.radius : ?;

rear_spar_fitting_1_casing.uid : "FC_008_rear_spar_1";
rear_spar_fitting_1_casing.channel_fitting_factor : ?;
rear_spar_fitting_1_casing.end_pad.height : ?;
rear_spar_fitting_1_casing.end_pad.thickness : ?;
rear_spar_fitting_1_casing.end_pad.bolt_hole.cross_section.diameter : ?;
rear_spar_fitting_1_casing.end_pad.effective_hole_offset : ?;

rear_spar_fitting_1_casing.base_wall.width : ?;
rear_spar_fitting_1_casing.base_wall.thickness : ?;
rear_spar_fitting_1_casing.base_wall.bolt_hole.cross_section.diameter : ?;

rear_spar_fitting_1_casing.side_wall.effective_height : ?;
rear_spar_fitting_1_casing.side_wall.thickness : ?;

rear_spar_fitting_1_bolt.uid : "FB_008_rear_spar_1";
rear_spar_fitting_1_bolt.head.radius : ?;

END_INSTANCE;

END_DATA;

```

### 3. Parametric entity tagging approach

#### a) Back plate

```
DATA;
INSTANCE_OF back plate;

length1 : 60.000000;
length2 : 20.000000;
length3 : 20.000000;
length4 : 20.000000;
width1 : 40.000000;
width2 : 20.000000;
thickness : 10.000000;
material : ? ;
area : ? ;
effective_span : ? ;
span_reduction_factor : 0.4 ;
critical_area : ? ;
part_number : "XYZ-901" ;
designer : ? ;
hole1.height : ? ;
hole1.volume : ? ;
hole1.origin.x : ? ;
hole1.origin.y : ? ;
hole1.origin.z : ? ;
hole1.cross_section.radius : 6.000000;
hole1.cross_section.diameter : 12.000000;
hole1.cross_section.area : ? ;
hole1.cross_section.origin.x : ? ;
hole1.cross_section.origin.y : ? ;
hole1.cross_section.origin.z : ? ;
hole2.height : ? ;
hole2.volume : ? ;
hole2.origin.x : ? ;
hole2.origin.y : ? ;
hole2.origin.z : ? ;
hole2.cross_section.radius : 4.500000;
hole2.cross_section.diameter : 9.000000;
hole2.cross_section.area : ? ;
hole2.cross_section.origin.x : ? ;
hole2.cross_section.origin.y : ? ;
hole2.cross_section.origin.z : ? ;
origin.x : ? ;
origin.y : ? ;
origin.z : ? ;
name : ? ;
youngs_modulus : ? ;
poissons_ratio : ? ;
cte : ? ;
length : ? ;

END_INSTANCE;

END_DATA;
```

## b) Flap link

DATA;

/\*

flap link catia request file

case 3 (or 4)

- dimension entity (or parametric) tagging
- see "XYZ-510" for further details
- NOTE: uses metric mm units (vs. XYZ-510 and others in English inch units)

part number "XYZ-620v1" (design change variation 1 on original 620 parameters)

19981204 7:15pm - R. Peak

- case 3 original

19990624a - A. Chandrasekhar, D. Ma, R. Peak

- successfully tested via case 4 approach (parametric)
- adjusted allowable\_inter\_axis\_length\_change\_factor to metric units

\*/

INSTANCE\_OF flap\_link;

/\* ---- variables requested from cad model ---- \*/

/\* - group 1 - \*/

part\_number : "XYZ-620" ;  
description : "connecting rod type 2" ;  
designer : "J. Bridges" ;  
material : "aluminum" ;  
allowable\_twist\_factor : 0.001;  
allowable\_inter\_axis\_length\_change\_factor : 0.025;  
origin.x : ? ;  
origin.y : ? ;  
origin.z : ? ;  
inter\_axis\_length : 190.000000;  
sleeve1.width : 70.000000;  
sleeve1.outer\_diameter : 50.000000;  
sleeve1.inner\_diameter : 30.000000;  
sleeve2.width : 80.000000;  
sleeve2.outer\_diameter : 80.000000;  
sleeve2.inner\_diameter : 50.000000;  
shaft.taper\_angle : 4.528063;

/\* - group 2 - \*/

shaft.critical\_cross\_section.design.flange\_width : 33.000000;  
shaft.critical\_cross\_section.design.flange\_base\_thickness : 5.000000;  
shaft.critical\_cross\_section.design.flange\_taper\_angle : 11.309932;  
shaft.critical\_cross\_section.design.web\_thickness : 3.000000;  
shaft.critical\_cross\_section.design.flange\_fillet\_radius : 1.000000;  
shaft.critical\_cross\_section.design.flange\_taper\_thickness : 3.000000;

/\* ---- some variables to be solved for after cad response received ---- \*/

effective\_length : ? ;  
sleeve1.wall\_thickness : ? ;  
sleeve2.wall\_thickness : ? ;  
shaft.critical\_cross\_section.design.area : ? ;

END\_INSTANCE;

END\_DATA;

## Appendix D

### Solved APM files for test cases

#### 1) Dimension entity tagging approach

##### **a) Back plate**

##### **Solved APM file for the back plate**

DATA;

```
INSTANCE_OF back_plate;
length1 : 60.0;
length2 : 20.0;
length3 : 20.0;
length4 : 20.0;
width1 : 30.0;
width2 : 15.0;
thickness : 10.0;
material.name : "steel" ;
material.yield_stress : 18000.0;
material.stress_strain_model.linear_elastic.youngs_modulus : 3.0E7;
material.stress_strain_model.linear_elastic.poissons_ratio : 0.3;
material.stress_strain_model.linear_elastic.cte : 1.0E-5;
material.stress_strain_model.linear_elastic.shear_modulus : 1.153846153846153E7;
area : 300.0;
effective_span : 17.4;
span_reduction_factor : 0.4;
critical_area : 220.0;
part_number : "XYZ-901" ;
designer : ? ;
hole1.height : 10.0;
hole1.volume : 502.6548245743669;
hole1.origin.x : 0.0;
hole1.origin.y : 20.0;
hole1.origin.z : 15.0;
hole1.cross_section.radius : 4.0;
hole1.cross_section.diameter : 8.0;
hole1.cross_section.area : 50.26548245743669;
hole1.cross_section.origin.x : 0.0;
hole1.cross_section.origin.y : 20.0;
hole1.cross_section.origin.z : 15.0;
hole2.height : 10.0;
hole2.volume : 196.349540849362;
hole2.origin.x : 0.0;
hole2.origin.y : 40.0;
hole2.origin.z : 15.0;
hole2.cross_section.radius : 2.5;
hole2.cross_section.diameter : 5.0;
hole2.cross_section.area : 19.6349540849362;
hole2.cross_section.origin.x : 0.0;
```

```

hole2.cross_section.origin.y : 40.0;
hole2.cross_section.origin.z : 15.0;
origin.x : 0.0;
origin.y : 0.0;
origin.z : 0.0;
END_INSTANCE;

```

```

END_DATA;

```

## b) Flap link

### Solved APM file for the flap link

```

DATA;

```

```

INSTANCE_OF flap_link;
part_number : "XYZ-510" ;
description : "flap link type 5" ;
designer : "J. Smith" ;
material.name : "steel" ;
material.yield_stress : 18000.0;
material.stress_strain_model.linear_elastic.youngs_modulus : 3.0E7;
material.stress_strain_model.linear_elastic.poissons_ratio : 0.3;
material.stress_strain_model.linear_elastic.cte : 1.0E-5;
material.stress_strain_model.linear_elastic.shear_modulus : 1.153846153846153E7;
allowable_twist_factor : 0.0010;
allowable_inter_axis_length_change_factor : 0.0010;
origin.x : 0.0;
origin.y : 0.0;
origin.z : 0.0;
inter_axis_length : 6.25;
sleeve1.width : 2.0;
sleeve1.outer_diameter : 2.0;
sleeve1.inner_diameter : 1.0;
sleeve1.wall_thickness : 0.5;
sleeve1.origin.y : 0.0;
sleeve1.hole.cross_section.radius : 0.5;
sleeve1.hole.cross_section.diameter : 1.0;
sleeve1.hole.cross_section.area : 0.7853981633974483;
sleeve1.hole.height : 2.0;
sleeve1.hole.volume : 1.570796326794896;
sleeve2.width : 2.5;
sleeve2.outer_diameter : 2.7;
sleeve2.inner_diameter : 1.5;
sleeve2.wall_thickness : 0.6;
sleeve2.origin.y : 6.25;
sleeve2.hole.cross_section.radius : 0.75;
sleeve2.hole.cross_section.diameter : 1.5;
sleeve2.hole.cross_section.area : 1.767145867644258;
sleeve2.hole.height : 2.5;
sleeve2.hole.volume : 4.417864669110645;
shaft.taper_angle : 3.210243;
shaft.critical_cross_section.design.flange_width : 1.5;
shaft.critical_cross_section.design.flange_base_thickness : 0.25;
shaft.critical_cross_section.design.flange_taper_angle : 10.0;
shaft.critical_cross_section.design.web_thickness : 0.25;
shaft.critical_cross_section.design.flange_fillet_radius : 0.13;
shaft.critical_cross_section.design.flange_taper_thickness : 0.05;
shaft.critical_cross_section.design.area : 1.175;
shaft.critical_cross_section.design.total_height : 2.0;
shaft.critical_cross_section.design.web_height : 1.3999999999999999;
shaft.critical_cross_section.design.flange_thickness : 0.3;
shaft.critical_cross_section.basic.total_height : 2.0;
shaft.critical_cross_section.basic.flange_width : 1.5;

```

```

shaft.critical_cross_section.basic.flange_thickness : 0.25;
shaft.critical_cross_section.basic.web_thickness : 0.25;
shaft.critical_cross_section.basic.area : 1.125;
shaft.critical_cross_section.basic.web_height : 1.5;
shaft.critical_cross_section.tapered.total_height : 2.0;
shaft.critical_cross_section.tapered.flange_width : 1.5;
shaft.critical_cross_section.tapered.flange_base_thickness : 0.25;
shaft.critical_cross_section.tapered.flange_taper_thickness : 0.05;
shaft.critical_cross_section.tapered.web_thickness : 0.25;
shaft.critical_cross_section.tapered.area : 1.175;
shaft.critical_cross_section.tapered.web_height : 1.3999999999999999;
shaft.critical_cross_section.tapered.flange_thickness : 0.3;
effective_length : 5.0;
rib1.height : 0.875;
rib1.thickness : 0.25;
rib1.base : 1.75;
rib2.height : 1.125;
rib2.thickness : 0.25;
rib2.base : 2.25;
allowable_twist : 0.0050;
allowable_inter_axis_length_change : 0.0050;
END_INSTANCE;

END_DATA;

```

## Appendix E

### XaiTools CATIA Adapter - User's Manual

#### CATGEO-based APM Interface to CATIA

##### Aim / Goal

The aim of this tool is to enable extracting attributes of geometric entities from a CAD model for the purpose of using them in possibly many analyses. In order to achieve this aim, the XaiTools CATIA adapter has been developed based on concepts from (Chandrasekhar 1999). It utilizes and manipulates built-in CATIA functions and was written in Tk/Tcl (scripting language).

Note: 'Identifiers', 'labels' & 'tags' are used interchangeably throughout this document.

##### Pre-requisites

Installation of GT\_Image (Hale 1998) and the XaiTools CATIA Adapter with CATIA version 4.1.9 compatible version.

##### Steps involved in extracting attributes of geometric entities

- 1) The first step is to define the APM schema (.cos file). The APM schema contains the definitions of parts and assemblies including geometric entities and their attributes. The schema also contains the relations of idealized attributes. Figure 68 shows APMs implemented as constrained objects in XaiTools, which was adopted in order to extract geometric dimensions and parameters from CATIA design models (numbered '1' in the figure). Please refer Sections 3.1, 3.3 and 6.1.4 for further information on the APM and XaiTools.
- 2) Once the APM has been defined, CATIA needs to be started and then the CAD design model needs to be opened. The designer then uniquely tags the geometric entities that are required for analysis purposes by using the 'IDENTIFY' function/icon in the CATIA graphic user interface (GUI). The tags must conform with those that were used in the definition of the APM schema. The naming

conventions for the tags have been explained later in this manual. This step is indicated in the figure as '0'. The tagged CATIA model has to be saved.

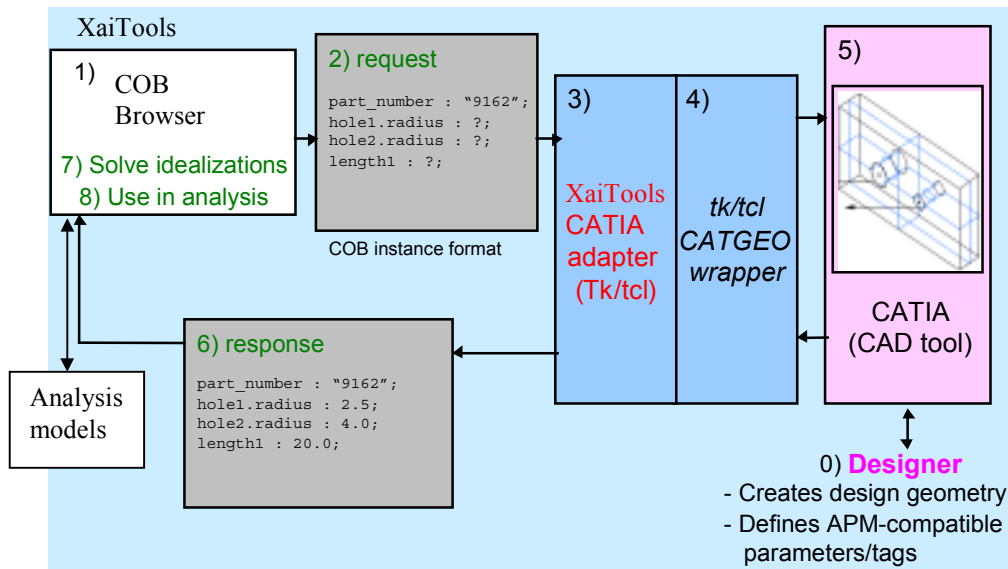
- 3) Once the APM has been defined and the geometric entities have been tagged, the COB browser is used to generate a '.coi' format request file ('2' in the figure). The request file consists of the list of attributes of different geometric entities that are needed for analysis. The naming conventions for different geometric entities have been listed later in this Section. Request files for three test cases are shown in Appendix B.
- 4) Information that may not be included in the CATIA model such as 'designer\_name' and 'span\_reduction\_factor' can be manually typed into the request file if needed or entered later in the COB Browser.
- 5) The next step is to feed the request file into the 'XaiTools CATIA adapter' in order to extract the requested attributes (numbered '3' in the figure). The XaiTools CATIA adapter is started by typing the following commands:
  - a) Logon to the CATIA machine either locally or from a network computer and enter the user name and password.
  - b) If the machine is being accessed through XWindows on a network computer, type 'export DISPLAY=computer\_name:0.0' or equivalent command and press the ENTER key. This UNIX command allows the user to view the CAD system and its XaiTools CATIA adapter on the monitor of the network computer.
  - c) Type '. catia.env' in order to allow the XaiTools CATIA adapter to be accessed via CATIA.
  - d) Type 'catia' in order to start CATIA
  - e) The tagged CATIA model needs to be opened.
  - f) The 'function keys' icon is selected in order to view all CATIA functions.
  - g) Mark Hale's GT\_IMAGE function will have to first be selected. Next, any other function can be selected from the default function menu on the right hand side of the screen. The GT\_IMAGE function will replace the second function that was selected from the CATIA default function menu.
  - h) Now, the GT\_IMAGE function is selected and a command prompt will appear in the window that was used while starting CATIA.
  - i) Type 'cd IMAGE.a0.4/tcl' in order to change to the directory in which the 'GIT\_Interface' adapter and its functions reside.
  - j) Type 'source tclIndex' in order to allow all 'GIT\_Interface' adapter functions to be accessed. The 'tclIndex' file is a file with a list of all functions that the 'GIT\_Interface' adapter uses. If the adapter is extended by adding other functions, the function added must be added to the text file in a format that is similar to that of all the other functions that are listed in the 'tclIndex' file.



- k) The 'coi' request file is read into the adapter by typing the following command:

IMAGE\_GTInterface <Name of the request file> <Name of the response file>

- 6) The XaiTools CATIA adapter reads in the attributes, queries the tagged CAD model via the Tk/tcl CATGEO wrapper, and extracts the requested attributes. The extracted attributes are then written out to the specified response file name (numbered '6' in the figure). Response files for the test cases are shown in Appendix C.



**Figure 68 : Methodology for obtaining attributes of geometric entities**

- 7) In the XaiTools COB Browser, load the APM response file and solve for idealized attributes and other unknown attributes that have been defined in the APM schema. The solved file contains all the geometric attributes that are needed for diverse analyses ('7' in the figure). These attributes are now used to drive a series of analyses ('8' in the figure).

After analyzing the design, the analyst may recommend changes in its geometric dimensions. In the case of a parametric CAD model, it is possible to change the dimensions by changing the appropriate values in the response file. The changed response file is then saved and fed into the CAD system and it automatically updates the design model with the new dimensions. This can be done by either extending the APM adapter or by using the graphic user interface (GUI).

## Importing and exporting parametric CAD data through the Native CATIA (CATIA format)

While using the gui to import the changed parameter file, the command to be used is as follows:

Type the ‘/paramimp’ import command and then type ‘ENTER’.

CATIA displays a window that allows the user to select the file that contains the changed attributes. The file is an ascii file and must conform with the CATIA format for importing CAD data. The file is read into the system and it prompts the designer to confirm if he/she would like the design to be modified. If the designer chooses ‘YES’, the design model gets updated (if the dimensions do not violate the constraints that are defined in the parametric design model). Finally, the ‘Update Solid’ icon is selected and the update design model can be viewed on the CATIA screen. The changed parameters of the design model can also be exported as an ascii file in the following manner:

Type the ‘/paramexp’ export command and then press the ‘ENTER’ key. CATIA displays a window that allows the user to select a file name. The parameters are written out to a file with the specified name.

## Tagging/Naming conventions

Geometric entities and other CAD entities have been uniquely tagged in the CATIA design models as explained in Section 5.2. The following rules need to be followed while tagging CATIA entities:

- a) The IDENTIFY function is used to tag any CATIA entity. The tags are referred to as ‘identifiers’ in CATIA manuals.
- b) By default, unique tags are automatically assigned to every entity in CAD systems, however, for purposes of clear identification and conformity with APM attribute names, the designer may change these tags and assign his own.
- c) The tag is comprised of a string of characters.
- d) The tag must have at least 1 character.
- e) The tag or identifier should not exceed 70 characters.

- f) The tag cannot begin with a '\$' or '\*' character; all other characters are allowed.
- g) Any blank character at the beginning of a tag is automatically removed by CATIA.
- h) Lower case characters are converted to upper case characters by CATIA (but the coi/cos models can use lower case names as CATIA considers such tags to be case-sensitive).
- i) A tag or identifier of a geometric entity can be changed at any time by the designer (but ensure conformity with any APMs that use it).

#### a) Geometric entity tagging approach

While obtaining the attributes of geometric entities, the entity is tagged. However, while querying the attributes of the entity, the following naming convention must be followed:

(Tag of the geometric entity).attribute : ? ;

For example, if a line entity is tagged as 'line1' and if its 'length' attribute is required, the request file should be in the following format:

line1.length : ? ;

The geometric entities and attributes that are supported by the XaiTools CATIA Adapter are listed below. Note that some of these attributes come directly from attributes of CATIA entities available via primitive CATGEO functions while others are calculated by the adapter based on such CATGEO functions.

#### a) Wireframe entities

##### 1. Point

The co-ordinate attributes of tagged point entities can be extracted. The attributes need to be queried in the request file as follows:

x (for the x-co-ordinate)  
 y (for the y-co-ordinate)  
 z (for the z-co-ordinate)

## 2. Line

The starting point entity co-ordinates, ending point entity co-ordinates and lengths of tagged line entities can be obtained. The attributes need to be queried in the request file as follows:

- start.x
- start.y
- start.z
- end.x
- end.y
- end.z
- length

## 3. Circle

The center (origin) co-ordinates of a circle can be extracted, along with its radius, diameter and area. They need to be queried in the request file as follows:

- radius
- diameter
- origin.x
- origin.y
- origin.z
- area

## b) CSG primitives

Sample CSG primitives that are supported are as follows:

### 1. Cylinder

The center (origin) co-ordinates of the base of the cylinder (point entity) can be extracted, along with the radius, diameter and height. It is also possible to extend the XaiTools CATIA adapter to obtain volume and surface area properties, if programmed accordingly. The attributes need to be queried in the request file as follows:

- inner.radius (if hollow)
- outer.radius (if hollow)
- radius (if solid)
- height
- origin.x
- origin.y
- origin.z

## 2. Sphere

The center co-ordinates of the sphere (point entity) can be extracted, along with its radius. It is also possible to extend the XaiTools CATIA adapter to obtain volume and surface area properties, if programmed accordingly. The attributes need to be queried in the request file as follows:

- inner.radius (if hollow)
- outer.radius (if hollow)
- radius (if solid)
- origin.x
- origin.y
- origin.z

Note: The XaiTools CATIA adapter can be extended for other CSG primitives as well, if necessary.

### b) Dimension entity tagging approach

When the value of a dimension entity from a 2D draft view is needed, first, the dimension entity has to be tagged. While querying the value of the dimension entity, the name of the entity must be listed in the request file. The query protocol must be as follows:

(Tag of the dimension entity) : ? ;

For example, if a radius dimension is selected in 2D draft views and uniquely tagged uniquely as 'radius\_for\_circle1', and, if the dimension value of the circle radius is needed, the request file would be as follows:

radius\_of\_circle1: ? ;

Normally the APM considers entities like circles as distinct entities with attributes (E.g. circle1.radius). Thus this would be written as 'circle1.radius: ?;'. Note however that in CATIA this tag refers to a single dimension entity.

### c) Parametric entity tagging approach

While obtaining the parameters from a 3D parametric CAD model, first, the parameter entities have to be uniquely tagged. While querying the value of the parameter entity, the name of the entity must be listed in the request file. The query protocol must be as follows:

(Tag of the parameter entity) : ? ;

For example, if a radius parameter is selected in the 3D parametric CAD model and uniquely tagged as 'radius\_for\_circle2', and if its parameter is needed, the request file should contain the following type of request:

radius\_of\_circle2: ? ;

This would normally be written as 'circle2.radius' in the APM as in the dimension tagging approach where the APM versus CATIA entity differences are similar.

### d) Combining the three approaches

All the above three approaches may be used simultaneously, in order to query the attributes of a 3D CAD model. For example, if a 3D parametric model along with its associated 2D draft views exist, it is possible to simultaneously obtain the attributes of parametric entities, dimension entities, as well as wireframe and CSG entities.

## REFERENCES

1. Arabshahi, S., D.C. Barton and N.K. Shaw (1991). "Towards Integrated Design and Analysis." Finite Elements in Analysis and Design 9(4): 271-293.
2. Arabshahi, S., D.C. Barton and N.K. Shaw (1993). "Steps Towards CAD-FEA Integration." Engineering with Computers 9(1): 17-26.
3. Armstrong, Cecil G. (1994). "Modelling Requirements for Finite-Element Analysis." Computer-Aided Design 26(7): 573-578.
4. Autodesk, Inc. (1998). Customization Guide. 1999.
5. Benzley, Steven E., Karl Merkley, Ted D. Blacker and Larry Schoof (1995). "Pre- and post-processing for finite element method." Finite Elements in Analysis and Design 19: 243-260.
6. Chandrasekhar, Ashok (1999). Interfacing Geometric Design models to Analyzable Product Models with Multifidelity and Mismatched Analysis Geometry. Mechanical Engineering Master's Thesis. Atlanta, Georgia Institute of Technology: 165.
7. Chandrasekharan, B. (1990). Design problem solving: a task analysis. AI Magazine. 11: 59-71.
8. Cook, Robert, David S. Malkus and Michael E. Plesha (1989). Concepts and Applications of the Finite Element Analysis, John Wiley & Sons.
9. DASSAULT, STSTEMS (1997). Overview. 3D Parametric Variational Modeler Guide: 1-10, Chapter 1.
10. Dennis, Tord (1999). Tutorial on 'Interfacing capabilities of CAD systems', Engineering Computing Services, Georgia Institute of Technology. (<http://www.cad.gatech.edu/support/support.html>)
11. Engineering Information Systems Lab, Georgia Institute of Technology (1999). XaiTools: An X-Analysis Integration Toolkit.

12. Finn, Donald (1993). "Introduction: Preliminary Stages of Engineering Analysis and Modeling." AI EDAM 7(4): 231-237.
13. Finn, Donald P. (1993). "A Physical Modeling Assistant for the Preliminary Stages of Finite Element Analysis." (AI EDAM) Artificial Intelligence for Engineering Design, Analysis and Manufacturing 7(4): 275-286.
14. Finn, Donald P., J.B. Grimson and N.M. Harty (1992). An Intelligent Modelling Assistant for Preliminary Analysis in Design. Artificial Intelligence in Design '92. J. S. Gero. Netherlands, Kluwer Academic Publishers: 597-596.
15. Finnigan, P.M., A. Kela and J.E. Davis (1989). "Geometry as a Basis for Finite Element Automation." Engineering with Computers 5: 147-160.
16. Gere, James M. and Stephen P. Timoshenko (1990). Mechanics of Materials. Boston, PWS-KENT Publishing Company.
17. Gero, J.S. (1990). Design prototypes, a knowledge representation schema for design. AI Magazine. 11: 27-36.
18. Hale, Mark, James Craig and Russell Peak (1999). On the role of geometry model in engineering design. CATIA Solutions. May/June 1999: 40-42.
19. Hale, Mark A. (1998). Dynamic CATIA CATGEO Access: An Interpretive Load Module Implemented With Tk/tcl. Atlanta, GA, Aerospace Systems Design Laboratory, Georgia Institute of Technology.
20. Hemmelgarn, Don (1999). ANSYS, Inc. Partners with International TechneGroup Incorporated (ITI) to Deliver Geometry for Analysis. 1999.
21. Hoimyr, Nils J., CERN (1996). CAD/CAM and the Exchange of Product data. 1999.
22. Hunten, Keith A. (1997). CAD/FEA Integration with STEP AP209 Technology and Implementation, Lockheed Martin Tactical Aircraft Systems: 1-9.
23. IBM (1991). Geometry Interface Reference Manual, CATIA Base Version 3.
24. ISO TC184/SC4/WG5 N243 (1995). EXPRESS-M Reference Manual. Surrey, England, CiMiO Ltd.
25. Mortensen, Michael E. (1997). Geometric Modeling, John Wiley and Sons, Inc.
26. Parametric Technology, Corporation (1997). Pro/Toolkit User's Guide, Fundamentals. 1999.



27. PDES, Inc. (1997). CAD data exchange standards. 1997.
28. Peak, Russell, Robert E. Fulton, Ashok Chandrasekhar, Selcuk Cimtalay, Mark Hale, Donald Koo, et al. (1999). Design-Analysis associativity Technology for PSI, Georgia Institute of Technology.
29. Peak, Russell S. (1993). Product Model-Based Analytical Models (PBAMS): A New Representation of Engineering Analysis Models. Mechanical Engineering Ph.D Thesis. Atlanta, Georgia Institute of Technology.
30. Peak, Russell S. and Robert E. Fulton (1993a). Automating Routine Analysis in Electronic Packaging Using Product Model-Based Analytical Models (PBAMs), Part I: PBAM Overview. New Orleans. Paper 93-WA/EEP-23.
31. Peak, Russell S. and Robert E. Fulton (1993b). Automating Routine Analysis in Electronic Packaging Using Product Model-Based Analytical Models (PBAMs), Part II: Solder Joint Fatigue Case Studies. New Orleans. Paper 93-WA/EEP-24.
32. Peak, Russell S., Robert E. Fulton, Ichirou Nishigaki and Noriaki Okamoto (1998). "Integrating Engineering Design and Analysis Using a Multi-Representation Approach." Engineering with Computers Volume 14, Number 2.: 93-114.
33. Peak, Russell S., Robert E. Fulton and Suresh K. Sitaraman (1997). Thermomechanical CAD/CAE Integration in the TIGER PWA Toolset. Advances in Electronic Packaging-1997. E. Suhir and et al. Kohala Coast, Hawaii. EEP-Vol. 19-1: 957-962.
34. Peak, Russell S., Andrew J. Scholand and Robert E. Fulton (1996). On the Routinization of Analysis for Physical Design. Application of CAE/CAD to Electronic Systems. D. Agonafer and et al. Atlanta. EEP-Vol.18: 73-82.
35. Peak, Russell S., Andrew J. Scholand, Diego R. Tamburini and Robert E. Fulton (1999). "Towards the Routinization of Engineering Analysis to Support Product Design." Intl. J. Computer Applications in Technology Vol. 12, No. 1 (Invited Paper for Special Issue: Advanced Product Data Management Supporting Product Life-Cycle Activities): 1-15.
36. Qu, Jianmin (Fall 1997). Elastic Yield Design Course, Georgia Institute of Technology, Atlanta.
37. Shephard, Mark S. and Mark A. Yerry (1986). "Toward Automated Finite Element Modeling for the Unification of Engineering Design and Analysis." Finite Elements in Analysis and Design 2: 143-160.

38. Shigley, Joseph E. and Charles R. Mischke (1989). Mechanical Engineering Design, McGraw-Hill.
39. Tamburini, Diego R. (1999). The Analyzable Product Model Representation to Support Design-Analysis Integration. Mechanical Engineering. Atlanta, Georgia Institute of Technology: 626.
40. Wilson, Miyako (expected 2000). Constrained Object Representation for Engineering Analysis. Mechanical Engineering Master's Thesis. Atlanta, Georgia Institute of Technology.
41. Wolfram, Stephen (1996). The Mathematica Book, Wolfram Media/ Cambridge University Press.