A Solid Model Comparison Approach Based on a Model Tree Analysis

JEAN-FRANÇOIS CHATELAIN (1), ROLAND MARANZANA (2), SERGE ST-MARTIN

Mechanical Engineering Department (1)
Production Engineering and Automation Department (2)
École de Technologie Supérieure, Université du Québec
1100 Notre-Dame Street West
Montréal, Québec, H3C 1K3
CANADA
Tél. : (514) 396-8512
Fax : (514)396-8530

Abstract: This work presents an innovative approach to the solid model comparison problem which aims to detect and extract all geometric differences between two successive versions of the same part. The automatic model comparator has been developed to offer a standardized information about solid model evolutions to all engineering teams concurrently involved in the product development process. The comparison approach is based on a systematic analysis of both solid model trees belonging to each compared part in order to detect any change in the primitives or Boolean operations used throughout the part modelling process. The model comparator has been successfully applied to various aircraft structural parts which have been affected of multiple incremental changes in their design. The comparator has been implemented to the CATIA v4 Computer Aided Engineering software.

Key-Words: Product development, CAD, Change management, Data management, PDM, Solid model, Checker

1. Introduction

The development cycle of a product produces lots of data that is dynamically modified through the complete cycle: each step going through the product definition creates new revisions of different documents. This data management becomes more and more important as the product complexity increases. On the other hand, to succeed in the current market competitiveness, the reduction of the product development cycle is a necessity and this one cannot be realized without an efficient and reliable data management.

The Product Data Management systems (PDM) are software tools that facilitate the management of all files involved in the product development cycle which includes all the design and the manufacturing processes required to deliver the product. Among their principal functions, there is the management of dependencies and links between files and the management of the product configurations and document releases [1]. The Computer Aided Design and Manufacturing (CAD/CAM) data, which defines the product as a geometric model, is one of the major and central information managed by these systems. The incremental design process generates multiple CAD revisions from which it becomes difficult to deal from a « client » point of view in a concurrent engineering environment. For example, a design revision may or may not alter an existing tool path generated from the previous part model revision. Without a systematic and reliable notification or annotation system, it may become very tedious to identify every change occurrence between two successive revisions of a same part. This task of identifying every geometric and cosmetic feature difference between two models is difficult to keep reliable and systematic under a manual process. It greatly depends on the good behalf of every designer. On the other hand, this comparison task cannot also be ensured through the PDM systems, because of the need to access the internal CAD database implementation to extract the geometric data representation for comparison. This can only be realized through extended programming with the Application Procedural Interfaces (API) of the particular CAD implementation.

1.1 Existing Approaches

The comparison of solid models has been a problem of interest in many researches. Most of the existing works are based on the internal representation of the

compared solids. The main problem related to this approach is that different data structures can represent the same solid in the case of Boundary Representation (B-Rep) models, while different modelling trees can lead to the same product involving the same primitives in the case of Constructive Solid Geometry (CSG). In this particular case, two different sets of primitives may represent the same part. Due to this plurality of solutions, most of the comparison algorithms refer to a standardization step for the models before their comparison. Among different techniques, Leinen [2] proposes an original method based on the graph theory to normalize models defined through a B-Rep data structure. For CSG models, Perng and al. [3] refer to a destructive solid geometry technique for the model normalization. This technique subtracts different primitives to a blank model in order to obtain the solid model being the object of comparison. Once normalized, the differences can be found and interpreted. For the comparison, the former research proposed by Leinen is based on the resolution of constraints to relate the "PSC" graphs of both compared models. Other techniques are rather based on form features extraction and comparison from the B-rep or the CSG structure of the solid models [4], [5]. Although all of these techniques have the benefits of being general with the alternative of being applied to totally different solid models, they seem to loose the design intent through the normalization process, which is a major concern in our comparison problem applied to models of the same part in revision. In this context, a counterbore added to a model is different than a Boolean subtraction of a cylinder, even if the geometrical result is the same. In the first case, the design intent is properly expressed. Our proposed algorithm is based on the systematic comparison of the historical modelling information based on the model trees. This approach properly translates the design intent found into each revision by comparing each primitive and operation, when required.

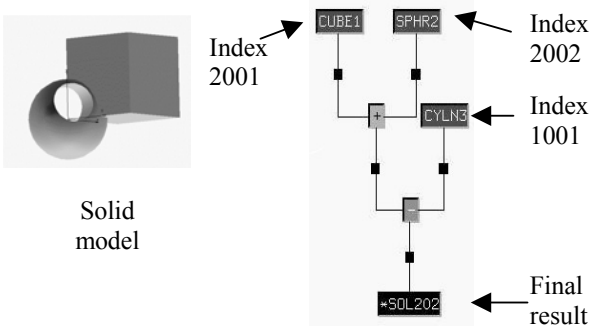
2. The Comparison Algorithm

An automated comparison approach is proposed in this work to systematically extract all differences between two successive revisions of a solid model. The algorithm is based on an exhaustive model tree analysis which verifies the correspondence between each primitive and the Boolean operations applied on them. From this approach the design intent related to the modification can be properly captured. The figure 1 shows a simple model with its corresponding model tree. From such a model tree representation, which is expressed though a matrix as the solid model internal representation, all the required geometrical information and Boolean operations applied on the primitives are extracted for the comparison process. The model tree and the matrix representation shown in figure 1 have been generated from the CATIA V4.0 CAD/CAM software within which the comparison algorithm proposed in this work has been fully implemented.

2.1 Parameters and Data Extraction

The algorithm is based on a systematic comparison of the model tree matrix representation related to each model being compared. It is based on an exhaustive comparison of four parameters defining the geometry and the location of each primitive in its corresponding model tree. The first parameter of importance is defined as being the type of primitive belonging to the model, like a cylindrical, a prismatic, a planar, or a lofted surface primitive, as an example. The second most significant parameter to consider through the comparison process is the transformation matrix expressing the location of the primitive with respect to the global reference frame. The geometrical definition of the primitives and the index number locating each of these into the model tree are the remaining parameters referred in the approach to identify any difference between the two models in comparison. The former includes the numerical values defining the primitives, e.g. the length, the width, and the depth of a cube for example, as well as the geometrical information related to more complex elements, let say the spine and the section of a swept primitive.

Depending on the CAD application, these comparison parameters can take any figure, or simply being inexistent. This is the case for the index numbers in our application which are proposed to locate each primitive in the model tree. In this particular case, the API available with the CAD system targeted for our application does not supply any function to extract this kind of information. Thus, a specially built routine has been developed to define an index number for each primitive belonging to a model tree. As shown in figure 1, the index is defined based on the level in which the primitive sits into the model tree, starting with level 1000 and incrementing a value of 1000 for each additional level of the tree.



	Operation	Entity 1	Entity 2
Line 1	1 (union)	45123 (sphere)	348 (cube)
Line 2	3 (subtraction)	-1	2316 (cylinder)

Fig.1 Model tree and its matrix representation

When two compared models are found to be identical based on these parameters comparison using the sorting mechanism explained in the next section, the Boolean operation affecting each leaf of the model tree are next compared to validate the similitude of both models.

The comparison parameters are extracted through the proprietary CAD Application Procedural Interface routines which give access to the model data. In this work, the CATGEO routines of the CATIA application software are utilized to extract the various parameters required to sort the primitives. Such routines require the type of primitive as input data in order to output the size and the reference elements required for its definition. The type, the transformation matrix expressing the location of the primitive with respect to the reference frame as well as the operation applied to the primitive is extracted from the model tree information expressed through a three-column matrix as the internal representation of the model tree. As shown in the matrix in figure 1, the first column relates to the Boolean operation number while the other two columns relate to an identifier from which the entity type and all the primitive definition data can be deduced. The negative value for entity 1 in line 2 of the matrix points toward the operation line number 1. In this case, the first line of the model tree represents the union between the sphere and the cube while line two relates to the subtraction of a cylinder from this last result expressed through the entity number -1.

2.2 Classification of the Primitives

The comparison of the solid models consists in matching pairs of primitives having some identical comparison parameters. It is a scanning process checking one primitive of a model against all the others belonging to the second model to find the maximum number of identical parameters as possible. Through the entire process, the sorting algorithm will order each pair of primitives in one of the 5 specific classes based on their similitude (figure 2). The primitives that are found to be exactly of the same type, with identical numerical/geometrical parameters and locating matrix, as well as being symmetrically located in their respective model tree (same index number) will be identified as class A primitives. All primitives that comply to class A except for the index number will be identified as belonging to class B. Similarly, the primitives complying to class B, except for the parameters, are identified to be part of class C while the class D includes primitives of the same type only. Finally, the class X is reserved for the primitives remaining after the sorting process is terminated.

The algorithm developed to find all the differences between two solid models thus stacks pairs of primitives into one of these 5 classes. This is done through an iterative elimination process, sorting all pair of primitives starting with class A and finishing with class D. As shown in the flowchart (figure 3), all the primitives are first stacked in class X for both model trees. Then, a first primitive of model1 “Pr1” is compared against all primitives of the revised model for a match complying to class A definition.

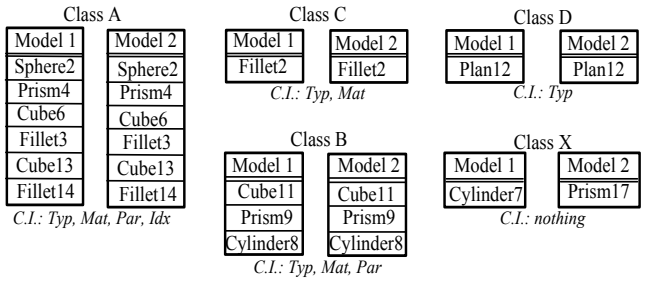


Fig.2 Primitives classification

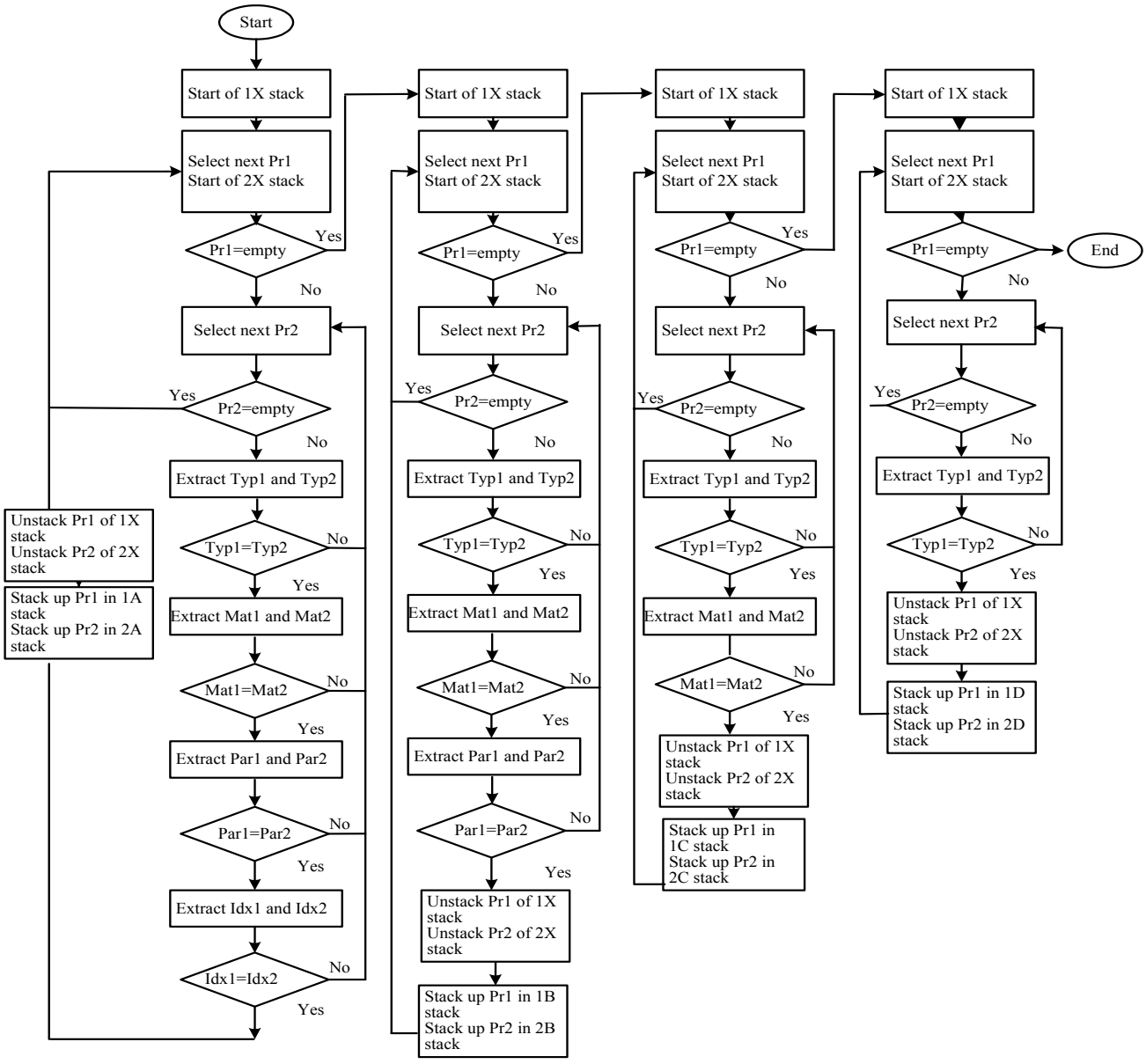


Fig.3 Sorting algorithm

The process is forwarded until all Pr1 are scanned. For any match found, the corresponding primitives Pr1 and Pr2 are un-stacked from their class X pile and are each stacked into their respective Class A. For two identical solid models, this process would lead to two identical piles of class A. The remaining primitives belonging to class X are then similarly compared for a match complying with class B definition. The process ends when there is no more primitive to sort or when a tentative for a match has been performed for all class definition. In this latter case, the remaining elements in class X mean there is some added or removed primitives to the revised version of the solid model.

3. Aeronautical Applications

The solid model comparison approach has been validated using several aeronautical examples related to the Bombardier Aeronautic production. One of these validation examples is presented below. For more details, one can refer to St-Martin [6].

The aircraft frame showed in figure 4 includes 102 primitives distributed in a 23 levels model tree. There have been four different revisions applied to the original model. The first revision includes a translation applied to one of its opening and a minor change in the profile of its openings.

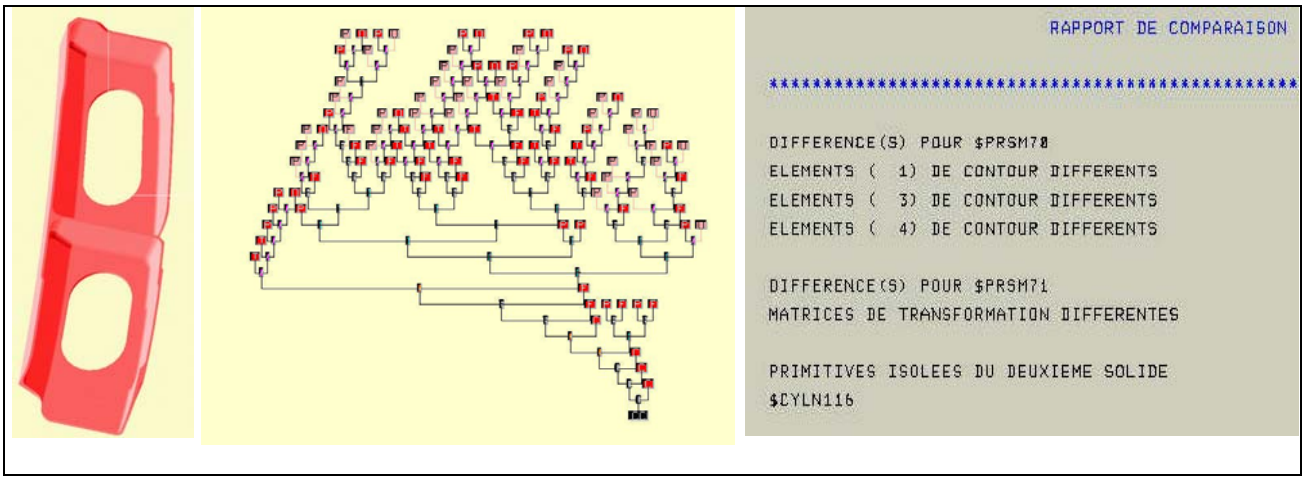


Fig.4 Aircraft frame solid model and its comparison report

These differences are so minor that they are imperceptible to the designer. The algorithm takes less than a minute to report these modifications. The second and third sets of modifications respectively add on top of the last ones, a primitive suppression and a primitive addition. Both are compared with success to the original model. To give an idea of the algorithm behaviour and performance, the second revision is about 2 minutes longer to compare than the third set of modifications. This is due to the scanning mechanism of the algorithm which significantly depends on the model tree ordering of the primitives (addition versus subtraction). Finally, the last revision is the same as the third one except for the added primitive which is moved to a higher level of the model tree. Based on the sorting scheme explained above, this actually accelerates the model comparison since the primitive is higher in the model tree related matrix from which the comparison is done. It takes just more than a minute to report all the differences.

The algorithm performance is found excellent when compared to the manual process of identifying such kind of modifications, which is quite tedious and very much unreliable. In fact, the few minutes required to compare complex models is nothing, considering this calculation time largely depends on the computer unit utilized for the implementation and also that the calculation is concurrently done to the designer tasks.

After the solid model differences are all found, the model checker generates a comparison report. This one takes two different aspects, a graphical and a textual form. The former allows for a rapid check through a colour coding identifying all types of differences, as previously discussed. The primitives found different are identified with a specific colour within the compared model as well as within the related model trees. For more detailed information, the textual form of the report can be referred (figure 4). This latter can give as much detailed information as it is available through the API routines utilized to extract the primitives geometric information. This important and very rich comparison information can then be fed to a PDM system for rigorous historical data archiving of the solid model evolution of a given part.

4. Conclusion

An automatic comparison approach for CAD solid models has been described in this paper. The proposed algorithm systematically and reliably extracts all the differences between models, based on a concurrent engineering context involving successive revisions of a same part. The model checker prototype integrated to the CAD/CAM software CATIA reports all the differences between two models in a few seconds with complete reliability as compared to a manual comparison which is tedious, time consuming and not error prone. The approach could also lead to an automatic transfer to a PDM system of all the differences found by the comparator to keep historical data related to each part model of a product.

The proposed prototype has been validated through various structural part models comparison belonging to the Bombardier Canadair aircraft manufacturer.

Acknowledgements:
This project has been realized throughout a research program regarding the CAD and PDM tools enhancement with the collaboration of Bombardier Aeronautic. We sincerely thank our partner Bombardier Canadair for their assistance and the Natural Science and Engineering Research Council for their financial support.

References:

[1] PELTONEN H., PITKÄNEN O., SULONEN R., *Process-based view of product data management*, Computers in Industry, vol. 31, no. 3, pp. 195-203, 1996.

[2] LEINEN S., JUNG J-P, GARDAN Y., *Comparaison de modèles de CAO par normalisation de graphes*, Revue de CFAO et d'informatique graphique, vol. 12, no. 1-2, pp.153-167, 1997.

[3] PERNG D.B., ZEN C., LI R.K., *Automatic 3D machining feature extraction from 3D CSG solid input*, Computer Aided Design, vol. 22, no. 5, pp. 285-295, 1990.

[4] LEE Y.C., FU K.S., *Machine understanding of CSG: extraction and unification of manufacturing features*, IEEE Computer Graphics and Applications, vol. 7, no. 1, pp. 20-32, 1987.

[5] GARDAN Y., MINICH C., *Feature-based models for CAD/CAM and their limits*, Computers in industry, vol. 23, no.1-2, pp.3-13, 1993.

[6] ST-MARTIN S., *Maîtrise du changement dans les modèles CAO*, Projet d'application, Maîtrise en technologie des systèmes, École de Technologie Supérieure, Janvier 2001.