Operating invalid feature-based models

This item was submitted to Loughborough University's Institutional Repository by the/an author.


Additional Information:

- This is a conference paper.

Metadata Record: https://dspace.lboro.ac.uk/2134/13877

Version: Accepted for publication

Publisher: Society for Design and Process Science, USA

Please cite the published version.
This item was submitted to Loughborough’s Institutional Repository (https://dspace.lboro.ac.uk/) by the author and is made available under the following Creative Commons Licence conditions.

For the full text of this licence, please go to:
http://creativecommons.org/licenses/by-nc-nd/2.5/
ABSTRACT

A valid feature-based representation is one where instantiated features in a model agree with the features' expected behaviours, available and defined as a library. Invalid feature-based models happen when manipulations on the model change the interrelationship among features therefore changing the behaviour of an instantiated feature.

Freedom of manipulation is an intrinsic advantage of using a CAD system and it is taken for granted. However, even the most basic manipulation, such as "adding" a feature to a model, is capable of disrupting the validity of a representation. Furthermore, invalid models could compromise the usefulness of any following analysis on it.

Thus, identifying means to operate on an invalid model to make it valid, through "revalidation operations", is a necessity in Feature-based CAD systems. It allows conventional CAD systems (usually more preoccupied with representing and producing feature-like shapes within a geometrically constrained environment) to interface more easily for example with CAPP systems (usually more preoccupied with planning problems than with the correctness of the representation).

The framework of a feature-based validation system, called FRIEND (Feature-based Reasoning system for Intent-driven Engineering Design), and a discussion on representation validity analysis is presented with emphasis on identifying and discussing "revalidation operations".

NOMENCLATURE

- CAD = Computer Aided Design
- CAE = Computer-Aided Engineering
- CAM = Computer Aided Manufacturing
- CAPP = Computer-Aided Process Planning
- DbF = Design-by-Features
- DI = Designer's Intent
- FBM = Feature-Based Modelling
- FPV = Feature Produced Volume
- GSM = Geometric Solid Modelling
- ICAD = Intelligent CAD Systems
- MDI = Morphological DI

INTRODUCTION

Design-by-Features (DbF) is one approach for implementing Feature-based Modelling systems (FBM, see Figure 1) that offers a set of high-level entities - the features - to the designer which are relevant to the domain target - type of product - and thus are more effective in capturing designer's intent and interfacing with other engineering activities such as Computer-Aided Manufacturing (CAM), Computer-Aided Process Planning (CAPP) and Computer-Aided Engineering (CAE).

FBM, and indeed DbF, systems are usually based on Geometric Solid Modelling (GSM) techniques. However, one basic element that makes GSM so well established, important, popular and powerful, namely Geometric Validation, lacks a sibling in the FBM world. This is so because features add a layer of complex semantics to CAD systems which make it difficult to establish measuring means and are subjective to implement. Feature-based representation validation is
very important because it is the process responsible for
guaranteeing the delivery of a valid representation (and
therefore verified, useful and misrepresentation free) to
downstream applications.

The validation is a required analysis of DbF
systems because invalid situations are likely to occur
even after basic editing manipulations such as "adding a
new feature to the model".

As DbF systems usually subsume an
implementation on top of a GSM scheme (such as CSG,
Brep or hybrid) then, they also subsume the availability
of low level modelling operators (such as Euler Operators
or Boolean Operators) as well as GSM validation (such as
Euler-Poincare formulae verification or Boolean
regularisation; Zeid, 1991). Therefore, these low-level
operators are not included in this study. Other approaches
(Stroud, 1993; Subrahmanyan et al., 1995) go into this
level.

In earlier studies (Hounsell and Case, 1996) on the
properties of DbF systems and their validation it was
made clear that not only is verification an important task,
but of equal importance and usefulness is the ability to
operate the model when an invalid situation is found.

This paper presents a simple but formal
validation framework for a DbF system that performs
self-validation without using parameter-constraining
techniques. The characterisation of the domain is
presented through the definition of Morphological
Designer's Intents. Groups of rules are presented to verify
the conformity of the model with their expected
behaviours - The Designer's Intents. These rules are also
responsible for firing operations that aim to make the
model valid if invalid situations are found. A discussion
and enumeration of these revalidation operations is then
given. Finally, a brief discussion of the implementation is
presented followed by the conclusions.

A VALIDATION FRAMEWORK

Representation Validation is a set of
verifications on the feature-based model to analyse its
conformity with the previously established domain-
dependent feature concept (its role as a 3D modelling
technique, common sense expected behaviour and extra
meanings). Therefore, it is called conceptual feature
validation (Hounsell and Case, 1996). Feature-based
representation validation is a necessary step because,
ultimately, downstream applications will work on the
representation and they expect correct data.

The following are the elements necessary to
come a feature-based representation validation
reasoning system (see left-hand side of Figure 2).

- Domain characterisation: features can be defined by
  their underlying "intents" from many
different perspectives. This characterisation must
  be made clear and verifiable.
- Validity conditions: define the means by which
  to verify if all features configured to model a
  component comply with their expected intents.
- Revalidation operations: represent means to
  operate on the model and to turn invalid
  representations into valid ones and are selected
  and requested by the "validity conditions".

DOMAIN CHARACTERIZATION

Capturing Feature-based Designer's Intents (DI's)
at early stages of the design through a more user-friendly
interface that includes a meaningful design vocabulary
are properties of a FBM system that could allow more
intelligent decisions and reasonings to be made and are
considered the only possible basis for Intelligent CAD
(ICAD) systems (Dixon et al., 1990).

"Designer's intents are of high importance to be
preserved but their understanding has a complicated
nature" (Yoshikawa and Ando, 1987). Although features
are a proclaimed and accepted means of capturing and
representing DI's, existing FBM systems do not deal with
DI's as a major concern for three main reasons: Firstly,
there still is a lack of a formal well-accepted definition
for features and their role as a geometrical modelling
technique. Secondly, there is also the same lack of
understanding of what DI's are, especially in the FBM
context and; Thirdly, identified intents are usually tied to
the application in particular implementations.

Feature-based constraint-driven approaches are
said to capture design's intents (DI's) via the use of
parameters and parameter dependency/relationships
(Sheu and Tin, 1993; Dohmen, 1994) but, this is achieved
mainly through the designer's understanding and explicit
assignment of how parameters are related to a specific
functional aspect. The approach presented here aims to
establish some of these functional aspects beforehand
through the definition of feature properties and
behaviours - called Designer's Intents (DI's) - regardless
of their type and positioning. Therefore, some level of
self-correctness can be expected when using features in a
way that affects those previously defined DI's. This approach establishing yet another definition for features.

Feature-based Designer's Intents (DI's) are defined as a variety of concerns that help decide on a specific geometric attribute or configuration. They are factual peculiarities of the geometric design that are intrinsic to features themselves or to the use of features in the design and have (especially manufacturing) engineering-related purposes. DI's are properties that are expected to arise in the model because of the use of a feature in a specific location or because a/the interaction that a feature provokes with the existing surrounding features in the model.

Designer's Intents include morphological functional, theoretical functional and relational functional ones. A taxonomy of DI's concerned with the feature-based geometric detail design phase for prismatic machining parts has been established (Hounsell and Case, 1977c) but emphasis here is placed on feature-based geometric detail design phase for prismatic functional ones. A taxonomy of DI's concerned with the volume). The feature's nature feature-based representation must result in a change in the

features in the model.

Morphological Designer's Intents

Morphological DI's (MDI's) are to be considered specially, but not solely, when an interaction between feature volumes occurs. Intermediate states, delete operations and editing manipulations have a direct influence over MDI's in design. To deal with MDI's the semantics of non-conflicting and conflicting interactions between features must be defined. Four MDI's can be implemented within the geometric realm of feature-based models:

Labelling MDI's identify the relationships between all features' faces and their attributes. Every feature has a set of labelling relationships that is kept as the feature's label. Labelling is implemented by defining a template of virtual and real faces that bound the produced volume of a feature. Virtual faces basically identify tooling external access directions and real faces identify surfaces to be imprinted on the part.

In addition to establishing a label-to-shape relationship, features are usually expected to imply a volumetrical behaviour, which is called the feature's nature by Lenau et. al. (1993), of adding material (when it is said to have a positive volume) or removing material (when it is said to have a negative volume) from the stock. A feature's nature is identified by a Boolean operation (union for a positive volume and difference for a negative volume). The feature's nature implies that a change in the feature-based representation must result in a change in the
to the desire to imprint specific shapes on the model.

The feature's

nature

negative

is identified by a Boolean operation

A feature's nature

volumetrical behaviour, which is called the feature's nature from the stock.

features in the model.

Morphological Designer's Intents

Morphological DI's (MDI's) are to be considered specially, but not solely, when an interaction between feature volumes occurs. Intermediate states, delete operations and editing manipulations have a direct influence over MDI's in design. To deal with MDI's the semantics of non-conflicting and conflicting interactions between features must be defined. Four MDI's can be implemented within the geometric realm of feature-based models:

Labelling MDI's identify the relationships between all features' faces and their attributes. Every feature has a set of labelling relationships that is kept as the feature's label. Labelling is implemented by defining a template of virtual and real faces that bound the produced volume of a feature. Virtual faces basically identify tooling external access directions and real faces identify surfaces to be imprinted on the part.

In addition to establishing a label-to-shape relationship, features are usually expected to imply a volumetrical behaviour, which is called the feature's nature by Lenau et. al. (1993), of adding material (when it is said to have a positive volume) or removing material (when it is said to have a negative volume) from the stock. A feature's nature is identified by a Boolean operation (union for a positive volume and difference for a negative volume). The feature's nature implies that a change in the feature-based representation must result in a change in the volume and surface of the component being modelled. This feature's requirement and ability to change the existing model is called the changeability MDI. The changeability requirement invalidates obsolete features (Shah, 1990) that occur when a feature is completely inserted into another and has the same nature. However, it does not require that all the boundaries of the feature's produced volume should be shaped into the part.

A feature must have adequate parameters to exactly fit and define the intended form (in the same way as an edge is limited by its two exact ends, called vertices) thus, the feature must fit within the limits of where it is intended to be placed. This ability to fit is called the fittability MDI. The fittability requirement invalidates feature's parameters made obsolete (Shah, 1990) where feature's parameters do not describe exactly the extent of what it imprints on the pan.

Furthermore, interesting and difficult situations arise when redundant intents are found. Features that have overlapping volumes usually present a redundant MDI. This is a feature interaction problem that has been receiving much attention in the literature as being of special difficulty to handle (see Mill et al., 1993: and The Contiguity Problem in Shah, 1990).

VALIDITY CONDITIONS

Validity Conditions translate Designer's Intents into verification statements. A human-based analysis of a feature-based model is usually done by searching for invalid situations and therefore much of the engineer's experience is built on the search for invalidity, rather than its validity. Thus, it is much easier to spot and devise tests for invalid situations than to identify/test the validity ones, especially in the context of abstract elements such as features that have no mathematical and well-accepted definition.

Also, invalidity tests rather than validity ones can easily be divided in sub-cases that correspond to specific remedies - the revalidation operations -, although the spectrum of invalid situations are extremely extensive and application-dependent. Therefore, it is pragmatically easier to perform feature-based validation on a model by invalidity tests. Nevertheless, from a logical point-of-view, if a model fails all invalidity tests it cannot be considered "completely" valid but, may be thought of as a non-invalid model for that specific set of criteria.
Four sets of reasonings were identified (Hounsell and Case, 1997b) related to Morphological DI's:

- **Simply geometrical** reasonings, are responsible for performing the geometric interaction scenario identification among features as well as performing other geometric reasonings defined by other applications.

- **Simply labelling** reasoning happens when, instead of volumetrical interaction, *labels* are the main focus of the reasoning, such as when the system is searching for the right *label* for a specific feature according to its faces' properties. Simply labelling rules include all those where low level interactions (face level) could result in a change in a feature's face property (from virtual to real, or vice-versa) and consequently a change in its labelling, regardless the feature's *nature*.

- **Simply volumetrical** reasoning happens when volumetrical reasonings and/or the feature's *nature*, despite the feature's *label*, are enough to fire an action such as when conflicting volumetrical intents (*hollows* or *satellite* volumes) appear in the model. Simply volumetrical rules also include those when an incoming feature interacts with the stock material, regardless of the former's *label*. This last reasoning example has priority because the stock material is considered to be the envelope of the whole component (and all its features), and thus any volumetrical analysis involving the stock would speed up the processing of the newly added feature.

- **Complex** reasoning happens when both the feature's volumetrical interaction and *label* determine the actions to be taken, as in "cut-out" cases (Mills et al., 1993). All other further interactions between features, except the stock, are also considered as complex rules.

The priority/sequence of these reasonings is depicted on the right-hand side of Figure 2: *Simply geometrical* reasoning performs GSM-based reasoning and generates the interaction scenario between features at various levels of interest (initially volumetrical interaction up to face interactions, as it is requested, Hounsell and Case, 1997 a). The interaction scenario is then considered by the subsequent set of reasonings. The first, is the *simply volumetrical* reasonings. If there is enough information, the *labels* are then verified and (re)assigned if required via *simply labelling* reasoning. If the model is not yet valid then, there will be enough information with both *labels* and geometric interactions defined. In such cases, face interactions are added an. *complex* morphological reasoning is then considered. These situations already consider some designer's experience, product type and application's constraints.

### REVALIDATION OPERATIONS

The identification of the above reasoning groups helped devise rules that pinpointed specific invalid situations and so atomic revalidation operations could be defined.

The following are the atomic revalidation operations identified:

1) **Add** volumetric intent. Similar to the Add Feature editing manipulation, but this revalidation operation manipulates FPV's that then will be later identified as a feature (via a proper label/orientation). Add volumetric intent is usually requested after other revalidation operations.

2) **Delete** volumetric intent. Similar to Delete Feature editing manipulation, but inactivated/deleted FPV’s are considered conflicting, therefore they will not reappear in the actual model again.

3) **Make Obsolete**. When a feature overlaps entirely the Morphological DI of another feature of the same nature, then this last is said to be obsoleted by the first and thus, it is removed from the model but is kept in a dormant status. Obsoleted features can become active and reappear in the model if the overlapping feature is later removed.

4) **Make Active**. Features that were made dormant in the model can become part of the actual model again via this revalidation operation. The situation that resulted in the dormant feature should have been resolved otherwise, a possible loop would arise.

5) **Split**. Split the FPV of one feature against the FPV of another producing two or three new "smaller" FPV's that should be properly labelled afterwards. This revalidation operation helps correct the feature's parameter made obsolete.

6) **Merge**. Merge the FPV's of two distinct and "touching" features producing one "bigger" FPV /feature that needs to be labelled afterwards.

7) **Label**. Responsible for operating on feature's parameters at the face level and finding a proper meaning for the result - a label according to a proper orientation. It consists of three atomic operations:
• Change face's property to Virtual.
• Change face's property to Real.
• Label search on the feature's library (find a label considering the pattern of Virtual and Real face codes) conforming to a particular orientation.

8) **Add** and **Delete Intent** relationship. Morphological DI's are kept in the model by the list of active and dormant features as well as intent relationships (such as merged_from, split_into, obsoleted_by) that help reasoning after later manipulations on the model. Therefore, Add and Delete are revalidation operations that help the management of the MDI's.

9) **Complement.** Is the operation that converts a feature with positive nature into a set of features with negative nature able to produce the same shape on the part. Together, positive and negative features form a better representation although only negative nature ones are used for machining purposes. The conversion has a plethora of possibilities that should be selected according to some criteria (Waco and Kim, 1994; Tseng and Joshi, 1994).

10) **Rigid Propagation.** Propagation extends the effects of an editing operation, such as move and rotate, towards dependant features. Propagation seems to be an important and valuable revalidation operation that should be carried out or suggested when there is a coupling relationship between features such as counter-bore or a T-slot or even a nesting relationship (that then originates a parent-child dependency) between features.

11) **General Propagation.** Manipulations on the stock-material and complex analysis (such as thin-wall identification and tool availability) that could simply render the model invalid were found to need complex editing manipulations as revalidation operations but, again, those can only be performed under the designer's assistance.

It is understood that the process of validation is a loop: once a particular intent is verified, the process analyses another intent on all features assigned to the model until all intents and all features are analysed. If a "revalidation operation" is performed, it creates a different scenario of features (hopefully a simpler one). The features involved in the revalidation operation and their adjacencies are then, once more, verified against all intents. When all intents are verified and no more new scenarios are produced, the validation process loop delivers the resulting feature-based non-invalid model to a downstream application.

**IMPLEMENTATION**

A prototype system called **FRIEND** (Feature-based Reasoning system for Intent-driven ENgineering Design), has been implemented with special concern for the validation of feature-based geometric design representations (Hounsell and Case, 1996). A clearer definition of feature semantics within FRIEND was achieved with the help of the morphological functional DI's (Hounsell and Case, 1997c), briefly presented here. The verification reasoning is based on the spatial geometrical feature interaction (such as those that are exemplified here: abuts, touches, inserted) applied at various levels, such as the feature volume and feature face levels (Hounsell and Case, 1997a),

**Figure 2** shows that to implement the reasoning loop in FRIEND, the four sets of validation reasonings, just described, are organised in a hierarchical fashion. There is also a priority relationship among the situations mentioned above and the feature interaction identification level (Volumetrical, Boundary and, Face - Hounsell and Case, 1997a). The reasoning goes deep into the interaction level if it cannot reason with the information and interaction already available, and this is another reason why the scheme **Figure 2** is in a loop.

In FRIEND, the first eight revalidation operations listed above have been implemented for features with negative natures as a DbF system for prismatic components, with the high-level editing manipulations Add and Delete Feature and is capable of assuring Morphological Designer's Intents on the feature-based model without any constraint-based analysis.

Taken together, the feature's interaction and feature's data structure (which includes feature's nature and label) offer a vocabulary that permits a knowledge-based system to reason and validate the feature-based model. A simply volumetrical reasoning is exemplified below.

**Rule**
(Feature1 has a different nature from Feature2)
(Feature1's volume "matches" Feature2's volume)
THEN
Ask "Is Feature1 being used to delete Feature2 ?"
Answer YES, delete Feature1, delete Feature2
Answer NO, delete Feature1 (it was a mistake !)
END-RULE
A simply labelling reasoning is exemplified below. If a face of a given feature "abuts" and is completely inserted into another feature's real face then, the former must be a virtual face (Silva et al., 1990).

IF
(Feature1's face FFI "abuts" Feature2's face FF2)
AND
(Feature1 AND Feature2 are active in the model)
AND
(FF1 has a Real code)
AND
(FF2 has a Virtual code)
THEN
Change FFI's property to Virtual
Label Feature1
END-RULE

DISCUSSION

The revalidation operations were observed to be dependent on a number of factors:

- Feature Interaction Identification (Hounsell and Case, 1997a). The revalidation operations reflect greatly the level in which features are defined to interact amongst themselves. High-level interaction vocabularies require high-level revalidation operations.

- Feature Representation. Besides the feature's nature and the face's virtual-real property, the actual primitive volume of the feature, sometimes called Feature Produced Volume (FPV, Shah 1990), and parameters such as fillet radius, primary axis should be considered.

- Editing Manipulations. Again, the variety and level of manipulations available to the designer would greatly influence the set of revalidation operations to be devised. Examples of editing manipulations include Add, Delete, Move, Modify and, Copy. However, a minimum set of manipulations consists of Add and Delete, which could be used internally to implement the other editing manipulations (Kim and O'Grady, 1996). If low level manipulations are considered, such as chamfering an edge, tapering a face, etc. this greatly adds to the complexity of the revalidation operations. Therefore, a taxonomy of operations would include very basic manipulations such as Euler and Sweeping Operators (Stroud, 1993; Subrahmanyan et al., 1995)

- Application Domain. Different characterisations of the domain would influence the size of the revalidation operation set. A broader domain possibly needs an extended set of operations.

General propagation poses the problem of identifying the variety of possible alternatives. For instance imagine enlarging the thickness (or height) of the part in Figure 1 where the holes were defined as "nested" (child) features to the step. A plethora of alternative scenarios can be used to revalidate the step/hole relationship (see Figure 3):

1) Change step's positioning; (keep step's parameters); change hole's positioning and labels to blind-holes; (keep hole's' parameters);
2) Change step's positioning; (keep step's parameters); increase the holes' height;
3) Change step's parameters; (keep hole's parameters and positioning);
4) Change step's label to a slot thru; (keep holes parameters and positioning); (step's parameters and positioning kept);
5) Invalidate height manipulation because of the functional significance of the hole/step arrangement

The general propagation revalidation operations could benefit from the use of a constraint-based environment where, besides the DI’s considered here, other DI’s could be explicitly captured beforehand as a parametric hierarchy-dependency.

It is believed that the intent-driven and constraint-driven approaches can work alongside and indeed, be complementary to each other. Also, an extended DI definition would embrace a larger functional requirement and therefore help automate even further the capturing and reasoning of designer's intent and would most probably require a broader set of revalidation operations.

CONCLUSIONS

A framework for an intent-driven Design-by-Features system has been presented alongside its main elements, which include revalidation operations. All elements (Designer's Intents, Interaction Identification, Invalidity Tests, Editing Manipulations, etc.) were found to influence the set of revalidation operations. Therefore, they have been presented in the context of those most
important concepts in this framework (Designer's Intents and Invalidity Tests).

Revalidation operations helped produce a DbF system that automatically revalidates a feature-based model but the supervision of the designer is required constantly because the system can capture and represent those Designer's Intents but not predict them.

REFERENCES


FIGURES

Figure 1: A feature-based model example.

Figure 2: Alternative revalidation operations.

Figure 3: Reasonings and Operations.