CHAPTER 8

Interactive feature definition

O.W. Salomons, F. van Slooten, H.G. Jonker, F.J.A.M. van Houten, H.J.J. Kals
University of Twente, Mechanical Engineering Department, Laboratory of Production and Design Engineering
7500 AE Enschede, The Netherlands

Present feature based CAD systems do not offer a facility to easily define one's own features, if they offer such a facility at all. The reason why present software tools are inadequate for feature definition is that defining features requires a lot of programming skills, making feature definition an error prone and difficult task. The people in charge of defining new features usually can be regarded as domain experts in the fields of design or manufacturing or both and not as programming experts. This chapter therefore elaborates on interactive feature definition, aiming at facilitating the definition of features by non-programming experts. Interactive feature definition is applied in a prototype of a re-design support system, called FROOM, in order to support feature based design. However, the feature definition as presented in this chapter can also be used in a computer aided process planning system for defining features to be recognized. Conceptual graphs are used as an aid in the definition of features and for representing the features.

8.1. INTRODUCTION

Present feature based CAD systems generally do not offer a facility to easily define one's own features. If CAD systems offer a facility for feature definition at all, it usually consists of some kind of programming interface to the geometry modeller. Often, the programming interface has limited access to the geometry processing functions of the modeller. In so-called Knowledge Based Engineering (KBE) systems, like e.g. ICAD^{TM}, the programming interface as well as the access to the geometry modeller is better when compared to "traditional" CAD systems. Nevertheless, even within KBE systems, feature definition is difficult and error prone, requiring extensive programming skills as well. The person in charge of defining new features usually can be regarded as a domain expert in the field of design or manufacturing or both. Present feature definition functionality requires this person to be a programming expert as well. In practice, this is hard to achieve.

This chapter therefore elaborates on interactive feature definition, thus facilitating the definition of features. It aims at providing domain experts (non programming experts) a tool with which feature definition can be performed relatively easily. The interactive feature definition is applied in a prototype of a re-design support system, called FROOM, in order to support feature based design. However, the interactive feature definition as proposed in this chapter can also be used in automated process planning systems for defining manufacturing features. The definitions of the manufacturing features can then be converted into feature recognition algorithms. Conceptual graphs are used for defining and representing the features. In the following introductory subsections we will elaborate on features, FROOM, and conceptual graphs. Finally, an overview of the remainder of this chapter will be provided.

8.1.1 Features

Features are being widely used in advanced CAD and CAPP systems. However, still a lot of different feature definitions exist. Shah defined 4 requirements that a feature should at least fulfil [SHA 90]; a feature:

1 has to be a physical constituent of a part (component).
2 ought to be mappable to a generic shape.
3 should have engineering significance
4 must have predictable properties.

For an overview of research in feature based CAD, reference could be made to [SAL 93a]. The lack of today's feature based CAD systems, is that they are focused too much on designing single components using generic built-in "user defined" features. These features allow the users only to parametrically modify the geometry of the features (parametric design). Topology and non-geometry related information cannot be defined as part of the feature. Therefore, in present CAD systems, features cannot be related (or are hard to relate) to a particular application domain. In Knowledge Based Engineering systems, like ICAD^{TM}, it is possible to define application dependent features. However, these systems lack ease of feature definition due to laborious programming.

An example of the use of features in CAPP can be found in [HOU 91], describing the PART system. Presently, the feature recognition language of PART implicitly also covers the feature definition; it is both a feature definition and a feature recognition language. In an integrated CAD-CAPP approach, the feature definition should be separated from the feature recognition algorithm. This would enable the features defined in the CAPP system to be used within the CAD system and vice versa.

Often the "difference" between design features and process planning features has been indicated, e.g. in [HUM 89]. This difference in view of designers and process planners towards features is known as the multiple views problem. However, in an observational study, conducted by some of the authors, the multiple views problem has been relativized somewhat [SAL 93b]. It was found that in the situation that was studied, the differences in view towards form features between designer and process planner were minor [SAL 93b]. It was suggested that naming conventions of both features and feature attributes could help in overcoming the multiple views problem. This means that by performing the feature definition well, the multiple views problem could - to some extent - be overcome. Another suggestion was the use of abstract features and assembly relations, from which - in the end - manufacturing features could be derived. This chapter, however, will be concerned with the definition of form features only. Form features are considered as a "generic shape that carries some engineering meaning" [WIN 91].

8.1.2 FROOM

The FROOM system is a prototype of a re-design support system, currently under development. The development of FROOM has been motivated by the following facts. Firstly, in mechanical engineering design a high percentage of all the design tasks can be considered as re-design tasks. For these tasks proper computer support tools are not yet available. Secondly, tools that integrate well with automated process planning (CAPP) systems are not yet available. The CAPP system that is of particular interest to us is the PART system [HOU 91], which has been developed within our laboratory and which is now commercially available.

FROOM is an acronym for Feature and Relation based Object Oriented Modelling. FROOM is a feature based system, allowing the modelling of both components and assemblies. Features in the FROOM context can be design form features, manufacturing form features and even abstract features. FROOM employs conceptual graphs for knowledge representation. Features, components and assemblies (concepts) as well as (conceptual) relations play a central role in FROOM. The next section will elaborate on the conceptual graphs as used in FROOM. In figure 1 the system architecture of FROOM is shown.

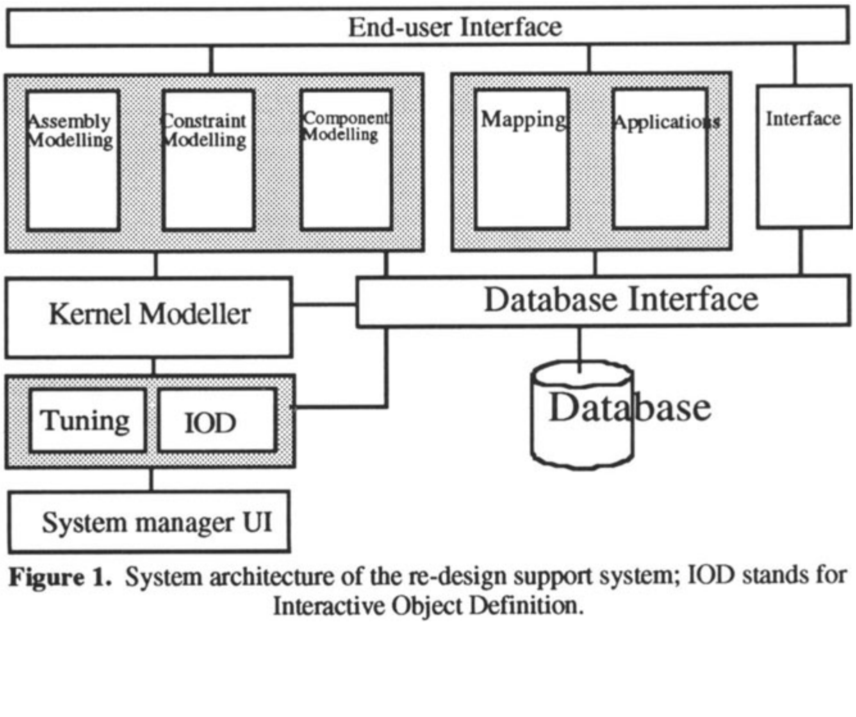

Figure 1. System architecture of the re-design support system; IOD stands for Interactive Object Definition.

As can be seen from figure 1, two different kinds of users are distinguished: end-users and system manager users. End-users are the designers who design assemblies, components etc.; they perform the actual design tasks. System managers customize the FROOM system to a certain application domain, company and user (group). They define the features to be used, the catalogues from which selections can be made and other information that can be useful for the end-users. The feature definition module is part of the Interactive Object Definition module. Apart from features, the module allows for the definition of assemblies and components. For the two different user groups, different user interfaces are available.

Another important part of FROOM is the modelling module in which both components, assemblies and constraints can be modelled. This module has access to the kernel modeller, which offers the basic geometry processing functionality. In FROOM, the ACIS^{TM} kernel modeller is used for this purpose. The application module includes the possible mappings to applications and the applications themselves. An example of an application might be manufacturability evaluation. For more details on FROOM refer to [SAL 93c] for its concepts and to [SAL 93d] for its methodic design.

8.1.3 Conceptual graphs

Conceptual graphs - or conceptual structures - have been proposed for use in natural language processing and for representing mental models by Sowa [SOW 84]. However, the number of applications of conceptual graphs in other domains is increasing. One of these domains is that of (re)design support in CAD.

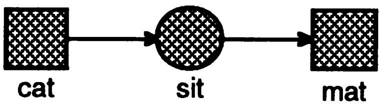

Figure 2. An example of a conceptual graph [SOW 84].

Conceptual graphs are graphs with two different kinds of nodes: concepts and conceptual relations. Conceptual graphs are therefore called bipartite. Sowa defines a conceptual graph as follows [SOW 84]:

Assumption. A conceptual graph is a finite connected bipartite graph.

1 the two kinds of nodes of the bipartite graph are concepts and conceptual relations.
2 every conceptual relation has one or more arcs, each of which must be linked to some concept.
3 if a relation has n-arcs it is said to be n-adic and its arcs are labelled 1,2..n. The term monadic is synonymous with 1-adic, dyadic with 2-adic and tri-adic with 3-adic.
4 a single concept may itself form a conceptual graph, but every arc of every conceptual relation must be linked to some concept.

In figure 2, a simple example of a conceptual graph - taken from [SOW 84] -is given. The blocks denote the concepts while the circle denotes the conceptual relation. Figure 2 may be read as "a cat sitting on a mat". The concepts (objects) are cat and mat while the conceptual relation is sitting. The deviations of the conceptual graphs as applied in FROOM compared to Sowa's original approach [SOW 84] are the following [SAL 94]:

1 Only assemblies, components, features and faces can be concepts that are shown in the graph. Attributes for instance, are part of the object oriented descriptions of concepts and relations and are usually not shown in the graph. That is, unless the constraint network of parameter constraints is shown.
2 The relations used in FROOM are only monadic, dyadic or tri-adic.
3 Directed arcs - as in fig.2 - are only used when necessary. Often relations just denote a general connection relation like "component A is connected to component B". Usually also "component B is connected to component A" holds equally well, so that arcs could have been directed in both directions. In these cases directed arcs are omitted.
4 In the graphical notation of concepts and relations, sometimes other symbols are used than the small rectangles or circles.

8.1.4 Organization

The organization of the remainder of this chapter is as follows. Section 2 briefly summarizes related literature. Section 3 elaborates on the design of the interactive feature definition module in FROOM. Section 4 describes the present implementation of this module in FROOM. Finally, in section 5 conclusions and recommendations are presented.

8.8.2. RELATED LITERATURE

Feature definition and feature representation are two closely related subjects. Feature definition is considered here as defining how a feature should look like, how it should behave, what its characteristics are etc. Feature representation is regarded as how all this feature information is represented computer internally. Feature representation is defined here as representing both the individual (form) features and the relations between the individual features that constitute a component. Often the way features are represented, determines the way in which they can be defined, and therefore these two subjects are often used interchangeably.

Pratt proposed a non-manifold feature representation based on the belief that B-rep is the preferable solids representation scheme, and that features should be volumetric [PRA 88], [PRA 90]. Pratt also introduced the notion of implicit and explicit feature representations. In the explicit representation, all geometric details of the feature are fully defined. In the implicit representation sufficient information is supplied to define the feature, but the full geometric details have to be calculated when required.

Shah et al. used a hybrid CSG/B-rep feature representation scheme for their feature based modelling system [SHA 88]. Wang and Ozsoy also propose a hybrid representation scheme [WAN 91]. Joshi and Chang proposed the use of an attributed adjacency graph for feature representation [JOS 90]. Fu et al. propose a graph grammar for feature representation [FU 93]. Van Emmerik used a CSG approach based on half spaces [EMM 90]. A more detailed discussion of these and other feature representation approaches will not be considered here due to space limitations. However, there are 4 approaches that do require more discussion. These are the approaches by Billo et al. [BIL 89a,b], Sheu et al. [SHE 93], Duan et al. [DUA 93] and Sreevalsan et al. [SRE 91].

Billo et al. would like to bring some more rigour into feature representation by using conceptual graphs [BIL 89a,b]. By doing this, feature recognition and feature mapping or conversion would be made easier. A simple example of feature transformation to Group Technology coding is given. The feature representation used by Billo et al. - in case of prismatic features - consists of face concepts and adjacency relations between them. The face concepts are part of the general feature concept. However, it is not fully clear whether other relations between faces are also possible such as for example perpendicular, concave, convex, parallel etc. Other features are defined using conceptual graphs for sweeps and ruled surfaces. It is not clear in what respect the concepts have been worked out in order to realize interactive feature definition.

Sheu et al. present a representation scheme that is suitable for representing and operating form features [SHE 93]. Five basic constituents, B-rep solid components, measure entities, size, location, and constraints are used to represent a single form feature. However, although the concepts seem promising, an interactive feature definition does not seem to have been realised.

Duan et al. employ generalized sweeping techniques for feature definition [DUA 93]. A feature is defined as a parametric shape-unit. It consists of a geometric description, attributes and application oriented mapping methods for design and manufacturing purposes. The geometry of a 3D feature is defined by sweeping a 2D shape along a guideline, which is variational-geometric and dimension driven. A 2D shape is sketched interactively first, and then the dimensions of the draft and the geometric constraints are determined. Then, the sweeping pattern is specified. The 3D geometry that is generated in this way is well parameterized and dimension driven. The solid model representation that Duan et al. use is a hybrid B-rep/CSG representation scheme.

A unification of form feature definition methods for integrating feature based design, automatic feature recognition and interactive feature identification has been discussed in [SRE 91]. Features for use in design are written in terms of their generic properties, construction procedures etc. Feature templates are used to interactively identify features. A set of standard functions is used to synthesize feature recognition algorithms in order to formalize features to be recognized. An interpreted feature language has been developed for expressing both the feature templates and recognition algorithms. Some problems were noted with regard to the unification of the three modes of feature definition. These problems are related to the lack of dynamic interfaces to geometry modellers and the restrictive access to the functions and data of geometric modellers. According to Sreevalsan et al., the problems can be alleviated by the use of open architecture geometric modelling kernels like ACIS^{TM}.

8.3. DESIGN OF THE INTERACTIVE FEATURE DEFINITION MODULE

FROOM has been designed using a methodic design approach which has been described in [SAL 93c] and [SAL 93d]. Therefore, the design of the Interactive Feature Definition (IFD) module of FROOM was performed employing a methodic design approach. The first "iteration" in this design approach has been described by Jonker in [JON 93]. The following sections are mainly based on Jonker's description of the design of the Interactive Feature Definition (IFD) module. However, some extensions and improvements to Jonker's design are also described in the following sections. These more or less represent a second "iteration" in the design process.

8.3.1 Problem definition

The problem can be defined as the design and implementation of a capability to interactively define features. The requirements to be met are the following:

1 interactive form feature definition (as little programming as possible).
2 geometry and non-geometry definition (constraints, tolerances, materials etc.).
3 definition of the behaviour of the feature.
4 except from design purposes, the feature definition functionality must be of use for recognition purposes.
5 feature description should be convertible into a standard format like STEP.

Besides these requirements, for implementation sake, it would be desirable to correspond with the FROOM data structures (conceptual graphs) and modeller (ACIS^{TM}). Also, it would be desirable if the system would allow extension to the definition of abstract features.

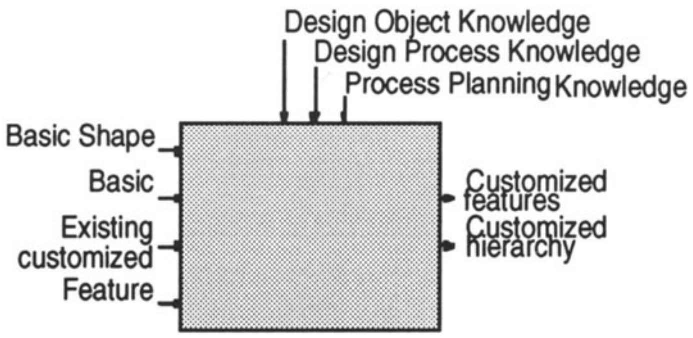

Figure 3. Black box description of the interactive feature definition functionality.

Looking at the feature definition module as a black box, it would look something like fig. 3. Before detailing the functionality of this black box, we will first consider its incoming and outcoming characteristics; the inputs and the outputs. Incoming characteristics are the basic shape elements and the basic (standard) features which can be used to define new features. Also use can be made of existing previously defined features and the feature hierarchy (assuming an object oriented application). The system manager has the responsibility to define features that make sense: the features should be of use in either CAD or CAPP applications or both. The system manager brings in design object knowledge, design process knowledge, and process planning knowledge that is related to the feature to be defined. Outcoming characteristics are the newly defined features and an updated feature hierarchy.

8.3.2 Conceptual design of the IFD module

The conceptual design mainly focuses on detailing the black box of fig.3 into functional modules and choosing operating principles for these subfunctions. The decomposition that has been derived in [JON 93] is shown in figure 4.

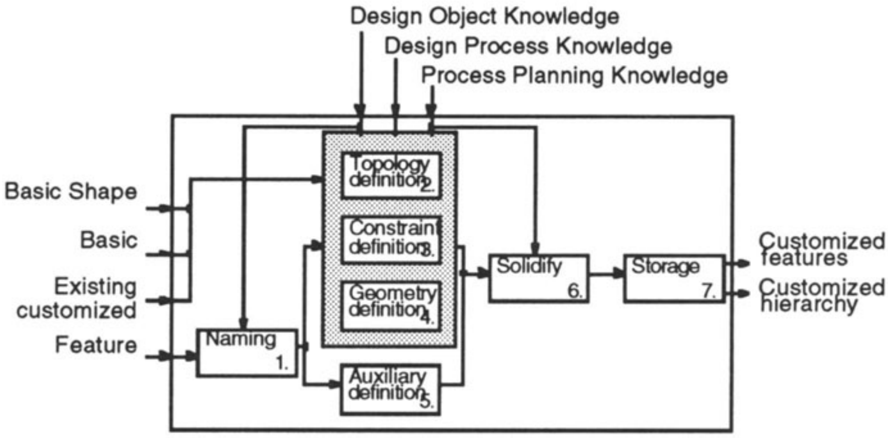

Figure 4. Functional decomposition of the black box description [JON 93].

Although it can sometimes be questionable whether the sequence of subfunctions as given in figure 4 is correct, the subfunctions as such will have to be performed in the feature definition process. Therefore, these subfunctions will be elaborated upon in the following.

Naming. The feature to be defined has to be given a proper name to be useful and to be able to identify it later. This holds especially for feature based design purposes. Naming includes placing the feature somewhere in the feature hierarchy and thus inheriting properties of the parent feature.

Topology, constraint and geometry definition. The definition of topology, constraints and geometry are closely related and have therefore been assembled in the grey block in fig. 4. Topology is the number and types of feature elements (faces, edges and vertices) and how they relate to each other. Geometry determines the actual shape, the dimensions. Constraints are regarded as desired relationships between two or more objects. The following constraint types are discerned: feature internal constraints, feature external constraints and feature tolerance constraints.

Feature internal constraints that were seen as useful are: adjacent (convex, concave and tangential), parallel, perpendicular and co-axial [JON 93]. These constraints merely address feature topology. Feature internal constraints, however, also can address feature dimensions (geometry). These are important in order to give the end-users useful parameters with which they can control feature size. Feature external constraints have not been addressed in [JON 93]. These constraints, however, were addressed in [SHE 93]. Feature external constraints are important for instantiating features and for defining their behaviour. Usually, feature external constraints can only be defined after a significant portion of the geometry and the topology of the feature have been defined. Therefore, feature external constraint definition could have been added separately in figure 4, after the solidify function. Also tolerance constraints (feature internal variational geometry constraints) are important for defining the feature's default tolerances. The latter constraint type will not be addressed further in this chapter as it is part of the tolerance specification functionality of FROOM which is now being developed and on which a separate article will appear.

Auxiliary definition. This comprises defining all other non-geometry related information such as affinity relations that features may or may not have with other features or components, icons, help texts, manufacturing information etc.

Solidify. Making a solid model of the feature elements.

Storage. Storing the defined feature in the right format.

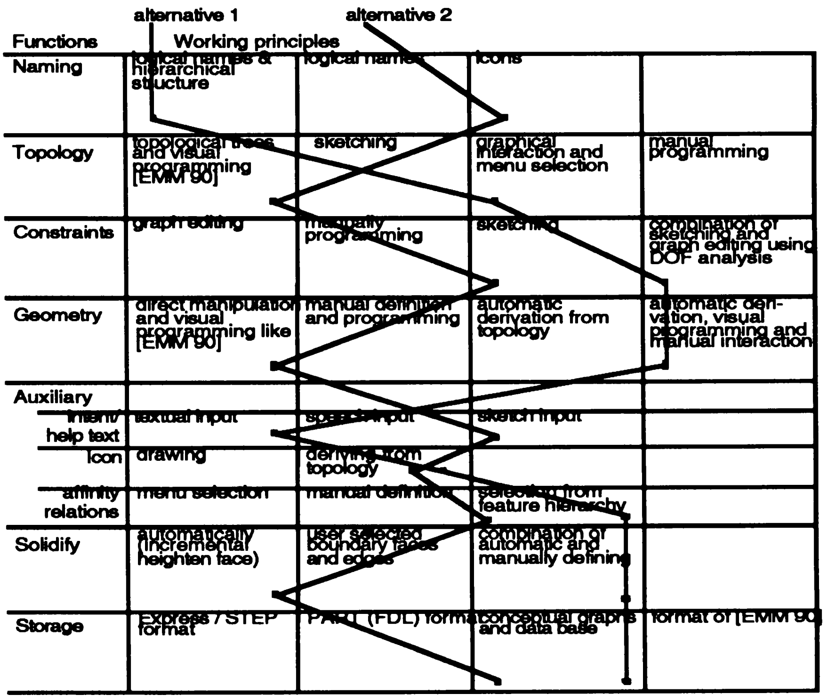
alternative 1
alternative 2
Functions
Working principles
Naming
logical names & hierarchical stucture
logical names
icons
Topology
topological trees and visual programming [EMM 90]
sketching
graphical interaction and menu selection
manual programming
Constraints
graph editing
manually programming
sketching
combination of sketching and graph editing using DOF analysis
Geometry
direct manipulation and visual programming like [EMM 90]
manual definition and programming
automatic derivation from topology
automatic derivation, visual programming and manual interaction
Auxiliary
intent/ help text
textual input
speech input
sketch input
Icon
drawing
deriving from topology
affinity relations
menu selection
manual definition
selection from feature hierarchy
Solidify
automatically (incremental heighten face)
user selected boundary faces and edges
combination of automatic and manually defining
Storage
Express / STEP format
PART (FDL) format
conceptual graphs and data base
format of [EMM 90]

Figure 5. Morphological overview for interactive feature definition subfunctions: alternative 1 has finally been chosen [JON 93].

In [JON 93], for each of the above mentioned subfunctions (except the feature external constraints), an "operating principle" has been chosen. This is summarized in fig. 5, indicating a morphological overview with all the previously denoted subfunctions together with possible ways to fulfil them. Fig. 5 also shows two alternatives that were considered (no. 1 and no.2). Alternative 1 has finally been chosen. In the following this alternative will be explained further.

For the feature representation conceptual graphs have been chosen. The reasons were the following. First, conceptual graphs allow for a representation that is both relatively easy to understand by humans and can easily be processed by computer [BIL 89b]. Another reason was the already existing incorporation within FROOM of conceptual graphs for assembly representation [SAL 94]. Furthermore, conceptual graphs do not significantly differ from the attributed adjacency graphs and other graphs commonly used for feature representation. The choice for using conceptual graphs has been influenced by its ability to represent the different kinds of feature constraints by means of conceptual relations.

For the naming function it was chosen to use logical names (responsibility of the system manager) and a hierarchical structure providing for inheritance.

For defining topology, constraints and geometry, it was chosen to use sketching in combination with (conceptual) graph editing and feature identification on a solid model. By sketching, geometry and topology can be defined easily and quickly. During sketching, the system should automatically make assumptions on the feature internal constraints intended by the system manager. At present the following feature internal constraints are anticipated: adjacent (convex, concave, tangential), parallel, perpendicular and co-axial. The set of feature internal constraints can easily be extended. The common denominator of the feature internal constraints is that they generally denote face associations or associations between edges and faces, edges and edges etc. These constraints become part of the conceptual graph: they are represented as conceptual relations. Sometimes, however, it might be necessary that the system manager operates on the conceptual graph directly, e.g. when the system has assumed the wrong constraint or when additional constraints have to be added in order to allow feature recognition not to fail when some parts of the feature are missing due to feature intersection. Identification of feature elements on a solid was perceived as a third approach of feature definition, especially interesting in case of CAPP when certain features are not recognized on a solid.

The feature external constraints are similar to the feature internal constraints: they generally denote face associations and dimensional constraints related to these face associations. Most of the geometry related constraints - also the feature internal and external constraints - are designed to be managed in a unified way in FROOM. This unified geometry constraint management in FROOM is a combination of degrees of freedom analysis as proposed by Kramer [KRA 92], Liu and Nnaji [LIU 91] and face association domain knowledge, similar to the approach applied to tolerances by Clément et al. [CLE 93]. This subject is part of FROOM's constraint management module. Therefore, it will not be elaborated here in detail; the subject of constraint management will be elaborated in a future article.

For the auxiliary feature definition functionality a number of possibilities were considered. Defining the intent of a feature was seen to be most suitable by textual input. Sometimes icons may be required for the features that are defined; this can be done by deriving icons automatically from the previously defined topology and geometry or by drawing icons manually with some computer tool. Affinity relations can best be selected from a feature hierarchy.

Solidifying can be performed automatically, using some of the ACIS^{TM} functions like incrementally heighten a face or by manually selecting the elements of a feature and "stitching" them together to form a solid. The combination of these two methods was seen as most favourable.

Storage of a feature can be performed in a number of ways (see fig.5). It was chosen to store the features as conceptual graphs in the first place as the graphs represent the internal, modeller independent, feature representation. Moreover, modification of a feature depends on having a conceptual graph of the feature. Other storage or representation formats can be derived from the conceptual graph format. A database seems most suitable to store a conceptual graph.

8.3.3 Embodiment and detail design of the IFD module: a session of the IFD module

How topology, constraint and geometry definition are envisaged from a system manager user perspective will be detailed in the following three paragraphs. Each paragraph focuses on one of three envisaged modes of feature definition: by sketching, graph editing and identification on a solid. Special attention is given to the design of the user interface that is to be provided to the system manager user.

8.3.3.1 Defining a new feature by sketching

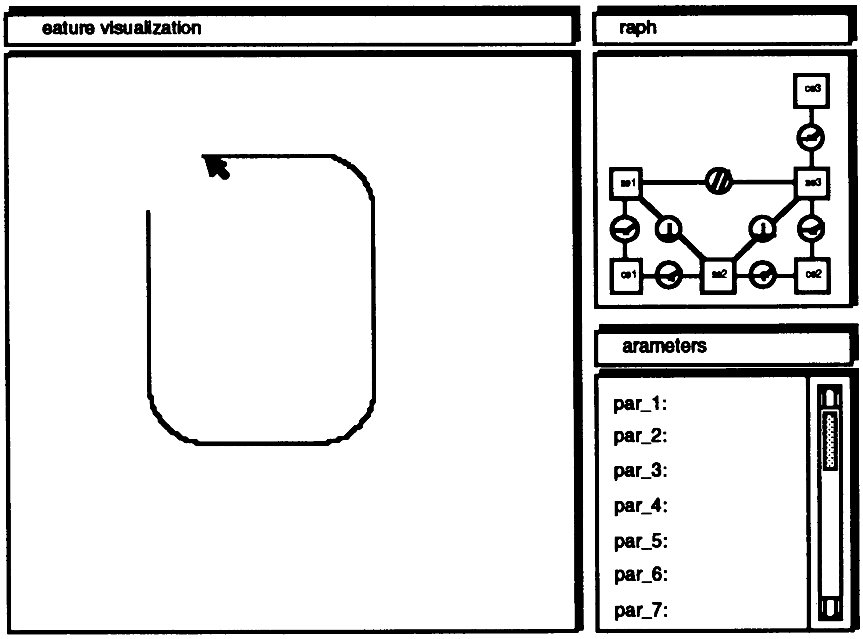

Figure 6. User interface design for defining a new feature by sketching.

In [JON 93] sketching was envisaged as a means for defining features. The system would then have to infer geometry, topology and constraint information automatically. In [JON 93] the sketching of explicit features has been stressed, i.e. sketching each face explicitly. However, sketching of implicit features would give even more advantages. Conventional CAD techniques like sweeping could then be used. Fig. 6 shows an example of a user interface that has been designed for feature definition employing a sketching mode. This user interface design was inspired by the existing FROOM user interface (see e.g. [SAL 93a,b]. The left side of fig. 6 shows the workspace while the right shows the corresponding graph which should be inferred automatically from the sketch. The degrees of freedom of the entities in the sketch can be kept by a degrees of freedom approach similar to Kramer [KRA 92] or to Arbab et al. [ARB 89] who addressed the 2D case.

8.3.3.2 Defining / modifying a new feature by graph editing

If the feature defined by sketching would need some changes or if the sketch capabilities are not sufficient, a feature can be defined or modified by conceptual graph editing. A design of a user interface for this mode of feature definition is shown in fig. 7. The work area is now on the right in which graphs can be edited. In the area on the left, geometry is automatically derived, based on the graph that is defined. The degrees of freedom have to be kept up to date when doing this in order to allow for changing from one mode of feature definition to the other while keeping the underlying feature description consistent.

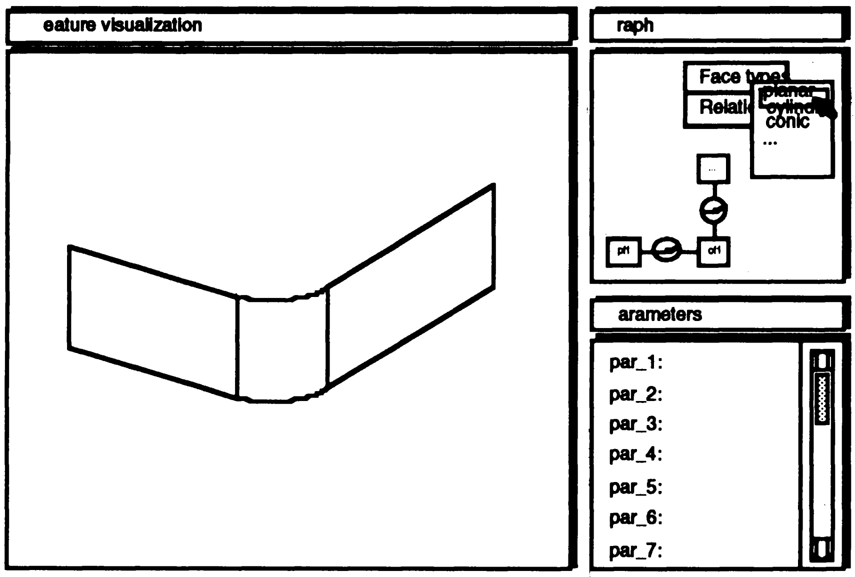

Figure 7. Design of a user interface for defining a new feature by graph editing.

8.3.3.3 Defining a feature by identifying edges and faces from a solid

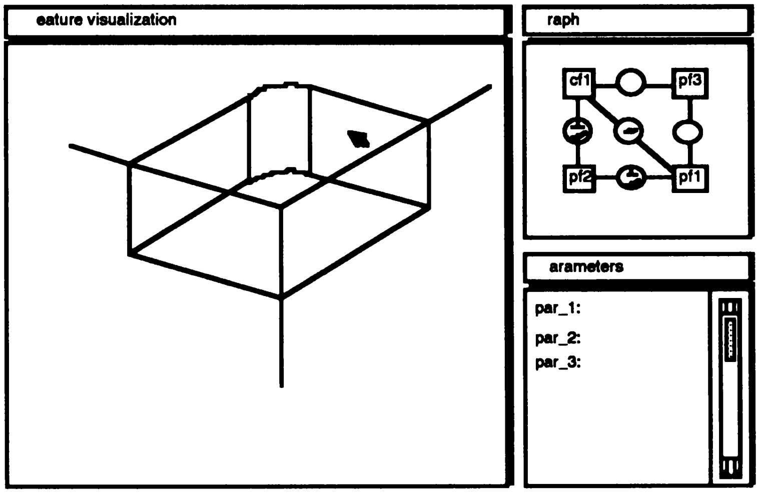

Figure 8. Design of a user interface for defining a new feature by identifying faces, edges etc. in a solid.

In CAPP it is possible that not all features of a product model are recognized directly. Because there is a product model with topology, geometry and implicit constraints, it makes no sense to sketch or edit a conceptual graph of the feature from scratch. It is possible that the features that were not recognized can be identified interactively by picking their faces, edges etc. A conceptual graph representation of the feature could then automatically be derived, making use of routines that are usually used in feature recognition like detecting convex/concave edges. A design of a user interface for this mode of feature definition is shown in fig. 8. The graph representation of the feature on its turn could subsequently be transformed into a feature recognition language.

8.4. IMPLEMENTATION OF THE INTERACTIVE FEATURE DEFINITION MODULE

In FROOM both the concepts and the conceptual relations of the conceptual graphs are programmed as C++ objects. The concepts that are currently implemented as part of the interactive feature definition module are planar face, cylindrical face and conical face. For each of these concepts, a class definition has been programmed from which the system manager-user can make his own instances. This also holds for the relations; currently implemented relations are adjacent (convex, concave or tangential are attributes of the relation), perpendicular, parallel and co-axial. As these relations can associate different concept types, e.g. planar face adjacent to planar face or planar face adjacent to cylindrical face, many different conceptual relations have to implemented. Presently, only the most common relations have been implemented. As for the concepts, the relation objects contain attributes and methods. The methods of the relation objects take care of instantiating relations and of deleting them. They are also used for updating the degrees of freedom of the concepts that are part of the association.

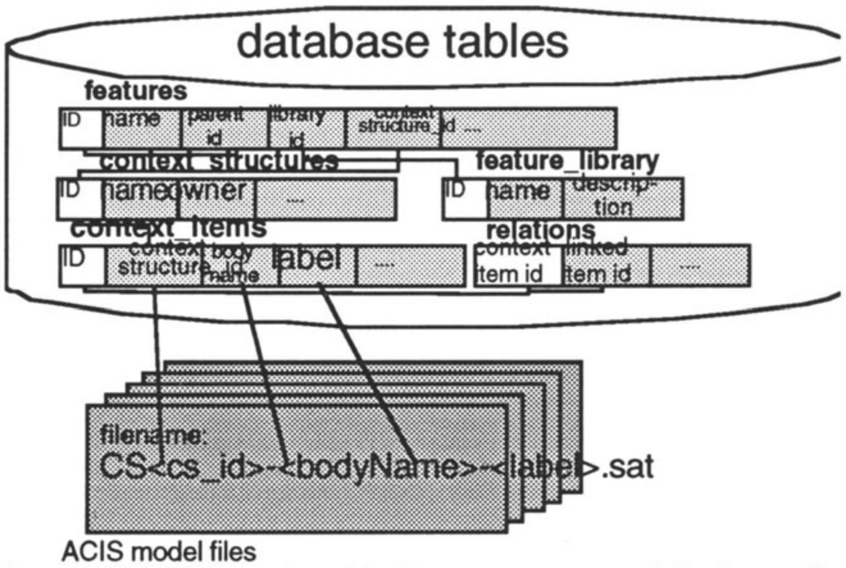

Figure 9. Simplified overview of the data structure as currently implemented.

The conceptual graphs in FROOM are stored in a relational database (presently Oracle^{TM} is used). The ACIS^{TM} modeller stores its models in files, however. Therefore, the features that have been defined in the interactive feature definition module are stored in a hybrid way using both a database (for the modeller independent conceptual graphs) and the files that contain the modeller dependent geometric information. Fig. 9 shows a somewhat simplified overview of the data structure that has been implemented.Fig.10 gives an example of the user interface that has been realized for the graph editing mode of feature definition, according to the design shown in fig.7.

Fig. 10 shows how features are currently interactively defined by the system manager. This is done by defining a conceptual graph representing the feature. The user instantaneously also gets sample geometry in the geometry window. This is considered to be very important as system manager users usually will have geometry in their mind of which the graph is an abstraction. At the moment it is possible to define relatively simple features, like a pocket rectangular straight within a few minutes using the graph editing approach. Using a sketching mode of feature definition the time needed to define a feature could be reduced further.

Note from fig. 10 that the pocket feature has been defined redundantly: pf#2 (planar face no. 2) is perpendicular to pf#3 which is perpendicular to pf#4, which is the same as pf#2 being parallel to pf#4. The reason why redundant feature definitions are allowed, is that in case of recognizing a feature based on a redundant feature definition, also possible intersections should be taken into account. In case of intersections, one or more of the feature's faces could become deleted as a result of which the previously redundant relations may be required in order to unambiguously define (recognize) the intersected feature.

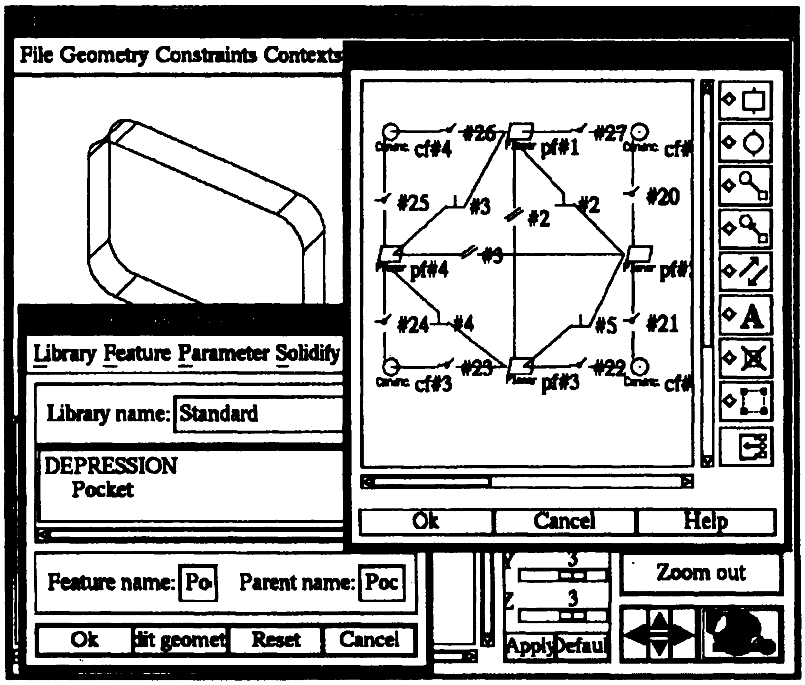

Figure 10. Feature definition employing graph editing as currently implemented in FROOM.

In the future the feature definition user interface will have to be extended to enable direct geometric input, by means of sketching, followed by automatic conceptual graph generation. This would also require a significant amount of intelligence built in into the feature definition module, which at the moment is not yet present. Also, feature definition by identification of faces and edges has not yet been implemented.

Future extensions will have to be made to the available concepts. For instance the definition of free-shaped features (sweeps, ruled surfaces etc.) and abstract features will have to be accounted for. The mechanism of managing the feature internal and external constraints also has not been fully implemented. The transformation of the feature graph to a standard format like STEP also has not yet been achieved. Presently work is starting in converting the conceptual graph of the feature to the feature recognition language of PART.

8.5. CONCLUSIONS AND RECOMMENDATIONS

The focus of this chapter has been on the interactive feature definition module in a re-design support tool, called FROOM. The feature definition is based on a conceptual graph representation of features. The design of the interactive feature definition module allows for defining features based on sketching and by graph editing. Also identifying faces on a product model is an option, especially suitable for CAPP systems in order to define features which have to be recognized. Presently implemented is the graph editing mode of feature definition. More work is required in defining features by means of sketching. Also work needs still to be done toward transforming the conceptual graph into a feature recognition language or STEP format. More intelligence could be brought in into the feature definition module.

6. ACKNOWLEDGEMENTS

This research is supported by the Technology Foundation (STW).

7. REFERENCES

[ARB 89] Arbab F., Wang B., A constraint-based design system based on operational transformation planning, proceedings of the 4th. International Conference on Applications of Artificial Intelligence, Gero J. (ed.), Springer Verlag, 1989, 405-426.

[BIL 89a] Billo R.E., Henderson M.R., Rucker R., Applying conceptual graph inferencing to feature-based engineering analysis, Computers in Industry, vol. 13, 1989, 195-214.

[BIL 89b] Billo R.E., An object-oriented modeling methodology utilizing conceptual graphs for form feature definition, PhD thesis, Arizona State University, 1989.

[CLE 93] Clément A., Rivière A., Tolerancing versus nominal modelling in next generation CAD/CAM system, CIRP Seminar on Computer Aided Tolerancing, ENS de Cachan, Paris, April 1993, 97 - 113.

[DUA 93] Duan W., Zhou J., Lai K., FSMT: a solid modelling tool for feature-based design and manufacture, cad, vol. 25, no. 1, 1993, 29 - 38.

[EMM 90] Emmerik M.J.G.M. van, Interactive design of parameterized 3D models by direct manipulation, PhD thesis, Delft University, The Netherlands, 1990.

[FU 93] Fu Z., De Pennington A., Saia A., A graph grammar approach to feature representation and transformation, International Journal of Computer Integrated Manufacturing, Vol.6, No.1, 1993, 137-151.

[HOU 91] Houten F.J.A.M van, PART, a computer aided process planning system, PhD thesis, University of Twente, Enschede, 1991.

[HUM 89] Hummel K.E., Brown C.W., The role of feature in the implementation of concurrent product and process design, ASME winter meeting, San Francisco, 1989, 1-9.

[JON 93] Jonker H.G., Interactive feature definition, MSc thesis, University of Twente, report no. SPA-93-23/PT465, Enschede, September, 1993.

[JOS 90] Joshi S., Chang T.C., Feature extraction and feature based design approaches in the development of design interface for process planning, Journal of Intelligent Manufacturing, Vol.1, 1990, 1-15.

[KRA 92] Kramer G., Solving Geometric Constraint Systems, a case study in kinematics,The MIT Press, Cambridge, Massachussetts, London, England, 1992.

[LIU 91] Liu H-C., Nnaji B.O., Design with spatial relations, Journal of Manufacturing Systems, Vol.10, No. 6, 1991, 449-463.

[PRA 88] Pratt M.J., Synthesis of an optimal approach to form feature modelling, Computers in Engineering Conference, 1988, 263-273.

[PRA 90] Pratt M.J., A hybrid feature based modelling system, Advanced geometric modelling for engineering applications, eds. Krause F.L., Jansen H., Elsevier Science Publishers, 1990, 189-210.

[SAL 93a] Salomons O.W., Houten F.J.A.M. van, Kals H.J.J., Review of research in feature-based design, Journal of Manufacturing Systems, Vol.12, No.2, 1993, 113-132.

[SAL 93b] Salomons O.W., Slooten F. van, Houten F.J.A.M. van, Observations from redesign and process planning practice, a study of human practice aimed at improving feature based CAD/CAPP, proceedings of MICAD, Vol. 1, 9-12 February, Paris, 1993, 115-129.

[SAL 93c] Salomons O.W., Kappert J.H., Slooten F. van, Houten F.J.A.M. van, Kals H.J.J., Computer support in the (re)design of mechanical products, a new approach in feature based design, focusing on the link with CAPP, Proceedings of KNOWHSEM '93, IFIP WG 5.3, North Holland, Budapest, April 20-22, 1993.

[SAL 93d] Salomons O.W., Slooten F. van, Houten F.J.A.M. van, Kals H.J.J., A computer support tool for re-design, a prototype system resulting from applying a methodic design approach, proceedings of the International Conference on Engineering Design, ICED'93, The Hague, August 17 - 19, 1993, Vol. 3, 1559 -1570.

[SAL 94] Salomons O.W., Slooten F. van, Koning G.W.F. de, Houten F.J.A.M. van, Kals H.J.J., Conceptual graphs in CAD, acc. for publ. in CIRP Annals V.43/1 and for presentation at CIRP general assembly, Singapore, August 1994.

[SHA 88] Shah J.J., Rogers M.T., Expert form feature modelling shell, CAD, vol. 20, no.9, 1988, 515-524.

[SHA 90] Shah J.J., Philosophical development of form feature concept, CAM-I report P-90-PM-02, 1990, 55-70.

[SHE 93] Sheu L-C., Lin J.T., Representation scheme for defining and operating form features, CAD, volume 25, no. 6, 1993, 333-347.

[SOW 84] Sowa J.F., Conceptual Structures, information processing in mind and machine, Addison-Wesley Publishing Company, 1984.

[SRE 91] Sreevalsan P.C., Shah J.J, Unification of form feature definition methods, IFIP WG 5.2 workshop on Intelligent CAD, Columbus, Ohio, September, 1991.

[WAN 91] Wang N., Ozsoy T.M., A scheme to represent features, dimensions and tolerances in geometric modelling, Journal of Manufacturing Systems, Vol. 10, no.3, 1991, 233-240.

[WIN 91] Wingard L., Introducing form features in product models, a step towards CAD/CAM with engineering terminology, Licentiate thesis, Royal institute of Technology, Stockholm, 1991.