Proceedings of DETC 2003
2003 ASME Design Engineering Technical Conferences
September 2-6, 2003, Chicago, Illinois USA

DETC2003/CIE-48197

BOUNDARY EVALUATION FOR A CELLULAR MODEL

Rafael Bidarra, Willem J. Neels and Willem F. Bronsvoort
Computer Graphics and CAD/CAM Group
Faculty of Information Technology and Systems
Delft University of Technology
Mekelweg 4, NL-2628 CD Delft, The Netherlands
Email: (Bidarra/Bronsvoort)@its.tudelft.nl

ABSTRACT

Feature modeling systems usually employ a boundary representation (b-rep) to store the shape information on a product. It has, however, been shown that a b-rep has a number of shortcomings, and that a cellular model can be a valuable alternative. A cellular model stores additional shape information on a feature, including the faces that are not on the boundary of the product. Such information can be profitably used for several purposes.

A major operation in each feature modeling system is boundary evaluation, which computes the geometric model of a product, i.e. either the b-rep or the cellular model, from the features that have been specified by the user. Because it has to be executed each time a feature has been added, removed or modified, its efficiency is very important.

In this paper, boundary evaluation for a cellular model is described. Subsequently, its efficiency is compared to the efficiency of boundary evaluation for a b-rep, on the basis of performance measurements and considerations for both. It turns out that boundary evaluation for a cellular model is in fact more efficient than for a b-rep, which makes cellular models even more attractive as an alternative for b-reps.

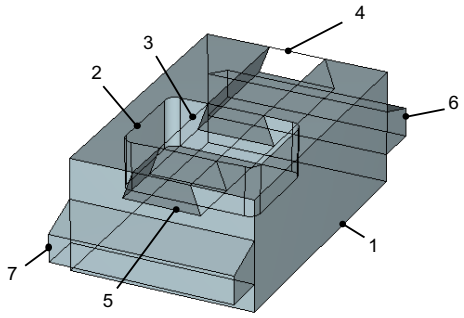
1 INTRODUCTION

Feature modeling is now the predominant approach for product modeling. Shape and related functional information can be stored in a single product model, which can considerably improve the product modeling process [1].

Most commercial feature modeling systems employ a boundary representation (b-rep) as geometric model to store the shape information. B-reps have been successfully used in traditional geometric modeling systems to support specification, visualisation and analysis of a model, but they are in fact not powerful enough to store all shape information that is relevant in a feature modeling context. Therefore, cellular models have been suggested as an alternative for b-reps. In a cellular model, more relevant shape information on features can be stored, e.g. not only the faces of the features that are on the boundary of the product, but also the faces that are not on the boundary; see [2] for an overview of several proposals.

The cellular model introduced by the Computer Graphics and CAD/CAM Group of Delft University of Technology [2], for example, has been profitably used for several purposes. First, it allows the semantics of features to be specified, and the validity of a feature model to be maintained, which is essential to make feature modeling really more powerful than geometric modeling [3]. Second, more functional information of a feature model can be visualised, e.g. the above-mentioned faces of features that are not on the boundary of the product, which can be very helpful during specification and analysis [4]. Third, it can serve as one of the main data structures for a multiple-view feature modeling system, which can support the integration of several product development phases [5].

Boundary evaluation is the process that computes the geometric model of the product, i.e. either a b-rep or a cellular model, on the basis of the set of features that has been specified by the user. It is a major operation in a feature modeling system, and, because it has to be executed each time a feature



cell 1 - <block>
cell 2 - <block, roundedRectPocket>
cell 3 - <block, throughSlot, roundedRectPocket>
cell 4 - <block, throughSlot>
cell 5 - <block, throughSlot>
cell 6 - <ribBack>
cell 7 - <ribFront>

Figure 1. Cellular model of a part and owner lists of its cells

has been added, removed or modified, its efficiency is of utmost importance.

In this paper, boundary evaluation for a cellular model is described in some detail. On the basis of the higher complexity of a cellular model compared to a b-rep, one can legitimately ask how efficient boundary evaluation for a cellular model is, compared to boundary evaluation for a b-rep. The main goal of this paper is to answer this question, by giving performance figures of, and considerations on, both types of evaluation. Performance of boundary evaluation for a b-rep has been measured with the commercial feature modeling system Pro/ENGINEER [6], and for a cellular model with SPIFF [3], a prototype feature modeling system developed at our group.

In Section 2, the cellular model of SPIFF and the basic operations on it are described. In Section 3, boundary evaluation for both b-rep and cellular model are discussed. In Section 4, the outcome of performance measurements for the two types of boundary evaluation is presented. In Section 5, the results are interpreted, and conclusions on boundary evaluation for the cellular model are given.

2 CELLULAR MODEL

In this section, the cellular model used in the SPIFF system is described, with a special emphasis on its functionality to modify model topology [2].

Basic notions

The cellular model is a non-manifold geometric representation of the feature model of a part, and integrates the contributions from all its volumetric features. It represents a part as a connected set of volumetric quasi-disjoint *cells* of arbitrary shape, and represents each feature as a connected subset of these cells. The cellular subdivision is determined by the property that two cells may never volumetrically overlap. So, whenever two features overlap, their cells are such that one or more cells are shared by the two features, and the remaining cells lie in either of them.

Any two adjacent cells are separated by a number of interior faces in the cellular model. Such faces can be regarded as having two 'sides', designated as partner *cell faces*. A face that lies on the boundary of the cellular model has only one cell face (one 'side'), that of the only cell it bounds. In either case, a cell face always bounds one and only one cell. As a consequence of this cell subdivision, each feature face is represented by a connected set of cell faces. See Figure 1 for an example of a cellular model containing some overlapping features.

To identify and analyze features in the cellular model, each cell has as attribute an *owner list* indicating which features it belongs to (see again Figure 1). Similarly, each cell face has an owner list indicating which feature faces it belongs to. Finally, the nature of a cell expresses whether its volume represents 'material' of the part or not, and, similarly, the nature of a cell face expresses whether it lies on the boundary of the part or not.

So the cellular model contains much more information than only the model boundary of the part. In particular, it contains explicit information on the 'not on boundary' faces of subtractive features and on intersecting features. The cellular model, including its attribute mechanism to maintain the owner lists of cells and cell faces, has been implemented in the SPIFF system using the Cellular Topology Component of the ACIS geometric modeling kernel [7].

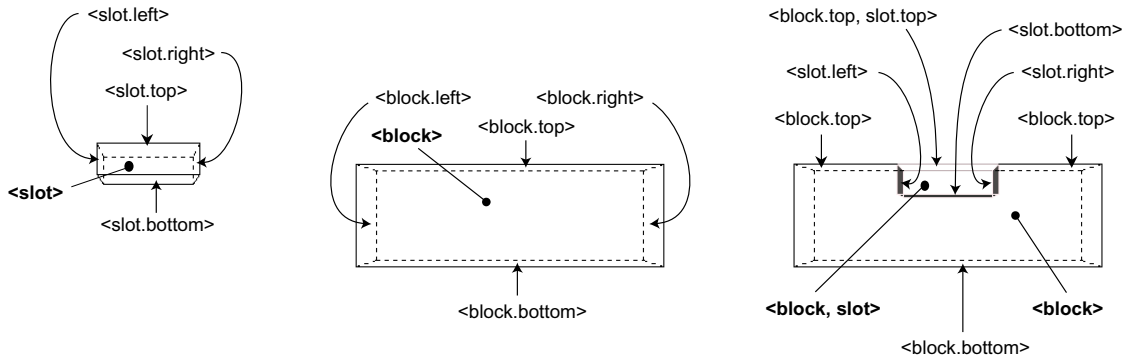
Two basic operations are defined that modify the cellular model: adding a new feature shape to the cellular model, and removing an existing feature shape from the cellular model. The effect of these operations is twofold: (i) they change the topology of the cellular model, and (ii) they update the owner lists of its cellular entities accordingly. Both aspects are described and illustrated for the two operations in the following subsections.

Adding a feature shape to the cellular model

The goal of this operation is to add the imprint of a new feature shape to an existing cellular model.

In the first stage, the new feature is represented by a one-cell shape, which is dimensioned and positioned relative to the cellular model according to the set of feature parameters. In the owner list of this cell, only a reference to the new feature is contained, whereas the owner list of each of its cell faces contains a reference to the respective feature boundary element. This is illustrated in Figure 2(a) for a rectangular slot feature.

In the second stage, a *non-regular cellular union* operation is performed, between the cell representing the new feature and the cellular model. This set operation computes the cellular de-



(a) slot cell before the operation (b) cellular model before the operation (c) cellular model after the operation

Figure 2. Adding a feature to the cellular model

composition described in the previous subsection, and changes the owner lists of the relevant cells and cell faces, e.g. whenever these are split, so that each entity always 'knows' precisely which feature shapes, or feature faces, it belongs to.

During a non-regular cellular union, first every cell (C_m) of the cellular model is identified that somehow intersects the new cell (C). Mutual cellular decomposition is then carried out between C and each C_m. This may occur in two ways:

(i) The two cells intersect only over their boundaries, in which case there are no new cells created; instead, their overlapping cell faces are decomposed, yielding partner cell faces that lie inside the boundary intersection (i.e. bounding both cells) and cell faces that lie outside it (i.e. bounding only one of the cells). Split cell faces get as owner list that of the cell face from which they originate.

(ii) The two cells volumetrically overlap, in which case the decomposition results in new cells that lie either inside the intersection or outside it. Mostly, a subset of the boundary of the two cells is also decomposed (except when one of the cells lies entirely inside the other), yielding cell faces that lie either on the intersection or outside it. Whenever two cells undergo this mutual decomposition, the owner lists of the new entities are determined as follows:

   (a) a new cell that lies in the intersection of C and C_m gets as owner list the union of the owner lists of C and C_m;
   (b) the other cells resulting from the decomposition get as owner list that of the respective cell from which they originate (either C or C_m);

   and, analogously:

   (a) a new cell face lying on the boundary of both C and C_m gets as owner list the union of the owner lists of the overlapping cell faces from which it originates;
   (b) a new cell face lying on the boundary of either C or C_m gets as owner list that of the respective cell face from which it originates;
   (c) the remaining new cell faces get an empty owner list.

Figure 2(c) illustrates both the cellular decomposition and the owner list propagation after the non-regular cellular union between the slot feature in Figure 2(a), and a cellular model consisting of a single block (see Figure 2(b)), which is an example of case (ii) mentioned above. The owner lists of both cells before the operation are also shown in Figures 2(a) and (b), though for the sake of legibility, only some cell face owner lists are depicted. After the operation, the block cell has been decomposed into two cells, of which one is shared with the slot. The owner lists of the two new cells, as well as the owner lists of some cell faces, are shown in Figure 2(c).

Removing a feature shape from the cellular model

The goal of this operation is to completely remove from the cellular model the imprint of a feature. This operation comes down to a selective removal and merge of topologic entities in the cellular model, and is conceptually much simpler than any conventional Boolean operation. It is carried out in three stages, all of which are confined to the set of cells owned by the feature to be removed.

In the first stage, these cells are walked through in order to remove from their owner lists all references to that feature. Similarly, all references to faces of that feature are removed from the owner lists of the cell faces bounding those cells.

In the second stage, the same set of cells is searched for cells exhibiting an empty owner list. Such cells are removed from the cellular model, which is easily accomplished by removing all one-sided faces bounding them.

In the third stage, each of the remaining cells in the same set is analyzed. Whenever it exhibits the same owner list as one of its adjacent cells, they are merged, which is easily accomplished by removing all two-sided faces that separate them. After all

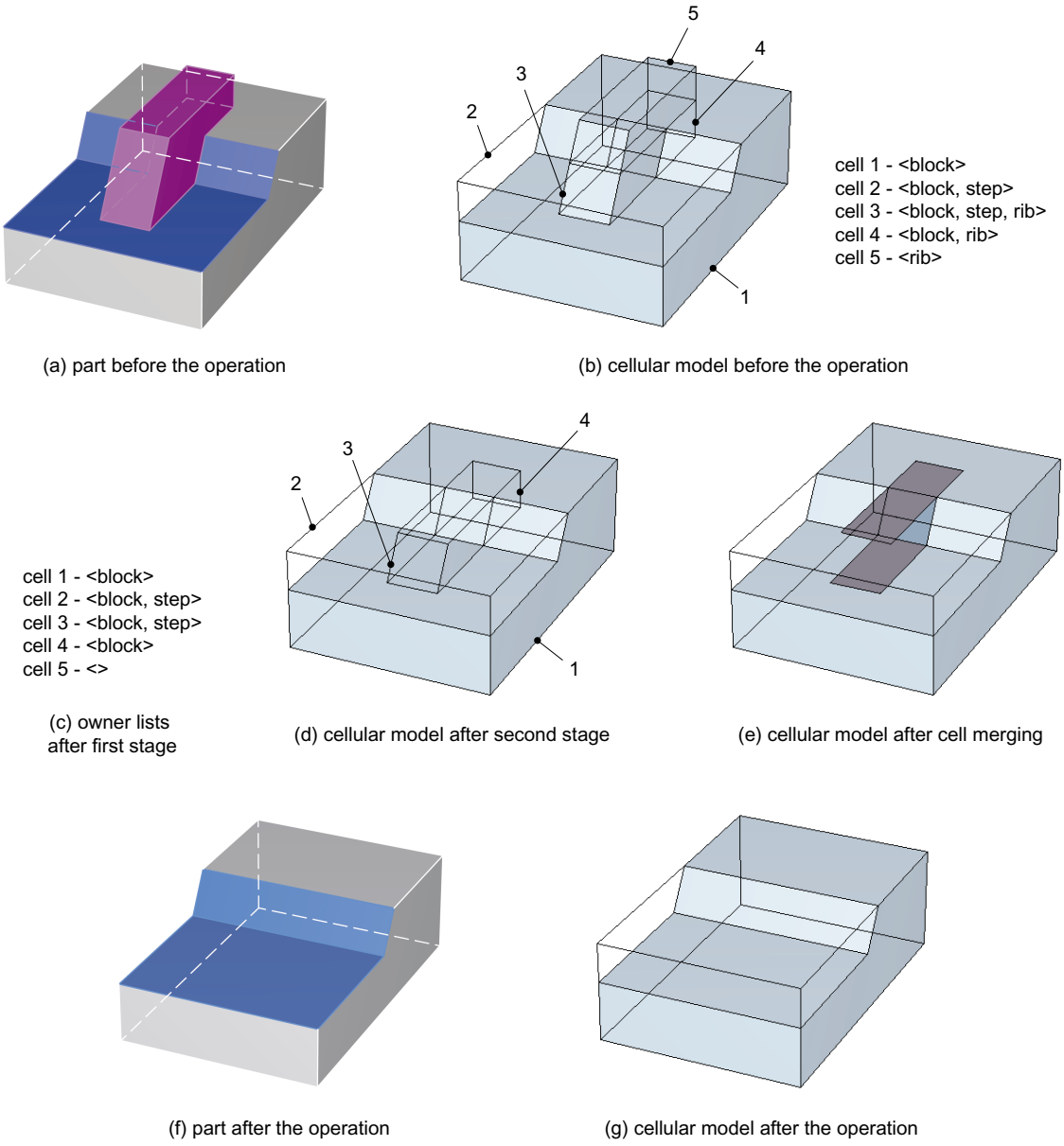


Figure 3. Removing a feature from the cellular model

such cells have been merged, the cellular model has the minimal number of cells required to represent the remaining features, but it may still exhibit a non-minimal number of faces and edges. A final clean up is therefore performed, merging any adjacent faces with the same geometry, whose cell faces have equal owner lists.

Figure 3 illustrates this operation for a simple part consisting of a block, a step and a rib (see Figure 3(a)). Figure 3(b) shows the corresponding cellular model, where its five cells are identified, together with the respective owner lists. When the rib is removed from the model, at the end of the first stage the cell owner lists are those shown in Figure 3(c). Cell 5, which was owned exclusively by the rib, has now an empty owner list and is therefore removed in the second stage, after which the cellular model shown in Figure 3(d) is obtained. In the third stage, adjacent cells 2 and 3, having equal owner lists, are merged, and the same is performed for cells 1 and 4, yielding the model in Figure 3(e). The redundant faces left, highlighted in Figure 3(e), are subsequently merged with their adjacent faces during the final clean up, resulting in the final part and cellular model shown in Figures 3(f) and 3(g) respectively.

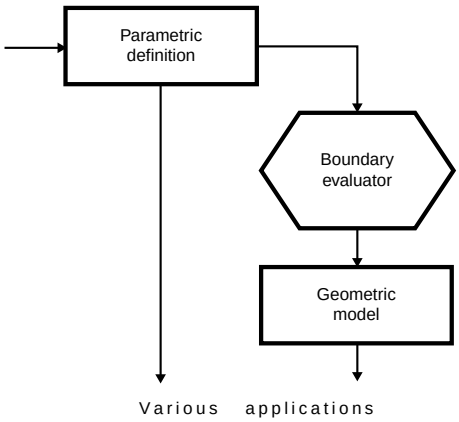


Figure 4. Generic scheme of current parametric modeling systems

3 BOUNDARY EVALUATION

Current parametric feature modeling systems use a dual representation for a product: on the one hand a parametric definition of the product, on the other hand a geometric model representing the resulting shape; see Figure 4. The parametric definition can consist of a large variety of information, usually organized in a graph structure. Typically, the graph represents relations among instances of parameterized features, constraints, set operations, auxiliary geometric entities, etc.

Boundary evaluation is the process that computes the geometric model on the basis of the parametric definition [8]. In the following subsections, this process is discussed for b-rep and cellular model, respectively.

Boundary evaluation for a b-rep

Most current modeling systems are called history-based systems, meaning that the most important relation in the parametric definition graph is the order of creation of feature instances, defining what is usually designated as the *model history* or *feature tree*. These systems typically use a b-rep as geometric representation.

In such systems, adding a new feature to the model, by providing a set of input values for its parameters, appends a new node to the model history, yielding a new parametric definition of the product. Similarly, feature instances can be modified by specifying new values for their parameters, or be deleted from the model. This is done by modifying, or deleting, the respective feature node in the model history.

The goal of the boundary evaluator in Figure 4 is to generate a b-rep that matches the parametric definition at each moment. Therefore, whenever a new parametric definition is obtained, it should be input to the boundary evaluator, to generate the corresponding new b-rep.

When a new feature is added to (the top of) the model history, all the evaluator needs to do is to combine the new feature shape into its current b-rep. For this, it uses a Boolean union to process additive features, and a Boolean difference to process subtractive features.

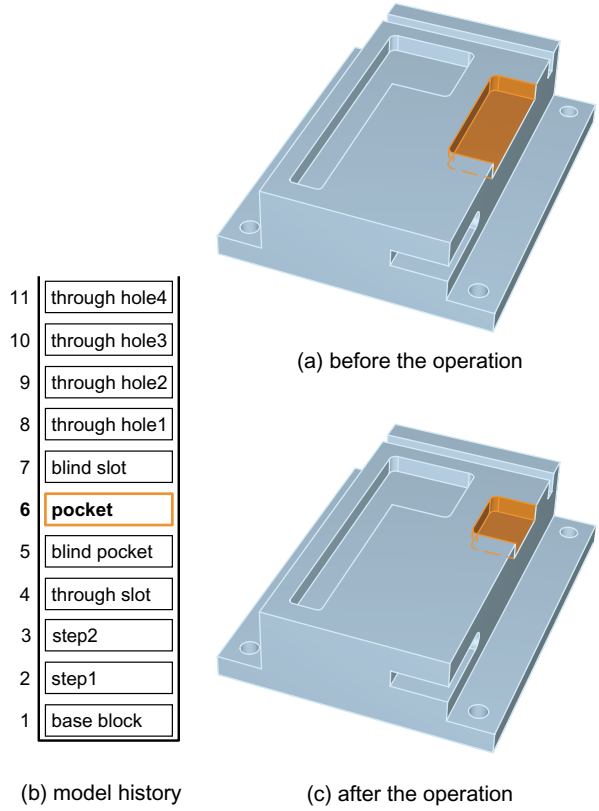


Figure 5. Re-evaluation after a feature modification operation

However, when a feature is modified in, or removed from, the model history, the current b-rep is, in general, of little use for the boundary evaluator. The simplest way to generate the new b-rep is to sequentially walk through the nodes in the model history, and re-execute the associated Boolean operations to build a new b-rep from scratch.

Figure 5 presents an example of this re-evaluation process. The width of the pocket highlighted in Figure 5(a) is decreased. For this, the whole model history in Figure 5(b) is sequentially re-executed, yielding the model in Figure 5(c). So, in this re-evaluation scheme, all features in the model history are processed.

Re-executing the whole model history after modifying, or removing, a feature has the shortcoming that its computational cost is roughly proportional to the model history size. An improvement to this method, which is currently used in most boundary evaluators, consists of keeping the *intermediate models* be-

tween all history steps, in which case only the history steps after the modified, or removed, feature node need to be re-executed. Using this method, the computational cost of boundary evaluation after modifying, or removing, a feature becomes dependent on its position in the model history, but the amount of memory required to store all intermediate models is significant. An alternative improvement consists of storing only the *deltas*, i.e. the model differences, between history steps. This alternative requires less storage, because deltas are typically much simpler than intermediate models, but more computation time is needed to rollback from the current b-rep to the state from which the model needs to be re-evaluated. In both alternatives, however, typically several features that are actually left unchanged by the operation have to be processed, as is, for example, the case for features 7-11 in Figure 5.

Boundary evaluation for a cellular model

This subsection describes the evaluation for the cellular model implemented in SPIFF. This system also follows the scheme given in Figure 4, the difference being that the boundary evaluator produces a cellular model instead of a b-rep.

The most relevant at the parametric definition level is here the Feature Dependency Graph (FDG), which relates all feature instances, each of them having its own set of parameter values, by means of the *dependency relation* [3]. A dependency is a directed relation between two features, and expresses an attach or positioning/orientation reference that a feature has to another feature in the model.

Adding a feature to the model typically creates one or more dependencies in the FDG between the new feature node and the feature nodes already there. Removing a feature requires eliminating its node from the FDG, though not without first making sure that it has no dependents. Modifying an existing feature, in turn, is done by editing its parameters in the respective node in the FDG. Whenever this feature has dependent features, it is likely that the modification will affect some of them as well.

After the FDG has been modified, the boundary evaluator is invoked to update the cellular model accordingly. This is carried out in two phases. In the first phase, the cellular model is incrementally re-evaluated. In the second phase, the cellular model is interpreted, i.e. the shape it represents is determined, according to the feature information stored in its cellular entities and the current dependencies among the feature nodes in the FDG.

Incremental re-evaluation In contrast with the systems described in the previous subsection, which employ two non-associative set operations (union and difference) to compute the b-rep, only one set operation is used by the SPIFF system to compute the cellular model from scratch: the non-regular cellular union of the shapes of all features (see Section 2). Because it is a union operation, and therefore commutative and associative, the order in which the shapes are processed is irrelevant for the final cellular model obtained.

The computational cost of building the whole cellular model from scratch is roughly proportional to the number of features in the model, as is the case for computing a b-rep. Fortunately, this is only required when the cellular model needs to be built in one step, e.g. when starting a modeling session with a model file containing a previously created parametric definition.

Once there is a cellular model, re-evaluating it after each modeling operation is done incrementally, i.e. only for the features whose geometry is actually involved in the operation. The computational cost of this incremental evaluation is dependent on the number of such features, which is usually very low, and is, therefore, independent of the total number of features in the model. The three modeling operations are performed as follows:

(i) Adding a new feature instance to the cellular model: the shape extent of the new feature is combined with the current cellular model. For this, the nonregular cellular union operation is used, as described in Section 2.
(ii) Removing a feature instance from the cellular model: the current cellular model is modified using the operation to remove a feature described in Section 2.
(iii) Modifying a feature instance in the cellular model: the modified feature, and all its dependent features affected by the operation, need to be taken into account. They are removed from the cellular model and then re-added with their new parameters, using the add and remove feature operations just mentioned.

History-independent interpretation Interpretation of the cellular model consists of determining whether the point set represented by each cell belongs to, or represents 'material' of, the product, i.e. the nature of that cell. This requires deciding which of the features in its owner list 'prevails', either as additive or as subtractive. To this purpose, precedences among features have to be established.

If, based on some precedence criteria, a global ordering is defined on the set F of all features in the model, say assigning to them unique, increasing precedence numbers, then every cell owner list (a subset of F) can be sorted according to these precedence numbers. The nature of a cell becomes, then, the nature of the last feature in its owner list, i.e. the feature with the highest precedence number. Appropriate precedence criteria produce an interpretation of the cellular model that is unambiguously determined without invoking any model history considerations [3]. This, together with the incremental character of model evaluation, are important advantages of a cellular model.

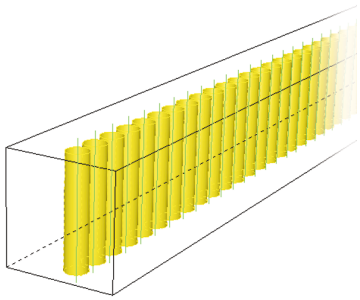


Figure 6. Model Holes

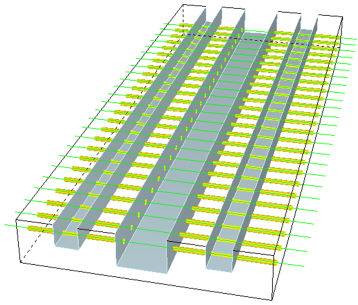


Figure 7. Model SlotsHoles

4 PERFORMANCE MEASUREMENTS

To evaluate the performance of boundary evaluation for the cellular model, and compare this to the performance of boundary evaluation for a b-rep, their behavior has been measured for series of operations on several models. In this section the outcome of the measurements is presented, whereas in the next section the results are interpreted and conclusions are drawn.

The performance measurements for boundary evaluation for the cellular model were done with the prototype feature modeling system SPIFF. Measuring evaluation times could simply be done by including timers in the source code of the system around its boundary evaluation algorithm. SPIFF is running under Linux, and the measurements were done on an Intel Pentium 4 computer.

The performance measurements for boundary evaluation for a b-rep were done with the commercial system Pro/ENGINEER [6]. Measuring evaluation times in this system was more complex than in SPIFF, because the source code of Pro/ENGINEER could not be adapted for this. However, Pro/ENGINEER uses a so-called trail file to record all user actions during a modeling session. Such a trail file can subsequently serve as a script file for another modeling session, possibly after changing it. By including timestamp commands in a trail file around the boundary evaluation step in the script, and executing this file, the evaluation times could be measured. The measurements were done with Pro/ENGINEER 2001 running under Windows 2000 on an Intel Pentium 2 computer.

A major problem was that, because of the measuring unit of the timing commands, the measured times had an inadequate precision compared to the order of magnitude of the times. Therefore, all measurements were done several times, and the results averaged. With some elementary statistical theory, it was determined that 100 measurements were needed to guarantee sufficient accuracy of the computed average time.

Because the time measurements for the two approaches had to be performed on two different computer systems, the times are not directly comparable. To make them more congruent, a simple benchmark was run on both systems to compare their performance in general. This resulted in a normalisation factor that was applied to the measured times for Pro/ENGINEER. Notice, however, that although the normalised times for Pro/ENGINEER are mostly in the same order of magnitude as the times measured for SPIFF, care should be taken in comparing their absolute values. They are still dependent on, for example, the available memory, the operating system, and the degree of optimisation of the implementation of the modeling software. However, the goal here is not to compare the two approaches in absolute computing times, but, instead, to compare trends in their behavior. The times presented here are perfectly suitable for that purpose.

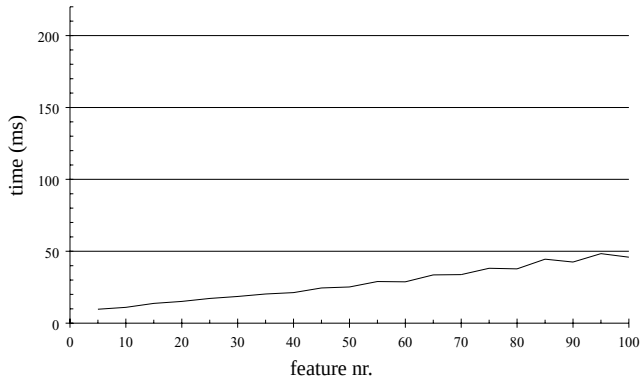
Measurements were done on several models; for two of these the results are presented here. The first model, subsequently called Holes, consists of a block with a row of 100 non-intersecting cylindrical hole features (see Figure 6). The second model, subsequently called SlotsHoles, consists of a block with three slots and a row of 24 non-intersecting cylindrical hole features, each of which intersects all slots (see Figure 7). The hole features in the complete Holes model are numbered from 1 to 100; the hole features in the complete SlotsHoles model from 1 to 24.

For each model, times required for a series of add, remove and modify feature operations were measured. The results are presented in Figure 8 for Holes with SPIFF, Figure 9 for Holes with Pro/ENGINEER, Figure 10 for SlotsHoles with SPIFF, and Figure 11 for SlotsHoles with Pro/ENGINEER.

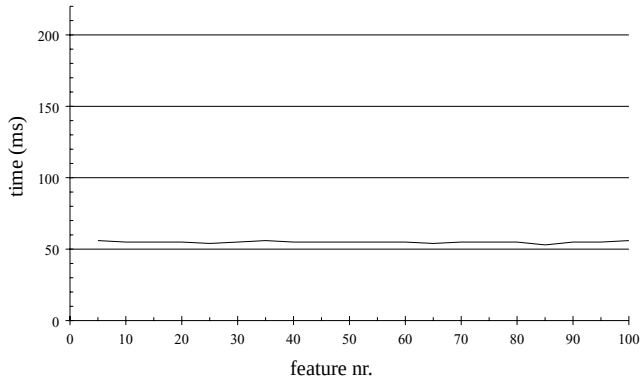
The three graphs in each figure show (normalised) computing times for the series of add, remove and modify feature operations. Depending on the operation, for position *n* on the horizontal axis of each graph, the evaluation time is depicted to respectively:

(i) add feature *n* to the model consisting of features 1 to *n-1*;
(ii) remove feature *n* from the complete model;
(iii) modify feature *n* in the complete model, i.e. change one of the parameters of feature *n*.

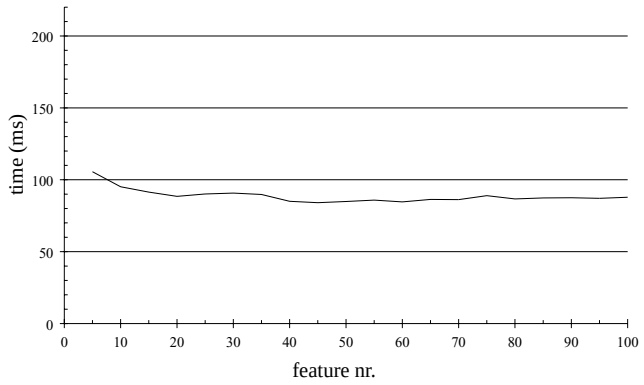
The graphs are quite regular, which is obviously due to the fact that all features in the models are identical. The relatively small irregularities in the graphs can be attributed to factors in-



(a) add feature

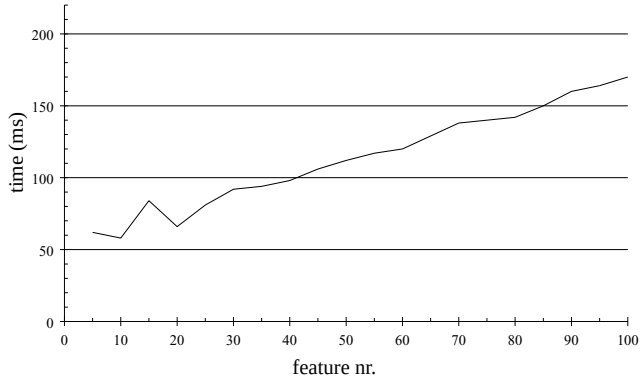


(b) remove feature

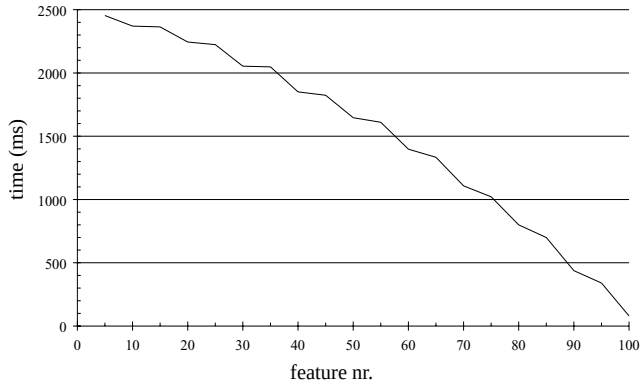


(c) modify feature

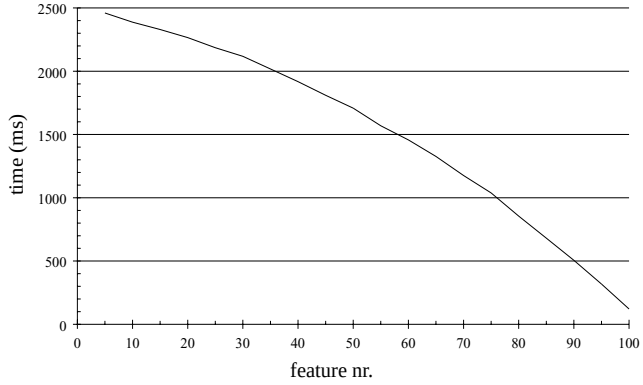
Figure 8. Measurements of boundary evaluation for Holes with SPIFF



(a) add feature

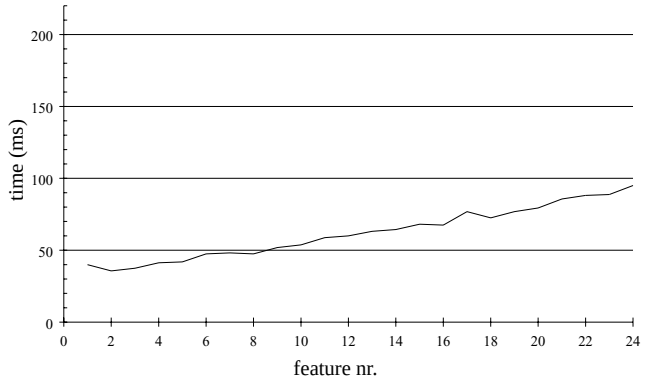


(b) remove feature

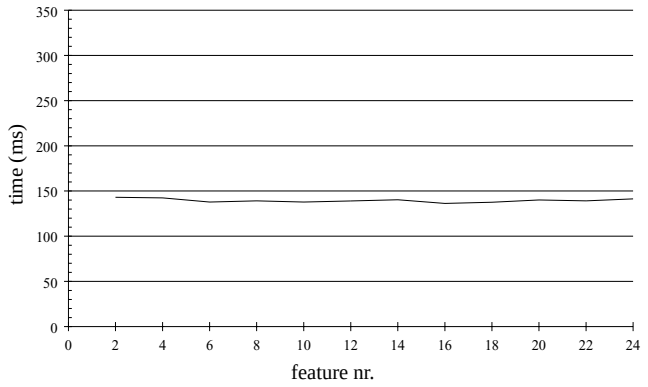


(c) modify feature

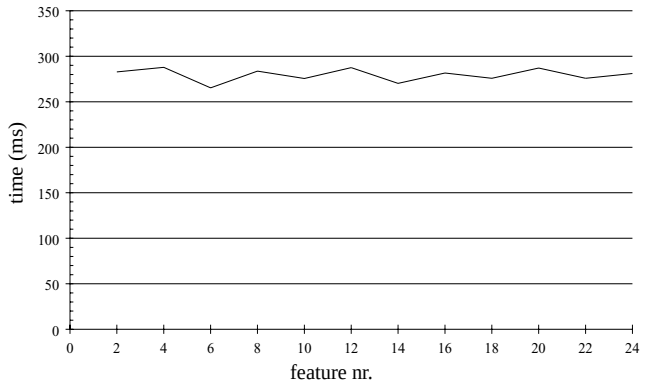
Figure 9. Measurements of boundary evaluation for Holes with Pro/ENGINEER



(a) add feature

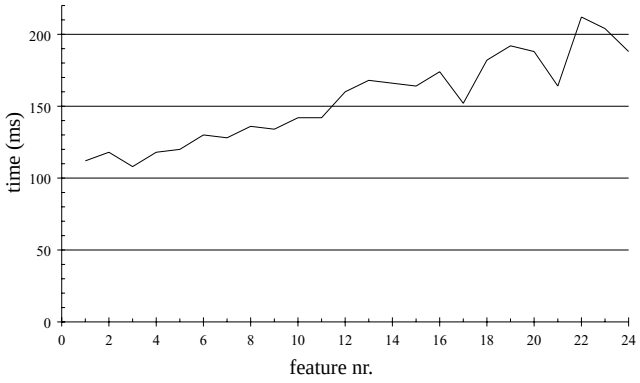


(b) remove feature

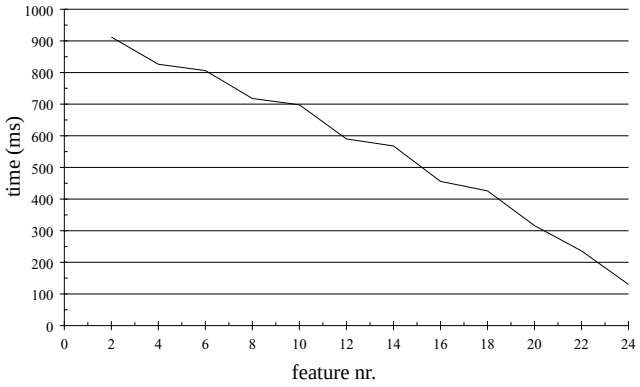


(c) modify feature

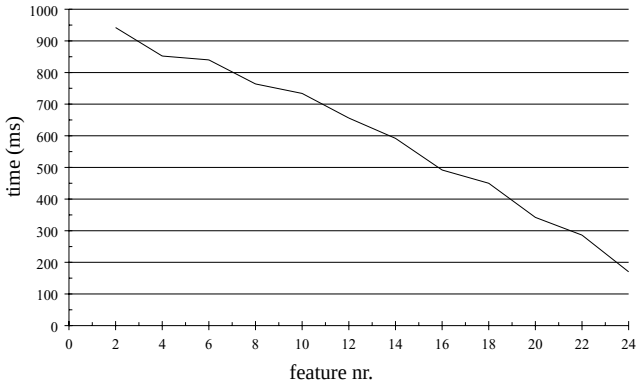
Figure 10. Measurements of boundary evaluation for SlotsHoles with SPIFF



(a) add feature



(b) remove feature



(c) modify feature

Figure 11. Measurements of boundary evaluation for SlotsHoles with Pro/ENGINEER

dependent of the modeling systems, e.g. the memory allocation strategy of the operating system.

5 RESULTS AND CONCLUSIONS

In the previous section, the outcome of performance measurements for boundary evaluation, both for the cellular model and for a b-rep, have been presented for two models. In this section, trends in the presented graphs will be deduced and interpreted, and conclusions will be drawn on the performance of boundary evaluation for cellular models.

Both models are regular, in the sense that they contain a set of identical features, and therefore suitable to deduce trends in the behavior of the boundary evaluation algorithms. The two models were used for deducing trends in two different types of models: the Holes model for trends in models without intersecting features, the SlotsHoles model for trends in models with intersecting features.

First, the Holes model will be considered. The trends for the three operations, add, remove and modify feature, will be deduced from the graphs for this model, and subsequently be explained. Notice that for the boundary evaluation for the cellular model in the SPIFF system, the right explanations of the trends can always be given, but that for the boundary evaluation for the b-rep in the Pro/ENGINEER system, only plausible explanations can be given. Insufficient information is available on the latter system to do better here.

Add feature operation

For position *n* on the horizontal axis of the add feature operation graphs (Figures 8(a) and 9(a)), the evaluation time is displayed for adding feature *n* to the model consisting of features 1 to *n-1*.

For both systems, the evaluation time turns out to (more or less) linearly increase with *n*. This can be attributed to the increasing model complexity: when feature *n* is added to the model consisting of features 1 to *n-1*, it has to be intersected, or at least tested for intersections, with the faces already present in the model.

For the SPIFF system, indeed the number of faces in the cellular model linearly increases with the number of features in the model (see Section 2). For the Pro/ENGINEER system, the number of faces in the b-rep also linearly increases with the number of features in the model.

Remove feature operation

For position *n* on the horizontal axis of the remove feature operation graphs (Figures 8(b) and 9(b)), the evaluation time is displayed for removing feature *n* from the complete model of 100 features.

For the SPIFF system, the evaluation time is nearly constant. This can be explained from the fact that removing a feature involves only removing its cells from the cellular model (see Section 2), which is similar for every feature being removed.

For the Pro/ENGINEER system, on the other hand, the evaluation time is not constant, but dependent on the sequence number of the feature that is removed. This can be explained by assuming the boundary evaluator stores intermediate models or deltas between history steps (see Section 3). When feature 1 is removed, the whole model has to be re-evaluated, which involves 99 features. When feature 100 is removed, no re-evaluation is required at all, because the last intermediate model in the model history contains the required result. For features 2 up to 99, the evaluation time is linearly dependent on the feature sequence number *n*, because this involves re-executing the model history from step *n+1* to the end, i.e. adding *100-(n+1)+1=100-n* features to the intermediate model stored when feature *n-1* was added.

Modify feature operation

For position *n* on the horizontal axis of the modify feature operation graphs (Figures 8(c) and 9(c)), the evaluation time is depicted for modifying feature *n* in the complete model of 100 features, i.e. for changing one of the parameters of feature *n*.

For the SPIFF system, the evaluation time is again nearly constant, but somewhat higher than for removing a feature. This can be explained from the fact that modifying a feature from the cellular model involves (i) removing the old instance of the feature from the model, and (ii) adding the new instance of the feature to the model (see Section 3). Both steps are independent of the sequence number of the feature concerned. For the remove step, the independency has been explained above. For the add step, it follows from the following argument: adding a feature to a model consisting of 99 features always has the same cost, because it always has to be intersected, or at least tested for intersections, with the same number of feature faces already present in the model. The cost of the modify operation is approximately equal to the sum of the cost of the remove feature operation from Figure 8(b) and the cost of the add operation for feature 100 from Figure 8(a), because the latter operation also consists of adding a feature to a model consisting of 99 features.

For the Pro/ENGINEER system, on the other hand, the evaluation time is again dependent on the sequence number of the feature that is modified. Again assuming that this system stores intermediate models or deltas (see Section 3), this can also easily be explained. When feature 1 is modified, the whole model has to be re-evaluated, which involves 100 features. When feature 100 is modified, this involves combining the intermediate model stored when feature 99 was added with feature 100 only. For the intermediate features, the evaluation time is again linearly dependent on the feature sequence number *n*, because this involves re-executing the model history from step *n* to the end, i.e. adding *100-n+1=101-n* features to the intermediate model stored when

feature *n-1* was added.

Behavior of the two systems

It is also interesting to compare the costs of the add, remove and feature operation for a single system.

For SPIFF, the following has already been observed:

(i) the cost of remove and modify feature operations (Figures 8(b) and 8(c)) is nearly constant;
(ii) the cost of a modify feature operation (Figure 8(c)) is approximately equal to the sum of the cost of a remove feature operation and the maximum cost of an add feature operation (Figure 8(a)).

For Pro/ENGINEER, on the other hand, the following can be observed:

(i) the cost of a remove feature operation (Figure 9(b)) is generally much higher than the cost of an add feature operation (Figure 9(a));
(ii) the cost of a modify feature operation (Figure 9(c)) is again generally much higher than the cost of an add feature operation (Figure 9(a)).

The latter behaviors follow from the fact that both a remove and a modify feature operation generally involve several add feature operations.

Results for model with feature intersections

The same trends as deduced above from the graphs for the Holes model, can be deduced from the graphs for the SlotsHoles model (see Figures 10 and 11). Stated differently, the same trends occur for models with features with intersections, as for models with features without intersections. Although the number of intersections of each feature in SlotsHoles is limited (to three), this is also typically the case in real-world models, because of the local character of features. Usually features are small compared to the whole product, and interact with a small number of other features only.

Conclusions

On the basis of the above observations, the following can be concluded. For the add feature operation, boundary evaluation for the cellular model has the same performance trend as boundary evaluation for a b-rep; for both, the time increases linearly with the number of features already in the model, and the times are in the same order of magnitude. For the remove and modify feature operations, however, the trends are different. Boundary evaluation for the cellular model has a constant cost that is in the same order of magnitude as the cost of an add feature operation. Boundary evaluation for a b-rep, on the other hand, has a cost that is linearly dependent on when the feature being removed or modified was added to the model, and is very high compared to the cost of an add operation.

Usually, most of the time during a modeling session is spent on adjusting features already in the model, rather than on adding new features. The conclusion is therefore that boundary evaluation for the cellular model is in fact more efficient than boundary evaluation for a b-rep. Considering this conclusion, and the applications and advantages of cellular models mentioned in Section 1, it can be expected that in the future such models will be increasingly used in feature modeling systems.

ACKNOWLEDGEMENTS

We thank Klaas Jan de Kraker for many valuable comments on a previous version of this paper.

REFERENCES

[1] Shah, J. J., and Mäntylä, M., 1995. *Parametric and Feature-based CAD/CAM*. John Wiley & Sons, Inc., New York.
[2] Bidarra, R., de Kraker, K. J., and Bronsvoort, W. F., 1998. "Representation and management of feature information in a cellular model". Computer-Aided Design, **30** (4), pp. 301–313.
[3] Bidarra, R., and Bronsvoort, W. F., 2000. "Semantic feature modelling". Computer-Aided Design, **32** (3), pp. 201–225.
[4] Bronsvoort, W. F., Bidarra, R., and Noort, A., 2002. "Feature model visualization". Computer Graphics Forum, **21** (4), pp. 661–673.
[5] Bronsvoort, W. F., and Noort, A., 2003. "Multiple-view feature modelling for integral product development". *Submitted for publication*.
[6] Parametric Technology Corporation, 2002. Pro/ENGINEER product information. http://www.ptc.com/products/proe/.
[7] Spatial Technology Inc, 2002. 3D ACIS Modeler, Version 8.0. http://www.spatial.com.
[8] Requicha, A. A. G., and Voelcker, H. B., 1985. "Boolean operations in solid modeling: boundary evaluation and merging algorithms". Proc. IEEE, **73** (1), pp. 30–44.