A framework for concurrent structure analysis in building industry

A. Niggl, R. Romberg & E. Rank

Lehrstuhl für Bauinformatik, Technische Universität München, Germany

R. P. Mundani & H. J. Bungartz

IPVS, Abteilung Simulation großer Systeme, Universität Stuttgart, Germany

ABSTRACT: In this paper, a software-framework will be presented, which helps to support the concurrent work of multiple planners in the construction industry. Basis of this work is a strictly volume-oriented building model. This model is stored in a central database, which supports the cooperative work by using object-based ‘check-in’, ‘check-out’ and ‘locking’ mechanism. Furthermore a decomposition algorithm will be presented, which automatically derives a hexahedral mesh for a finite element computation from this central building model.

1 INTRODUCTION

The efficient and accurate exchange of data is an important basis for a successful cooperative work in the field of computer-supported planning and design in building industry. This fact not only holds for the exchange inside a homogenous group of planners, it is also important for the data-transfer between planning processes within different working-domains.

In this paper, a software-framework will be presented, where a central volume-oriented geometric model is considered as a basis to support the cooperative work and the integration of different planning-processes. The central data set is thereby given by an explicit geometric B-Rep (Boundary-Representation) model associated with semantic product data attributes, which is originally derived from an IFC product model.

Using a classical client-server structure, the building model is provided centrally and can be accessed by different planners. The concurrent access to the database is organized similar to classical software-management-systems, where single entities can be ‘checked out’, locally modified and ‘checked in’ by the user. To ensure geometric consistency of the common data model, an octree-based algorithm is applied, which was developed in this project.

The geometric building model is the starting point for various subsequent tasks in the planning process. For the structural analysis, we present an automatic generation of a volume-oriented finite element mesh, consisting of solid hexahedral elements. In addition to the finite element analysis, an indoor air flow simulation was also connected to the
framework in an other project (v. Treeck et al. 2004). Figure 1 shows a schematic view of the framework.

The outline of this paper is as follows: In the next section, the software-structure and techniques used in this work will be presented. Then in Section 3 the automatic derivation of the hexahedral finite element mesh from the architectural model will be described in detail. Finally, in Section 4 the cooperative work of two planners will be demonstrated in an example.

Figure 1. Schematic view of the framework: an IFC-model is converted into a geometric B-Rep model, which is saved in a central database; different clients can access in parallel the central data; a finite element model can be derived and analyzed automatically.
2 A VOLUME ORIENTED GEOMETRIC MODEL

2.1 From IFC to an explicit geometric model

In this project, a commonly shared geometric model is used as a basis for various tasks in the planning and simulation process. The geometric data of a building model given by the IFC-standard (IAI 2003) is usually not directly adequate for a numerical simulation, as it only describes the topology and the mutual connections of different structural components. For example, the geometry of a wall is described by a 2D profile together with an extrusion direction or a window is given by its relative position to an ‘anchor point’. But for the automatic derivation of subsequent simulation models, for example the generation of a finite element mesh, we need an explicit description of the geometry. Thus, we use the geometric modeller ACIS (Spatial 2004) to create a geometric model from the IFC-product model, where each construction unit is described by a single B-Rep object. The IFC data is accessed by using the Eurostep IFC-Toolbox (Eurostep 2000), which is an object oriented C++ implementation of the IFC scheme representation and which provides interface functionalities to access and manage instances of the product model. The semantic data contained in the IFC object model is added as attributes to the B-Rep entities and is saved parallel to the geometric data in an additional database.

2.2 Organization of the concurrent access

The technical basis of our cooperative workspace is a classical client-server architecture. In order to ensure consistency of the central data model, the concurrent access by the clients is being controlled by an intermediary management layer. Similar to well known software-management systems, like CVS (concurrent version system), the server provides methods for downloading, managing and uploading data. The smallest organizational entity in the exchange is thereby one single B-Rep object. Using an octree-based algorithm, the management layer ensures geometric consistency of the internal data model.

In order to share information among the concurrently working planners, notification services were developed in this project. The user can activate a locally working software module, which connects to the server and informs him about modifications in the central data model caused by other planners. In a configuration menu, the user can choose among different notification levels. He has also the possibility to reduce the number of objects, he wants to be informed about, to a subset of the complete model.

During runtime and depending on the type of user access, a single object on the server can remain in one of three different states: clean, shared and locked. An object is clean, if no client has accessed this object for modification. It is in state shared, if at least one user has accessed the object in read- and/or write-mode and it is in state locked, if just one client has claimed exclusive write permissions for an object.

The operational procedure in the user-side workspace will be demonstrated in an example in the last section in detail. Thereby, one thing is always the same: In order to
upload (check-in) modified or new objects to the central database, the user has to checkout the selected objects first.

Based on the internal states mentioned before, for each object, the user can choose among three possible modes of access:

- **Read-only**: in this mode, a modification by the user is not allowed. However, the user will be registered on the server and has the ability to activate the notification service for the selected object.
- **Read-write**: the user gets read and write access for the data, which allows him to check in modified objects to the server. Modification by others is also possible.
- **Exclusive-write (lock)**: the user gets exclusive write access to the object. Modification by others is not allowed.

Only in case of read-write mode, a concurrent access of objects is possible. In this case, each upload by a user will overwrite the current version in the central database. To avoid confusion, the notification service may help, so that the user can stay informed about the actual state and can react accordingly. This makes sense especially in cases, where one user (e.g. a designer) changes the geometry of an object while an other user (e.g. a structural engineer) only wants to modify some attributes like loads or material. In such cases, an exclusive lock by one user would only hamper the work of the other. In cases where one wants to prohibit concurrent access by others completely, e.g. when substantial modifications must be applied, exclusive-write access should be used instead.

After modifications in the local workspace are finished, the user will check in his object to the central database. Each upload of a modified or new building object will thereby initiate a consistency check on the server, according to the method which will be explained in the next sub-section. In the case of geometric collisions or insufficient access rights, the upload will be rejected and the user will be informed about the problem.

### 2.3 Consistency check

Before any consistency check can be performed, volume-oriented models have to be derived from the respective surface-oriented ones. In (Mundani et al. 2003) we presented an algorithm for the generation of octrees by intersecting half-spaces, allowing us a fast and efficient derivation both in real time and on-the-fly.

Applying the Boolean operator ‘intersection’ on two octrees, collisions of type ‘overlap’ and ‘gap’ can easily be detected. Whenever two voxels—volume elements—of two arbitrary octrees claim for the same space an overlap occurred and the algorithm can stop at this point. Depending on the maximum depth of recursion \( d_{\text{max}} \) overlaps up to \( h=1/2 \ d_{\text{max}} \) on the finest resolution level can be found. In this case, the check-in of modified parts is rejected by the server.

When no overlap could be detected, the two parts or the two volume-oriented models, respectively, are either lying perfectly together side by side or are disjoint. In the latter case, a gap among these two parts exists; only gaps of certain sizes are of further interest. Therefore, the algorithm has to determine the maximum depth \( d_{\text{eff}} \) reached during the intersection calculation, not to confuse with the maximum depth of recursion \( d_{\text{max}} \). By specifying \( d_{\text{gap}} \), the maximum gap size, only in case of \( d_{\text{gap}} \leq d_{\text{eff}} < d_{\text{max}} \) a gap has been
detected. The corresponding part is allowed to be checked in but further user feedback is necessary, because most gaps unintentionally occur due to design or round-off errors.

3 STRUCTURAL ANALYSIS

3.1 A volume oriented finite element approach

In contrast to the classical way in finite element analysis using dimensionally reduced models (e.g. 2D-plates, shells or beams), in this project the structural analysis is performed in a fully volume-oriented approach. The complete structure is discretized consistently with solid hexahedral elements and the computation is carried out by using higher order elements of the so-called p-version of the finite element method (Szabó et al. 1991, Szabó et al. 2003, Düster et al. 2003). This approach has some important advantages:

- The automatic derivation of a finite element model from the original (product-) model is simpler, if this transition can be done consistently in the same volume-oriented way.
- The possibility of such an automatic model derivation releases the engineer from a ‘manual’ reconstruction of various numerical systems.
- Using solid finite elements, possible three-dimensional stress conditions can be resolved.
- There is no need for coupling different dimensionally reduced mechanical models.

3.2 Automatic derivation of the finite element mesh

An important basis of our consistent volume-oriented approach is the automatic derivation of the finite element model (Romberg et al. 2004). The basic idea of the underlying algorithm is to decompose the given geometric building model into a set of simpler geometric objects and decompose each of these objects again into hexahedral elements. To ensure compatibility (i.e. no hanging nodes) of the final mesh, the whole procedure is carried out in a set of steps, which are mainly used to find a common discretization at the interface of different entities. It should be mentioned that this approach does not aim at meshing general spatial volume-structures but is capable of decomposing a typical building model, which usually consists of objects like plates, beams, columns and slabs.

3.2.1 Connection model decomposition

As a first step in the process of creating a hexahedral mesh, the given geometric building model is decomposed into a so-called connection model using boolean operations. Figures 2-4 illustrate the basic idea.

Starting from the set of building models \( M_b \), these elements are partitioned into a set of coupling objects \( M_k \) and a set of difference objects \( M_d \). The set of coupling objects \( M_k \) are then again recursively decomposed into coupling objects of different levels (\( M_{kl} \ldots M_{kl} \)).
Each coupling and difference object is itself a closed B-Rep body being described by nodes, edges and faces. After decomposition, the intersection between difference objects or coupling objects of the same level is given in points and edges only, whereas adjacent difference and coupling objects intersect in faces, edges and nodes.

After applying this decomposition algorithm, the resulting elements have some important characteristics with respect to the following steps in finite element

Figure 2. Initial configuration $M_b$ with three objects.

Figure 3. Creation of connection objects ($M_{kl}$, $M_{k2}$) using boolean operations.
mesh generation: Each coupling object $M_c$ possesses hexahedral structure, thus, it can be easily partitioned into smaller elements, whereas each difference object $M_d$ is ‘plate shaped’ and can be assumed to be obtained from sweeping a two-dimensional polygonal domain.

3.2.2 *Generation of hexahedral finite elements*

The connection model shown in the previous section is the starting point for the automatic generation of hexahedral elements in the next step. Thereby we use either elementary three-dimensional meshing macros mainly applied to the coupling elements or, in case of the plate-like difference objects, hexahedral elements are obtained by creating a 2D quadrilateral mesh on the polynomial mid-face and sweeping this mesh to the third direction. Most crucial in meshing is yet the question of generation of compatible elements. For this, we apply a two-step approach. In a first run, a reasonably refined mesh for each (separate) difference object is defined. According to the different discretization on the boundaries of adjacent elements a compatible discretization must be determined. When this common discretization is found on the boundary,
Figure 5. Original 3D volume model given by the CAD system.

Figure 6. Decomposed connection model.
a new mesh is created on the difference objects in a second run, which is then compatible with its neighbours, i.e. the resulting mesh has no hanging nodes.

### 3.3 A complex example

In this section, the process from the original geometric building model to the finite element results is demonstrated in an example of a realistic office building. The building is constructed by reinforced concrete and consists of two massive inlying building cores, six floor plates and supporting columns. It has dimensions of about 40×30 meters in the ground view. Figure 5 shows the geometric model.

In Figure 6, the decomposed connection model can be seen, which was derived from the original model according to Section 3.2.1. One can see easily the connection elements on the top floor plate, created at the intersection of the inlying building cores and the plate.

Figure 7. Finite element mesh.
Figure 7 shows the finite element mesh, derived from the connection model according to subsection 3.2.2. It consists of 8313 hexahedral elements. In Figures 8 and 9 the finite element results can be seen. First, in Figure 8, the displacement plot is depicted. Figure 9 shows mean stresses (v. Mises stress) in a zoomed detail. For the computation, vertical loads on the floorslabs and horizontal wind-loads were considered. The results were computed using the $p$-version of the finite element method with a polynomial degree of 3. This resulted in a computation with 269,043 degrees of freedom, which took about 2 h of time on a Pentium IV with 1.7 GHz.
Figure 9. Zoomed detail of stress plot (v. Mises stresses).

Figure 10. Complete model in workspace A.
4 CONCURRENT WORK EXAMPLE

In this section, the concurrent work in a group of multiple planners will be demonstrated in an example. The office building shown in the previous section is now modeled and stored in the central database, which was described in detail in Section 2.2 and 2.3. In this context the database is also referred to as ‘server’.

Let us assume, that planner A (e.g. an architect) decides to carry out some modifications in the model. In a first step, he may request for an update in order to get the newest model version from the server (Fig. 10). Then, he checks out some objects with exclusive-write access in order to apply some modifications, e.g. move a column to another place (Figs. 11, 12).

In the meantime, planner B (e.g. a structural engineer) has also checked out some other objects and changes load attributes (Fig. 13).

Figure 11. Check-out of selected items by planner A.
Figure 12. Change of column by planner A.

Figure 13. Change of load attributes by planner B.
Planner $A$ has finished his work now. So, he selects the modified objects and tries to check them in to the server (Fig. 14). Unless any other user has locked one of the objects exclusively or the consistency check detects an intersection, the check-in is successful and the objects will be stored in the database and overwrite their current version there.

Figure 14. Check-in of modified column by planner A.

Figure 15. Local database agent informs planner B about the modified objects.
This occurrence of objects checked in is now reported automatically to planner B, because he has activated his local notification agent with the order to listen for geometric modifications (Fig. 15).

Technically, every check-in is thereby posted to a message queue on the server, where each message contains information about the type of modification (geometrically, only attributes, type of attribute…), the user, etc. This message queue can be read out by the remote, client-side notification agent, which prefilters the messages according to the user’s settings and sends a signal to the user.

Back to planner B, the structural engineer, after updating his local workspace, he initiates a new finite element computation on basis of the modified model.

![Figure 16. Resulting finite element mesh of the modified model.](image)

Again, the derivation of the finite element mesh and the computation is performed completely automatic and needs no further manual interaction by the user. Figure 16 shows the finite element of the modified model, where a complete line of columns was changed.

5 CONCLUSION

We have presented an approach, which may help to support the co-operation between different planners in building construction. The concurrent work is organized, using a
central database with its access management layer in combination with a consistency check and notification services.

The automatic generation of a finite element mesh, based on a strictly volume oriented model, releases the engineer from manually transferring design models to the numerical simulation model. This helps to speed up the design process, especially in cases, when modifications and different design variants must be investigated.

ACKNOWLEDGEMENT

This research has been supported by the Deutsche Forschungsgemeinschaft (Priority program 1103, “Vernetzt kooperative Planungsprozesse im Konstruktiven Ingenieurbau”) to which the authors are grateful.

REFERENCES


