AN OBJECT ORIENTED MODELLING APPROACH FOR STRUCTURES WITH STATISTICALLY DISTRIBUTED DEFECTS

EIN OBJEKTORIENTIERTER MODELLIERUNGSANSATZ FÜR STRUKTUREN MIT STATISTISCH VERTEILTEN DEFEKTEN

UNE APPROCHE ORIENTEE-OBJET POUR LA MODELISATION DE STRUCTURES AVEC DES DEFAUTS A DISTRIBUTION STOCHASTIQUE

Marco Schrank, Gerhard Dill-Langer, Simon Aicher

SUMMARY

In the following, we present an object oriented approach to the modelling of structures with stochastically distributed defects. With increasing computer power and the availability of parallel computers and clusters, finite element models become more complex. Up to now the techniques to design complex models has not improved significantly. A way to handle this increasing complexity is to model object oriented. This article gives an introduction to the object oriented FEM modelling by the example of a Monte-Carlo-simulation of a glulam beam with the commercial FEM tool ABAQUS.

ZUSAMMENFASSUNG

Im folgenden wird ein objektorientierter Ansatz zur Modellierung von Strukturen mit stochastisch verteilten Defekten vorgestellt. Mit zunehmender Rechenleistung und der Verfügbarkeit von Parallelcomputern und Clustern werden auch FE-Modelle immer komplexer. Die Techniken zur Modellierung haben sich jedoch nicht im gleichen Maß weiterentwickelt. Eine Möglichkeit die wachsende Komplexität zu beherschen ist die objektorientierte Modellierung. Dieser Artikel führt in die objektorientierte FE-Modellierung am Beispiel einer Monte-Carlo-simulation eines Brettschichtholzträgers mit dem kommerziellen FE-Programm ABAQUS ein.

RESUME

L'article présente une approche orientée-objet pour la modélisation de structures avec des défauts à distribution stochastique. L'augmentation de la puissance de calcul ainsi que la disponibilité d'ordinateurs parallèles et de grap-

pes de serveurs mènent à des modèles par éléments finis toujours plus complexes. Cependant, les techniques de modélisation n'ont pas évolué au même rythme. Une possibilité de maîtriser cette complexité croissante est la modélisation orientée-objet. Cet article constitue une introduction à la modélisation par éléments finis à l'exemple d'une simulation Monte-Carlo d'une poutre en bois lamellé-collé avec le logiciel éléments finis commercial ABAQUS.

KEYWORDS: Object oriented modelling, Finite Element Method, glued laminated timber, defekts

1 INTRODUCTION

In recent years the finite element method has become the state of the art tool for the solution of mechanical problems. Huge progresses have been made with respect to material modelling or solving large systems of equations. Homogenization techniques or the solution of highly nonlinear problems are only a few to name. In contrary, the modelling of mechanical structures has only slightly changed. The common way to design a FEM problem is to assemble it with boolean operations of geometric primitives. Afterwards material properties and boundary conditions are defined. This is a very intuitive approach, but becomes more difficult with increasing complexity. Considering the example of the glulam beam, knots, finger joints and further defects have to be added to the model. Especially for stochastically distributed defects like the afore mentioned, it is complicated to get a realistic and reliable solution. Further problems arise if the FEM simulation is embedded in a Monte-Carlos-simulation or other techniques where an automated modelling is required. Another point that should be considered is the reusability of the code. The modelling of a knot is in principle identical for different problems. With the currently used procedural programming practice, the best way to reuse the code is copy and paste, which needs adjustment and is error prone. The object oriented programming paradigm is a promising approach to overcome the shortcomings of procedural programming.

2 OBJECT ORIENTATION IN THE CONTEXT OF FINITE ELEMENT MODELLING

The main drawback of the procedural programming of finite element models is that the model can hardly be divided into logical units. It is mostly coded as a monolithic block of code. From the human view, a problem is easier to

oversee if it is divided in logical units. For the glulam beam example three tasks can be identified which constitute self consistent parts of the problem. The Monte-Carlo-simulation which creates the samples and evaluates the 'numerical' distributions, the finite element model as a framework for the sample solution and the defects which can be taken out of the finite element model for the sake of maintainancy and reuseability.

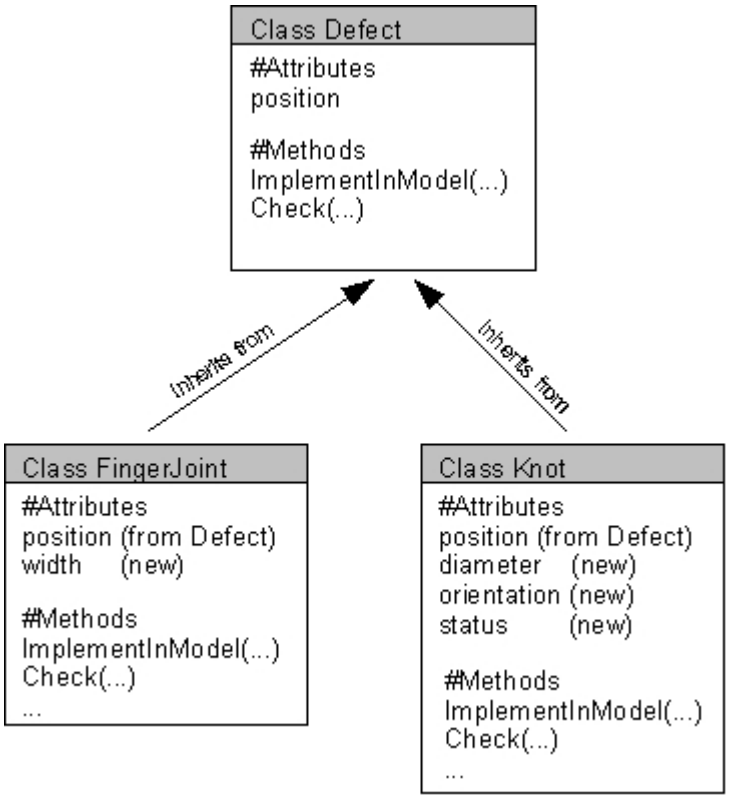

Figure 1: Hirarchy of the Defects

Those three subtasks of the simulation are shortly described. The Monte-Carlo-algorithm generates random samples from statistical distributions. This samples have to be solved, e.g. with a FEM-computation . It collects the results of the samples and generates the distributions of the results. During the model generation process the afore mentioned defects are included in the model. Without any knowledge of the particular implementation, the user knows how the three components work together. This is where the object orientation comes into play. The afore mentioned logical units, namely the Monte-Carlo-algorithm, the model and the defects can be realized without the knowledge of the implementation of the remaining parts; only an appropriate 'communication' between the different units is necessary. This has the advantage that the implementation of one part can be changed without interference of the remaining parts. This encap-

sulation principle is described for a knot example. Knots in timber have mainly four important attributes, the diameter d, the position X, orientation φ and the status s (living/dead knot). By the definition of an interface, the model submits the knot the afore mentioned information and the knot builts itself into the model. The logic of the knot-implementation is completely hidden to the model. The same holds for the Monte-Carlo-simulation which does not know about the Finite Element Model. It just submits the sample definition and obtains the result. In the following those logical and self consistent units are called 'classes'. The classes define attributes, which are comparable to common variables in procedural programming, and the methods are comparable to functions in the procedural programming terminology. Further important principles are inheritance and polymorphism. Again the knot example is stressed for the explanation. We start with the class Defect, Fig. 1, that defines the attributes and methods that all defects share. For example every Defect needs to implement the method *ImplementInModel (...)* to create itself in the model object. Now the two subclasses FingerJoint and Knot are derived from Defect and inherit all attributes and methods from Defect. In the case of *ImplementInModel (...)* it is necessary to overwrite this method from the superclass Defect, because the implementation in the model for a Knot is different from the implementation of a FingerJoint. Further the class Knot needs additional attributes like diameter and status. It is further possible, to assign a subclass object like a Knot in the model object as Defect (superclass of Knot) and call the method *ImplementInModel (...)* of Defect. The Knot will automatically use the right method for the implementation.[1] This principle is called Polymorphism.

3. THE GLULAM BEAM MODELLING FROM THE OBJECT VIEW

With the basic definitions at hand, the object oriented modelling is described for the example problem of a glulam beam. Timber as a construction material lacks from the drawbacks of its natural defects. It has a natural variability of strength and stiffness, the cross section is weakened by knots and the tensile strength is reduced e.g. by compression wood[2]. Because of the limited length of timber boards, they have to be combined by finger joints which have a lower strength than the surrounding clear wood. All this defects are statistically

---

[1] In a real world application, more advanced techniques like interfaces or abstract classes would be used

[2] Compression wood is a reaction of the tree to a constant pressure loading of parts of the trunk

distributed. The new German timber design code DIN 1052:2004 considers this implicitly with characteristic strength values based on the 5%-quantile of the strength distributions. It considers further the homogenization effect of the parallel system of boards in a glulam beam by a coefficient k_h.

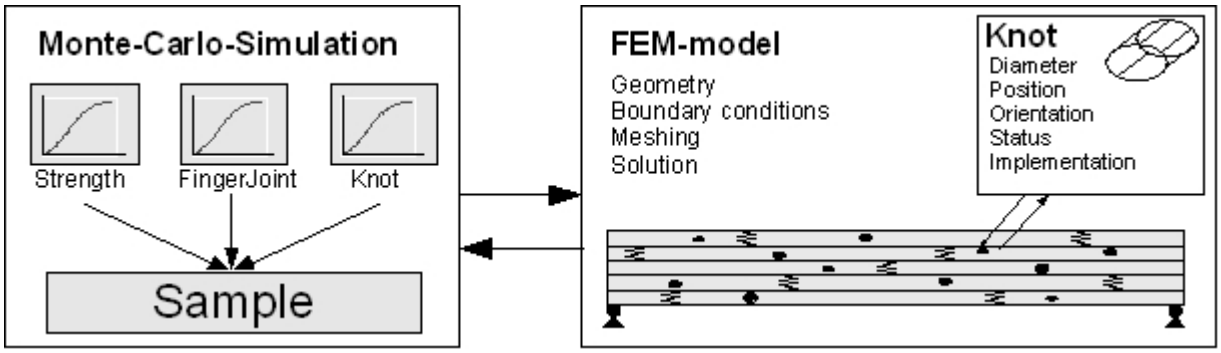

Figure 2: Classification of the problem in subunits

A more realistic consideration of glulam beams made from a population of boards, where the material properties and the distributions of the defects are known or can be estimated, is possible with a Monte-Carlo-simulation (MCS) yielding the 'numerical' strength distributions for the beam, Fig 2. The MCS consists mainly of the statistical distributions of interest, an algorithm for sample creation and the evaluation of the sample results. The MCS can be defined as follows: A set of distributions and the number of samples is given. With the method *CreateSample(...)* the sample, which consists of the material parameters and defect distributions, is created and sent to the model. The method obtains the results and stores it for later evaluation. The model gets the sample informations by a get-method that is called from *CreateSample(...)*. The get-method stores the information of the sample in the model and starts the computation process. The geometry is generated, the material properties from the sample are assigned and defects are created. It is again emphasized that the MCS does not know the implementation of the FEM-model. It only knows how to call the get-method from the FEM-model. This means further that the MCS can be exchanged by any other feasible procedure, for example Fuzzy Arithmetic, that is able to submit a sample with a call of the get-method. On the other hand it is also possible to substitute the FEM-model by another numerical method or an analytical solution, which provides a get-method to receive the sample data. The FEM-model of the glulam beam has now received all information about its material properties and defects. For the material properties, once they are stored in the model, there is no difference to the static modelling. Defects are included

after the geometry is defined. At this point the model runs a loop over the different sets of defects. For each knot, for example, the model instanciates a Knot object and supplies the required information by calling the get-method of the Knot object. Again, the implementation of the Knot logic is not visible to the FEM-model. The Knot object can implement complicated algorithms or just reduce the stiffness and strength in a certain region. The Knot implementation can be completely changed without any changes in the FEM-model. On the other hand, the Knot class can be used for every FEM-model that includes knots, for example solid wood or formwork beams, also without any changes. The exemplary implementation of a Knot is given subsequently. After the inclusion of the knot, a check should be performed, to assure the required construction and grading rules and also the model consistency. Meshing and computation is identical to the procedural modelling. When the computation is finished, the results are returned to the process that initiated the FEM computation, e.g. the MCS.

4. THE EXEMPLARY KNOT IMPLEMENTATION

A knot is a complex inclusion in the fiber structure of timber solids. In the new timber design code DIN 1052:2004 the influence of knots is implicitly considered in the characteristic strength values and material safety factors. If the knot shall be explicitly considered in a simulation, there are several approaches. A simple method is the reduction of strength and stiffness in a certain area around the knot. It is also possible to model the knot as a hole in the structure. Both approaches do not really resemble reality. The modelling as a hole for example leads to a significant overestimation of the stresses at the interface of the knot, [2]. A promising approach is to model the branch as a hole, but consider a realistic fiber flow around it. [1] shows that the orientation of fibers around a knot can be described by a laminar stationary flow around a cylidrical obstacle. This has been used in [2] with good results. In the following, the implementation of this approach is shown. As already known from the preceding section, the model submits the Knot object a diameter, a position, an orientation and a status variable with the get-method.

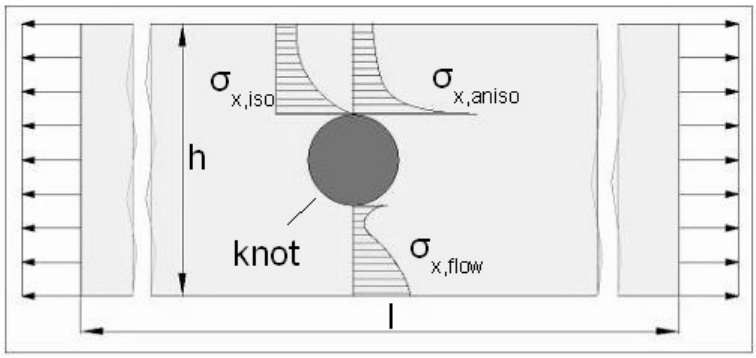

Figure 3: Comparison of different knot models: Isotropic material with a hole, anisotropic material with a hole and anisotropic material with a hole and 'fiberflow'

Afterwards *ImplementInModel (...)* is called and the knot implementation starts. At first a geometrical object, the cylinder with the given diameter, is created. Then it is moved to its final position X and rotated by the orientaion vector φ. Next the cylinder is cut from the respective geometric entity of the model object. The affected body of the model object is now divided into an appropriately fine grid of cells, where every cell receives a material orientation from equation (1) to approximate a realistic fiber orientation.

$$\phi = U_{\infty}\left(r + \frac{r_0^2}{r}\right)\cdot\cos(\varphi) \tag{1}$$

After the check of the implementation, the knot object returns a status if the creation of the knot was successful or not. In the first case, the implementation of defects progresses over the remaining set of knots an fingerjoints and the computation continous as in the procedural modelling. In the second case, the model decides with respect to the error type how to progress further.

5. CONCLUSIONS

The preceding sections show the development of an object oriented framework for the simulation of structures with stochastically distributed defects illustrated by the example of a glulam beam. Because of the statistical nature it is not possible to solve the problem with a single deterministic model. This involves a stochastic procedure that envelops the finite element problem. To get a consistent system that is most flexible and reuseable the problem is divided into logical units which are self consistent and reusable. This leads to an increased quality of the simulation, saves design work and keeps the problem manageable.

REFERENCES

[1] Christina Foley (2003) Modelling of knots in structural timber, Doctoral dissertation, Lund university, Sweden

[2] S. Aicher, G. Dill-Langer (2006) Einfluss des Astabstandes auf die Keilzinkenfestigkeit von Brettschichtholzlamellen, Forschungsbericht Nr. 9008507000, MPA Stuttgart, Germany

[2] B. Oestereich (2006) Analyse und design mit UML, Oldenburg Verlag, Germany

[3] DIN 1052 (2004) Entwurf, Berechnung und Bemessung von Holzbauwerken

[4] Abaqus scripting users manual (2004), Abaqus inc, Usa

[5] J. H. Spurk (1996) Strömungslehre, Springer Verlag, Germany