"THE CONCEPT OF HIERARCHICAL LEVELS; AN OVERALL CONCEPT FOR A FULL AUTOMATIC CONCRETE DESIGN INCLUDING THE EDUCATION OF CONCRETE. THE CASE MatrixFrame® VERSUS EuroCadCrete."

Ir. Ron Weener

ABSTRACT:

1. The exception proves the rule; Knowledge Based Automatic Concrete Design
From the early 80 till the second half of the nineties the software Matrix developed for civil engineers distinguished itself by an extremely enforced integrated approach. A complete structural design including the generation of drawings could be realized at once, with one press on the button. In this concept the user had little or no influence on unforeseen situations or shortcomings in the automatic analysis of boundary conditions or the automatic design. The fact that we were secured of the cooperation of civil engineers (experts) concerning improvements makes it possible for us to make our knowledge based system even more complete.

2. The exception becomes the rule; Interactive Concrete Engineering
A disadvantage of a full automatic structural design is the existence of exceptional cases. Every case needs to be programmed which leads to a huge programming effort. In the new concept the software is based on a structure very close related to the level of code checking. All the relevant parameters can be manipulated. The link to the code is absolutely clear by the visualization of the applied code article as well as the provided value and the required value.

3. The 80-20 rule; The concept of hierarchical levels
80% of his time a civil engineer is using only 20% of the functionality of his software for structural analysis. A program doesn't need to be too complex for daily use. When you think in different levels you can manage the 80% for daily use, as well as the 20% for the advanced topics in 1 program.
The computer, using generative processes, without intervening interactions can work out 80% of all calculations. When you think in levels it is possible to work out the other 20% by the same program.

4. Ruling by exception; Computer Aided Learning system
10 years ago the TU-Delft developed a CAD exercise. This CAD exercise was developed in order to support students in dimensioning, analyzing and detailing concrete structures, after the introductory in their third academic year. EuroCadCrete is a continuation of this CAD exercise and is based on MatrixFrame® 2D-Frame and on the experience of the TU-Delft during the lessons of the CAD Concrete exercise. A learning system like EuroCadCrete is a combination of the lowest level of concrete code checking, and the advanced level of parametric study on the other side, within the concept of hierarchical levels.

KEYWORDS: Structural Analysis, CAD, Concrete structures, Eurocode, CAL, Computer Aided Learning, Knowledge Based Systems

1. THE EXCEPTION PROVES THE RULE; KNOWLEDGE BASED AUTOMATIC CONCRETE DESIGN

Our company MATRIX Software BV was founded in The Netherlands in 1983. The first years we developed structural analysis software for 2D-Frame, Grillage and 1D-Beam, called MatrixFrame® [1)]. So far, so good, not really startling news until now.

In the second stage of MatrixFrame® we developed code checking software for concrete and steel structures as well as for designing joint connections. According to our slogan "*We continue, where others stops*", we developed code checking software fully integrated with the structural analysis part. Here we met the first problems.

In comparison with structural analysis software, software for code checking is rather fuzzy. Of course it is possible that inconvenient things may occur in structural analysis, like singular matrices or loss of accuracy, but these things are related to the mathematical bases of the structural analysis theory themselves. In the course of time we solved most of the problems. For example, loss of accuracy in case of large differences in stiffness or in case of non-linear analysis we solved by automatic division. Singular matrices in case of two or more releases acting on the same node by automatic modification of the stiffness matrix, and singularity in case of loss of stability by suggestion of additional supports.

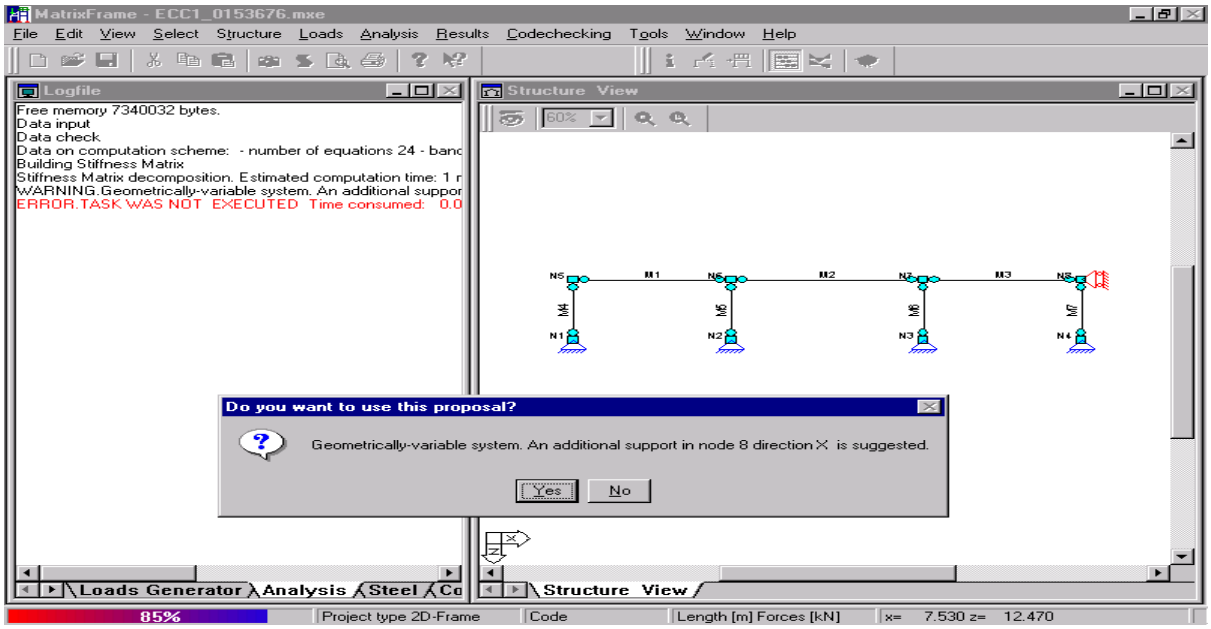

Figure 1. Practical solution for singular matrices

Back to the problems related to code checking. Actually we can distinguish five types of data for code checking:

1. Geometrical data (system length, boundary conditions, ..)
2. Section and material properties
3. Forces (internal forces, modified forces)
4. Additional data (like lateral buckling supports for steel structures and reinforcement for concrete structures)
5. Calculated data (formulas in the code, algorithms, results between)

[1)] The previous trade name of MatrixFrame® was BouwRaam® (MsDOS)

When we compare a stand-alone program with an integrated program, the stand-alone approach has a lot of disadvantages. The user should enter manually all data for data types 1 to 4. Especially for Forces (type 3) this is not so convenient, because each time the user can check only one combination of forces. The integrated version can check all combinations on all positions to determine the decisive combination and position. Of course the program should know which combination is responsible for which check, but MatrixFrame® knows the difference between combinations related to Ultimate and Serviceability Limit State.

Joining the code checking part to the structural analysis part is done very often in a straight way by copying data from one application to another one. After selection of a member, entering Additional data (type 5), some Geometrical data like boundary conditions (type 1) and modified forces (type 3), the necessary data from the structural analysis part is copied to code checking part, after which the check for this single member can be performed. This way is a Step-by-step approach.

We decided to join the code checking part to the structural analysis part in a different way. We moved forward all Additional data entry (type 4). In this way the user will be able to enter all the necessary information at once, after which the check for all members can be performed at the same time. To get a higher level of integration we tried to get as much information as possible from the geometry and make this data suitable for the code checking. This is what we called the really fully integrated approach.

Interpretation of geometry is not so easy as it looks like. Of course, for standard situations the only thing you have to do is to look to the end of a member and determine the boundary conditions. But, when boundary conditions are not clear, or boundary conditions are dependent on connecting members, there are a lot of uncertain situations.

In this example the field definition, which we need to know for deflections check and lateral buckling check, is not clear: do we have one or two fields?	*In this example the boundary conditions we need for calculating buckling length are not clear.*

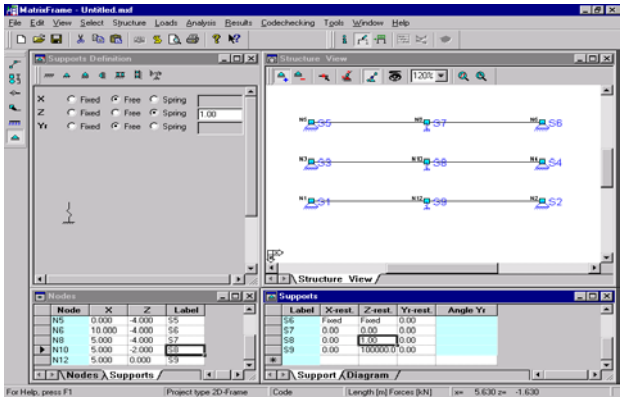

Figure 2. Boundary conditions are not clear

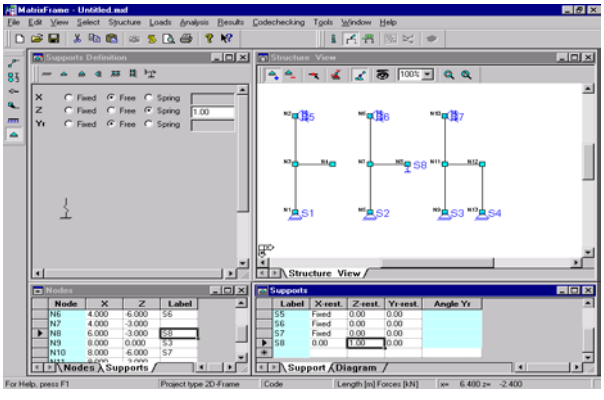

Figure 3. Boundary conditions dependent on connecting members

It's obvious that I can give a lot of examples related to the difficulties we met during interpretation of geometrical data. During implementation we got a lot of examples from our customers which gave bad interpretation of geometrical data. Every time we managed to solve these exceptions, together with our customers. It was interesting to see the great diversity of structures our customers send for review.

To improve the efficiency of MatrixFrame®, we also tried to reduce Additional data entry (type 4). Generating detailed additional data for code checking, for example reinforcement

bar combinations and the position of reinforcement bars, actually we automated the engineering design process. At first we started to examine a lot of our customers concerning the design process in civil engineering. In conclusion we can say that every customer has his own specific way of designing, but that there is a correspondence in the parameters; only the value differs.

After this investigation we started programming. To make the program easy to learn and easy to use, all information was generated without interaction of the customer. Most of the user related settings are stored in a general maintenance part, because we figured out that the value of the design parameters doesn't change very often.

In these examples the allowed size of the gap and differences between bar combinations on one position, as well as the trade length do have different values.

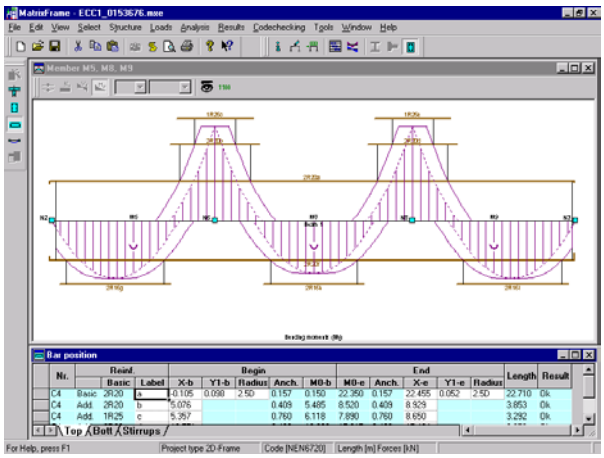

Figure 4. Concrete design based on theoretical reinforcement

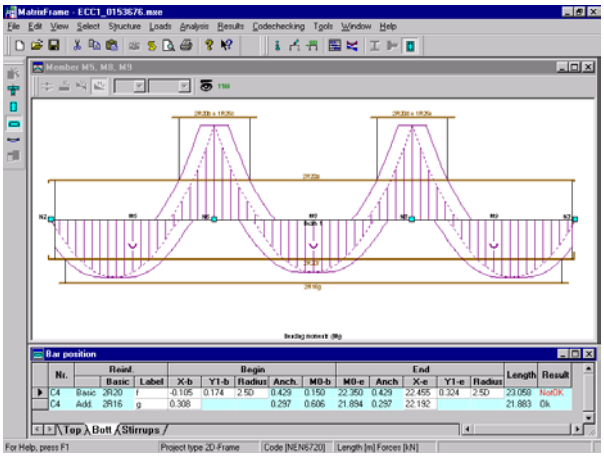

Figure 5. Concrete design based on practical reinforcement

The decision to generate all data without user interaction introduced a secondary effect we never expected. Hundreds of customers designed steel and concrete with our software, and several times they encountered the limits of the design algorithm and the relative lack of flexibility of the design parameters. The exception proves the rule. Our customers sent us their examples of structures, and every time we added new design rules to our software. The fact that we were secured of the co-operation of civil engineers (experts) concerning improvements made it possible for us to make our knowledge based system even more complete.

The final result of this MsDOS based program was a complete structural design program for automatic generation of climatic and live loads and design of concrete, steel, joint connection, including the generation of drawings which could be realised at once, with one press on the button.

2. THE EXCEPTION BECOMES THE RULE; INTERACTIVE CONCRETE ENGINEERING

A disadvantage of a full automatic structural design is the existence of exceptional cases. Every case needs to be programmed which leads to a huge programming effort. Sometimes customers became discontent, because a work-a-round was not available, and the engineer has to deliver his job in time. Who can wait on the upgrade of the software, if the contractor already has started?

Another point of view is the economical aspect. In order to complete the last 20% you need a programming effort of 80% of the total period. A lot of time should be spent on programming for situations that may never occur for the average group of customers.

The last disadvantage I will mention is the imposed way of practising engineering: only the most economical solution fits. In renovation jobs for example, the civil engineer has to accept the existing situation. Also civil engineers, working for the government, approving the calculations for building allowance are absolutely not interested in economical design.

In 1997 we started redesigning our software to the Windows platform. In the new concept the software is based on a structure very close related to the level of code checking. All the relevant parameters can be manipulated. The link to the code is absolutely clear by the visualisation of the applied code article as well as the provided value and the required value.

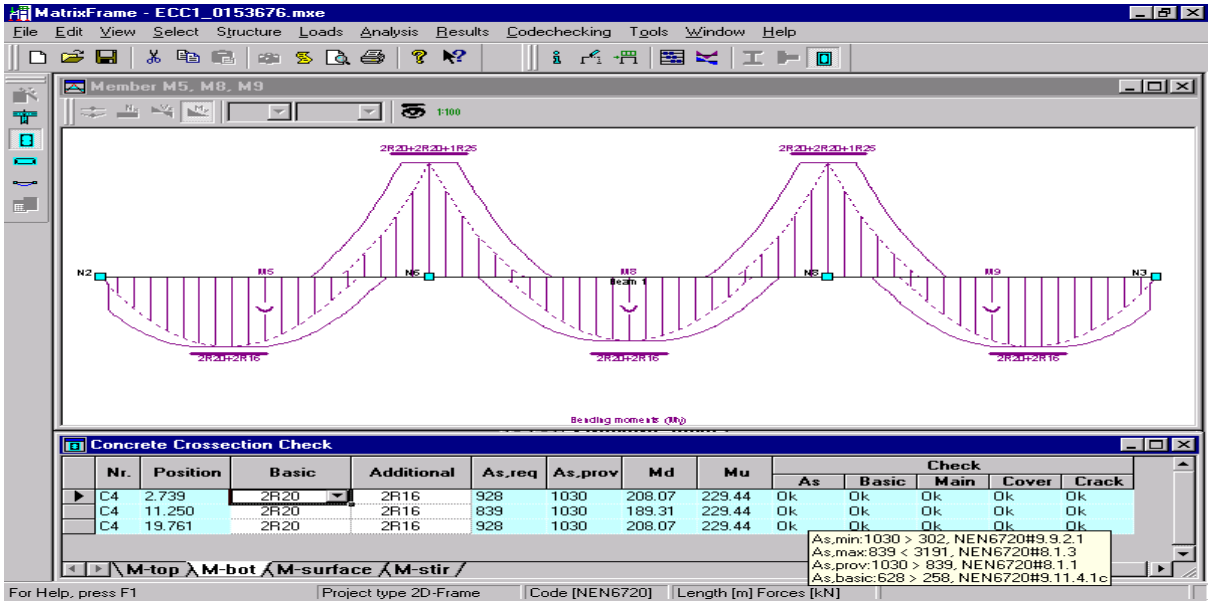

Figure 6. Close relation to the code

The interactive approach has given the right answer towards exceptions, because the exception becomes the rule. It doesn't give the right answer on the benefits of the good old DOS software: Velocity and economical automated design.

3. THE 80-20 RULE; THE CONCEPT OF HIERARCHICAL LEVELS

80% of his time a civil engineer is using only 20% of the functionality of his software for structural analysis. A program doesn't need to be too complex for daily use. When you think in different levels you can manage the 80% for daily use, as well as the 20% for the advanced topics in 1 program.

The basis of the concept of hierarchical levels is the calculation core with a complete functionality. When an easier variant of the calculation is needed, certain steps are omitted or the intermediate values are generated with a separate algorithm. In that case the same basic algorithm can be used, with one or more intermediate generation procedures.

A user interface very close to the core needs a lot of data entry, because there are a lot of parameters. A user interface for the more simple related calculations only needs a couple of entry fields. For the train of thought the 80-20 rule can be illustrative.

- In 80% of the time simplified calculations are performed, in 20% the advanced ones,
- The simplified calculations only need 20% of the total data entry, the other 80% are needed for advanced calculations
- The standard simplified calculations cover 20% of the functionality, the advanced ones 80%

We can classify the data entry very close to the core as level zero in the hierarchy. The standard input of simplified data is classified as level two in the hierarchy. Between these two levels we can classify level one for advanced input

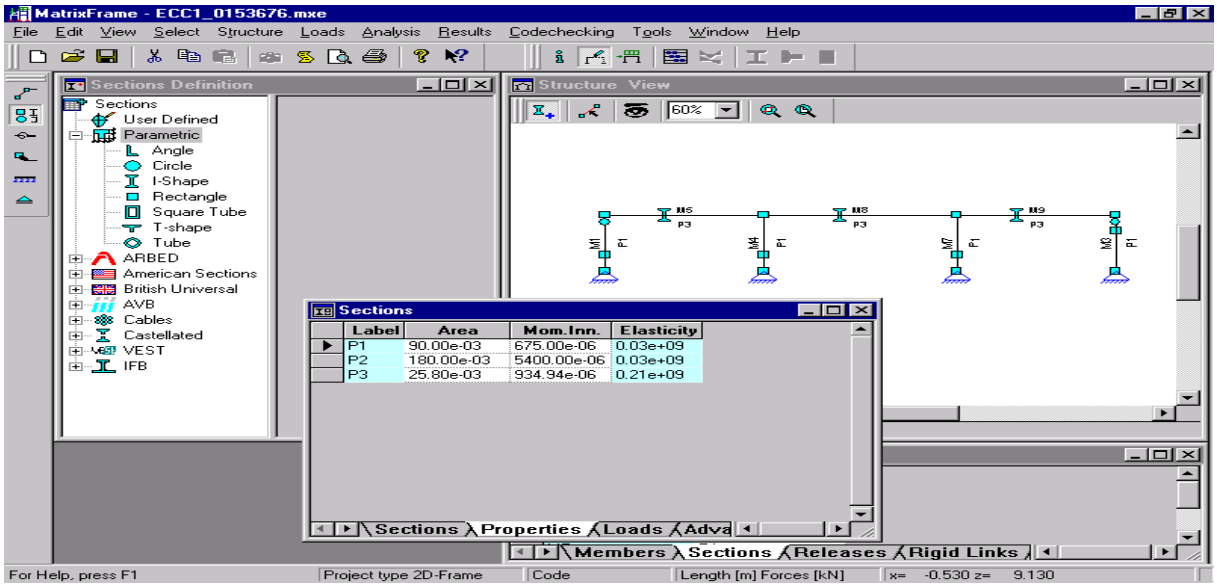

Figure 7. Level 0 "Sections properties"

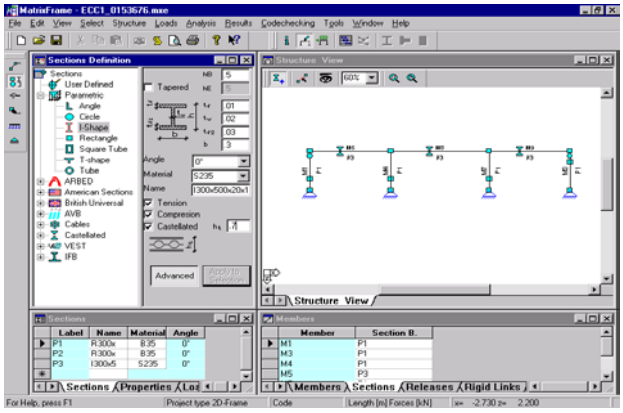

Figure 8. Level 1 "Advanced parametric sections"

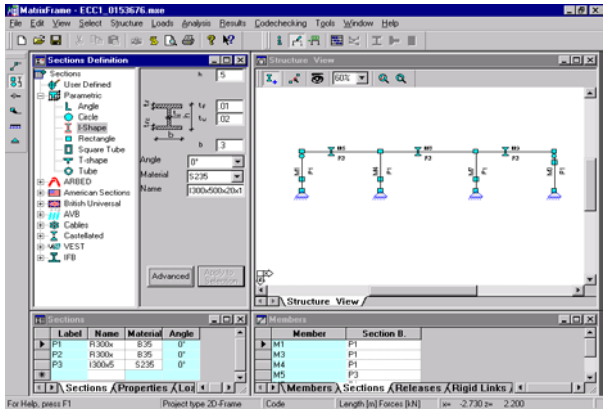

Figure 9. Level 2 "Standard parametric sections"

Every time, when data is generated for the benefit of a lower level, a new level is introduced. This is the real challenge for the software developer. Although the functionality of each higher level decreases, the performance and efficiency for the user increases. From level 0 "Core", level 1 "Advanced" and level 2 "Standard simplified" the next level 3 will bring the generation and optimising procedures. Our MsDOS software was based on level 3.

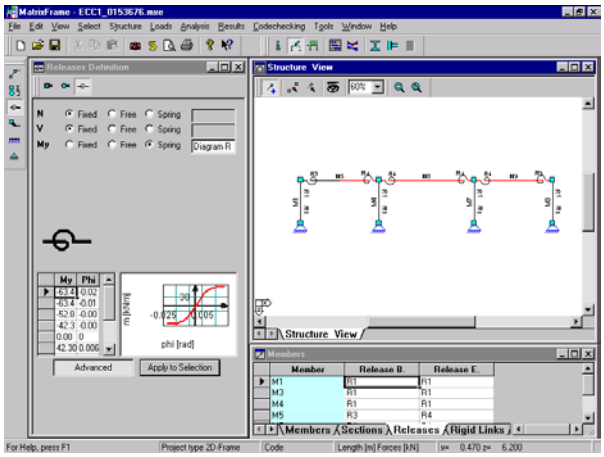

Figure 10. Level 1 "Advanced releases, input of non- linear spring diagram"

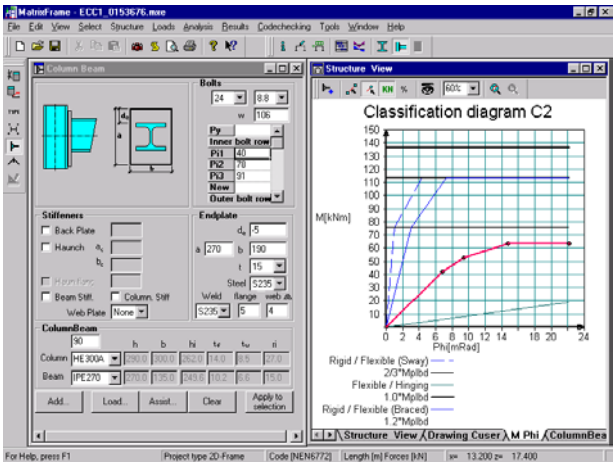

Figure 11. Level 3 "M-ϕ diagram of joint connections for generation of the non linear spring diagram"

In comparison with our MsDOS based software the introduction of hierarchical levels gives more flexibility and more functionality. During the design process the civil engineer starts on the highest level, when the functionality doesn't fit or when the answers are not in the right way, he descents to a lower level and he should modify only the specific items. This is a perfect combination of optimal user interaction and functionality on one side and on the other side a very fast and optimised design process.

It is not always necessary to split different levels in the user interface; sometimes it is more convenient to combine more levels in one window. This doesn't undermine the concept, if the basic rule is taking into account. According to the basic rule it always should be possible to enter data on the lower lever. This concept of hierarchical levels is very clear implemented in the loads part of MatrixFrame®:

- Level 0 (Core): Nodal load, imposed deformation (Primary forces)
- Level 1 (Extended) Trapezoidal load
- Level 2 (Simplified) Distributed load, Triangular load, Temperature load
- Level 3 (Generation) Load generator for Dead load, Live load, Wind Load and Snow load

The load generator generates loads according to the code. These loads are stored in the standard interface for load definition. So, after a generated procedure the user can change, remove and add manually all generated loads.

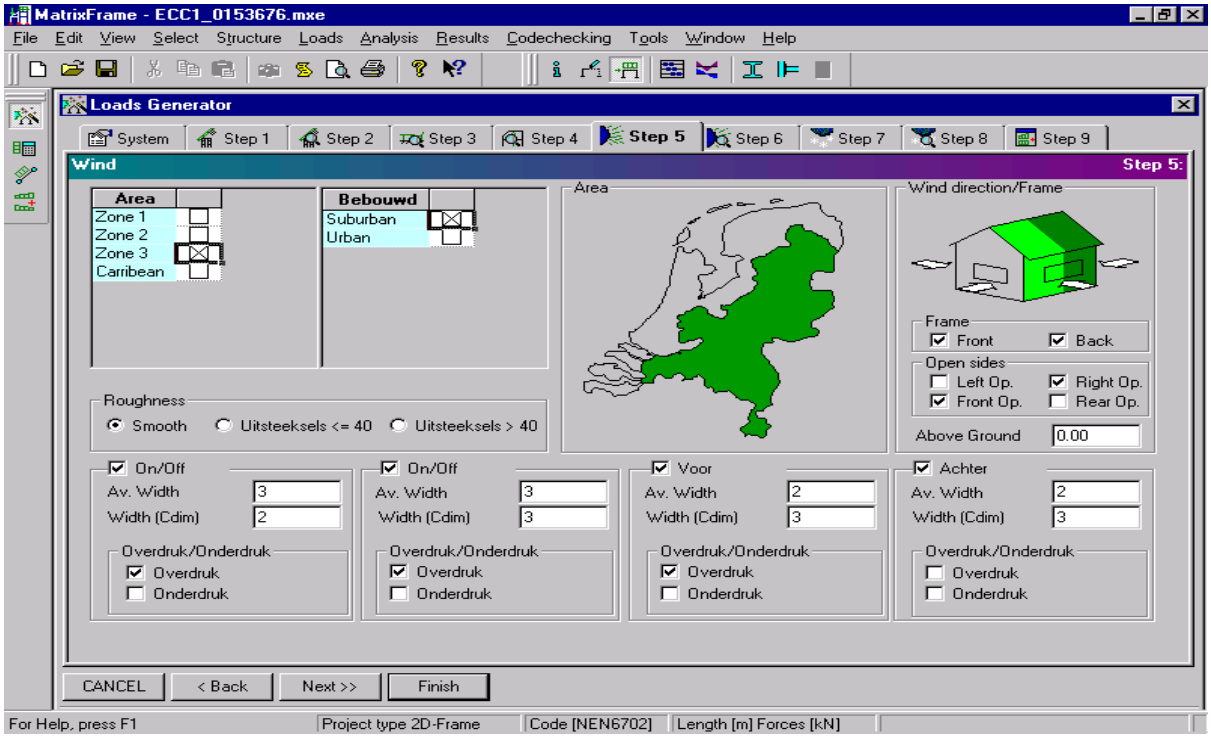

Figure 12. Level 3 "Load generator for Wind loads"

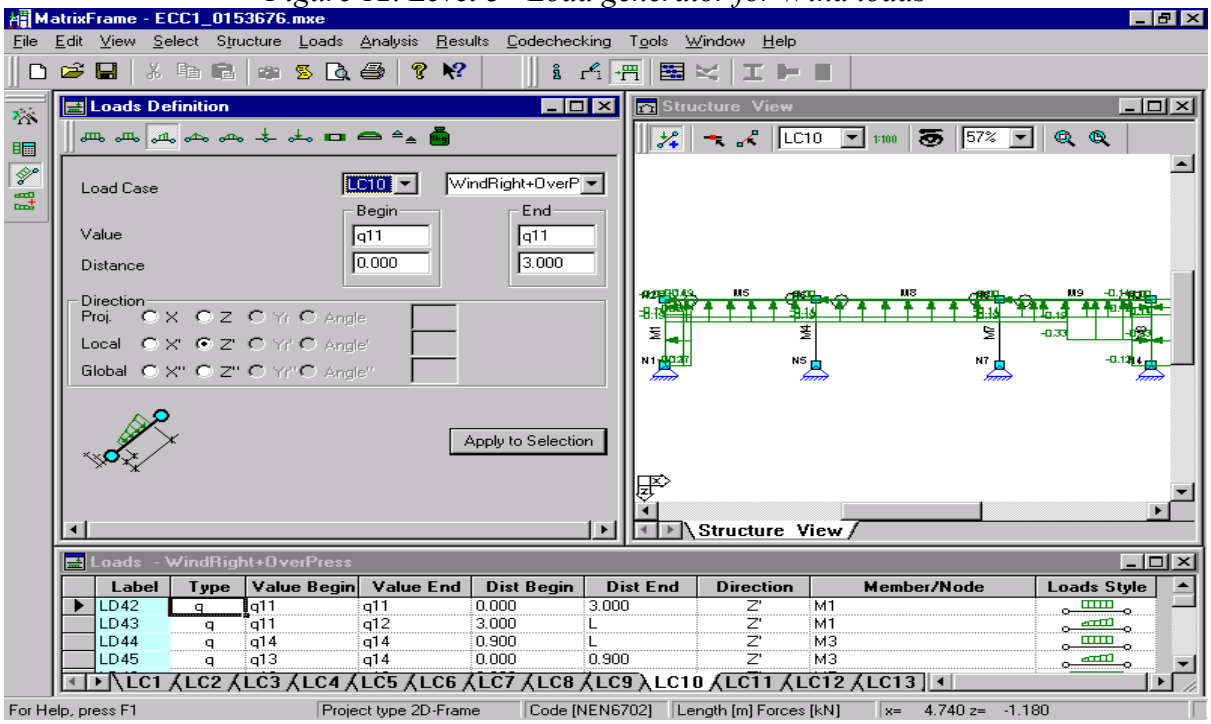

Figure 13. Level 2 "Load definition window" and Level 1 "Loads table"

In the next version of MatrixFrame® we will start with implementation of level 4. To be continued…

4. RULING BY EXCEPTION; COMPUTER AIDED LEARNING SYSTEM

10 years ago the TU-Delft developed a CAD exercise. During this period more than 1500 students used these exercises for their training. This CAD exercise was developed in order to support students in dimensioning, analysing and detailing concrete structures, after the introductory lecture in designing and constructing concrete in their third academic year.

EuroCadCrete is a continuation of the CAD exercise mentioned above and is based on the educational version of MatrixFrame® 2D-Frame and on the experience of the TU-Delft during the lessons of the CAD Concrete exercise. Students can define the structural analysis part of the exercise at home. Then the prepared job can be worked out according to the Eurocode[2)] in the EuroCadCrete wizard. By means of exercises and by providing interactive tools students gain a clear insight in the nature of reinforced concrete, which is the aim of this job.

The check mechanism of EuroCadCrete is based on the concrete module of MatrixFrame® used by the civil engineers (further the name MatrixFrame® is used for this application). If the code changes, the modification is done for MatrixFrame® as well as for EuroCadCrete in one time. In EuroCadCrete the student has to enter primary data step by step on a very low level. The entered data is proved by EuroCadCrete immediately by using the next level mechanism we use in MatrixFrame®. For example MatrixFrame® the covering is generated automatically after entering the environment class. In EuroCadCrete the student has to enter the covering too after entering the environment class. The covering is lower level information, because it can be generated in the same way as in MatrixFrame®. That's the way the check mechanism is working. Of course the example of the covering is quite simple, but the same mechanism is implemented for other steps. Much more complex is the check on possible bar combinations. Here the algorithm calculates all possible bar combinations that meets the requirements concerning capacity, minimum bar distance, maximum bar distance, bar diameter, cracking, etc. In EuroCadCrete the entered value is coloured red in case of the wrong answer. In MatrixFrame® the user can select the proper bar combination from a drop list (this drop list contains all possible bar combinations). In MatrixFrame® we check also the user data, but we show this information in a tooltip with all intermediate results as well as the link to the code (see figure 5). In EuroCadCrete the student can't continue with the next step, until all values are good in this step. In that case he can consult assist. According to educational backgrounds, this assist will give the student additional information in 3 steps. The last stage of consulting this assist will deliver him the right answers, in the same way we display the result of the check in the tooltip in MatrixFrame®, and the student can continue with the next step in de EuroCadCrete wizard. In this way the student is able to navigate through the exercise without intervention of a teacher: Ruling by exception.

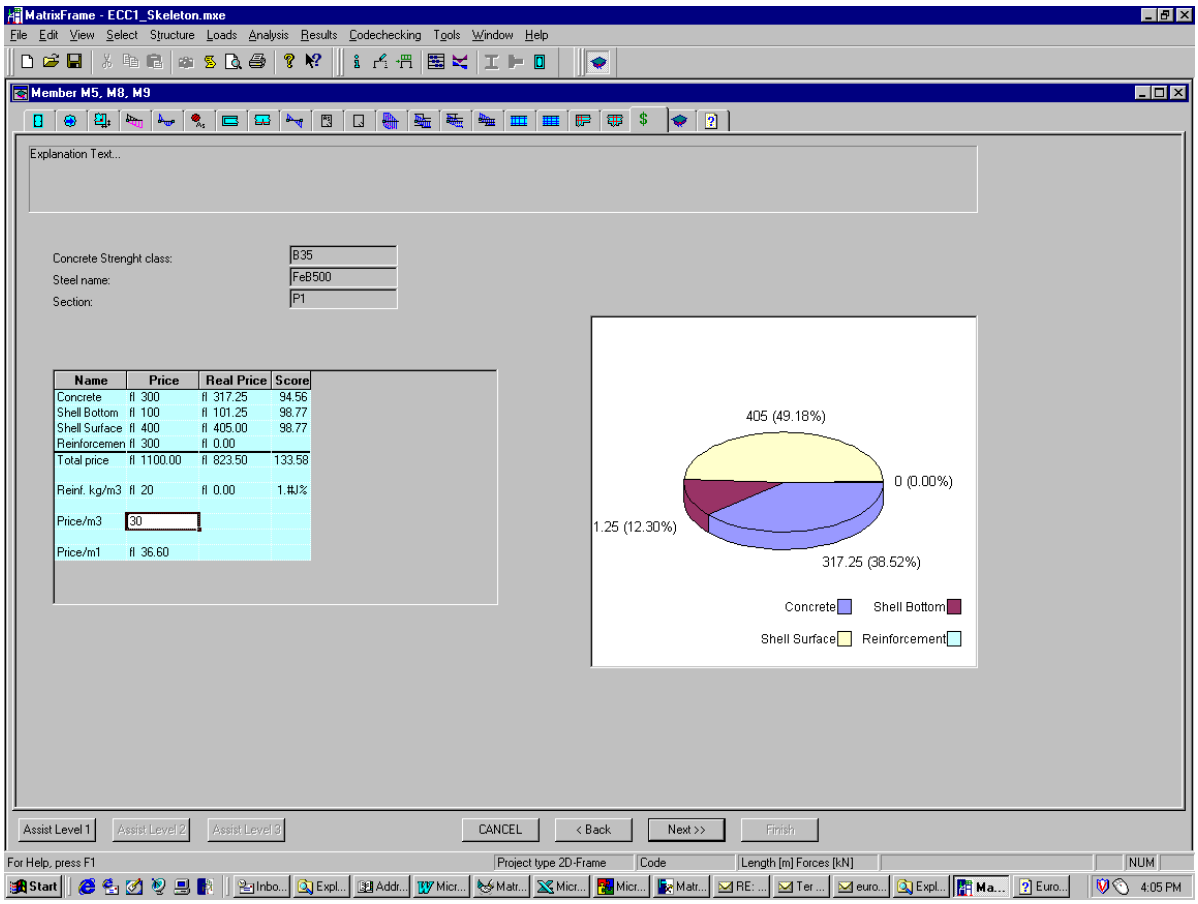

Figure 15. EuroCadCrete

The last part of the exercise gives the student the opportunity to perform parametric studies, using the "What if" analysis. The student can see the cost price effect when:

- What if: "the height of the beam is the minimum height"
- What if: "the height of the beam is the maximum height"
- What if: "the load factor increases"

EuroCadCrete is using the level 3 based MatrixFrame® automatic concrete design algorithm for complete (hidden) redesign of the structure with the new parameters. After redesigning, including bar combination and positioning, the cost price estimation is performed. This calculated cost price is used to show the effect of the changes of the parameters.

A learning system like EuroCadCrete is a combination of, on the one side, a Graphical User Interface based on the lowest level, and a check mechanism and parametric study on the other side, which is based on the higher levels within the concept of hierarchical levels. In April 2000 more than 100 students will perform their practical exercise with EuroCadCrete. After this fire-baptism the program is available for Universities and Technical schools all over Europe.

CONCLUSION

The case MatrixFrame® versus EuroCadCrete proves that the concept of hierarchical levels is suitable for different purposes. This concept is an overall concept for a full automatic concrete design including the education of concrete. Of course the concept is also suitable for other applications in the field of civil engineering, like steel and structural analysis.

[2)] The first version of EuroCadCrete was according to the Dutch code

REFERENCES

Galjaard ir. J (Hans).C., Vos Prof.ir. Charles J., Kunst Drs. Sabine:
"EuroCadCrete, an improved design exercise in reinforced concrete",
CIB W78 IABSE EG-SEA-AI, Reykjavik June 28-30, 2000