IEA

International Energy Agency

#5862

INTERNATIONAL ENERGY AGENCY

ENERGY CONSERVATION IN BUILDING AND COMMUNITY SYSTEMS PROGRAMME

EXECUTIVE COMMITTEE MEETING

TECHNICAL DAY - SOPHIA ANTIPOLIS

2nd JUNE 1992

IEA ANNEX 23

MULTIZONE AIR FLOW AND POLLUTANT TRANSPORT MODELLING

R. PELLETRET, S. SOUBRA, W. KEILHOLZ, A. MELOUK
(CSTB, France)

IEA-Energy Conservation in Buildings and Community Systems

Annex 23 Multizone Air Flow and Pollutant Transport Modelling

Executive Committee Meeting
June 2-4, 1992
Sophia Antipolis France

Annex 23, Subtask 1: System Development. An Intelligent Simulation Environment for Multizone Air Flow Modelling.

Dr. Roger Pelletret, Souheil Soubra, Werner Keilholz, Ahmed Melouk
Centre Scientifique et Technique du Bâtiment
BP 209
06904 SOPHIA ANTIPOLIS CEDEX
FRANCE

1. Introduction.

The objective of Annex 23 is to study physical phenomena causing air flow and pollutant transport in multizone buildings and to develop modules to be integrated in a multizone air flow modelling system.

The Annex is shared in three subtasks:

- subtask 1; System Development;
- subtask 2; Data Acquisition;
- subtask 3; Evaluation.

The goal of subtask 1 is to develop modules to be integrated in a Multizone Air Flow Modelling system (which will be based on the existing COMIS program), to store the physical knowledge associated to each module in a standard format in order to ease the reuse of this knowledge in various contexts (the models will be documented with 'PROFORMAs' [Pelletret 92.04]), to develop a prototype of an Intelligent Simulation Environment in order to facilitate access to the modelling system, and to demonstrate the coupling of a multizone air flow modelling system with Building Energy Performance Simulation codes.

This paper focuses on the development of the Intelligent Simulation Environment adapted to Multizone Air Flow Modelling; this task will mainly be performed at CSTB with contributions of other Annex 23 participants and in relation with other tasks of Annex 23. It deals also with the problem of coupling COMIS with Building Energy Performance Simulation tools, which is an integral part of the Intelligent Simulation Environment.

The concept of Intelligent Simulation Environment is based on some main ideas which are presented in this paper. Anyway, an ISE involves a graphical front-end; an example of a graphical front-end for COMIS is also presented.

2. The concept of Intelligent Simulation Environment

The concept of Intelligent Simulation Environment is under development at CSTB in the frame of the IISIBât project [Pelletret 90.06, Soubra 91.11]. On the first glance, the IISIBât project might look like just another attempt to build a graphical front-end around existing Building Performance Evaluators. It is indeed a major goal of the ISE to provide a sophisticated graphical environment to allow the user to enter information in a straightforward way, instead of dealing with 'macro-languages' or other highly confusing forms of input files needed by common building simulation tools. So, the ISE is also a graphical front-end. But is more.

The ISE will allow the sharing of data between various simulation tools, the coupling of simulation tools; it will involve build-in checking and helping functions in order to assist the user when modelling the system, simulating it or analyzing the results. The ISE will also include a complete model documentation; the models are documented with the standard format, so-called 'PROFORMA'; the ISE will allow the connection with external data bases.

2.1. Sharing data via the 'Integrated Data Model'

The ISE will enable simulation tools to communicate on different levels (sharing data and sharing part of assemblies).

Sharing data is the simplest way of communication. The starting point for this concept is the trivial observation that all the simulation tools involved operate the same 'object': a building.

It is an obvious demand of users who apply a set of tools that information commonly needed by several tools should not have to be entered more than once. This requirement is met by Integrated Data Model (IDM) [Dubois 92.01] which is an integral part of the ISE. The Integrated Data Model is a description of a building at such a high level of abstraction that it can be used as a common base for all Building Simulation Evaluators that are operated through the ISE; that means, among others, that the sum of all fields of all classes of its internal object oriented representation contains all information ever needed by any of the tools. Adding new tools to the ISE means, among others, adding new fields and/or classes to the Integrated Data Model used by the ISE.

Thus the IDM concept allows exchanging data between simulation tools on the level of components (the ISE build-in representation of the components is independent of the simulation tools): a library of components administered by the ISE can be shared by a set of simulation tools. Each component has a piece of information attached to it that consists of a list of simulation tools to which it can be applied. 'Shared' components can be used by more than one tool to build up systems of interconnected models and macro-models in an assembly window, without entering the model parameters each time a component is used in an assembly.

Furthermore, the translation of entire assemblies (of models) between different simulation tools can be considered: an assembly describing one aspect of a system (such as a thermal model of a building) could ideally be translated into an assembly describing another aspect of the same system (e.g. a description of the same building in terms of COMIS objects). In the case of some simple, straightforward assemblies, an automatic translation of both the models and the logical links between them may be feasible. In most cases, however, the best we can hope for will be a semi-automated translation: many models used in one view of a given problem do not have counterparts in other views. In such cases, the assembly yielded by the translation will be a rather sparse network of models because of missing correspondences between the components involved. Missing components will have to be transferred manually, and missing links will have to be defined.

2.2. Coupling of Simulation Tools

In general, the coupling of simulations tools aims to enlarge the range of possible applications and to increase the accuracy of the results. For instance, the coupling of COMIS with Building Energy Performance computation codes will enable accurate performance assessment of active air heating and cooling systems (i.e. calculation of the heat gains and losses of the building and of the cooling or heating power provided by the HVAC system), passive cooling (i.e. determination of the energy released by passive cooling during the night), heat transport between zones (i.e. calculation of the heat transported by air circulating between zones: accurate computations are useful for passive solar buildings, where excess heat in irradiated part should be brought to the shaded part of the building by natural convection), ventilation heat losses, etc.

The goal of task 1.07 'Use of COMIS for Building Energy Performance computations' is to bring new ideas about the advantages and inconveniences of the various possibilities which exist to couple an Air Flow computation code with a Building Energy Performance simulation code. There are four possibilities of 'coupling' but only the first three of them will be considered in the frame of Annex 23.

- Sequential coupling.
  Sequential coupling is the most straightforward and also most time-efficient coupling method. It simply consists of invoking a first simulation tool (COMIS), yielding a huge matrix where one dimension is time. This result is fed into another simulation run, invoking a Building Energy Performance computation code. This coupling method is too crude for many problems, but is sometimes sufficient. This work should be done in the frame of the task 1.07 of Annex 23 by coupling COMIS with, among others, DOE-2 and BLAST.

- Incorporation of COMIS into a Building Energy Performance computation code.
  Incorporation of COMIS into a BEP computation code means in fact using COMVEN as a huge subroutine. This work should be done in the frame of the task 1.07 of Annex 23 by translating COMVEN as a TRNSYS type.

- 'Ping-pong coupling'.
  The "Ping-pong" method consists of alternately starting two simulation tools, each using variables calculated by the other. A supervisor should be built in the ISE; the supervisor should generate a time-loop and alternately starts a one-step simulation run with the two codes involved. Each tool solves its own problem with its own solving method. At each time step, a decision has to be made, whether or not the simulation converges. In this method, the supervisor compares, at each time step, the results of iteration i with those of iteration i-1 and decides whether or not the results are valid. The advantage of this method is that when a generic coupling environment exists, various codes can be coupled without having to rewrite specific subroutines. This work should be done in the frame of the task 1.07 of Annex 23 by coupling TRNSYS and COMVEN.

- The fourth method should yield the most accurate results; it consists of extracting the knowledge contained in various Building Performance Evaluators and translating this physical knowledge into an Integrated Data Model, which can then be translated into models of a generic solver (domain independent). This method might be pursued in future versions of the Intelligent Simulation Environment.

2.3. Assisting the user during the process of Modelling / Simulating / Analyzing the results.

A user can interact with an Building Performance Evaluator at different levels:

- he can try to improve the program by modifying existing components or adding new components;

- he may want to describe a real building and simulation parameters;

- he may want to simulate the system under investigation;

- he may need help to analyze the results or to get as easily as possible the right solution (this is the idea of optimization).

At each stage, he may need information both on the simulation environment (how to use the simulation tool? What to do next?) and the physics involved (what are the limits of application? What are the constraints when using this component? etc).

For each of these problems, the ISE can bring an answer.

2.3.1. Introduction of new models by the user

The ISE should provide a simple mechanism for introducing new models for the Building Performance Evaluators involved. Since, however, the user who wants to add new models has to write subroutines in a programming language such as FORTRAN or C and link these subroutines to the respective Building Performance Evaluators, some expertise is required for this. Nevertheless, the ISE can ease the creation and linking process by providing standardized interfaces to the Building Performance Evaluators involved (e.g., a text editor containing a standard function header in FORTRAN, listing all the pass parameters available). The compiling and linking process of the user written routines can be automated to a certain extend.

On the other hand, Building Performance Evaluators like COMIS make an extension by the user difficult due to their non-modular structure. We call this kind of BPE 'monolithic'. Monolithic simulation codes will be available to the user through the ISE. However, introduction of new models requires a detailed knowledge of the overall structure of such a BPE: this has to be described in a detailed programming guide. Administration and exchange of models will be more difficult than with modular codes, since new models (created by different authors) will have to be merged taking into account all the implications of the merging.

2.3.2. Modelling of a complex system

We assume that any system can be modelled by assemblies of components or sub-systems. Then to describe a complex system, a user will have to choose components in models libraries and to link them in an assembly. If the project under investigation already exists, the user can request its IDM representation to start the assembly and, at least, to fill the slots which have already been valued. But, in addition to that, an ISE can provide more helping or checking functions to the user.

To ease the use of the ISE it-self, helping functions have to be included; they make the simulation environment self-explanatory.

Checking functions will assure that the values entered by the user always meet the requirements of the BPE used; this can be done taking into account the various units that the user can use.

To allow even unexperienced users to take profit of the advanced features of building simulation tools we hope to put as much expert knowledge into the ISE as possible. However, the nature (i.e. the degree of complexity, the structure, etc.) of expert rules to be included in the ISE are not fixed yet, due to the genericity of the ISE: since the ISE should be open for other BPEs than the ones we are using at the moment (TRNSYS and COMIS), possible extensions to the expert system component are difficult to foresee. Even if one limits the application of a future expert system to assisting the user in assembling systems with TRNSYS and COMIS, the rules found so far are rather vague. So far, a number of possible categories for expert rules in the frame of the IISIBât project have been listed, but no concrete rules have been formulated yet:

- Input data rules
  Expert knowledge could be used to help the user in choosing adapted values considering the simulation goal and the default values stored in the PROFORMAs.

- Simulation goal rules
  To develop simulation goal rules needs the definition of the possible simulation goals [Roulet 91.11]. Then, starting from a simulation goal specified by the user, the system could ease the assembly of models in several ways:

  - by not allowing to use a certain number of models which are not in line with the simulation goal (the information needed for this should be contained under the heading "Application List" of the PROFORMA file);

  - by helping the user to choose modules adapted to his simulation goal (the information needed for this should be contained under the heading "Application List" of the PROFORMA file);

  - by providing for each simulation goal standard assemblies that the user can take as a starting point for his own assemblies;

  - by assisting in the choice of the solving method for the numerical problems (e.g., the system could provide answers to questions like "Is sequential coupling sufficient in the given assembly / simulation goal combination ? Which solver should be used ?");

  - by checking the consistency of the numerical parameters needed in relation to the simulation goal and the assembly

- Linking rules
  The goal is to assist the user in correctly linking models and choosing models that can be linked to a given model. This information should be taken from the PROFORMA file description attached to each model (this information is stored, in the PROFORMAs, under the headings 'Upstream models' and 'Associated models'). Linking rules can be considered on various levels. The simplest approach would be two lists of models attached to each model, one consisting of possible precursors (i.e., upstream models) and the other one listing possible successors (this information could be generated by an inference engine from the lists of upstream models). Furthermore, rules for linking variables (for instance, in the TRNSYS style) between models could be automated once a link between two models is established.

- General Modelling Rules (Meta-rules)
  On another level, an expert system could run through the knowledge base to propose entire networks of components, starting from some given components (at least one). Rules to ingeniously restrict this search of models are needed, since considering all possible combinations would yield a combinatorial explosion. These rules could, for example, use the data on the simulation goal.

- Consistency checking
  An expert system could apply rules to a given assembly which are to ensure the consistency of the links the user defined between models. This would go beyond the simple rules used to check if a link between two given models is allowed. More complex rules, taking several hops in a network (or the entire network) into consideration could exist (for this kind of check, the information from the PROFORMAs, under the heading 'Compatibility rule', is needed)

2.3.3. Performing the simulation

An important function of the ISE is, of course, to start simulation runs of the BPEs involved by executing the appropriate Operating System commands. For this function, the ISE will translate the internal representation of a project described by the user (which will consist of an instance of the Integrated Data Model) into an input file that can be processed by the BPEs (e.g., a 'COMIS Input File').

2.3.4. Analyzing the results

The ISE should provide comfortable tools for analyzing the results of these simulation runs, such as a curve editor.

From the user's point of view, the most interesting part of a simulation is of course the result. The standard working procedure is to run a sequence of simulations, varying a couple of parameters to reach a desired configuration. A lot of expert knowledge can be involved in this process, too. An expert system, fed with the simulation goal and rules about interdependencies of parameters could assist the user in his choice of parameters to modify and in what way. This is a design-like process: given a number of constraints, a system has to be designed that satisfies as many constraints as good (optimal) as possible.

In addition, functions to ease sensitivity analysis or error propagation study could be involved (for instance, a function could take in charge the preparation of the sets of parameters needed to do an optimization; for that the concept of Multi-run Interface for sensitivity Analysis developed in the frame of subtask 3 of Annex 23 could be reused).

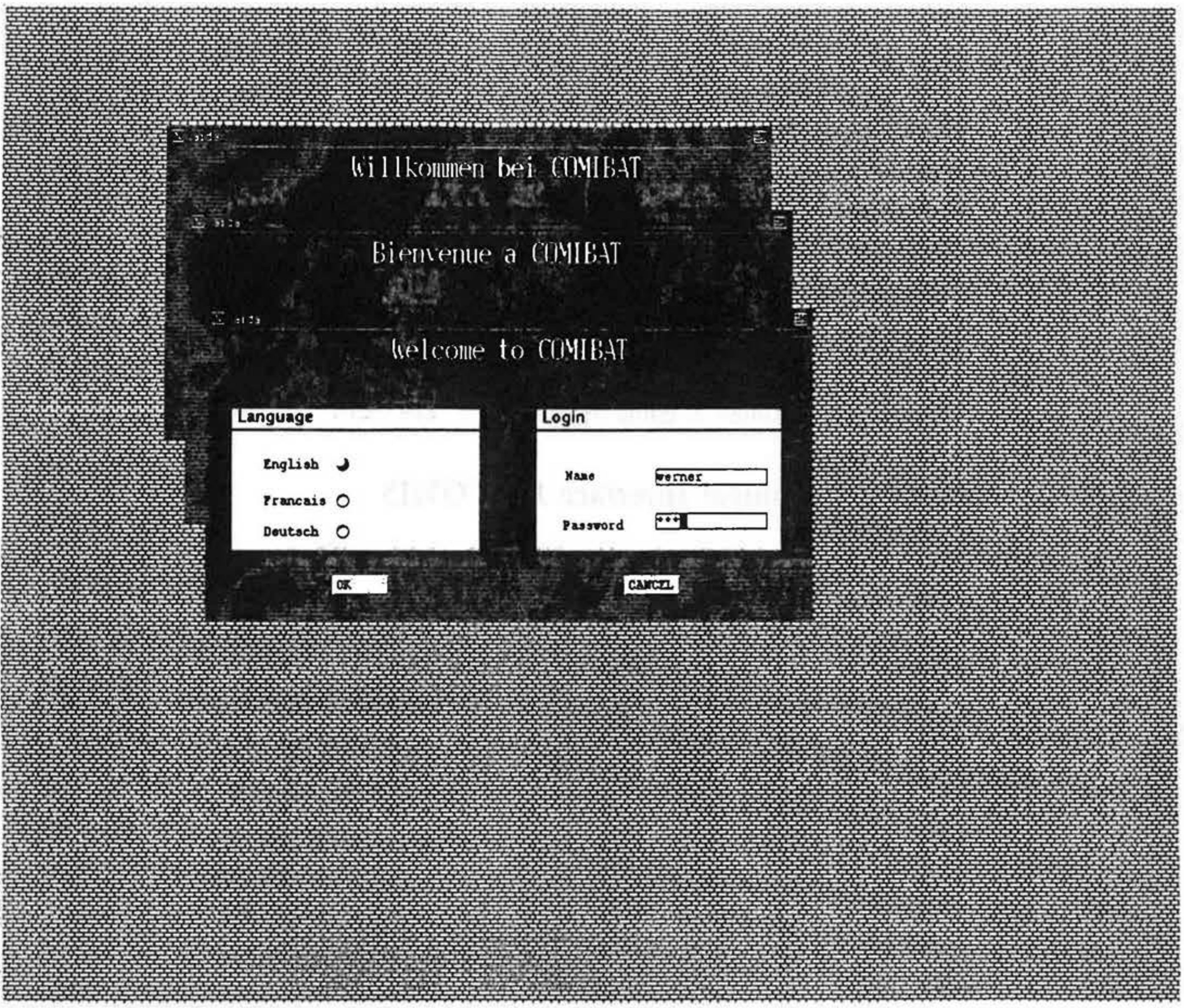
3. Example of a Graphical Interface for COMIS

We have designed an Object Oriented Model of COMIS [Keilholz 91.12] and, with this model, we have built, in the frame of the IISIBât project, an example of a Graphical Interface for COMIS [Keilholz 92.03] called COMIBât. COMIBât is a very first prototype of a graphical front-end of which the goal was to define the particularities of the Human/Machine dialogue due to the incorporation of COMIS and to test new solutions for the Human/Machine dialogue, different from those already tested with the software IISIBât/TRNSYS.

IISIBât(1) is based on the programming language Le_Lisp(2) and uses the development tools Aïda(2) and MIPS(1). This choice ensures a portability on any Unix(3) machine with the graphical standards X-Window(4) and Motif(5).

In the next pages, an example of a working session with COMIBât is presented.

(1) IISIBât and MIPS are software products developed by CSTB
(2) Le_Lisp and Aïda are commercial products distributed by ILOG S.A.
(3) Unix is a registered mark of AT&T
(4) X-Windows is a registered mark from MIT
(5) Motif is a registered mark from the Open Software Foundation


Willkommen bei COMIBAT

Bienvenue a COMIBAT

Welcome to COMIBAT

Language

English [✓]

Francais

Deutsch

Login

Name: werner

Password: ***

Fig.1: COMIBât can welcome you in three languages. The "dictionary" of COMIBât can be easily extended by any language that uses the standard ASCII character set.

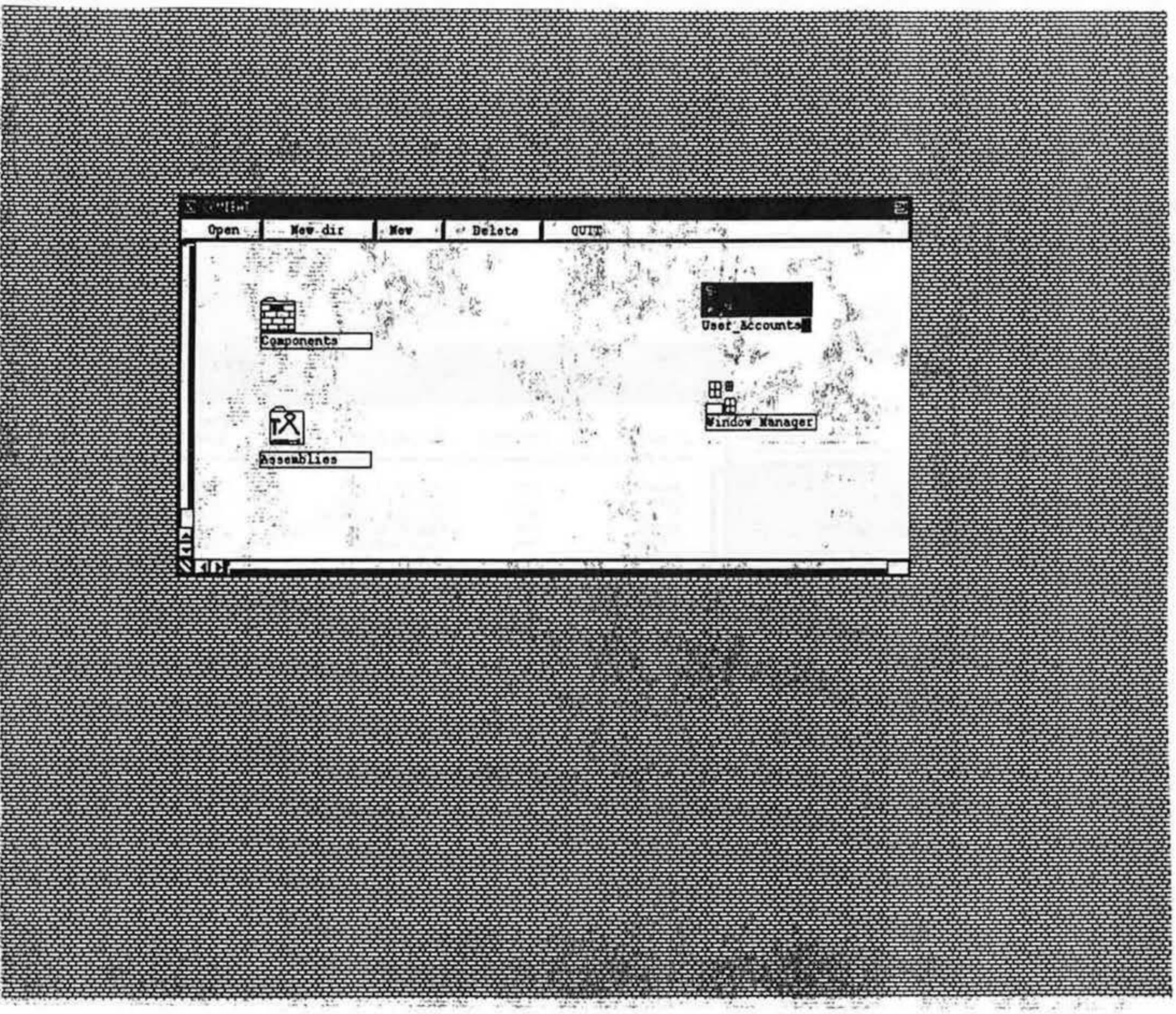
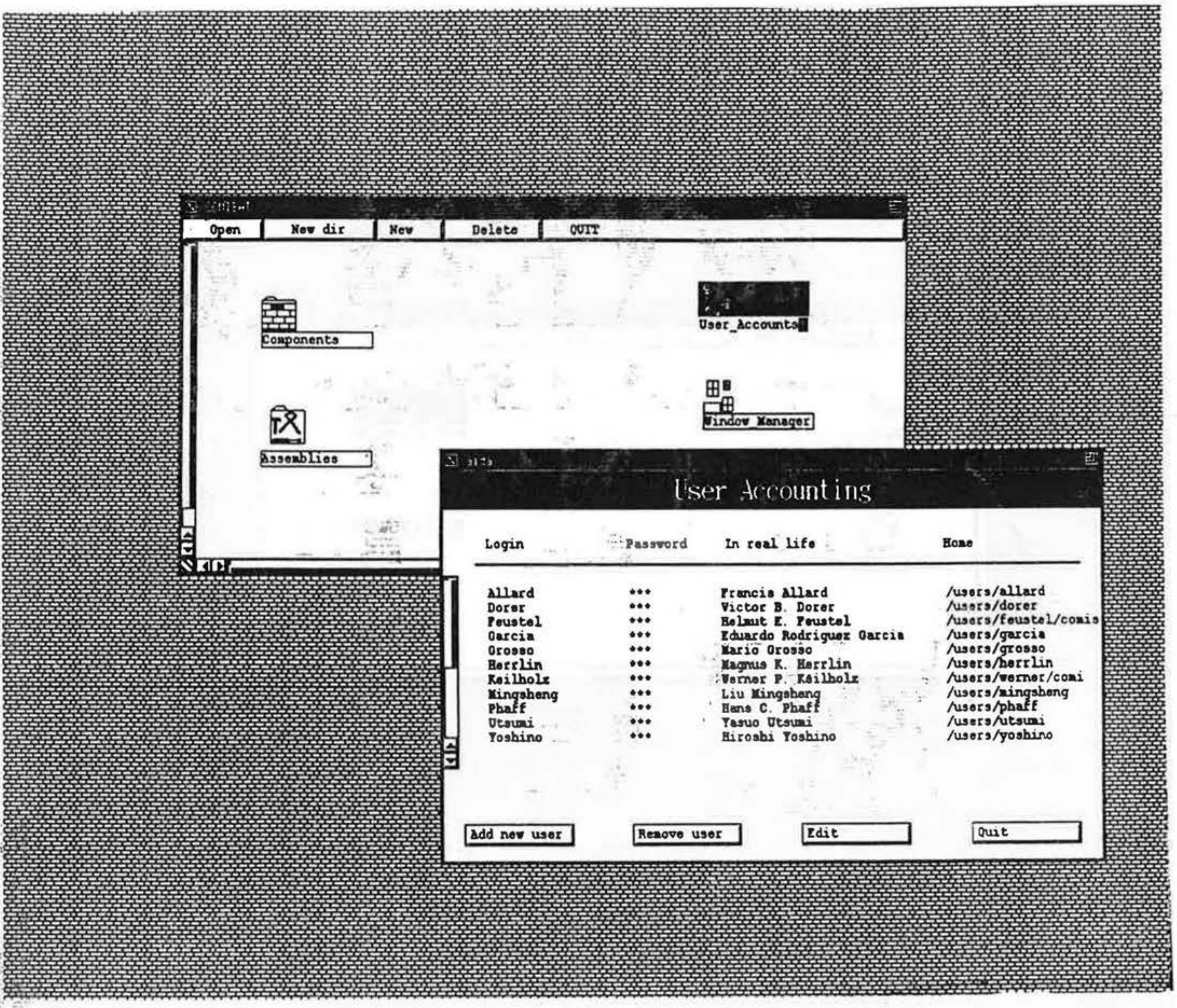


Fig.2: The main window of COMIBât contains two kinds of objects: directories (there are two on the left hand side in this example) and applications. In this example, the application 'User_Accounts' is selected (i.e., the user has clicked on this icon).



Login	Password	In real life	Home
Allard	***	Francis Allard	/users/allard
Dorer	***	Victor B. Dorer	/users/dorer
Feustel	***	Helmut E. Feustel	/users/feustel/comis
Garcia	***	Eduardo Rodriguez Garcia	/users/garcia
Grosso	***	Mario Grosso	/users/grosso
Herrlin	***	Magnus K. Herrlin	/users/herrlin
Keilholz	***	Werner P. Keilholz	/users/werner/comi
Mingsheng	***	Liu Mingsheng	/users/mingsheng
Phaff	***	Hans C. Phaff	/users/phaff
Utsumi	***	Yasuo Utsumi	/users/utsumi
Yoshino	***	Hiroshi Yoshino	/users/yoshino

Fig.3: Here the user 'opened the icon' that has been selected in figure 2; the application 'User_Accounts' is started. The user can now add new users, delete accounts, assign passwords and perform other administrative tasks. Note that this application was accessible to that user only because he was known by COMIBât as a 'super-user'; various categories of users can be defined according to working procedures; different rights to write, delete, open, etc, applications or directories are attached to each users category.

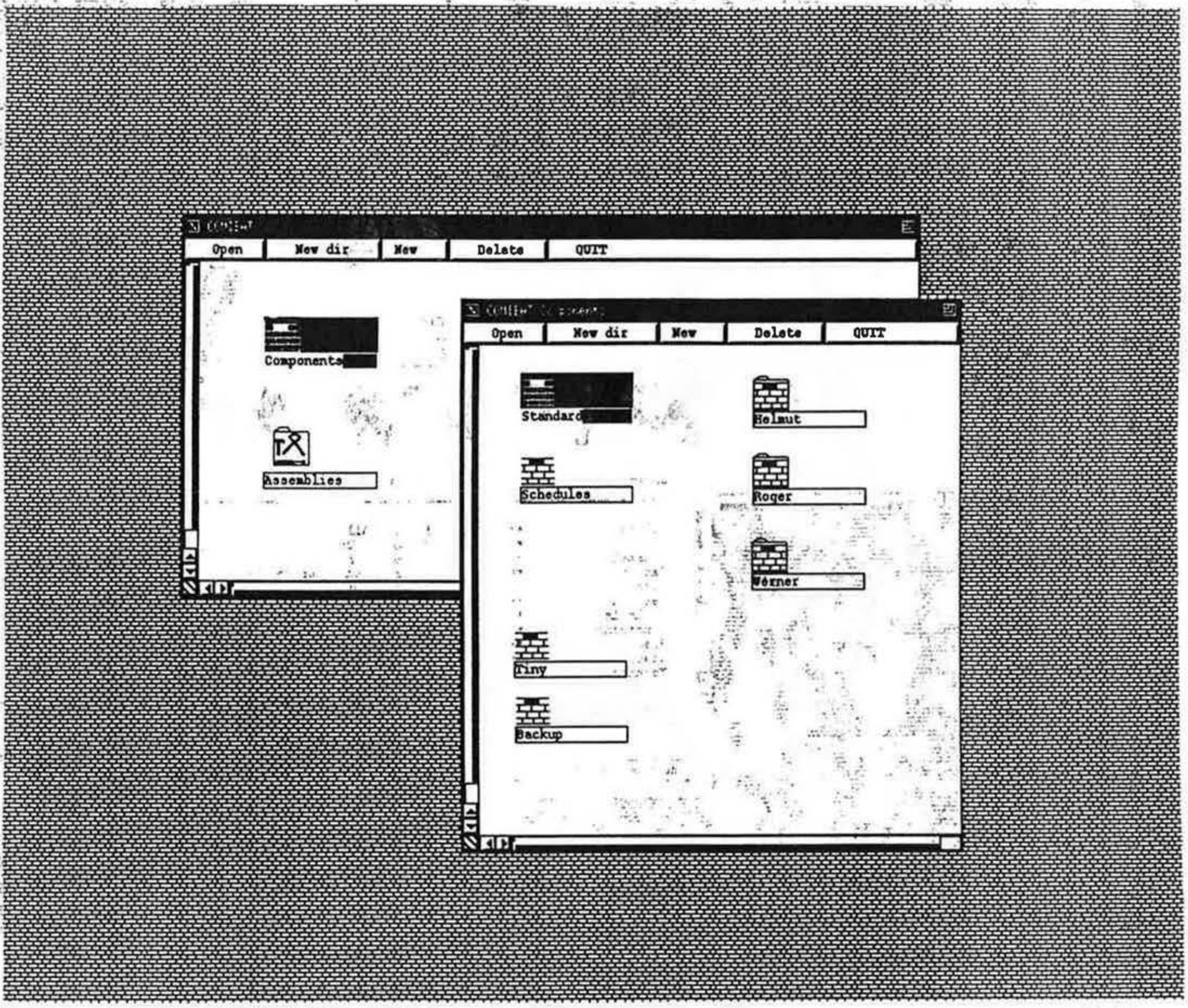


Fig.4: COMIBât directories are 'typed': in this example a possible subdirectory of the directory 'Components' is shown; it contains files (on the right hand side) which may only consist of libraries of components (not, for example, assemblies) and nested subdirectories. We see three subdirectories here: 'Helmut', 'Roger' and 'Werner'.

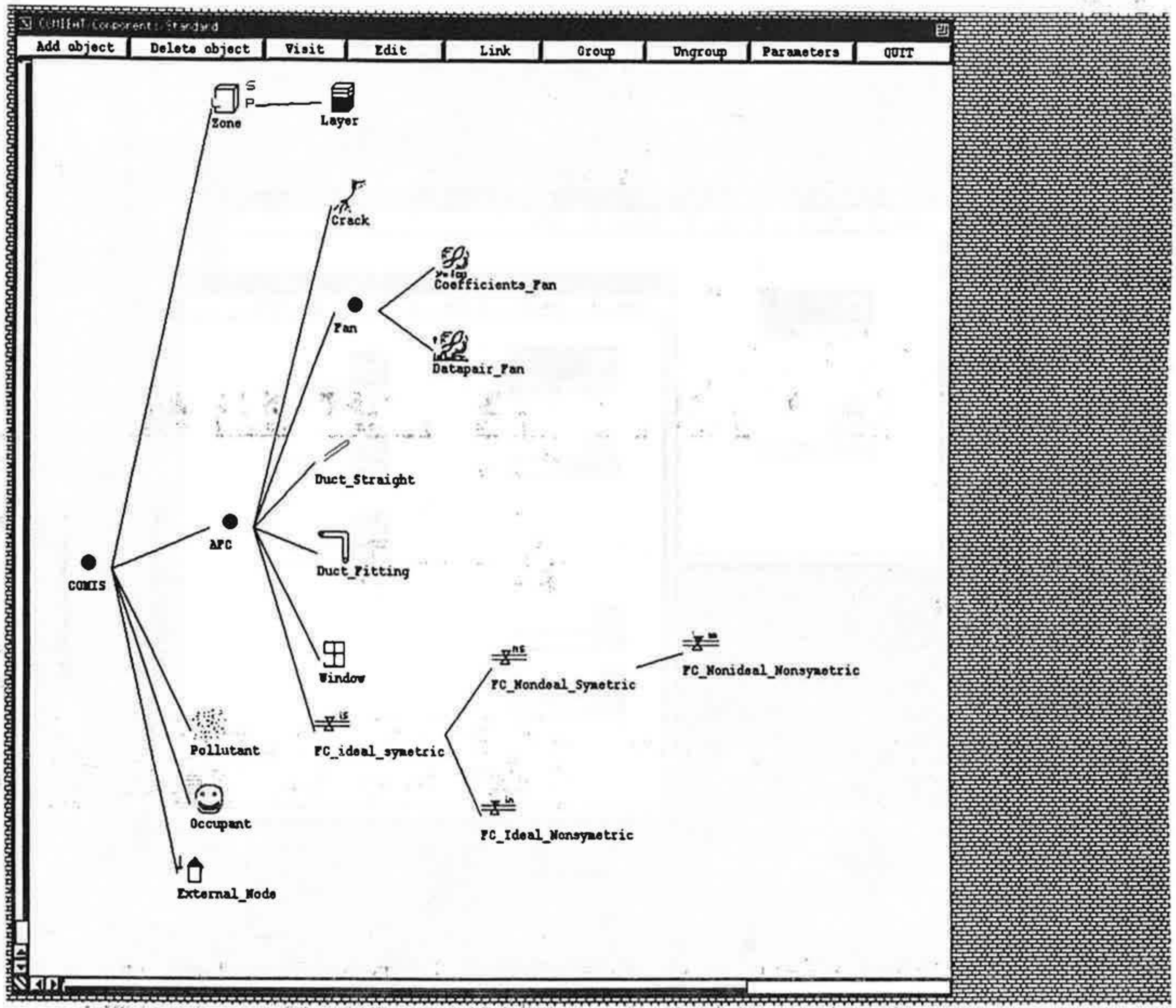


Fig.5: The standard library of COMIBât contains all the objects described in the COMIS User Guide. Their parameters have been set to the default values contained in the COMIS User Guide. These objects can be used to create assemblies, which describe a given building -see Fig.6-.

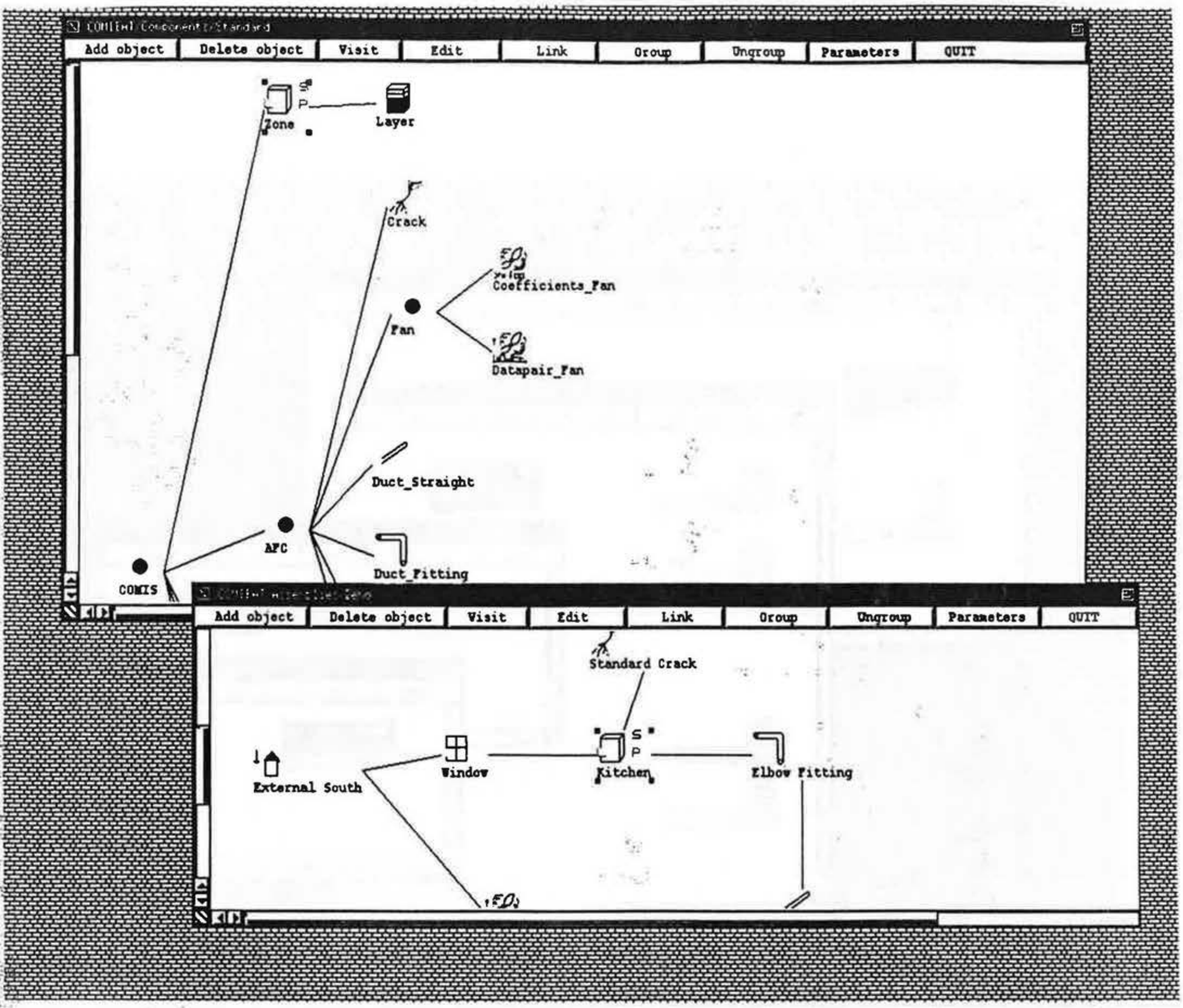


Fig.6: On this screen, a library window (the standard library, on top of the screen) and an assembly window (on the bottom of the screen) have been opened at the same time. This allows the user to transfer objects from the library window to the assembly window. Next the logical links between the objects are defined in the assembly window.

Here we see an example where the Object Oriented approach could be useful for the common user too; in the library of components, each node is a class which has its specific characteristics; these characteristics are automatically inherited by any component attached to the class; then, if the user creates a new object, he can store it in a library, under a certain class, and the new created object will inherit the properties of its mother class (of course, the developer of a new object has to store it cleverly).

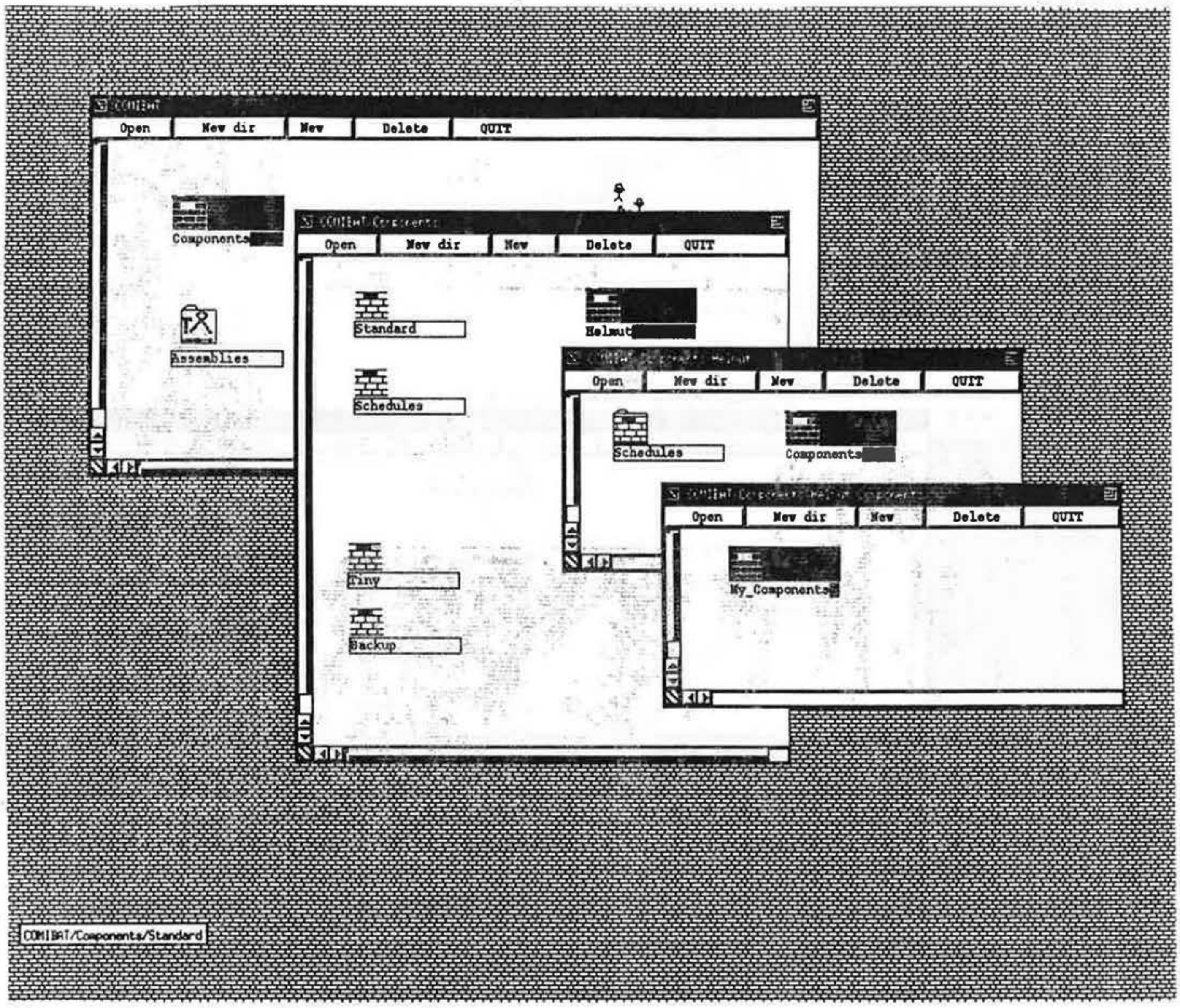


Fig.7: A user can create his own hierarchies of subdirectories, containing library or assembly files. In this example, the user 'Helmut' decided to create a subdirectory in the 'Components' directory, which he called 'Helmut'. There he created two more subdirectories, to store 'Schedules' and 'Components'. In his newly created 'Components' directory he keeps his own, personalized version of the standard library, and calls it 'My_Components'.

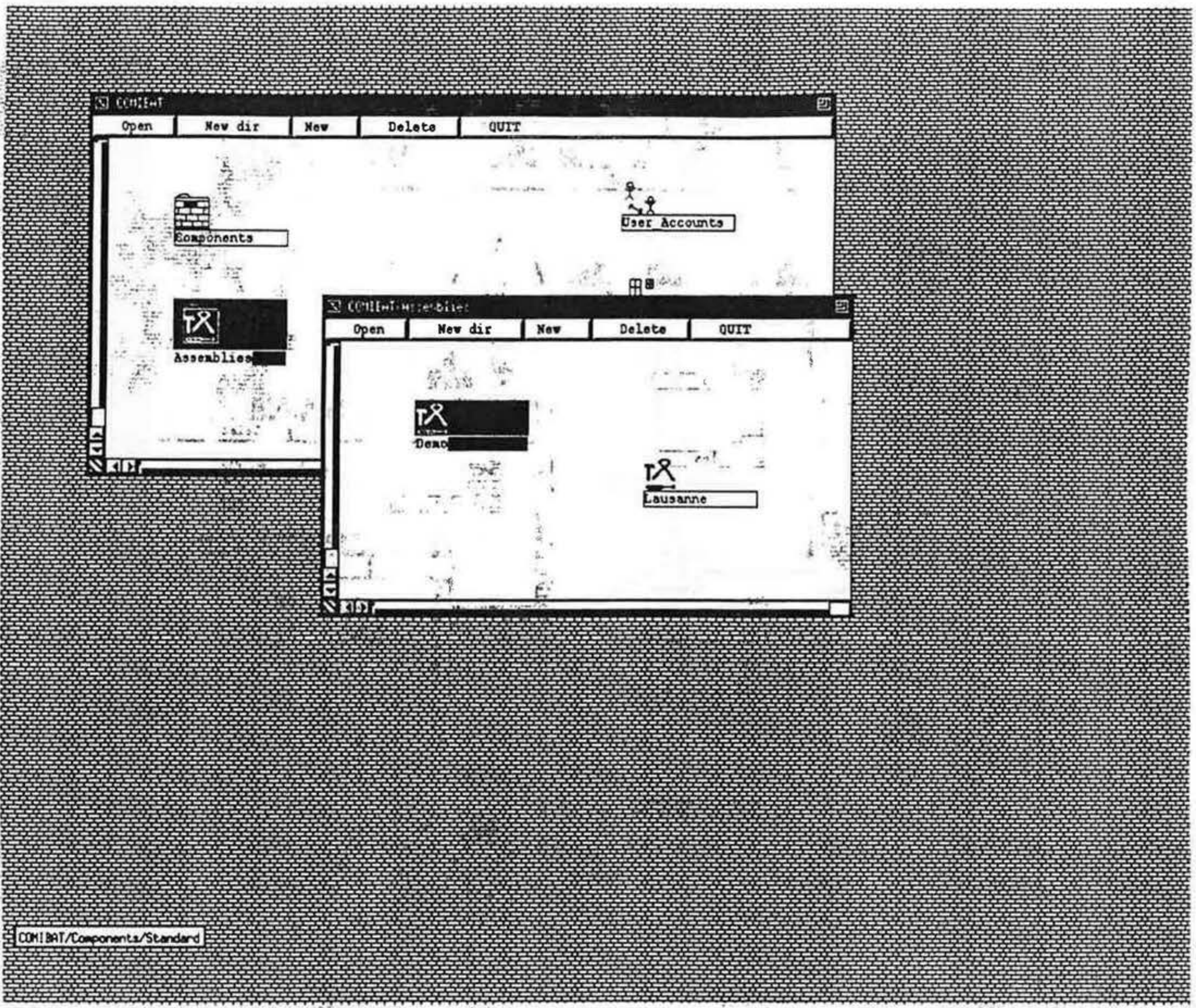


Fig.8: The 'Assemblies' subdirectory could also contain subdirectories (of assemblies). In this example, it is not further divided: there are only two assembly files, 'Demo' and 'Lausanne'. At the left bottom of the screen the iconified standard library is visible. It can later be used to complete one of the assemblies.

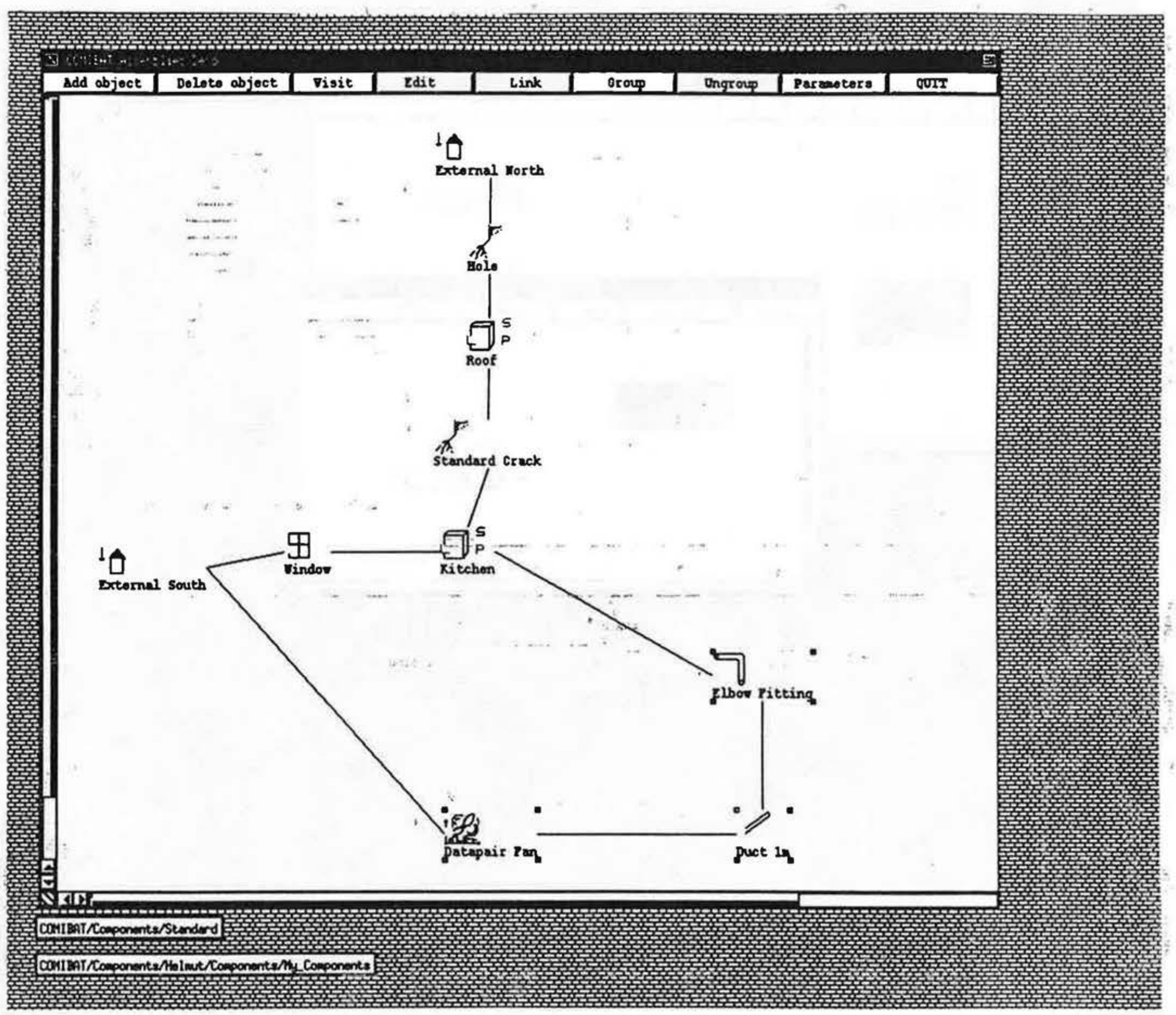


Fig.9: This screen shows the contents of the assembly file 'Demo'; it is an example of a simple building, consisting of two zones. The zone 'kitchen' has a solid 'S' attached to it, which means that one or several schedules have already been defined for this zone. For the other zone ('Roof') no schedules have been defined yet. Three objects are selected in this assembly.

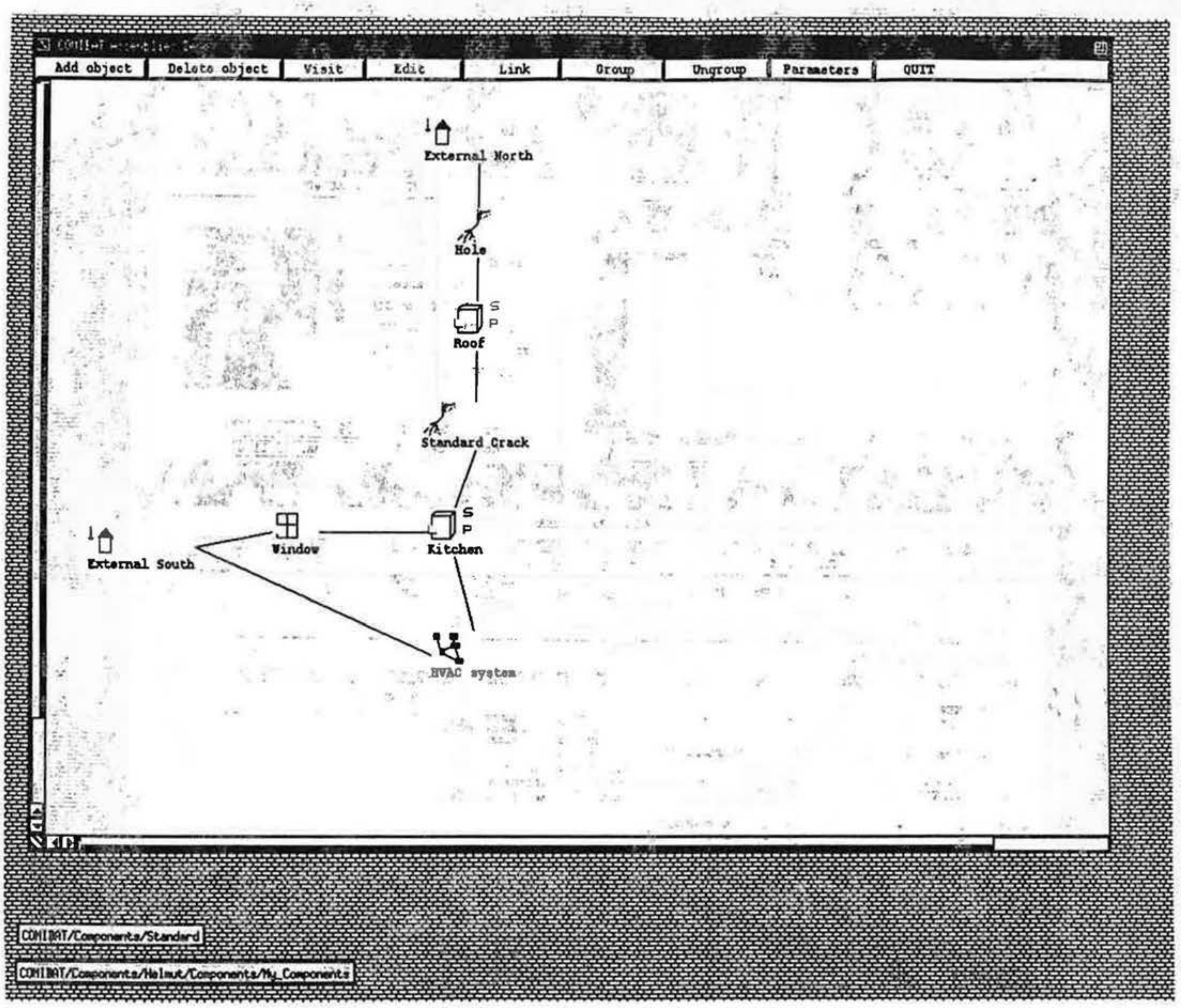
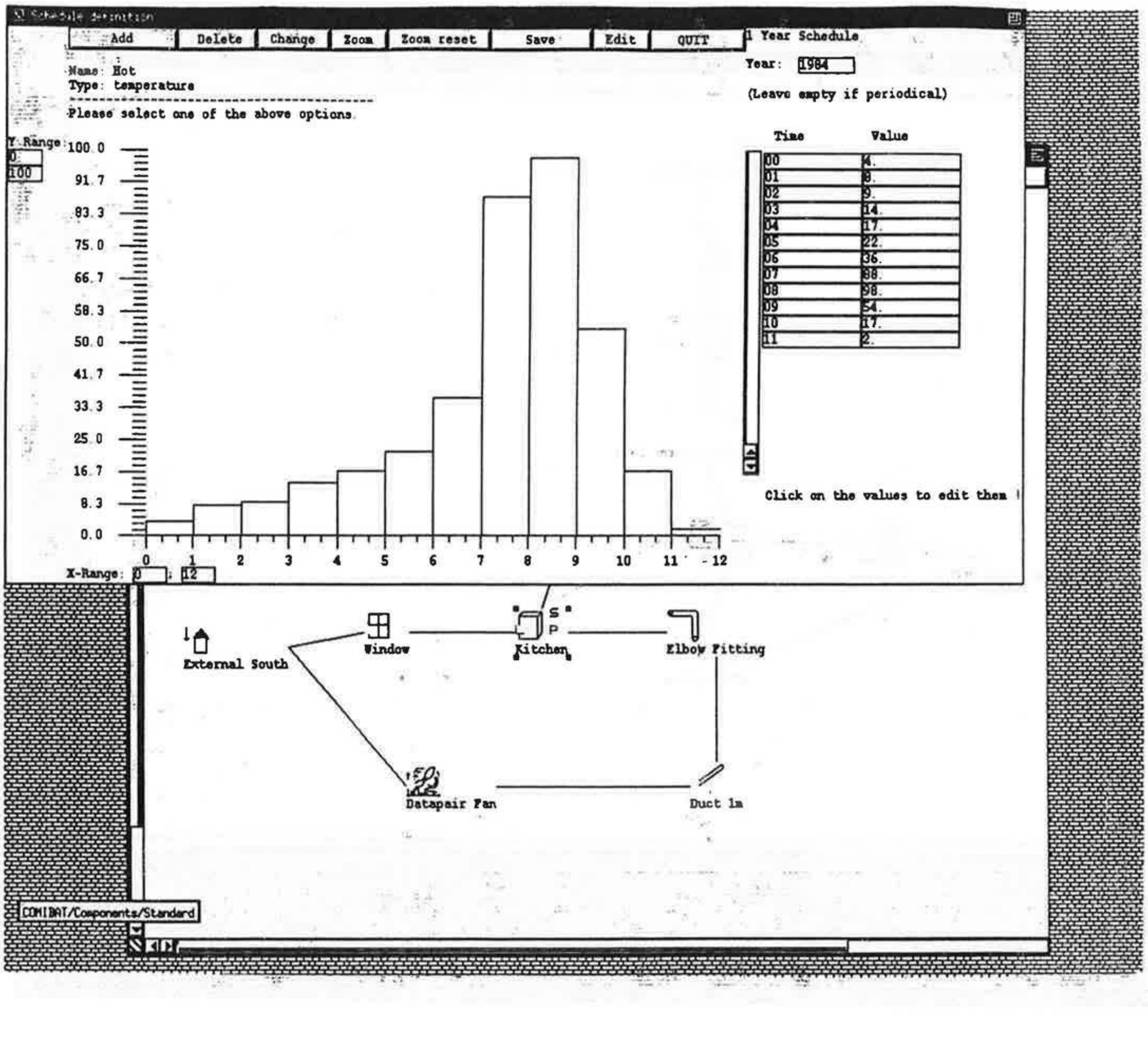


Fig.10: This screen illustrates the concept of macro-models; the three objects that have been selected earlier (cf. Fig.9) are grouped together into one object. This procedure not only simplifies a given assembly, but also allows to transfer entire structures between windows, together with all the information attached to them. One can, for example, transfer macros to a user defined library of standard macros and reuse them as needed.



Name: Hot
Type: temperature

Please select one of the above options.

1 Year Schedule

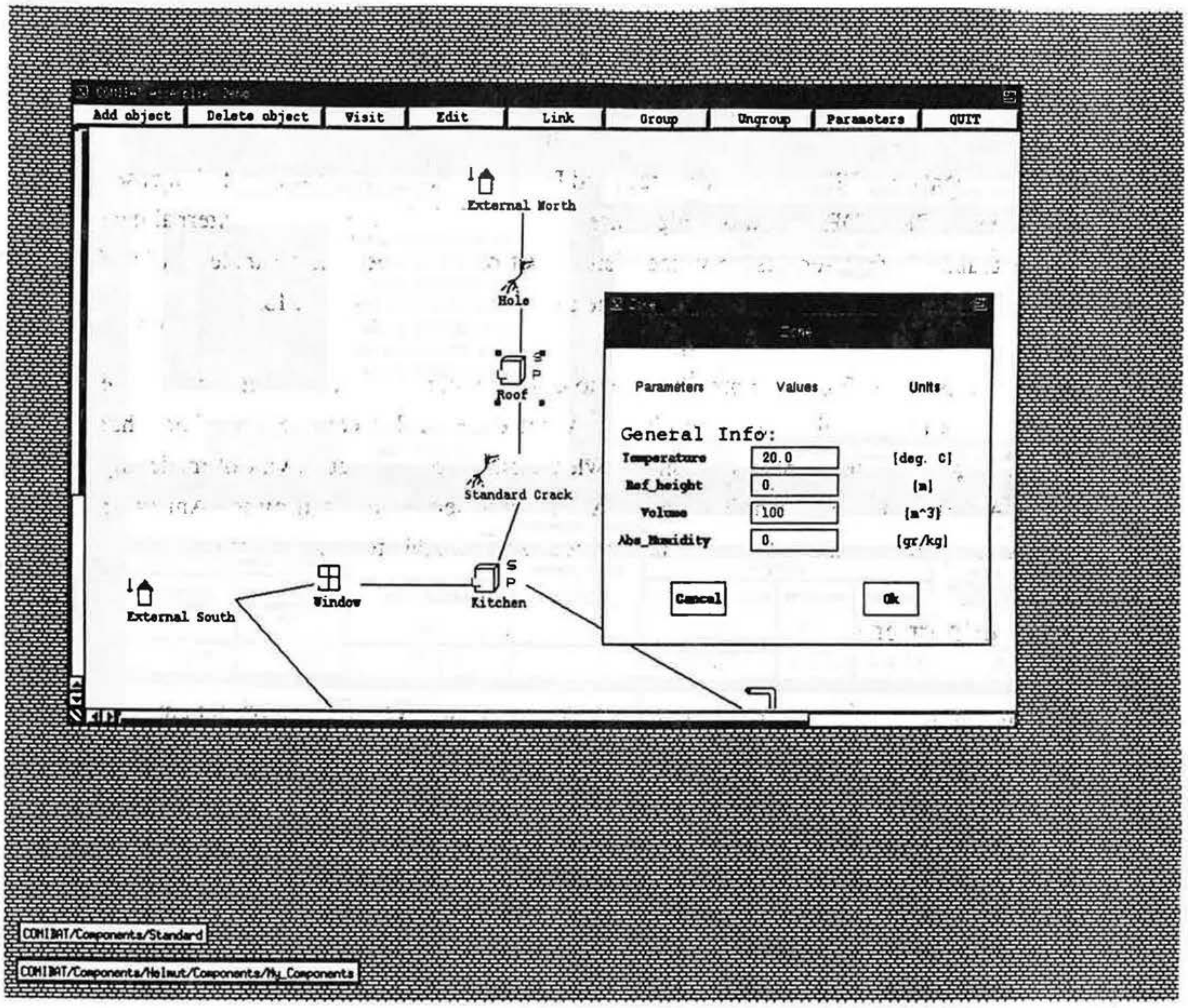
Year: 1984

(Leave empty if periodical)

Time	Value
00	4.
01	8.
02	9.
03	14.
04	17.
05	22.
06	36.
07	88.
08	98.
09	54.
10	17.
11	2.

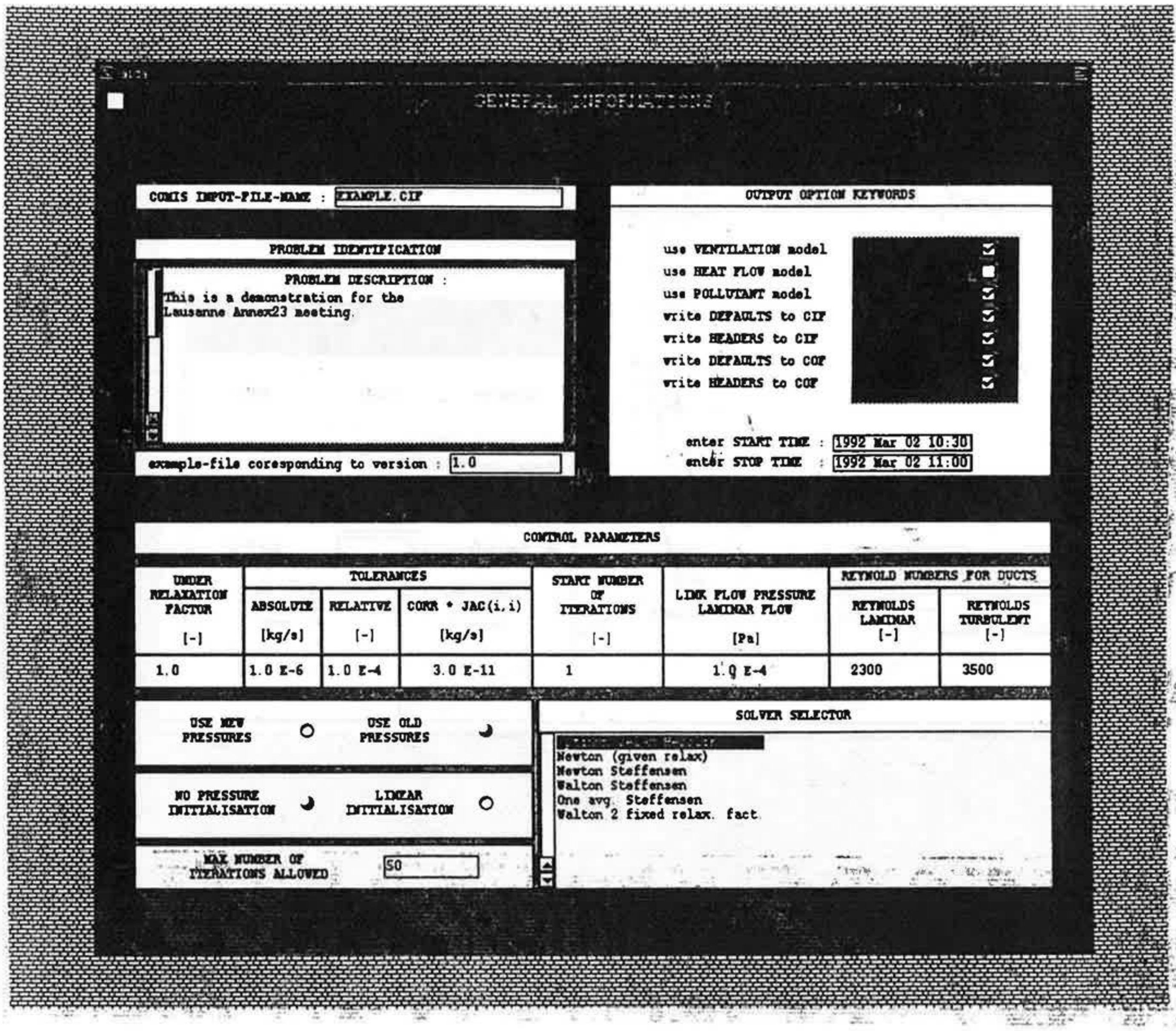
Click on the values to edit them !

Fig.11: To attach a schedule to an object, one can either pick one from the libraries or use an interactive application to define a new schedule. The second option has been chosen in this example: the user clicked on the S-tag of the 'kitchen' zone and specified that he wanted to create a new schedule. Now he can enter and edit values either with the mouse or with the keyboard (in the table on the right hand side of the schedule window).



Parameters	Values	Units
General Info:		
Temperature	20.0	[deg. C]
Ref_height	0.	[m]
Volume	100	[m^3]
Abs_Humidity	0.	[gr/kg]

Fig.12: At any time, the user can edit the parameters of an object; the dialogue window already contains the default values (e.g., the standard values from the COMIS User Guide if the object was copied from the standard library).



GENERAL INFORMATIONS

COMIS INPUT-FILE-NAME : EXAMPLE.CIF

PROBLEM IDENTIFICATION

PROBLEM DESCRIPTION :
This is a demonstration for the
Lausanne Annex23 meeting.

example-file coresponding to version : 1.0

OUTPUT OPTION KEYWORDS

use VENTILATION model	✓
use HEAT FLOW model	☐
use POLLUTANT model	✓
write DEFAULTS to CIF	✓
write HEADERS to CIF	✓
write DEFAULTS to COF	✓
write HEADERS to COF	✓

enter START TIME : 1992 Mar 02 10:30
enter STOP TIME : 1992 Mar 02 11:00

CONTROL PARAMETERS

UNDER RELAXATION FACTOR	TOLERANCES			START NUMBER OF ITERATIONS	LINK FLOW PRESSURE LAMINAR FLOW	REYNOLD NUMBERS FOR DUCTS	
	ABSOLUTE	RELATIVE	CORR * JAC(i,i)			REYNOLDS LAMINAR	REYNOLDS TURBULENT
[-]	[kg/s]	[-]	[kg/s]	[-]	[Pa]	[-]	[-]
1.0	1.0 E-6	1.0 E-4	3.0 E-11	1	1.0 E-4	2300	3500

USE NEW PRESSURES ○ [✓ USE OLD PRESSURES ●]

[✓ NO PRESSURE INITIALISATION ●] LINEAR INITIALISATION ○

MAX NUMBER OF ITERATIONS ALLOWED 50

SOLVER SELECTOR

[✓ [illegible]]
Newton (given relax)
Newton Steffensen
Walton Steffensen
One avg. Steffensen
Walton 2 fixed relax. fact.

Fig.13: Once an assembly is finished, the user can customize the 'global parameters' attached to the assembly. From this point, the system can directly create a COMIS Input File and start a simulation run.

4. Conclusion

The CSTB's contribution to Annex 23 consists not only in coordinating the subtask 1 but also to incorporate new modules in COMIS (Large vertical and horizonatal openings [Pelletret 91.09]) and, mainly, to incorporate COMIS into an Intelligent Simulation Environment. The goal of the ISE is to ease the access to COMIS, to ease its use, to provide helps to the user, to allow the sharing of data between various computation codes, to enable the coupling of COMIS with Building Energy Performance Simulation tools.

To reach these goals, the ISE involves an Integrated Data Model, a model documentation based on the concept of PROFORMA, knowledge bases and means of connections to external data bases. The graphical front-end is only 'the visible part of the iceberg' nevertheless it is of primary importance because it is through it that the user will dialogue with the ISE.

A difficult task of the ISE development will be to design expert systems in order to help the user in his tasks of modelling and simulating. What expert rules can we use? For what purposes? The goal will be to create relevant knowledge bases about the good way of modelling 'Multizone Air Flow and Pollutant Transport'. All the knowledge provided by the Annex 23 participants in the framework of subtask 3 is expected to be valuable for this purpose.

Acknowledgements.

The participation of CSTB to Annex 23 is supported by Ademe (Agence de l'Environnement et de la Maîtrise de l'Energie -Environment and Energy Management Agency-).

References.

[Pelletret 90.06] The IISIBât project : Intelligent Front-End Shell for Building Simulation Codes.
R. Pelletret & S. Soubra
2nd French-Finnish Seminar. Espoo. Finland. June 1990.

[Pelletret 91.09] Modelling of large openings.
R. Pelletret, F. Allard, F. Haghighat, G. Liebecq, J. Van der Maas
12th AIVC conference. Proceedings p.99-109.
Ottawa. Canada. September 1991.

[Soubra 91.11] Intelligent Simulation Environments for Building Modelling and Simulation.
S. Soubra & R. Pelletret
EuropIA'91. Athens. Greece. November 1991.

[Roulet 91.11] Applications of Multizone Air Flow Simulation Models.
Claude-Alain Roulet, November 1991.
Annex 23 Working document

[Keilholz 91.12] Modelling of the COMIS Data Structure.
Werner Keilholz, December 1991
Annex 23 Working document (CSTB's reference: DPE/91-825)

[Dubois 92.01] COMBINE IDM / V3.3
Anne-Marie Dubois & al., January 1992
CSTB's report: MGL/92-1264

[Keilholz 92.03] Prototype Development for a Graphical Interface for the COMIS Multizone Infiltration Simulation Code.
Werner Keilholz, March 1992
CSTB's internal report: TTA/DPE/92-1218

[Pelletret 92.04] Model Documentation. The Annex 23 PROFORMA.
Roger Pelletret, April 1992
Annex 23 Working document.