case/4/0

The Introductory Version

microTOOL

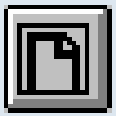
Copyrights

Reading the Documentation

This part of the **case/4/0** description should be read first to ensure a good start with the Introductory Version of **case/4/0**. Here, we'd like to illustrate how you use this online document. This document was created with Adobe Acrobat 2.0 and is contained in a PDF file (Portable Document Format). You will need the Acrobat Reader to be able to read the PDF file. The Reader is supplied together with Introductory Version of **case/4/0**. We will explain here the essential Reader functions. Detailed instructions can be obtained with the F1 function key or over the ***Help*** menu. The corresponding online handbook (help_r.pdf) is opened in either case.

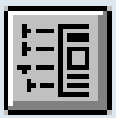
Default Settings That Make Reading Easier:

The increased use of 256 color displays induced us to setup our documentation in accordance with to this standard. Of course our documentation can still be read well with a 16 color display. If it is possible, however, we recommend a configuration with 256 colors.
The following settings should be made after opening a document to achieve optimal reading results:
The magnification level of the document should be adapted to the size of your screen: Click this button to zoom the document to fill the width of the screen.
With the ***Zoom*** function you can change the magnification level of the document's screen representation. Comfortable reading results can be achieved with the ***Fit Visible*** command.
Have an outline of the document displayed; click this button in the Acrobat toolbar. The display window will be divided vertically, and the bookmarks defined for the document are displayed with a page icon in an overview area on the left. To move to one of the displayed topics, just click it or the corresponding page icon.



Fill the width of the screen

Zoom function



Show bookmarks

This icon to the left of a bookmark means that subordinate bookmarks have be defined. Click the icon to show the subordinate bookmarks.
Click this icon to hide the subordinate bookmarks again.

Show subordinate bookmarks

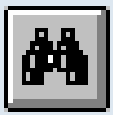
Hide subordinate bookmarks

Hypertext Links

You can jump to any topic of interest from any item in the table of contents with a click of the mouse.
We have also defined hypertext links for individual terms, which we set off in color and italic style. Just move the mouse pointer to such a term; the mouse pointer becomes a pointing hand. With a mouse click you "open" that page in the document linked to the colored term, where the term could appear again – possibly in another context – or where it is illustrated in detail.

Text Search

If you are looking for a specific term, you can look it up in the alphabetically ordered index of the corresponding document. You can move quickly to the given page by moving the scroll bar to the indicated page with the mouse: The beginning of the document is at the top and the end is at the bottom. Stop when the number of the desired page is displayed.
Or select the "Find" function from the Acrobat Reader toolbar. That is another way to find any term quickly. To find the next instance of the term, press the F3 function key.



Search

Printing Documents

To print a document, select first the ***Print Setup*** command from the ***File*** menu, and specify the appropriate settings in the dialog box that appears. Afterwards, select the ***Print*** command in the ***File*** menu.

case/4/0

The Introductory Version

Welcome to the world of **case/4/0**. Is professional software development your daily business? Then take a look at our ICASE-Tool **case/4/0**. We have developed it for you.

Discover case/4/0 ...

... and convince yourself just how simple and effective this tool is. A **case/4/0** demonstration version is available to you on the enclosed CD-ROM. Take a good look at every aspect. Try structuring, animating, printing and generating – from Analysis through to Source-Code.

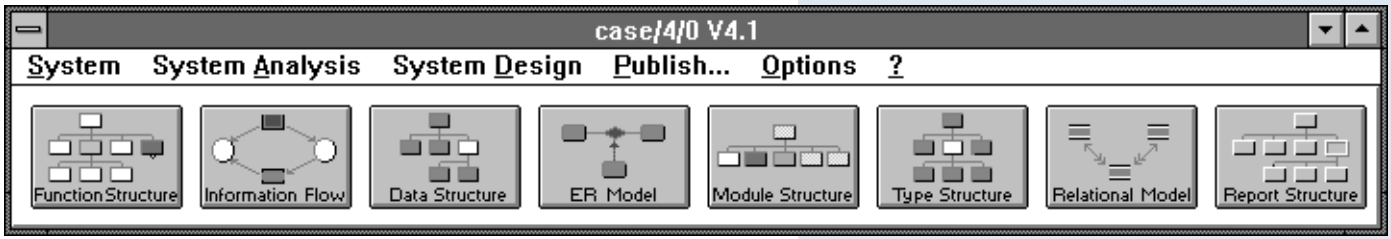
If you want to know more about case/4/0 ...

... then you will find an overview of the product and comprehensive information contained in the following pages, with which we hope to stimulate your journey through **case/4/0**.

case/4/0 – Analysis, Design and Programming Tools for LAN Operation

What is **case/4/0**? What can it do for you? A look at the menu provides an answer to this question:
case/4/0 integrates

- Tools for ***System Analysis***
- Tools for ***System Design*** and
- A Publisher for the organization of target group documentation



into a complete software development environment. Whether you are developing commercial or technical applications, whether you are specifically concerned with dialog processing, data base communication or real-time applications, **case/4/0** offers you comprehensive tool support.
From the toolbar you are able to see that **case/4/0** offers a number of graphical plans for System Analysis and System Design. This is not without good reason: **case/4/0** is conceived for task-share projects in which complexity should be reduced, where progress is made according to plan and architectural decisions are made consciously. It is with this sort of project that the model driven approach shows its strength. The concentrated and graphically structured knowledge in the models is actively employed by **case/4/0** for quality security and the generation of further used results.

Some of the **case/4/0** results – highlights from System Analysis and System Design – are comprehensively shown in the following pages.

Central Repository for Multi-User-Operation

As soon as you begin constructing a model, **case/4/0** automatically makes sure that your results fit in with those from all the other project members without inconsistencies. This is possible because **case/4/0** was conceived for multi-user-operations. It possesses an integrated LAN-Repository which actively secures the consistency of all the results. The **case/4/0** LAN-Repository has a relational meta-model. The complete description of the meta-model belongs to the **case/4/0** product package, as well as a simple language (we call it the *Generation Language*) with which you can gain direct access to the repository – to create evaluations of the development results, for example (***Evaluate...***). **case/4/0** aids the division of a system into sub-systems and the logical, consistent integration of distributed results with its ***Import*** and ***Export*** functions. **case/4/0** is flexible enough to fit into the most different forms of project organization. Apart from consistency **case/4/0** also guarantees the formal correctness of your results because it possesses...

... a safe foundation: The Structured Methods of case/4/0

case/4/0 offers two mutually co-ordinated methods: **System Analysis** and **System Design**. These combine the well-tried concepts of

- *Structured Analysis* from DeMarco with Real-Time Extensions from Ward and Mellor,
- *Entity Analysis* from Chen,
- *Structured Systems Design* based on Page-Jones and Yourdon,
- *Relational Data Modelling* from Codd

Two sample systems are contained in the demonstration version's repository, which you may work with as you please. Click on ***System/Select...*** *and select* ***EASY_C*** *if you are interested in designing with Microsoft C/C++, Microsoft Foundation Class Library (MFC) and ODBC-Interface. If you are interested in the design of a Micro Focus COBOL-Application with Micro Focus Dialog System, then select* ***EASY_COB****.*

By the way, ***case/4/0*** *also generates Microsoft Visual Basic Applications and with the aid of its generation language, it is compatible with many other target systems.*

into a universal process. Strong practical impulses have lead to continuous development updates of the structured concept. Therefore, for application development in C or COBOL on relational data bases they are, now more than ever, the ideal structuring aid. They are perfect for planning large pro-jects – without strategic risk.

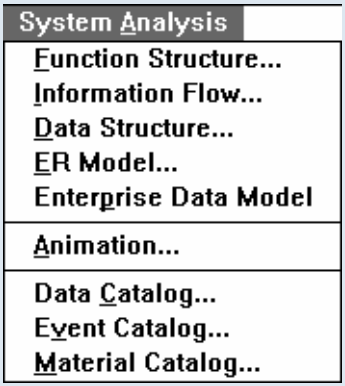
System Analysis Highlights

What does System Analysis in **case/4/0** actually mean? Let's take a look at the menu element of the same name. Functions, data and behaviour of an application system are modelled from a problem domain point of view by the graphical System Analysis tools. To this extent, **case/4/0** offers five diagram types:

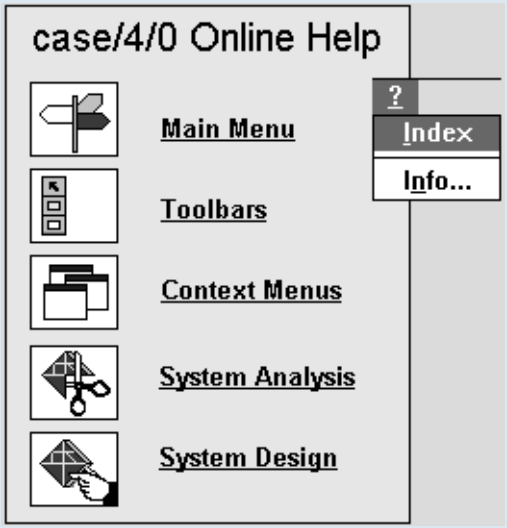
- The ***Function Structure***: a simple tree diagram for the functional decomposition of a system,
- The ***Information Flow***: a data flow diagram extended by the presentation of control and material flows,
- The ***Data Structure***: for structuring data hierarchically and describing views on entity types,
- The ***Entity Relationship Model (ER Model)***: illustrates data objects and their relationships,
- The ***Enterprise Data Model***: which offers graphical editing functions, navigation and zoom aids in order to integrate entity relationship models redundancy free into a complete model.

Data, ***event*** and ***material catalogs*** extend the graphical models by means of formal and textual descriptions.
Some things that the menu does not display: **case/4/0** provides three further tool components when developing information flows:

- The ***State Transition Diagram***: for system control specification,
- A ***Dialog Designer***: for the graphical design of the user interfaces,
- A syntax and context sensitive ***Editor***: for the description of the so-called *transformation behaviour*, i.e. the principle working process of the processing functions.



*Selecting either **EASY_C** or **EASY_COB** transposes you directly into the work which is being carried out by a project team. The task is to improve the functional organization of the fictive EASY Furniture Company. Some results are already present, so that you will find at least one example for each result type in System Analysis.*



*Questions concerning operation are answered in the **case/4/0** Online-Help. You can find out more about this by clicking on the question mark in the main menu.*

If the functionality, control and user interface of a system have been structured from a problem domain point of view then the organization of the specified enterprise processes can be dynamically simulated and verified. **case/4/0** supplies you with a true to life impression of the new system under the menu point ***Animation...***.
With the aid of System Analysis **case/4/0** offers you a system specification, which

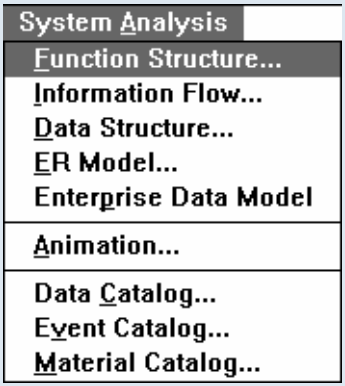
- consists, in the sense of modularization, of components with high functional binding and low data coupling – a necessary requirement for easy maintenance of a system,
- already works, that is to say it describes the organizational control of the functions found from a problem domain point of view,
- establishes the redundancy free, logical order of data and therefore the strategic potential of the new system.

The Function Structure

A function structure provides a structured overview of all of the problem domain functions. With **case/4/0** you are able to distinguish between several function types in a function structure due to...

...control, presentation, processing – Function is not equal to function

case/4/0 knows:
Processing functions (light yellow): these change the contents or structure of data or material.
Dialog functions (green): these give the future users of the system the chance to intervene in the system control by means of menus and dialog boxes.
Control functions (yellow): these co-ordinate the activities of the processing and dialog functions and react to events in the system.



*On selecting **Function Structure...** you are presented with a list from which a diagram of this type may be selected for editing purposes, or from which a new diagram may be created. In the sample systems you will find the functions structures from the EASY Furniture Company.*

And this is the toolbar which is used for editing function structures:

Function

Recurrent Structure

Part: Separate sub-tree

Block: Structuring aid

Condition

Action: Link elements

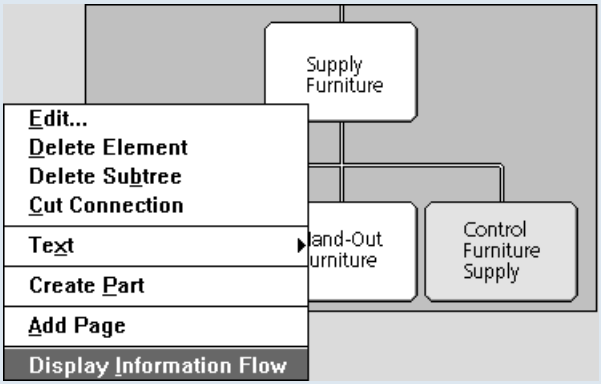
The reuse of system components can be planned and modelled with the help of *Recurrent Structures* (white).
For the analyst the function structure represents the thread through the system specification, for the project manager it is the basis of the task division, project planning and control.

The Information Flow

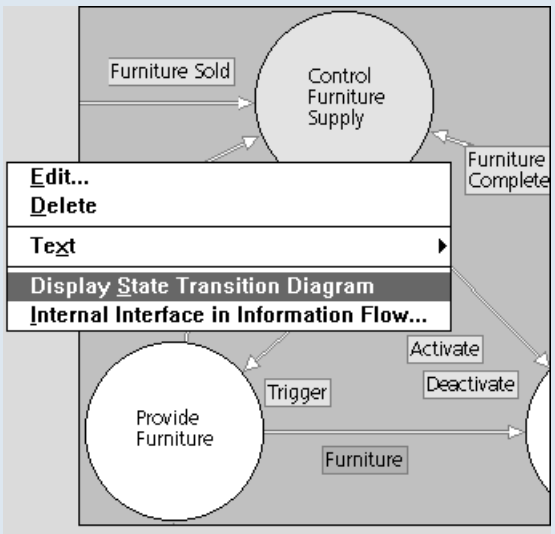
Information flows (Data flow charts) are the most flexible and well liked elements from Structured Analysis. This is not without good reason: their notation is very simple and they are extremely suited for communications – even for "method laymen". Therefore it is no great surprise that event oriented extensions to the information flows were suggested by various authors at the end of the eighties, so that they could be implemented even further – for example for the specification of real-time applications.

Structuring a Transparent Business Process

We at microTOOL have gone one step further and have integrated the development of dialog applications withgraphical user interfaces in the structured method world. As well as the exchange of data, material and events between functions, interfaces and stores, **case/4/0** also specifies the "internal life" of functions.
Processing functions can be divided into sub-functions whose interactions are illustrated in an information flow. The consistency of the resulting information flow hierarchy is taken care of by **case/4/0** top down.
Dialog functions are specified by the graphical design of the corresponding dialog boxes and menus. Furthermore the definition of Callbacks establishes how dialog functions should react to user actions. In the first instance this happens independently of the target system with the **case/4/0** *Dialog Designer*.
Control functions are specified in *state transition diagrams*. They illustrate how control functions react to events dependent on the actual state of the system.



If you click on an element with the right hand mouse key then it will display exactly what you can do with it. Try it for your-self! Click on Supply Furniture *and select the command* ***Display Information Flow****, in order to see the corresponding information flow. A double click on* Supply Furniture *also has the same effect.*



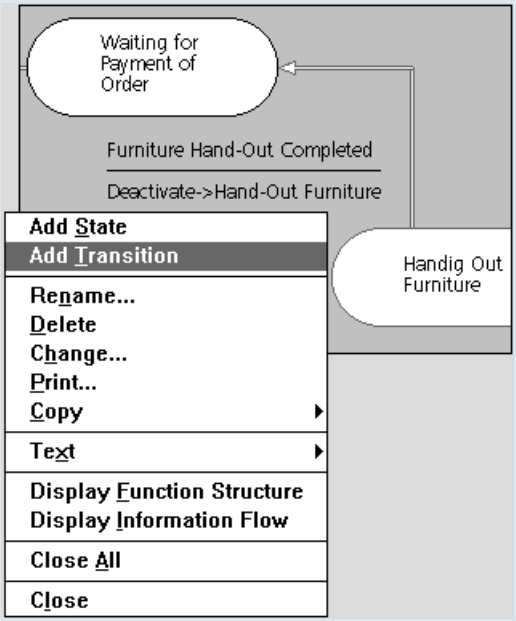
A state transition diagram exists for the Control Furniture Supply *function. Take a good look a it!*

The notation of the state transition diagram is quite simple: oval elements represent the system states and arrows represent the transitions. A transition is labelled with the condition which must be fulfilled in order for the transition to take place. Under this – and separated by a line – the action to be carried out by the system in order to reach the new state is registered.

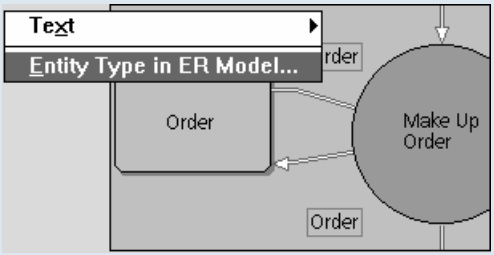
The Entity Relationship Model

Information flows show which data is exchanged, processed or created by the functions of a system. Data from the problem domain must first be structured before detailed information flows can be created. **case/4/0** supplies you with the entity relationship model (ER Model for short) to take over the task of semantic data modelling. The name "entity relationship model" expresses exactly the elements which are contained in a diagram of this type, i.e.

- *Entity types*, symbolized by rectangles. These represent collections of similar problem domain objects and consist of attributes which refer to uniquely defined data elements in the **case/4/0** data catalog,
- *Relationships*, which are symbolized by diamonds and connected to entity types, represent the problem domain relationships of the entity types. They are labelled with their cardinality and semantic,
- *Associative entity types*, i.e. elements, which can be interpreted as relationships on the one hand, but also as entity types on the other because they possess self describing characteristics, i.e. attributes. They are represented by rectangles which are connected to diamonds by arrows,
- *Sub and super relationships* which illustrate specialization or generalization of entity types and which can be recognized by a blue triangle.



If you wish to add a new transition, then the name of the condition can only be the name of an event which is registered to the corresponding control function in the information flow by an ingoing arrow. Such dependencies between diagrams are automatically taken care of by **case/4/0**.

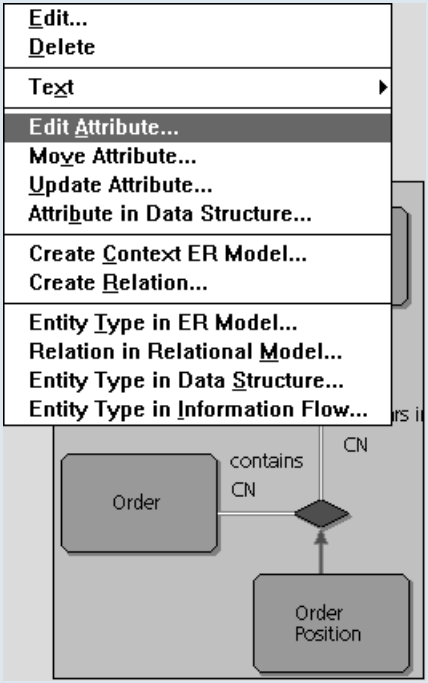


In the Sell Furniture *information flow you will find an inconspicuous, grey box – the* Order *entity type. If you click on this with the right hand mouse key then the path to the ER Model, from which this entity type originates will be displayed. Follow it!*

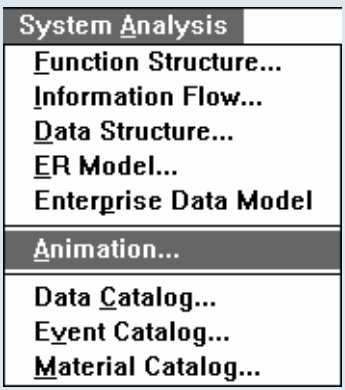
Entity types from the ER Model can be inserted as data stores in information flows. Later updates to an entity type are consistently passed on by **case/4/0** to all information flows affected.
How do you structure ER Models? We suggest a subject oriented development of your ER Models. Redundancies between the ER Models are valid and are automatically taken care of by **case/4/0**.
If you want to find out, away from all ER Models, which relationships an entity type is bound in, then you can generate a *context ER Model* which especially displays this aspect.
All entity types at once? No problem for **case/4/0**. ER Models can be integrated, redundancy free, into an *Enterprise Data Model (EDM)*.
The construction of ER Models by **case/4/0** comprises not only conceptual clarity, but also optimal use for software design and implementation: From an ER Model you can derive a relational data base design in the form of a relational model by a click of the mouse. (We will deal with relational models comprehensively later). And again – on the push of a button – you can generate a SQL-Data Base description.

Your Specification Learns to Operate

When system control, user interfaces and data storage are correctly specified within the context of an information flow as shown above, then it is just a small step towards ***Animation*** of the system behaviour. By this we mean the systematic run through of the business processes which are described by one or several information flows.
But there is still something left to do! First of all the following question must be answered: How do processing functions react principally to input data, events or incoming material? The answer to this is given by the definition of the *transformation behaviour* of the elementary processing functions in the information flow. In the form of very simplified rules and with active support from a context and syntax sensitive editor, the inputs, which a function creates outputs from, can be established. In the



This is where the Order *entity type comes from: the* Order Transaction *ER Model. With the* ***Edit Attribute...*** *command you can check what has been placed behind the box.*



You can also test out the Animation. You will find the corresponding menu point under ***System Analysis*** *in the main menu.*

animation, the transformation rules appear in place of the detailed algorithmic logic of the functions. If this requirement is met then **case/4/0** just needs information concerning which *scenario* you would like to dynamically examine in order to start. A scenario consists of one or more information flows which describe a business procedure and is extended by outline conditions. An outline condition is, for example, the number of external events which should happen in the process of the business procedure.

And this is what the animation looks like: For the purpose of demonstrating a business procedure the use of information paths and activation of functions in information flows is marked by moving highlights. Ever more so, if a dialog function is activated during the animation then the corresponding user interface appears on the screen.

Now you can intervene in the business process – just as the user would later – by clicking on a dialog element, for example. In the information flow you can then follow the moving highlights and see which functions are activated. If another dialog function is activated, then the corresponding dialog box will appear on the screen. The business process can therefore be abstractly followed by information flows in the form of moving highlights and concretely by dialog flow.

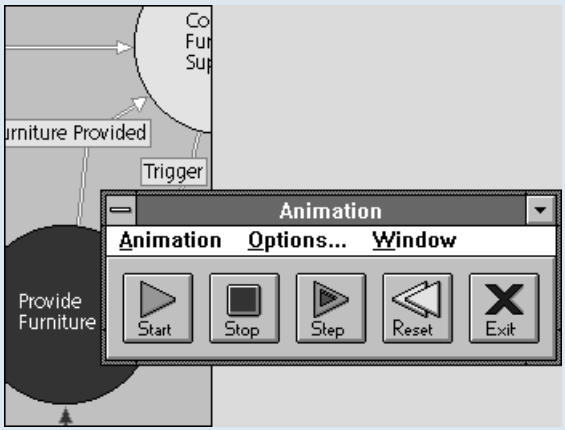
If the animation was successful then the result of the System Analysis with **case/4/0** is a complete and functional specification from the problem domain point of view. An optimal basis for software design is therefore created by **case/4/0**.

System Design Highlights

System Design with **case/4/0** means

- *Design in large*, i.e. data base design and development of software architecture based on the results of System Analysis,
- *Design in detail*, i.e. detailed specification of the algorithms of all complex functions as precondition for the implementation.

We suggest that you animate the Furniture Sales and Deliveries *scenarios. The project team from the EASY Furniture Company has already defined the outline conditions. With* ***Store Values...*** *100 pieces of furniture are defined as being available in* Furniture Warehouse. *Select* ***Start*** *in order to begin the animation.*



On the ***Start*** *command the animation begins running automatically. But don't forget to assume the role of the future user.* **case/4/0** *expects a reaction from you when a dialog box appears.*

A glance at the main menu point ***System Design*** reveals that **case/4/0** offers three diagram types:

- The ***Module Structure***: a tree diagram that, on the one hand, presents a graphical contents list for a module, and on the other acts as a sort of "container" for the Source Code. Who calls who – this aspect is also displayed by module structures,
- The ***Type Structure***: the technical opposite to the logical data structure and
- The ***Relational Model***: for the design of the relational data base.

A fourth diagram type is also accessible, not from the main menu, but by means of the module structure:

- The ***Implementation Tree***: illustrates the algorithmic logic of complex functions of the module structure in the form of graphical pseudo code.

The Windows version of **case/4/0** generates code from these graphical results for the following target technologies:

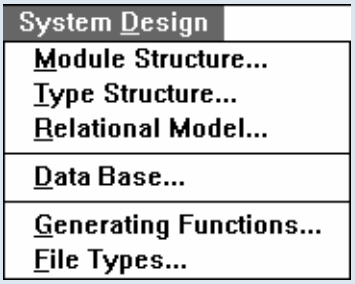
- **Micro Focus COBOL** with Micro Focus Dialog System and Embedded SQL for IBM Database 2,
- **Microsoft C/C++** with Microsoft Foundation Class Library and Microsoft ODBC-Interface,
- **Microsoft Visual Basic**.

This doesn't correspond to your package? With the aid of the built in ***Generating Functions*** – under the same name in the menu – you can extend the generation capabilities of **case/4/0** to incorporate other target systems.

Data Bases are generated on the basis of relational models.

System Design's primary objective is to transpose the analysis results into software design without loss of problem domain knowledge and architectural characteristics. To reach this objective, **case/4/0** offers three different procedure alternatives:

- *You develop modules explicitly*, i.e. you set up module structures and transfer the required functions and data, with the aid of **case/4/0**, from System Analysis into your module design.
- *You automate recurrent design steps.* If you realize that you often require modules of a similar structure, then you should design a generally valid set-up – we call this a *Module Template* –



Before you begin to explore in detail exactly what is hidden behind these menu points, let **case/4/0** *do the hard work for you once again by generating the System Design results from the System Analysis documents, because ...*

for this particular structure. Module templates are structure instructors which can reach up to the level of code. If required **case/4/0** creates concrete module structures from module templates. To do this you only have inform the tool as to which up to date results should be used in the module structure. Module templates are therefore a suitable instrument for illustrating and carrying out enterprise specific structure and programming standards. This is due to the fact that **case/4/0** independently takes care of their compliance.

- *You let* **case/4/0** *take care of the modularization work and go ...*

... from Analysis to Design by mouse click

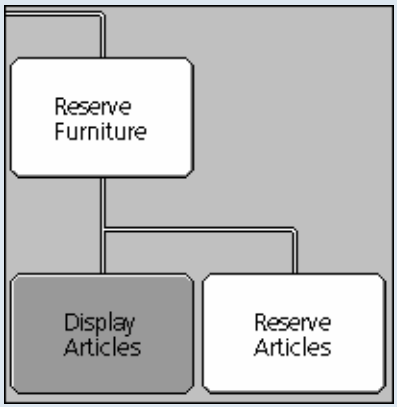
The **case/4/0** tool component, which relieves you of the design work, has been called ***Design Assistant***. The Design Assistant generates module structures from the information flows of System Analysis, relational models with attributes and primary & foreign keys from ER Models, and type structures from data structures. The Design Assistant requires two things for software design "at the push of a button":

- a fundamental architectural concept for structuring a software system,
- detailed construction plans for software structures, i.e. module templates.

The Design Assistant works on the basis of a distributable layer architecture, which also offers itself for the development of Client/Server applications. It is equipped with standard templates for user interface, processing and data accessing modules, which can be modified, extended or replaced specific to your application at any given time.

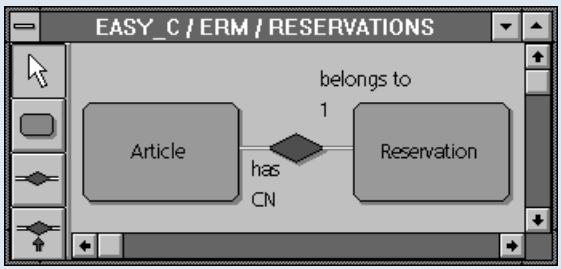
This is how the Design Assistant works: For each entity type in the information flow a check is carried out to see whether it has already been transferred into a relation, i.e. the design of a data base table. If this is not the case then the Design Assistant makes up for this design step and automatically creates a data base design from the ER Models in the form of relational models.

... software design at the click of a mouse key is something you absolutely have to try! Best of all on your own "piece of System Analysis". We suggest the following:



Step 1: Extend the EASY Furniture Company *function structure by setting up a third function –* Reserve Furniture *– next to* Sell Furniture *and* Supply Furniture*. Then divide this function into a dialog function –* Display Articles *– and a processing function –* Reserve Articles*. Link* Reserve Furniture *with* Self Pick-Up Furniture Sales*.*

Step 2: Set up a new ER Model by the name of Reservations*, which could look something like this:*



After this preparation, the Design Assistant is able to generate a module structure *for each* entity type for access to the corresponding relation. All data structures, which describe the input/output data of the information flow functions, are transferred into type structures, which are subsequently used by the Design Assistant to create data definitions in module structures. *All* processing functions, which appear in an information flow, are summarised in the module structure of a processing module. For each dialog function a module structure of an interface module is created. Finally the relationships used between interface, processing and access modules are derived from the information flow and illustrated in the module structures. Lets take a look at a few generated results.

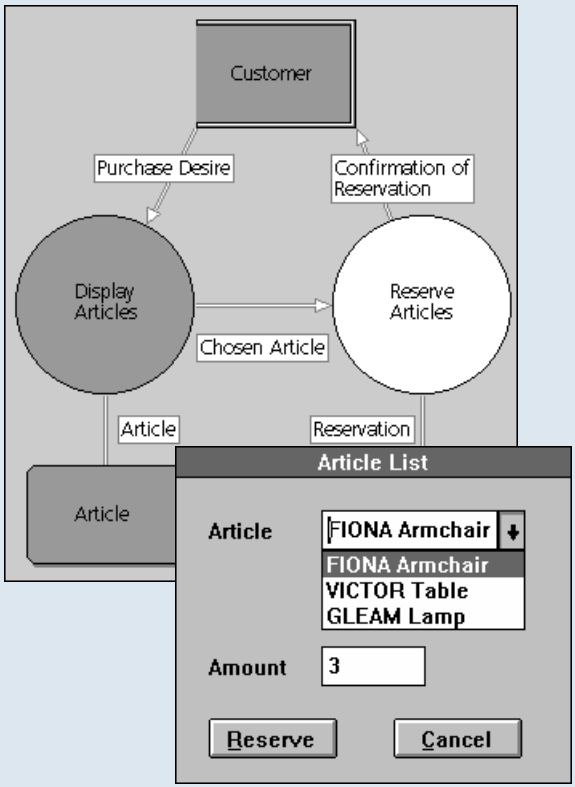
The Relational Model

The relational model serves the data base design. From its elements – the relations – **case/4/0** generates SQL table definitions. You remember the menu point ***Data Base...*** in the ***System Design*** menu? Relations can be described by their attributes, primary keys and indices.
Apart from the rectangular symbols for relations, the relational model also displays arrows. These represent the relationships between the relations which should be used as access paths. These are described by foreign keys.
You can also derive a rela-tional model from an ER Model directly – without the Design Assistant. And what if there are later corrections to be made to an ER Model? No problem! A mutual update of the relational and ER Models can be made at any time.

The Module Structure

Module structures make the design of software architecture into a conscious, structured process. A module structure is a graphical presentation of a C-Module or of a COBOL- or Visual Basic-Program and displays:

Step 3: Using the right hand mouse key, click on Reserve Furniture *in the function structure to open the context menu of the function. Select* ***Display Information Flow*** *and create this diagram:*



Customer *is an external interface,* Article *and* Reservation *are entity types of the ER Model which has just been set up.*
Construct a dialog box for Display Articles *and define (using* ***Assign/Callbacks...****) that a click on the* Reserve *button by the user results in the execution of* Reserve Articles*. Your piece of System Analysis is then complete. You can now let* **case/4/0** *take over the software design.*

- each function or set of data which realize a common problem domain or technical task or which are coupled together, and
- all foreign modules or called-up programs which are used in the module.

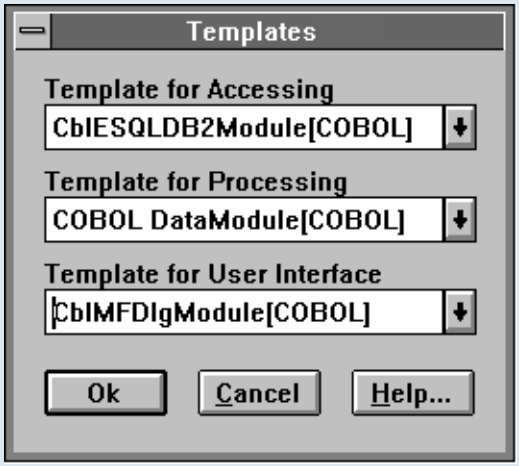
A *function* in a C-Module structure or a *section* in a COBOL-Module structure can be traced back to a problem domain function from a function structure. A special role is played here by the dialog functions. These are characterised by the "Dlg" abbreviation. If they have been taken over from a function structure then they will bring the design of the user interface created in System Analysis with them. Of course, menus and dialog boxes can also be worked upon within a module structure. With this, user interface elements, which the **case/4/0** dialog designer does not contain as standard, are bound in. If you are working with MFC-Interface Classes or with the Micro Focus Dialog System, then you will no longer need to code the interface – this is done automatically by **case/4/0**. The key to code generation – and not only for user interfaces – is generating functions, which can be assigned to the elements of a module structure.

Let case/4/0 do the programming ...

... by developing reusable generating functions for all common recurrent tasks within a software system and assigning them to the elements of a module template or module structure. As already mentioned, generating functions are formulated in **case/4/0**'s own generation language. This consists of simple, BASIC similar constructions and accesses to the repository, with which **case/4/0** ascertains the knowledge required in the current processing context. As opposed to triggering a macro, **case/4/0** does not insert a predetermined piece of code into the source, but carries out a generating function whenever you wish to see or work on the code in a module structure. **case/4/0** reverts back to the current repository contents so that all updates in the specification are automatically valid in code. If

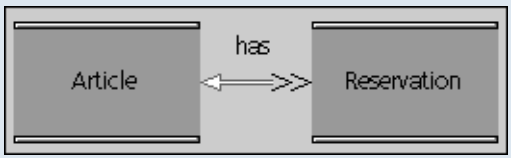
Step 4: In the context menu of the information flow you will find the menu point ***Design Assistant ...***

Step 5: The Design Assistant requires construction plan details from you, which will be used for the module design. Select the ***Templates...*** *button.*



If the templates are defined (as above for Micro Focus COBOL), then the Design Assistant will display exactly what it will generate: a relational model called Reservations, *a processing module called* Reserve Furniture, *an interface module called* Display Articles, *two access modules ... have a look for yourself!*

Step 6: When you end the dialog with the Design Assistant by clicking **OK**, *then, amongst other things, it creates this relational model:*

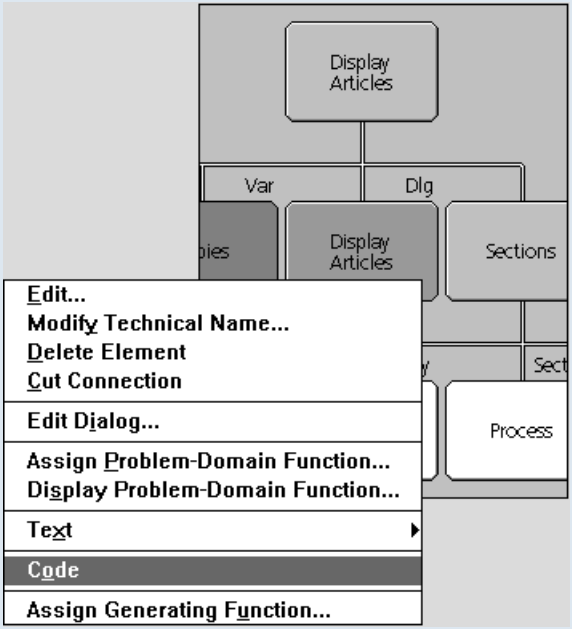


you use generating functions in module templates then you will receive complete construction plans for modules.
case/4/0 is supplied with a whole set of such. The effect of this: Design and programming of standard tasks such as interface treatment or data access is completely taken over by **case/4/0** with the aid of module templates. Where the development of complex processing logic is concerned, then the responsibility is, as before, yours. First of all you can specify processing functions from module structures with graphical pseudo code in the form of implementation trees and sub-sequently fill up the pseudo code structure with code. Also, to work in code, it is not necessary to exit **case/4/0**. The objective of this procedure is to link the specification and code so close to each other that they both stay up to date and consistent for the whole software life cycle. Code updates outside of **case/4/0** are also possible and can be subsequently taken over into the tool. Maintenance in **case/4/0** therefore always builds upon the up to date, easily readable, graphical documentation.

From Analysis to Maintenance – A Universal Route with case/4/0

At this point we will finish our inspection of **case/4/0**. Unfortunately we could not show you all of the functions offered by **case/4/0**. There still remains a lot to discover which will convince you that:

case/4/0 stands for ...



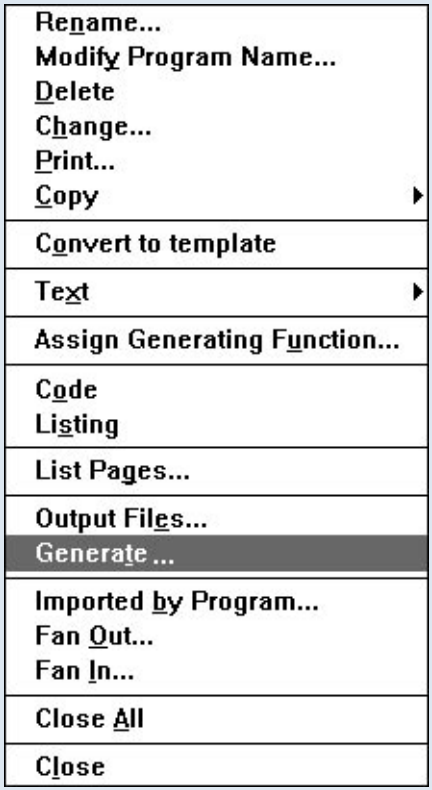
The Design Assistant has relieved you of even more work by generating a considerable part of the source code. Take a look behind the boxes of the Display Articles *module structure, for example, using the menu function* ***Code****. Or create a* ***Listing*** *by using the menu point of the same name, which you can find in the context menu of the module structure header box.*

... Great Advantages for your Projects ...

... due to the fact that the graphical results of System Analysis lead to better communication between all members of a project team and therefore to a faster and more efficient project progress. This then leads to high problem domain precision and provides strategic security well beyond individual projects.
The verification of system behaviour – before a single line is coded – also provides a certain security: Everything conceived will definitely work.
Software design and implementation with **case/4/0** means standardization of results and considerable reduction in work, measurable by the amount of statements generated. Above all other project steps the repository ensures the high quality of the formal results.
Convince yourself that **case/4/0** fulfils all that the demonstration version promises! We're looking forward to helping you develop your own specialised way to bring **case/4/0** into operation in your business. Please give us a call!

microTOOL GmbH
Voltastrasse 5
D-13355 Berlin
Phone (+49 30) 467 086-0
Fax (+49 30) 464 47 14
CompuServe: 100272,1713

microTOOL Sp. z o.o.
ul. Wołodyjowskiego 64
PL-02-724 Warszawa
Phone (+48 22) 43 52 76
Fax (+48 22) 43 81 01
E-Mail: mtool@ikp.atm.com.pl



In order to create source files from a module structure you will find two menu points in the context menu of a module structure header box: ***Output Files...*** *for the definition of generating target and* ***Generate...*** *for the output of the compiler compatible source.*

Something else you absolutely must test out – how you can create and update, for example, an analysis and requirements document by using ***Publish...*** *with Microsoft Word and OLE.*