case/4/0 goes Internet – Web Publishing with case/4/0 Version 4.3

More and more frequently, software development is being carried out in the form of a distributed project. One reason for this is intensive competition, no longer regionally bound, in such areas as public tenders. However, a new orientation to the question of location is also responsible for this trend: Know-how distributed in various locations, either due to large organization size or to outsourcing, is becoming more and more the rule. Simultaneously, a tendency towards globalization of software production can also be observed. The Internet offers interesting perspectives to distributed projects – direct access to development-results documentation, for example. **case/4/0** *takes the communication and information pathways of the Internet and Intranet into consideration: Special functions have now been integrated for publishing design documents for the web.*

case/4/0 manages all software development results centrally in its repository, ensuring their consistency in this way. The user can have the repository evaluated by **case/4/0**'s Web-Publisher at any time, creating HTML documents based on repository data. These documents can be made available to all project team members over the net.

case/4/0 remains true to its basic precept "A picture is worth a thousand words", even during web publishing. The graphical results of system analysis can be taken directly into the HTML document. In doing this, not only is a HTML page created for every diagram, but the pages are also linked with each other: From the start page, which is based on **case/4/0**'s main-menu user interface, the reader can quickly reach all contextually relevant information.

Application Example

What support does **case/4/0**'s Web-Publisher offer to software developers in everyday use?

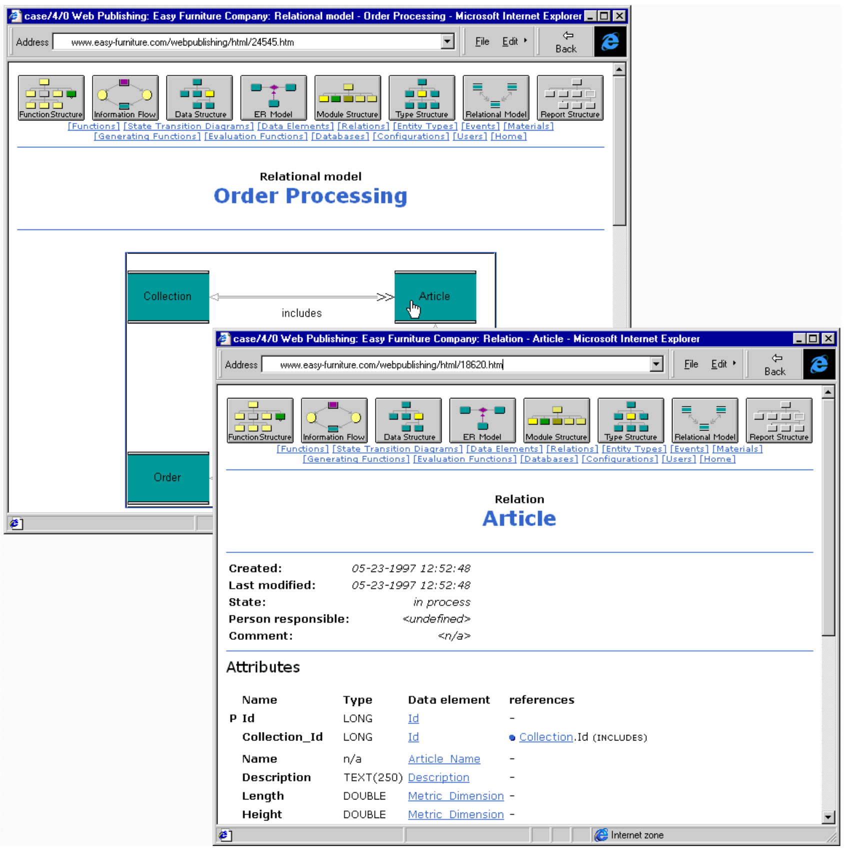
- **The Domain Specification in the Internet**
  The Web-Publisher allows you to publish the domain specification in the Internet and Intranet. The pages generated with the Web-Publisher can be displayed with any commonly available web browser, and help to stem the paper flood.
- **Decentralized Application Development**
  With **case/4/0** you can store part-systems separately, aiding you in dividing up tasks. Using the Web-Publisher, up-to-date evaluations of part-systems can be produced at any time, and complete specification and design knowledge made easily accessible to all project team members over the Internet/Intranet.
- **Informative Documents through Hyperlinks**
  HTML is particularly well suited for documenting development results, because of its ability to use links to depict relationships between individual diagrams in the repository.
- **Easily Customizable**
  HTML-page generation – just like source-code generation – is based on scripts. With the help of **case/4/0**'s proprietary script language, the user can control the output's layout and scope. This allows you to implement both functional extensions to the HTML language, and company-specific style guides.

Web Publishing Made Easy with case/4/0

After calling the Web-Publisher, you select the diagram types to publish, and the particular diagram whose characteristics are to be described in the HTML document.

The Web-Publisher generates an HTML page for every diagram. The graphic is automatically provided with hotspots, allowing easy navigation between individual pages.

In a relational model, for example, hotspots are defined for every relation. When clicked, the hotspots lead the reader directly to further information about the chosen relation. Clicking the relation "article" in the relational model leads directly to this relation. There, among other things, are listed all the relation's attributes with their references and data elements – all of which can also be reached by clicking a link.


case/4/0 Web Publishing: Easy Furniture Company: Relational model - Order Processing - Microsoft Internet Explorer
Address
www.easy-furniture.com/webpublishing/html/24545.htm
File
Edit
Back
Function Structure
Information Flow
Data Structure
ER Model
Module Structure
Type Structure
Relational Model
Report Structure
[Functions] [State Transition Diagrams] [Data Elements] [Relations] [Entity Types] [Events] [Materials]
[Generating Functions] [Evaluation Functions] [Databases] [Configurations] [Users] [Home]
Relational model
Order Processing
Collection
includes
Article
Order
case/4/0 Web Publishing: Easy Furniture Company: Relation - Article - Microsoft Internet Explorer
Address
www.easy-furniture.com/webpublishing/html/18620.htm
Internet zone

Relation
Article

Created: 05-23-1997 12:52:48
Last modified: 05-23-1997 12:52:48
State: in process
Person responsible: <undefined>
Comment: <n/a>

Attributes

	Name	Type	Data element	references
P	Id	LONG	Id	-
	Collection_Id	LONG	Id	● Collection.Id (INCLUDES)
	Name	n/a	Article_Name	-
	Description	TEXT(250)	Description	-
	Length	DOUBLE	Metric_Dimension	-
	Height	DOUBLE	Metric_Dimension	-

Figure 1: Web-Publisher output, displayed in the Internet Explorer

Custom Documents: Code-Scripting with case/4/0

case/4/0‘s repository is a relational database. For accessing this database, microTOOL has developed a simple script language. Using this language, the user can access the tables and attributes in the repository. To do this, a code script is defined, which as a rule selects a relation from the repository, and then carries out a series of operation on it.

The scripts developed for web publishing read-in information exactly in this way, and write it in HTML format into files. To ensure the compatibility of the generated pages with most browsers, the code scripts which come with the program intentionally use common HTML language elements only.

In the following, we use an example to show how such a script is created, and how the Web-Publisher’s output can be customized for special requirements, by making small modifications to the scripts.

Example – Relational Model

Our example requirement is: For every relational model in the system, generate a HTML page. Every page should include a graphic with links to all relations in the relational model.

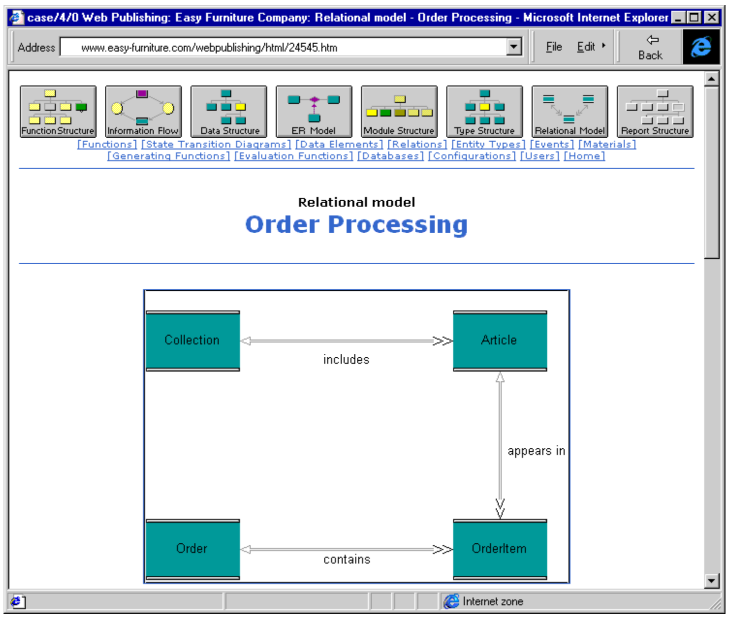

case/4/0 Web Publishing: Easy Furniture Company: Relational model - Order Processing - Microsoft Internet Explorer
Address
www.easy-furniture.com/webpublishing/html/24545.htm
File
Edit
Back
Function Structure
Information Flow
Data Structure
ER Model
Module Structure
Type Structure
Relational Model
Report Structure
[Functions] [State Transition Diagrams] [Data Elements] [Relations] [Entity Types] [Events] [Materials]
[Generating Functions] [Evaluation Functions] [Databases] [Configurations] [Users] [Home]
Relational model
Order Processing
Collection
includes
Article
appears in
Order
contains
OrderItem
Internet zone

Fig.1: The HTML page at an early stage

The appropriate code looks like this:

```
PROCEDURE main()
BEGIN
    FOREACH Relarea DO
        Filename$ = Str$(Relarea.Object)+".htm"
        CALL HTML_Relarea FOR Relarea INTO Filename$;
    END
END
```

Explanation: main() evaluates, one after the other, all the relational models in the system, and for each model calls the procedure HTML_Relarea. All instructions within the FOREACH loop relate to the relational model currently being processed. In each loop iteration, the variable Filename$ is assigned the attribute object of the currently chosen relation. Object possesses a system-wide, unique identification number, and thus can be used as the file name for the HTML page.

Finally, HTML_Relarea is called for the currently chosen relational model, and its results saved in a HTML file.

We will take a closer look at the script HTML_Relarea:

```
PROCEDURE HTML_Relarea FOR Relarea
BEGIN
   CALL Header("Relationalmodel");
   CALL Title("Relationalmodel", Conv$(Boxname));
   CALL Relation_Map(Name);
   PRINT {HTML} "<P><IMG SRC="+Str$(Object)+".png USE
                MAP=""#map""></P>", NL;
END
```

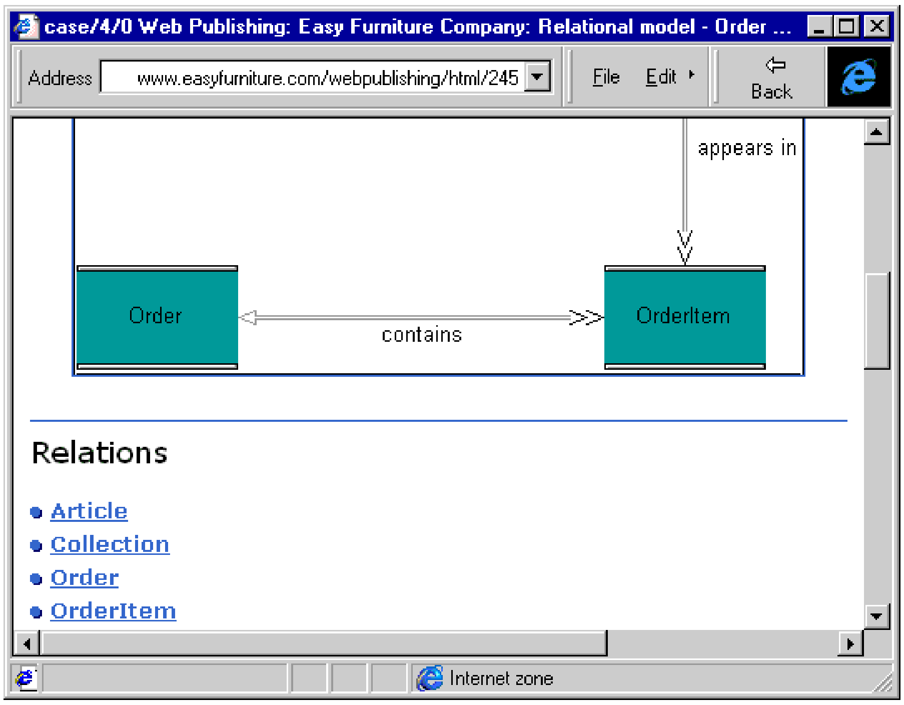
Explanation: HTML_Relarea creates the HTML code for the page:

After initializing the web page file header, the title is created with Title(). In Relation_Map the hotspots are calculated for the PNG illustration, for insertion with the next line's HTML code into the web page.

How Page Layout is Modified

Do you want all relations in the model to be listed below the diagram, with links to the appropriate HTML pages?

Then the lines marked with color in the script HTML_Relarea must be altered – that's all:


case/4/0 Web Publishing: Easy Furniture Company: Relational model - Order ...
Address
www.easyfurniture.com/webpublishing/html/245
File
Edit
Back
appears in
Order
contains
OrderItem
Internet zone

Relations

- Article
- Collection
- Order
- OrderItem

Fig.2: The HTML page after the changes

```
PROCEDURE HTML_Relarea FOR Relarea
BEGIN
   CALL Header("Relationalmodel");
   CALL Title("Relationalmodel ", Conv$(Boxname));
   CALL Relation_Map(Name);
   PRINT {HTML} "<P><IMG SRC="+Str$(Object)+".png USE
                MAP=""#map""></P>", NL;
   PRINT "Associated relations:", NL;
   FOREACH Relnode WHERE Area = Name ORDER BY Relation DO
      FOR Relation WHERE Name = Relnode.Relation DO
         CALL Link_Name(Relation.Name, Relation.Object);
      END
   END
END
```

Explanation: The code in the newly inserted lines ensures that all relations which can be found in the diagram are selected. It is possible for a relation to occur in several relational models. Such an occurrence is recorded in **case/4/0**'s information model as a further relation (Relnode). Finally, Link_Name() lists all the relations found, as links beneath the diagram in the HTML page. Link_Name() is also where you determine the layout of the listed links.