The design of an online help facility for ExSpect

Houben, G.J.P.M.

Published: 01/01/1993

Document Version
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 11. Dec. 2018
Eindhoven University of Technology
Department of Mathematics and Computing Science

The Design of an Online Help Facility for ExSpect

by

G-J. Houben

Computing Science Note 93/19
Eindhoven, June 1993
This is a series of notes of the Computing Science Section of the Department of Mathematics and Computing Science Eindhoven University of Technology. Since many of these notes are preliminary versions or may be published elsewhere, they have a limited distribution only and are not for review. Copies of these notes are available from the author.

Copies can be ordered from:
Mrs. M. Philips
Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB EINDHOVEN
The Netherlands
ISSN 0926-4515

All rights reserved
editors: prof.dr.M.Rem
         prof.dr.K.M.van Hee.
The Design of an Online Help Facility for ExSpect

Geert-Jan Houben

Abstract

Designing an online help facility is different from writing a paper manual. The design principles for online documentation differ from those for written documentation, although often the same people are responsible for both tasks. In this paper we shortly discuss general principles and guidelines for the design of online documentation and we illustrate them by evaluating the design of an online help facility for ExSpect, a software tool for writing executable specifications.

1 Design of Online Documentation

Designing online documentation for software products is quite different from writing a paper manual. While the writing of paper technical documentation is extensively studied and the resulting guidelines and principles have been tested in practice frequently, similar results are lacking for the design of online documentation. Only recently general theoretical results emerged and can be verified in practice. In this paper first some general guidelines and principles are presented. Subsequently, the actual design and implementation of an online help facility for the ExSpect software environment are considered in detail.

1.1 Online Documentation

The use of a software product generally requires a certain amount of user knowledge concerning that product. The user must know for example what buttons to press or what options to use in order to have the software product do what he or she wants it to do. This information about the use of the software product is called its documentation. The documentation specifies for all kinds of users, end-users and developers, how they can use the product. It includes help facilities, manuals, tutorials, error messages, system feedback, read-me files, demonstrations, etc.
This documentation is in general available in paper. Most of the software products today are accompanied by a number of books that contain all the relevant details about the use of the product. More and more products offer certain parts of that documentation also on screen. So, users are able to obtain the information from the documentation on the same computer screen on which they are using the software product itself. This is called online documentation (Rockley, 92).

For (end-)users online documentation reduces the number of aids that they need in order to work with the software product: besides the computer or terminal on which they are actually using the product, they do not need a large number of books telling them how the product needs to be used. Furthermore, the online documentation can make the information accessible in an easier way: with a proper design of the online documentation users should be able to find the information much quicker than in the paper version. Also, using both the product itself and its documentation on the same computer platform implies that the presentation in the documentation can be enhanced with simulations and animations close to the real product. So, benefits for end-users include increased usability, increased retrievability, and enhanced presentation.

For developers the main benefit of online documentation is that the online version can be managed better in case the information changes rapidly. Better management means less money, time and effort spent on the documentation’s maintenance. Also, if the documents can be organized into small chunks of information that each have their own purpose, an online version can be better designed to meet the users’ requirements and expectations. The information typically found in technical documentation (e.g. a reference manual) is ideally suited for an online presentation as a large collection of small individual pieces of information. Searching for an individual piece of information is easier online than on paper, as browsing through the vast amount of information, jumping from one piece to the other, can be realized faster online than on paper. Not every piece of text published on paper can be read as easily online, but the characteristics of software documentation imply that the use of online media has several benefits like lower maintenance costs and increased user satisfaction.
1.2 Use of Online Documentation

Studies show mixed and contradictory results concerning the preference of users for paper or online documentation (see (Schneiderman, 87)). (Charney, 87) claims that paper is faster and easier to read, but that despite the performance users often prefer online. According to (Nielsen, 89) older audiences prefer paper, just as inexperienced computer users (see (Charney, 87)).

Practice has shown that readers of technical documentation primarily want to get in and out of information quickly: reading-to-do, not reading-to-learn (see (Rockley, 92)). In this case the information should be chunked in an effective and efficient way in order to quickly supply the user with exactly the required part of the help information. Sometimes, readers want to explore the online information, e.g. when they are browsing the manual to see the product’s capabilities. In that case it is necessary to provide a variety of entry points and to create structured paths to related information.

Most of the time readers of technical documentation are familiar with the information and they know what they want after a repetition of searches. However, readers that are approaching new material cannot figure out what information they need. To help (new) users find their way a good organizational overview, for example by means of maps, is as helpful as the creation of logical paths through the information.

In the specification of a document writers should follow a number of steps (see (Rockley, 92)).

1. Analyze the future use of the document.
2. Define the content.
3. Based on this content organize the document by establishing links between pieces of the information.
4. Specify the screen layout.
5. Evaluate sample sessions with future users.

In their analysis of the future use of documentation two aspects are relevant for writers: the readers and the tasks they have to accomplish. In this reading analysis the questions to be raised are: Who are the users, how are they going to use the information, and what do they already know?
For the definition of the content of the documentation writers have to consider a number of problems when starting from an existing paper manual. Information must be rewritten for the screen if the hardware or software does not support technical documentation well. For most technical documentation one consequently has to deal with the real trade-off between autonomy and coherence. It is vital to chunk the information on a logical basis, but documents that are overchunked loose their coherence and their hierarchy. Possibly even the author's message can get altered. Writers must learn a whole new way of writing when they are inexperienced at writing *logically-organized* or modular documentation, as is typical for online (technical) documentation. So, when changing from paper to online a thorough analysis of the original paper document is necessary: otherwise, the required *logical chunking* is impossible.

In the organization of the document the writer uses *links* to tie the information chunks together. These links can be organizational or associative. The definition of links should start with the paper based equivalents. Additional ones should be added based on users' information needs. It is considered essential that enough information should be available for the readers to allow an informed choice between links. As (Rockley, 92) suggests writers should be selective in providing only the most important links, since information linked for unclear reasons only frustrates the user.

In relation to the screen layout of the documentation it is important to know that the hardware and software technology for publishing online is changing rapidly. Today few online publishing systems for technical documentation are available, and workstations suitable for presenting online manuals are not yet in widespread use. However, for a proper acceptance by the users the software and the hardware must meet both readers' and writers' needs.

**2 Design of an Online Help for ExSpect**

Above, a number of general principles and guidelines with respect to online documentation design have been discussed. Now, a concrete design for an online help facility for the *ExSpect* software is used to illustrate the relevant practical issues in detail.
2.1 ExSpect

ExSpect is a software tool that supports the design and the subsequent execution of formal specifications. The aim is to obtain specifications that due to their formal nature are suited for several kinds of analysis, and that also can be used as a basis for running simulations. Both aspects enhance the quality of the communication between designer and user of a specification.

The underlying model is based on a formalism called coloured Petri nets, a kind of Petri nets, with an extended data model. In ExSpect this language is supported with an environment primarily consisting of a design interface, a simulation interface, a number of analysis tools, and a large collection of libraries with reusable specifications.

The tool is developed by the Eindhoven University of Technology and now marketed by Bakkenist ExSpect. The entire tool is written in C and runs under UNIX (SunOS) in an OpenWindows environment.

2.2 Requirements

The project that we want to consider here aimed at developing a help facility for the ExSpect tool. The original and prime goal in that project was to supply the information available in the reference manual on an online basis. As a secondary goal it was allowed to offer in the help facility other information from the set of user manuals (see (ExSpect, 93)), e.g. tutorials, if this was possible elegantly.

Initially, the project was defined such that the first prototype version of the help facility should supply the necessary information from the paper reference manual in an electronic format. In later versions the quality of the help facility could then be improved on the one hand by augmenting the amount of information available online, and on the other hand by enhancing the coherence of the available help information.

As it turned out during the project the developers of the ExSpect software considered the interaction of the online help facility with the existing tools more important than the choice for a perfect platform for technical documentation. This influenced the quality of the software platform for the online documentation. The developers knew that from the documentation point of view the software eventually used for the help facility had limitations in presenting some of the basic elements from the paper manual. However, as they wanted to consider the first version of the help facility as
a prototype, they were satisfied with the expected quality. This software would offer them two prime features by enabling an easy interaction with the different ExSpect tools, and allowing easy prototyping. The software platform included filters to convert the paper text code into code for the online publication.

At the start of the project the information in the manual was already available in an electronic format (for word processing purposes). However, the original manual had been written purely for publication in a paper version. So, although the content of the information was available electronically, it was clear that its organization needed to be adapted to online publication.

Basically, the purpose of a typical chunk of information from the ExSpect reference manual was to specify the exact functionality of a menu item or the exact meaning of an element displayed on screen. So the help should just simply give an explicit specification in a case where the user in general would have a rough idea of the functionality or meaning (e.g. by having read the exact specification during the phase of learning ExSpect). Figure 1 shows the help available for the translate option from ExSpect, while Figure 2 shows the help for the main design window in ExSpect. In general the reading in the help facility would be reading-to-remember, as the user simply would want to refresh his memory with the exact details. It was not considered a principal goal to use this help facility in learning the users the ins and outs of the ExSpect software. This was left a goal for a subsequent release, in which several tutorials could become available online.

2.3 Realization

2.3.1 Conversion

In the ExSpect help project the online version was based on paper documentation written in Latex. This documentation consisted of a number of manuals specifically written for paper publication. The reference manual, the part that needed to be published online in the initial stage of the project, was organized as a sequence of more or less independent pieces of information, each giving a short, but explicit description of one of ExSpect's menu items or concepts. This meant that the reference manual was already logically-organized, as required for a representation on a chunked online basis. So, the amount of energy that had to be put into converting the original pieces of text into texts suitable for the online version appeared to
-- Translate (Shell) --

Menu sequence: 
-----------------
translate

-- Translate (Design interface) --

Menu sequence: 
-----------------
translate

Functionality: 
-----------------

If you select 'translate', the file corresponding to the current module is fed to the type checker.

* without optimizer: this menu item does not use the optimizer when translating a file

* with optimizer: selecting this item tells the ExSpect system to optimize. Optimization is done by an optimizer that works on the object file generated by the type checker

If the specification has been changed, the system asks if it has to be saved first.

Module has been changed, save first?
(seen)

If the specification is consistent, a message is displayed in the status line: 'Translating file...Done'. When the type checker finds errors, they are displayed in a separate window, see Translation errors. You can also start translate from UNIX.

Menu:
-----

define/install/view
close

Figure 1: help for the Translate option
Functionality:

---------------

The heading of the Main Design window is Design editor and the window has the module name in it. In this window the design functions like making definitions and declarations, saving and translating a specification etc. can be activated from the menu.

Menu:

-----

save
change module name...
translate =>
  without optimization
  with optimization
properties =>
  check completeness on (off)
  set confirm off (on)
  set export default on (off)
  implicit definitions in where part (..)
  default size of objects
  set grid default on (off)
definitions =>
  system definitions...
  processor definitions...
  function definitions...
  type definitions...
  definition hierarchy...
declarations =>
  system declarations...
  processor declarations...
  function declarations...
  type declarations...
include files...
module comment...
quit design

Figure 2: help for the Main Design window
be minimal beforehand.

2.3.2 Analysis

The analysis that the developers of the ExSpect help facility carried out to specify the expected usage of that help facility focused on a definition of the kinds of users and their specific requirements. Two main user groups were identified, the (end-)users and the developers of ExSpect tools. The end-users typically wanted to use the help facility for obtaining an explanation of a given ExSpect feature or for finding the ExSpect feature associated with a given functionality. Besides, they claimed that they wanted to be able to use it for getting introductory overviews and tutorials on some of the ExSpect tools. The ExSpect developers primarily wanted the same kind of use, but they also wanted to be able to deal with multiple releases of ExSpect's features. Also, they wanted to use the online version for a better management of the paper publication, in the sense that the paper version should be obtained by simply printing part of the online version.

For each of the user groups and for each type of usage the required functionality was analyzed. The functionality was also considered for its relevance to the goals of this first phase of the project. This led to a definition of the content of the chunks of information for the first prototype version. This definition was based on the users' idea that it would be sufficient to chunk the text of the reference manual into the parts evolving from the reference manual's structure, and to offer these chunks online.

The organization, the links (connections) between the individual pieces of information, evolving from this analysis appeared to be simple (in comparison to structures as in (DeBra, Houben, Kornatzky, 92)). First, the (associative) links were taken that existed already in the paper version. Subsequently, (organizational) links had to be added in order to be able to access each chunk.

The organization that resulted from this process consisted of a mix of associative and organizational links. It turned out that for the users this mix was rather non-coherent. So, during the analysis the proposed design changed as a couple of indices was introduced: a global index for accessing all chunks directly, and a small number of local indices that would offer access to the information specific for a given subject. Figure 3 shows the help information on the indices that are available for the different topics in the help facility. A consequence of this use of indices was that some of the
Functionality:

A separate index for the following ExSpect topics is available:
- Shell Index
- Design Index
- Runtime Index
- Functions Index
- Global Index

Figure 3: Topics available in the help facility

Organizational links turned out to be redundant and were removed. That new design appeared to give a better mix between associative and organizational links. The developers had the feeling that the semantical difference between both kinds of links could be better understood by the readers and that a proper access to each individual piece of information was guaranteed.

The analysis also showed that calling the help from within an ExSpect tool had to be possible through a menu option, just as the user would choose other actions in ExSpect. In general calling the help through the menu would lead to presenting the user with the main index to search for a particular piece of help. On the other hand users wanted to be able to obtain help related to a specific menu item or ExSpect element by using a simple keystroke while pointing at that item or element. So, two ways of calling help were needed: directly (with a keystroke) and indirectly (through the menu).

2.3.3 Representation

After the content and the organization of the information were discussed the attention focussed on the layout of the help facility on the screen. The first decision to take was the representation of the information chunks and their links.

For the chunks a window size and a font were chosen such that a meaningful amount of information could be presented in a readable fashion. This meant that the help facility obtained a window with a size sufficiently large
to present helpful information, while not interfering with the windows of the original ExSpect tools too much.

The connections between chunks of information were realized as *links* in a *hypertext*. This meant that while reading the text in a given chunk (a *node* in hypertext terminology) the reader could jump from a given place in the original chunk to another chunk. This place from which the user can jump to other information is referred to as an *anchor* in hypertext terminology. The jump (a link in hypertext terminology) was realized in the typical hypertext fashion such that the reader is presented with the text of the destination node, when he has activated the anchor inside the source node. The links themselves were made invisible, and only their anchors were made visible to the reader. For the links it was decided that after a jump the text of the destination node would replace the text of the source node. The anchors were represented by underlined text. It was also decided that the anchors would be placed inside the text, and not as separate buttons outside the text. This meant that most of the associative links were represented easily as they often evolved from references in the original text. For a lot of the organizational links some effort had to be spent to represent them by in-text anchors. It was decided that links to an index or to nodes previously displayed would be activated through menu options: no separate anchors, but using the menu just as in the ExSpect tools themselves. Another design decision was to use the name or title of an information chunk (visible in the frame of the node) as the exact text of the anchor referring to that chunk. All together, the mere choice for this representation implied more effort spent than was expected beforehand by the developers.

### 2.3.4 Implementation

In principle, any format could have been chosen as the format for the information in the help facility, as long as it satisfied two conditions. First, a conversion had to be possible from Latex to that format. Secondly, a parser had to be available to store the information in the database for the help facility.

In this project it was decided to use the format of *Info*, a public domain hypertext system. A self-made parser for filling the help database was available: this parser for the Info format was quick enough to be able to present help information without any unnatural delay. Moreover, the major part of the conversion from Latex to Info was covered by *makeinfo*, a public domain
software program. This program could convert from Texinfo to Info, while Latex and Texinfo are closely related as they are both based on Tex. So, the ExSpect programmers only needed to develop a routine from converting from Latex to Texinfo.

The original aim was to obtain one source from which both the paper and the online version could be constructed automatically. So, despite the conversion routine, the final source code would become an integration of code for the two versions. The developers tried to keep as much as possible of the code at a unique place. They wanted to have as less as possible manual effort spent on the future maintenance by minimizing the work for updating both versions of the documentation in case of new releases.

Therefore, the original approach was to just add the definition of nodes and links into the code used for the existing paper manual and have the (source-code-to-online-code) conversion routine ignore the paper markup. This meant that hypertext formatting commands invisible for the word processor needed to be added to the code for the definition of nodes, links and anchors. This would be a one-time job, only to be done for the very first version. For later versions an addition of text to the help information would then consist not only of the necessary word processing formatting commands, but also of a small number of hypertext formatting commands.

The quality of the paper documentation was sufficient, and the omissions that came forward after the new analysis were fixed leading to an even higher quality. Yet, it turned out that the original approach was too optimistic: part of the information for the online documentation had to be kept in separate nodes without equivalent sections in the paper documentation.

A first cause for the differences was that some text was not present in the paper documentation. Some of it was simply forgotten and was included in later releases, but some of it was typical for the online help. This latter text included information that was not in the reference manual but in other parts of the paper documentation, and that had to be included for the sake of completeness. Information that was also available online but not on paper concerned help on the help facility itself and on the use of the online indices.

A second cause for differences was due to the quality of the software for the help facility. Some information could not be represented effectively online on the basis of the format used for paper publication. The main problem with the software was the primitive layout on screen. Some information represented elegantly on paper could not be displayed adequately on screen.

The amount of text exclusively present online was larger than expected,
perhaps even larger as effectively manageable. That parts of the paper manu-

ual would have no counterpart in the online version was expected and part of the project's definition. That the opposite would hold in this magnitude was not expected. This undesirable phenomenon could not be avoided while trying to uphold certain quality standards for the online documentation, and it caused an unacceptably high amount of maintenance in case of future changes.

The help facility was designed to offer its help information in one window separate from the other ExSpect windows. So, the first time help was asked this window had to be opened, and subsequently this help window would stay open as long as the user wanted. The interaction between the help facility and the other ExSpect tools was realized such that the help window could be opened with the proper information right away: starting up the help facility would not cause any major delays. Remember that this was one of the reasons why the developers chose this software for the help facility.

The design implied that the user would be able to start the help facility in two ways. By choosing first the help button on the keyboard and subsequently an item from the ExSpect menus the help facility would open the help window with the information on that menu item directly available. A similar way would be followed for help on ExSpect elements in windows on the screen. By double clicking the help button on the keyboard (or choosing the main index item from the ExSpect menu) the help facility would open with the main index such that the user would be able to search for desired information.

3 Evaluation

In the evaluation of the ExSpect online help project a number of stages in the project can be considered:

- converting the existing paper manual to a new online manual;
- analyzing the expected use of the new online help facility;
- defining the nodes and links for the help information;
- choosing representations for those nodes and links;
- implementing the help facility;
• using that implementation.

3.1 Conversion from Paper to Online

The original aim when converting from paper to online was to extend the code for the paper version with some (hypertext) markup such that both the paper and online version could be generated more or less automatically from one source code. The major problem to deal with was the software that the ExSpect developers decided to use for the word processing and for the online facilities. This choice appeared to be in conflict with the requirement to obtain easy maintenance, both during the first realization\(^1\) and in the future.

During the first realization the text for the paper manual had to be extended with a portion of information that was specific for the online help. In order to try to minimize the effort spent on future changes, effort was put into avoiding unnecessary redundancy as much as possible. The larger problem, however, turned out to be the necessary mix of word processing formatting commands and hypertext formatting commands in the source code. This difficult interaction between word processing and hypertext would be reduced with the use of software better suited to serve both needs (at the same time).

While some of the effort was typical for a first release of the online facility, it is also interesting to look at the effort to be spent on maintaining the information both on paper and online in the future.

The first aspect related to future maintenance concerns the conversion routine from the unique source format to the paper and online formats. Although the entire routine manages to do a major part of the conversion automatically, it causes a lot of changes still to be made by hand. If this version of the conversion routine, specifically the self-made conversion from Latex to Texinfo, has to be used at every release of the documentation it will demand an unnecessarily high effort in preparing the paper and online documentation. So, as long as this software platform is being used and a conversion routine is necessary, it is vital to improve the quality of the

\(^1\)Note that the ExSpect programmers had built the conversion from Latex to Texinfo. This had been more difficult than expected: this had not been caused by Info or Texinfo, but by the possibility of nesting commands in Latex. Also, they had developed the parser. This parser had to be quick enough to show information in a natural way: explicitly reading and parsing the Latex information was not good enough.
routin such that more is done automatically.

The second aspect in connection to future maintenance is the addition of new information to the contents. The effort for adding new text can be kept to a minimum if the necessary markup for the hypertext formatting is minimal and if the redundancy in the source code is minimal.

The necessary hypertext markup is comparable with the markup needed for word processing. So, in view of the used software the additional markup can be considered (close to) minimal.

As far as the redundancy is concerned, two aspects need to be mentioned. First, the original paper version still contained redundant information: converting this to online did not remove this automatically. This aspect can be avoided in the future by a higher quality of (logically-organized) writing. Secondly, as some of the graphics from the paper manual could not be displayed with the online software new representations had to be chosen that sometimes caused redundancy. This will disappear when the proper software (capable of displaying graphics) is used.

So, concluding, with the current software the effort to be spent on maintenance is higher than necessary. In a next phase of the project the use of other software should be considered that is able to display the same quality on screen as on paper while maintaining a simple input format, and that can interact with the ExSpect tools as easily as the current software. As long as the same software platform is to be used, then the associated conversion routine has to be improved in the sense that more can be done automatically.

3.2 Analysis of the Expected Use

The goal of this analysis was to identify the user groups and their requirements.

The main problem encountered was the lack of knowledge of some of the intended users with respect to hypertext and its possibilities: the concepts on which this help facility is based come from the field of hypertext, but most users had no proper idea of the possibilities and impossibilities of hypertext solutions (e.g. users not familiar with the link concept tend to define too many links). This problem was solved by holding sample sessions and presenting demos of comparable systems. Thus they got an insight into the limitations of hypertext. This was important as most of them were definitely aiming too high in specifying the requirements for the help facility in comparison to the effort it would cost and the benefits they would get. Since the
advantages and disadvantages of the used software were not properly known at the time of the analysis, it was vital that the future users were given a good conception of what they could expect in the realization of their wishes.

Concluding, one can state that the use of sample sessions on the future platform and demonstrations of comparable software are essential as part of the analysis to convey to the future users the possibilities and impossibilities of using hypertext solutions.

3.3 Definition of Nodes and Links

In the third stage of the project the contents of the nodes needed to be defined, just as the links between nodes.

The contents of the nodes were basically determined by the way in which the original document was divided into chunks. The chunks corresponded to the sections from the reference manual. This reference manual was already structured and organized: some large sections needed to be split, while some small ones needed to be merged, but there was no real problem in trying to obtain one format and one size for the chunks. A fair amount of work had to be put into rewriting the text to make it suitable for reading both on paper and online: no place references, other figures, etc. Most of this effort was typical for the first time. Future text changes will cause a limited effort, but not a minimal effort.

The definition of the links was started from the relationships explicitly present in the text. A thorough analysis of associative relationships produced many new links. Additional links (organizational links) were added to obtain a logically-organized structure. Indexes were added, one global index and some local ones, to complete the organization such that easy access was possible: organizational links for accessing the information, and associative links for reading the information.

All together the production of links was a huge first-time effort. The maintenance of links will probably stay a serious effort, as the organizational links need to be revisited every time to see whether the access to all information is good enough.

The difference between the use of organizational links and associative links was not clear enough for naive users: this caused a wrong perception of searching vs. reading. This aspect had to be discussed extensively during
sample sessions and demos. The difference between searching and reading is important as in the organization of the document organizational and associative links have a different purpose: the users must know this difference as they can not properly exploit the link structures otherwise. E.g. the use of indices or other organizational links is different from the use (reading) of standard text chunks.

Another problem in specifying both paper references and online links was the effort to be spent on expressing those two relationships in the unique source code. This was caused by the fact that the conversion routine software did not work effectively and properly.

The conclusion is that the definition of nodes and links took a huge, but unavoidable first-time effort. Due to the software problems a significant effort will be needed for maintenance. However, the paper manual has gained quality due to a thorough analysis of its structure.

3.4 Representation of Nodes and Links

As far as the representation of nodes is concerned the requirements said that nodes should look like pages from the reference manual. The problems in this respect were the window size, the font size, the use of special symbols, and the use of graphics.

The size of the help window and its font were chosen such that the help text would be elegantly readable and that the window could mix well with the other ExSpect tools on the screen: the possible chunk size that followed from this did not cause a real problem.

The use of software that could not cope with special symbols, nor with graphics, resulted in a decrease in the educational quality of the help facility and a decrease in the quality of the association between the help information and the actual tools. This was significant as the text-without-graphics online facility caused a rewriting of the help information (to a lower quality).

For the representation of links it was necessary to define the representation of anchors and the visibility of links. The anchors were put in the text in different style (underlined), which required a limited rewriting and which purposely made no difference visible between the two types of links (organizational and associative). Alternatively, links could be activated by the use of menu options, just like in the ExSpect tools themselves. In particular
this was necessary for the links that make it possible to jump to an index or to nodes previously displayed: links without an explicit anchor. The use of a link was defined to cause a new help text to replace the old one: only one help window would be available and thus enough space would be kept for the ExSpect windows.

The conclusion is that there have been no problems with the link representation, but that the node representation suffered heavily from the lack of graphics support.

3.5 Implementation

In the implementation phase the information had to be realized in the software.

As the software was simple but poor, it turned out to be easy to use, but with limited capabilities. A lot of energy had to be spent on aspects that were not first-time-only. The coding of links and anchors was difficult. The text, including the text that had to be added, had to be (re)written as text without graphics. The conversion routine was not error-free, so a lot of damage repair was needed.

Concluding, the used software platform is not meant for easy maintenance, nor for high-quality information representation.

3.6 Use of ExSpect's Help

The evaluation of the use of the first version of the ExSpect help facility learned that the calling of help was simple: the call for help was integrated in the ExSpect tools as a simple menu item and as a simple keystroke command. The reading of large chunks was sometimes difficult, as no good layout was possible.

The users appeared to read more in a reading-to-learn than a reading-to-remember fashion, which was in contrast with claims made beforehand. So, although this phenomenon could be related to the fact that users had to learn to use the facility, for the prototype it held that the expected use differed from the actual use: there existed a wrong conception of the use of paper and online manuals. One can assume that this phenomenon may disappear over time.
Important lessons were learned by the manual writers for improving the quality of the manual in order to adapt the manual text to the users' needs (irrelevant of the conversion to online).

The online facility is easier to access, but not better to read or to understand, nor is it better maintainable. The general concepts are good, but the specific software platform is not good enough. So, as a prototype the implementation appears to function properly, but as a final product it is certainly not sufficient.

4 Conclusions

The prime goal in the ExSpect-help project was to publish ExSpect's technical documentation online. As a start the first stage of the project aimed at realizing the reference manual online. In order to minimize the future maintenance one of the principal requirements was the use of a unique source format for both paper and online version. As a consequence the ExSpect developers chose to use Info, as this offered an "easy" conversion (from the original situation) and a hypertext-like functionality. The developers started with a thorough analysis of the future use. Then, the nodes and links were defined and their representations. After the realization each of these steps was evaluated, as was the use of the prototype online facility.

The evaluation learned that the required functionality had been realized, but the presentation of the information had suffered and more effort than expected had to be spent on the conversion of the contents. Primarily these problems had been caused by the software that was used, although one must admit that the software had offered the easy interaction and prototyping it was chosen for. An advantage in this project had been the structure of the existing reference manual. As it was logically-organized the conversion was semantically relatively simple. The evaluation resulted in a number of possible goals for next releases: the use of graphics (and thus probably of other software), a higher quality of the text (as the rewriting sometimes was cumbersome for text that was not logically-organized), and better support for future maintenance (by using a good hypertext (management) system).

As one of the general conclusions from this project one can see that a conversion from paper documentation to online documentation takes a significant (manual) effort even if the text is logically-organized. A good
thorough analysis is vital, and in the analysis it is important to know the advantages and disadvantages of the used software. The future users have to be familiar with the limitations of hypertext in general and the software in particular, when they formulate their requirements. A solid management is needed for maintaining the paper and online documentation in parallel: preferably by using one source format with minimal redundancy in markup for both media. The rewriting of the original text is difficult for new, inexperienced writers and for writers not familiar with writing texts that are not logically-organized.

One of the general lessons learned from the project is related to the analysis for determining requirements for a hypertext solution. In the user analysis "real" sample sessions need to be held with the users in order to help those that are not familiar with using hypertext-like technical documentation. Readers less familiar with hypertext tend to abuse the hypertext concepts.

Whenever the text that has to be converted to online is not logically-organized, it is wise to rewrite the text before the conversion. In the rewriting the writers should maintain a clear distinction between associative and organizational links, as searching and reading are two separate issues in using a hypertext.

Using software that enables (almost) the same layout possibilities for paper and online reduces the effort in converting from paper to online and in maintaining paper and online versions.

For the future of the project the developers of ExSpect want the next releases of ExSpect’s help facility to have an enhanced functionality: other parts of the current paper documentation should be incorporated in the online facility, the reading quality should be upgraded, and the maintenance routines should become simpler. In the long term they think of choosing another software platform that can better serve their needs: the online help facility requires a platform that makes effective and efficient maintenance possible.
References


In this series appeared:

<table>
<thead>
<tr>
<th>Issue</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>91/02</td>
<td>R.P. Nederpelt, H.C.M. de Swart</td>
<td>Implication. A survey of the different logical analyses &quot;if...,then...&quot;, p. 26.</td>
</tr>
<tr>
<td>91/03</td>
<td>J.P. Katoen, L.A.M. Schoenmakers</td>
<td>Parallel Programs for the Recognition of P-invariant Segments, p. 16.</td>
</tr>
<tr>
<td>91/05</td>
<td>D. de Reus</td>
<td>An Implementation Model for GOOD, p. 18.</td>
</tr>
<tr>
<td>91/06</td>
<td>K.M. van Hee</td>
<td>SPECIFICATIEMETHODEN, een overzicht, p. 20.</td>
</tr>
<tr>
<td>91/07</td>
<td>E. Poll</td>
<td>CPO-models for second order lambda calculus with recursive types and subtyping, p. 49.</td>
</tr>
<tr>
<td>91/12</td>
<td>E. van der Sluis</td>
<td>A parallel local search algorithm for the travelling salesman problem, p. 12.</td>
</tr>
<tr>
<td>91/14</td>
<td>P. Lemmens</td>
<td>The PDB Hypermedia Package. Why and how it was built, p. 63.</td>
</tr>
<tr>
<td>91/16</td>
<td>A.J.J.M. Marcelis</td>
<td>An example of proving attribute grammars correct: the representation of arithmetical expressions by DAGs, p. 25.</td>
</tr>
</tbody>
</table>
91/18 Rik van Geldrop  
Transformational Query Solving, p. 35.

91/19 Erik Poll  
Some categorical properties for a model for second order lambda calculus with subtyping, p. 21.

91/20 A.E. Eiben  
R.V. Schuwer  

91/21 J. Coenen  
W.-P. de Roever  
J.Zwiers  
Assertional Data Reification Proofs: Survey and Perspective, p. 18.

91/22 G. Wolf  

91/23 K.M. van Hee  
L.J. Somers  
M. Voorhoeve  
Z and high level Petri nets, p. 16.

91/24 A.T.M. Aerts  
D. de Reus  
Formal semantics for BRM with examples, p. 25.

91/25 P. Zhou  
J. Hooman  
R. Kuiper  
A compositional proof system for real-time systems based on explicit clock temporal logic: soundness and completeness, p. 52.

91/26 P. de Bra  
G.J. Houben  
J. Paredaens  
The GOOD based hypertext reference model, p. 12.

91/27 F. de Boer  
C. Palamidessi  
Embedding as a tool for language comparison: On the CSP hierarchy, p. 17.

91/28 F. de Boer  
A compositional proof system for dynamic process creation, p. 24.

91/29 H. Ten Eikelder  
R. van Geldrop  
Correctness of Acceptor Schemes for Regular Languages, p. 31.

91/30 J.C.M. Baeten  
F.W. Vaandrager  
An Algebra for Process Creation, p. 29.

91/31 H. ten Eikelder  
Some algorithms to decide the equivalence of recursive types, p. 26.

91/32 P. Struik  

91/33 W. v.d. Aalst  
The modelling and analysis of queueing systems with QNM-ExSpect, p. 23.

91/34 J. Coenen  
Specifying fault tolerant programs in deontic logic, p. 15.

91/35 F.S. de Boer  
J.W. Klop  
C. Palamidessi  
Asynchronous communication in process algebra, p. 20.
92/01 J. Coenen
J. Zwiers
W.-P. de Roever

A note on compositional refinement, p. 27.

92/02 J. Coenen
J. Hooman

A compositional semantics for fault tolerant real-time systems, p. 18.

92/03 J.C.M. Baeten
J.A. Bergstra

Real space process algebra, p. 42.

92/04 J.P.H.W.v.d.Eijnde

Program derivation in acyclic graphs and related problems, p. 90.

92/05 J.P.H.W.v.d.Eijnde

Conservative fixpoint functions on a graph, p. 25.

92/06 J.C.M. Baeten
J.A. Bergstra

Discrete time process algebra, p. 45.

92/07 R.P. Nederpelt

The fine-structure of lambda calculus, p. 110.

92/08 R.P. Nederpelt
F. Kamareddine

On stepwise explicit substitution, p. 30.

92/09 R.C. Backhouse


92/10 P.M.P. Rambags

Composition and decomposition in a CPN model, p. 55.

92/11 R.C. Backhouse
J.S.C.P.v.d.Woude

Demonic operators and monotype factors, p. 29.

92/12 F. Kamareddine


92/13 F. Kamareddine

Set theory and nominalisation, Part II, p. 22.

92/14 J.C.M. Baeten

The total order assumption, p. 10.

92/15 F. Kamareddine

A system at the cross-roads of functional and logic programming, p. 36.

92/16 R.R. Seljée

Integrity checking in deductive databases; an exposition, p. 32.

92/17 W.M.P. van der Aalst

Interval timed coloured Petri nets and their analysis, p. 20.

92/18 R.Nederpelt
F. Kamareddine

A unified approach to Type Theory through a refined lambda-calculus, p. 30.

92/19 J.C.M. Baeten
J.A. Bergstra
S.A. Smolka

Axiomatizing Probabilistic Processes: ACP with Generative Probabilities, p. 36.

92/20 F. Kamareddine

Arc Types for Natural Language? P. 32.

92/21 F. Kamareddine

Non well-foundedness and type freeness can unify the interpretation of functional application, p. 16.
<table>
<thead>
<tr>
<th>Year</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>92/22</td>
<td>R. Nederpelt, F. Kamareddine</td>
<td>A useful lambda notation, p. 17.</td>
</tr>
<tr>
<td>92/23</td>
<td>F. Kamareddine, E. Klein</td>
<td>Nominalization, Predication and Type Containment, p. 40.</td>
</tr>
<tr>
<td>92/24</td>
<td>M. Codish, D. Dams, Eyal Yardeni</td>
<td>Bottom-up Abstract Interpretation of Logic Programs, p. 33.</td>
</tr>
<tr>
<td>92/25</td>
<td>E. Poll</td>
<td>A Programming Logic for F0, p. 15.</td>
</tr>
<tr>
<td>93/01</td>
<td>R. van Geldrop</td>
<td>Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.</td>
</tr>
<tr>
<td>93/02</td>
<td>T. Verhoeff</td>
<td>A continuous version of the Prisoner's Dilemma, p. 17</td>
</tr>
<tr>
<td>93/03</td>
<td>T. Verhoeff</td>
<td>Quicksort for linked lists, p. 8.</td>
</tr>
<tr>
<td>93/04</td>
<td>E. H. L. Aarts, J. H. M. Korst, P. J. Zwietering</td>
<td>Deterministic and randomized local search, p. 78.</td>
</tr>
<tr>
<td>93/05</td>
<td>J. C. M. Baeten, C. Verhoef</td>
<td>A congruence theorem for structured operational semantics with predicates, p. 18.</td>
</tr>
<tr>
<td>93/06</td>
<td>J. P. Veltkamp</td>
<td>On the unavoidability of metastable behaviour, p. 29</td>
</tr>
<tr>
<td>93/07</td>
<td>P. D. Moerland</td>
<td>Exercises in Multiprogramming, p. 97</td>
</tr>
<tr>
<td>93/08</td>
<td>J. Verhoosel</td>
<td>A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, p. 32.</td>
</tr>
<tr>
<td>93/10</td>
<td>K. M. van Hee</td>
<td>Systems Engineering: a Formal Approach Part II: Frameworks, p. 44.</td>
</tr>
</tbody>
</table>
A useful lambda notation, p. 17.
Nominalization, Predication and Type Containment, p. 40.
Bottum-up Abstract Interpretation of Logic Programs, p. 33.
A Programming Logic for Fω, p. 15.
A modelling method using MOVIE and SimCon/ExSpect, p. 15.
A taxonomy of keyword pattern matching algorithms, p. 50.
Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.
A continuous version of the Prisoner's Dilemma, p. 17
Quicksort for linked lists, p. 8.
Deterministic and randomized local search, p. 78.
A congruence theorem for structured operational semantics with predicates, p. 18.
On the unavoidability of metastable behaviour, p. 29
Exercises in Multiprogramming, p. 97
A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, p. 32.
Systems Engineering: a Formal Approach Part II: Frameworks, p. 44.
93/15 J.C.M. Baeten
J.A. Bergstra
R.N. Bol


93/16 H. Schepers
J. Hooman

A Trace-Based Compositional Proof Theory for Fault Tolerant Distributed Systems, p. 27

93/17 D. Alstein
P. van der Stok

Hard Real-Time Reliable Multicast in the DEDOS system, p. 19.

93/18 C. Verhoef

A congruence theorem for structured operational semantics with predicates and negative premises, p. 22.