Computational Chemistry Network Services and User Interfacing

A. H. M. Thiers, J. A. M. Leunissen, T. M. Miller, G. Schaftenaar, and J. H. Noordik*

CAOS/CAMM Center, University of Nijmegen, Toernooiveld, NL-6525 ED Nijmegen, The Netherlands

Received May 13, 1993®

The best use of computer-aided techniques by chemists naturally depends on the quality of available (network) facilities. Specifically, the optimization of user support and network access to integrated systems of data and tools has taken on increased importance in recent years with the increasing (internet) network capacity and availability. This paper reports on our ongoing development of a general user interface to chemical network services, where the emphasis is on the ease of maintenance for the Center, the service provider, and the ease of use for the chemist/molecular biologist, the end-user.

INTRODUCTION

Academic research institutions in the Netherlands are fully connected by a firmly established medium-to-high speed network, SURFnet, which allows reliable internode communication. Computer systems at some of these nodes, such as the CAOS/CAMM Center, have been established as part of a national strategy to centralize the "information" services for certain areas of science and technology; for the CAOS/CAMM Center this area comprises a number of chemical disciplines. For each of these disciplines, the Center is the "information services" provider via the network to end users at all Dutch universities. In this context, information services should be interpreted as an integrated system of tools, centered, around factual and/or knowledge databases and consisting of data access, retrieval, manipulation, calculation, and visualization facilities. For access to these facilities, we have developed a menu system which provides our users with a connection to a "virtual computer": that is, many computers appear as one to the user, each of them offering a dedicated service or a group of services and all services being accessible through a general interface. This concept of centralization of basic chemical information services for academic chemical research, on a network accessible "virtual computer" system, has several justifications.

Academic software distribution to interested academic parties often still occurs for a nominal fee, and little effort is required to acquire such software. However, installation and maintenance of this software requires many more (personal) resources, and we frequently find a scenario wherein at least one person in a research group or department is "lost" to computer maintenance and program updates. Commercially available software (e.g. modeling packages or reaction retrieval systems) also requires considerable manpower in maintenance and installation, as well as access to expert knowledge on the particular programs and databases installed. This personnel expense comes on top of specific hardware requirements for particular software systems, which could mean the purchase of new computers and disks. With commercial software, there are additional software licensing costs, which are rapidly increasing and are sometimes already becoming prohibitive for departmental installations. Centralization of services, which are network accessible and offered through an intuitive interface, alleviates many of these problems.

Building a virtual computer interface is rather straightforward if the number of user initiated processes, the application programs, is limited and if these processes are executed on computers within one vendors' product line, e.g. a VAX cluster.[1] If the interface has to cope with many different platforms, e.g. VAX/VMS[2], Unix/Ultrix,[3] and Convex,[4] plus a large spectrum of chemical databases and computational chemistry and molecular biology tools, the situation becomes much more complicated. This latter situation has evolved in many chemistry departments since the mid-1980s, and this is the situation we have successfully tackled at the CAOS/CAMM Center. Much of the Center's chemistry software was originally made available for VMS-based machines but currently more often for Unix. Computer sales trends show researchers and chemistry software development moving more and more to Unix-based systems because of machine speed and versatility. In order to follow and enhance this move to Unix-based machines, the Center had to extend its "virtual machine" beyond the VMS platform. What was once a VAX cluster here at the CAOS/CAMM Center has been expanded with the addition of CONVEXes, Ultrix workstations, Silicon Graphics, and Sun systems. This conglomerate of machines now forms the hardware core of the services offered by the Center over the network, with the understanding that the services are not necessarily limited to machines located at the Center but may even be network accessible elsewhere. This concept of a virtual computer is important in the discussion of the integration of systems and chemistry software tools at the Center. Our "one big machine" avails the user of all possible resources at the same time. If remote user selected resources include programs running interactively on the other side of the world via the network, the Center's computers merely pass characters. Users come into the system and are routed out again to run specific programs at special sites; however to them it still is "one stop shopping". The interface developed for this situation and described in this paper concerns a menu system running, and keeping the user, in a single operating system environment only (VAX/VMS cluster) but accessing all available services on the different machines. Developments are currently in progress to make the actual menu system and the user environment fully system independent.

THE CAOS/CAMM CENTER MENU SYSTEM

An Overview. Figure 1 shows the current composition of the Center's "virtual computer hardware". It consists of an ethernetted VAX-cluster (nodes CAOSxx), a DEC Ultrix "cluster" (nodes cammsx), two CONVEX C1 systems, two Silicon Graphics nodes, and two SUN nodes. The different software tools or services offered by the Center are each executed at specific (dedicated) nodes or hardware boxes in this configuration but accessed by the user from one main menu.

® Abstract published in *Advance ACS Abstracts,* October 15, 1993.

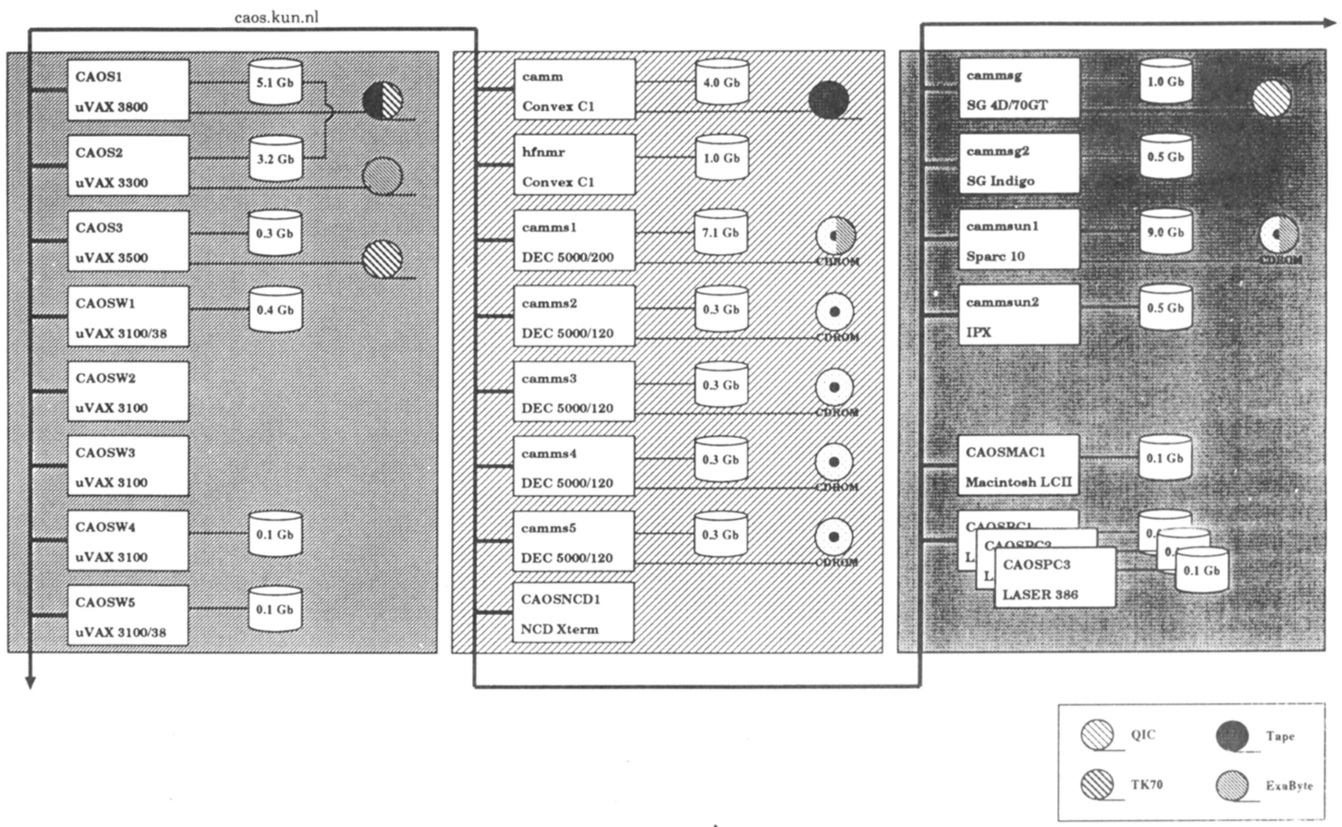
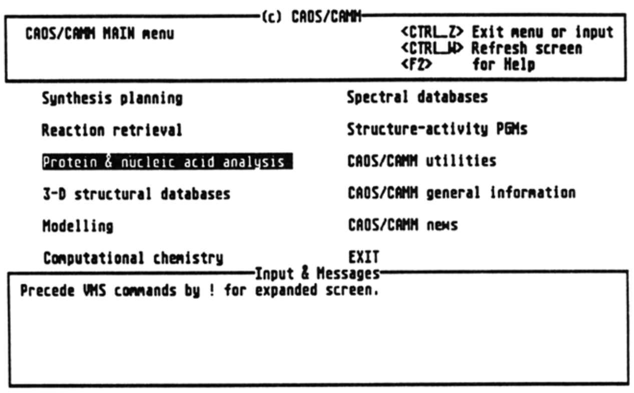
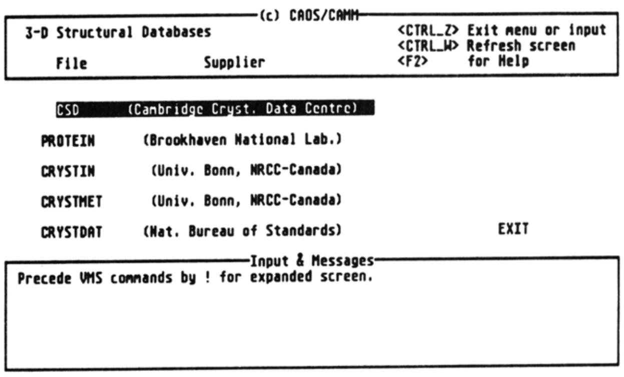


Figure 1. The Center's current computing environment. Users login to the domain caos.kun.nl, where they enter into a menu system (Figure 2) which allows the execution of specific tools on specific hardware boxes of the "big machine". Services on external nodes are not included in this figure.


```
                              (c) CAOS/CAMM
CAOS/CAMM MAIN menu                          <CTRL_Z> Exit menu or input
                                             <CTRL_W> Refresh screen
                                             <F2>     for Help

  Synthesis planning                    Spectral databases
  Reaction retrieval                    Structure-activity PGMs
  Protein & nucleic acid analysis       CAOS/CAMM utilities
  3-D structural databases              CAOS/CAMM general information
  Modelling                             CAOS/CAMM news
  Computational chemistry               EXIT
                                  Input & Messages
Precede VMS commands by ! for expanded screen.
```

Figure 2. Main menu screen. Allows user selection of specific tools grouped together under a "button" in the button window of the screen.


```
                              (c) CAOS/CAMM
3-D Structural Databases                     <CTRL_Z> Exit menu or input
                                             <CTRL_W> Refresh screen
   File             Supplier                 <F2>     for Help

  CSD       (Cambridge Cryst. Data Centre)
  PROTEIN   (Brookhaven National Lab.)
  CRYSTIN    (Univ. Bonn, NRCC-Canada)
  CRYSTMET   (Univ. Bonn, NRCC-Canada)
  CRYSTDAT  (Nat. Bureau of Standards)                  EXIT
                            Input & Messages
Precede VMS commands by ! for expanded screen.
```

Figure 3. Submenu screen for 3-D structural database services.

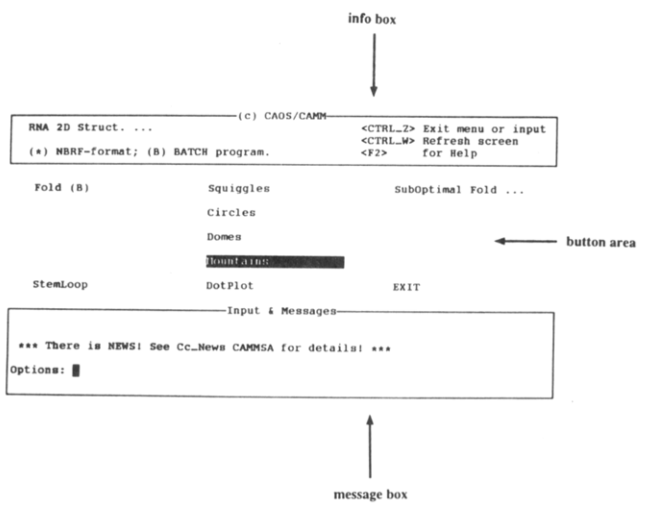
Typically a remote user would access the Center's services by opening a Telnet session[5] to the Internet domain *caos.kun.nl* and be supplied with a menu screen as shown in Figure 2. The screen layout will be discussed below. By selecting a service with the keyboard cursor control keys (highlighted button) and activating it by a ⟨CR⟩, the user will interactively execute a specific program set at a specific hardware box, generally thru a submenu.

As an example, Figure 3 shows the submenu screen which is presented to the user upon activation of the *3-D structural databases* button. Of the services accessible from this screen, the first one, CSD, is actually executed on one of the CONVEX systems through a task-to-task communication protocol[6] between VAX and CONVEX. However, the user is unaware of these underlying layers and will only see hits on his search actions being displayed in the window of his current session. The last service presented in the screen of Figure 3 is actually executed at a machine at the National Research Council in Ottawa, Canada, by establishing an automatic login[7] to that system. Again, the user is "unaware" of this action and simply continues his on-line dialogue.

Basic Functionality. The function and form of the menu system, which has been under development since 1988, is illustrated in Figure 4 which displays one of the molecular biology services screens and shows some of our basic requirements for a menu system. Each menu screen displays three distinct windows.

The uppermost window is an *information window*. It displays mainly static information like the current menu header name (uppermost left hand corner) as well as some keyboard definitions used to control the program (upper right corner) plus button headers for the buttons controlling application programs which are displayed in the middle of the screen.

At the bottom of the screen there is a *message window*. This window acts as an operating system shell within the menu and has the functionality of the full screen display. It displays output from actions and receives input from the user, both when prompting for additional parameters and when the user enters a command directly from the keyboard. VT100 type



```
---------------------------(c) CAOS/CAMM---------------------------
 RNA 2D Struct. ...                       <CTRL_Z> Exit menu or input
                                          <CTRL_W> Refresh screen
 (*) NBRF-format; (B) BATCH program.      <F2>     for Help

 Fold (B)            Squiggles            SubOptimal Fold ...
                     Circles
                     Domes
                     Mountains
 StemLoop            DotPlot              EXIT
--------------------------Input & Messages--------------------------

*** There is NEWS! See Cc_News CAMMSA for details! ***
Options:
```

Figure 4. Molecular biology submenu illustrating the layout and the function of the menu system.

screens are supported to serve low-end terminals.

The middle portion of the screen, the *button window*, is used for the buttons themselves, which are defined via a menu definition file (MDF). The buttons in this window activate other menu screens or application programs. The help functionality (VT100 F2 key) on each of the buttons will call-up a window overlaying the menu screen and displaying some text on the action of the button. Some buttons have added functionality, e.g. a file selector function, which also overlies the current screen. Internal menu commands, specified in the MDF file and user commands typed directly on the menu are executed in the *message window*. Help strings are also passed to this window for button descriptors with a H/I (help specification/internal help function; see Figure 5) qualifier in the MDF file. MDF files are simple ASCII files, specifying the appearance of the menu and the actions on the button activations. A part of the MDF file for the screen of Figure 4 is shown in Figure 5. We developed a Fortran program, *MENUXS*, which operates on the MDF file and controls the various actions specified in this file. MENUXS incorporates VAX/VMS screen management routines to depict information, message, and button windows from within VMS. Several keys are defined to control special menu functions, and the standard VT100 keyboard layout is incorporated.

Description of Menu Structure. In principle menu actions are executed in MENUXS by looping over the buttons of the MDF file, which is no more than the description of the buttons appearing on the screen. The formal description of an MDF file is shown in Table I.

As illustrated in the example MDF file of Figure 5, column 1 is reserved for the start of a description of a button, a message box, or an information box; all other lines start with a blank character (tab or space). Lines with an exclamation mark ("1") in column 1 are comment lines which are not interpreted by MENUXS. For each button description, the integer number in the first column indicates the level in the tree of menus. Each menu screen is rooted at a button higher in the tree hierarchy. The parent button of a particular submenu has a level which is one (1) less than its children, which should be described immediately after the parent. In effect, all buttons with a higher level number between two buttons of the same level are successors of the first one. This hierarchy is illustrated in Table II and can easily be followed in Figure 5.

Menu buttons are of a specific type; in our menus we support the following types:

(a) Push buttons (default): when activated, the button appears pressed; when the corresponding action has finished, the button pops back again.

(b) Toggle buttons: when activated the button will appear pushed if formerly popped up and vice versa.

(c) Radio buttons: when activated the button will appear pushed. Any other buttons belonging to the same group of buttons will pop back. The radio group is designated by referring to the name of one of its buttons. A new radio group can be created by letting the first button refer to itself.

The screen position (Figure 5 lines starting with C) is the only parameter required for a button description. It consists of a pair of integers denoting the *xy*-position relative to the screen (the top-left corner is (1,1)). The dimensions of the button (width and height) may be entered on the same line. The width defaults to the length of the button name or 14, whichever is larger. The default height is 1. In each menu tree there is only one information box description, and most of its contents are defined in the MDF file. The message box is the part of the screen which displays output from actions. The default label of this box ("input and messages") may be changed with the label specifier. The interpretation of the coordinates for these boxes is slightly different from the one in the button description. By default both boxes are centered horizontally whenever possible. If both *x*-coordinate and width are not specified (or have a value of 0), then the boxes occupy the full width of the display. The information box is placed by default at the top of the screen with a height of 2. The default height of the message box is 6. If the *y*-position of the message box is 0 (default), its placement is at the bottom of the screen.

With each button an action may be specified to be executed when the button is activated. By default the action will be sent to DCL, the command interpreter under VMS. However, provisions have been made to link another command interpreter to MENUXS. This alternative interpreter will be called when the /NOVMS qualifier is added to the action string specifier. The default behavior is that all output resulting from the action will be shown in the small message box, usually in the lower screen half. In the case of interactive (graphical) programs or programs which generate large amounts of output, this may be inconvenient. The /INTERACTIVE qualifier will change this behavior; the screen is cleared before the action, and afterward the menu will be restored.

The help specification is similar to the action specification. In addition to the /(NO)VMS and the /(NON)INTERACTIVE qualifiers, the qualifier /FILE changes the meaning of the action string into a file name. If help on a button with this qualifier is required (the PF2 key pressed), the menu is overlaid with a scrolling window showing the contents of the specified help file.

It is possible to supply an action string with additional parameters. If, for a certain button, a prompt string is given in the MDF file, the user will be prompted with this string for input. The effect is that the action string and the user input will be concatenated and passed to the command interpreter.

Design. The basic design criteria for our menu system were ease of maintenance for the network services provider and ease of use for the user of these services. With respect to the latter, the system has certainly been very successful to our users. For a very large majority of the interactive sessions on our system, currently around 40 000 per year, the menu system

```
2       RNA 2D Struct. ...
        H   write sys$output 'RNA Secondary Structure Prediction'
        C   5 13 20

!= BEGIN ===== RNA Secondary Structure Prediction ===========================

3       Fold (B)
        A/I Fold /Batch
        P   Options:
        H/I genmanual /page rna_secondary_structure fold
        C   5 7 20
3       StemLoop
        A/I StemLoop
        P   Options:
        H/I genmanual /page rna_secondary_structure stemloop
        C   5 15 20
3       Squiggles
        A/I Squiggles
        P   Options:
        H/I genmanual /page rna_secondary_structure squiggles
        C   30 7 20
3       Circles
        A/I Circles
        P   Options:
        H/I genmanual /page rna_secondary_structure circles
        C   30 9 20
3       Domes
        A/I Domes
        P   Options:
        H/I genmanual /page rna_secondary_structure domes
        C   30 11 20
3       Mountains
        A/I Mountains
        P   Options:
        H/I genmanual /page rna_secondary_structure mountains
        C   30 13 20
3       DotPlot
        A/I DotPlot
        P   Options:
        H/I genmanual /page comparison dotplot
        C   30 15 20
3       SubOptimal Fold ...
        H   write sys$output 'SubOptimal RNA Secondary Structure Prediction'
        C   57 7 20

!= BEGIN ===== MFold =====================================================

        4   MFold (B)
        A/I MFold /batch
!       H/I genmanual /page rna_secondary_structure mfold
        C   5 7 20
4       PlotFold
        A/I plotfold
!       P   Options:
!       H/I genmanual /page rna_secondary_structure plotfold
        C   5 9 20
4       EXIT
        H   write sys$output 'To leave this menu'
        C   57 15 20

!== END ====== MFold =====================================================

3       EXIT
        H   write sys$output 'To leave this menu'
        C   57 15 20
! ! !== END ====== RNA Secondary Structure Prediction ===========================
```

Figure 5. Contents of the MDF file (partially shown) for menu of Figure 4.

is used to navigate to the wanted (group of) tools, the chemical applications, and to start a specific application. Even though all tools are also directly accessible at the operating system command level, the large variety of the (network) services offered by the Center[8,9] makes the menu the preferred choice for access to both on-line sessions and to start-up batch applications. This applies equally to novice and experienced users of the system.

With respect to maintenance and development, a major part of the development consisted of the embedding within VMS (for a UNIX version see below). Many of the applications which currently run under MENUXS are

Table I. Formal Description of the Menu Definition File

```
Note on the syntax:
<x>    non-terminal; not part of the language
x | y  x or y
[x]    x is optional
{x}    1 or more occurences of x
X      literal,  part of the language (not case-sensitive)


mdf_entry:
<info_desc>  |
<msg_desc>  |
<button_desc>
info_desc:
INFO_BOX  [<x> [<y> [<w> [<h>]]]]
[ {<info_line>} ]
msg_desc:
MESSAGE_BOX        [<x> [<y> [<w> [<h>]]]]
[<label>]
button_desc:
<name>
<position>
[<action>]
[<prompt>]
[<help>]
[<type>]
info_line:
<blank>        <string containing non-blanks>
label:
LABEL          <string>
name:
<decimal digit> <string>
position:
COORDINATES        <x> <y> [<w>[<h>]]
action:
ACTION         [/INTERACTIVE | /NONINTERACTIVE]
               [/VMS | /NOVMS]     <string>
prompt:
PROMPT         <string>
help:
HELP           [[/INTERACTIVE | /NONINTERACTIVE]
               [/VMS | /NOVMS]] | [/FILE]      <string>
type:
TYPE           PUSH  |
TYPE           TOGGLE           |
TYPE           UNUSED           |
TYPE           RADIO            <string>
```

Table II. Tree Hierarchy of Menus

level	button name		
0	entry menu		
1		1st in menu	
1		2nd in menu	(has a submenu)
2			1st in submenu
2			2nd in submenu
2			exit from submenu
1		exit from menu	

accompanied by a rich environment with process logicals and symbols. The difficulty was to find a way to be able to enter and leave MENUXS while keeping this environment intact, i.e. the conservation of the user environment through successive menu calls. The solution we devised was to run the menu system as a subprocess and maintain a command procedure, which executes the VMS commands, running in the main process.

It is possible to run up to three instances of the menu system nested, i.e. menus may be invoked from within another menu. This is different from specifying submenus in an MDF file: the whole layout of nested menus may differ substantially, whereas the placement and contents of the information and message boxes are the same for submenus from the same MDF file. Note, however, that the actions specified in the MDF files of each instance of the menu will be executed in the main process, which ensures the proper adjustment of the symbols and logicals environment.

Future Development. To broaden the scope and the applicability of our current (VMS-based) menu system, the Fortran MENUXS program was ported to a UNIX environment. In this pilot project, a C version of the program was coded, maintaining the overall structure of the original program and offering exactly the same functionality, but replacing VMS-based routines by CURSES (public domain Unix) equivalents. This version of the menu system keeps the users in a Unix operating system environment while accessing the different services. Although operational in prototype our experience with this port, in particular the difficulty to conserve the user environment through successive menu calls, prompted us to redesign the structure of the MENUXS program, with even more emphasis on maintainability and hardware independence. Currently an object oriented design, building on our experience with the code described in this paper, is in progress. Through partition of the code we will separate the basic menu system and the actual user interface which will allow us to develop small front-end specific menu user interface objects for many PC's and X11 workstation and allow for a real system independent client/server implementation.

CONCLUSION

The more demanding the task, with many different hardware platforms, many different application programs, many different user terminal facilities, and increasing network dependency, the more difficult it is to construct the seams of a virtual computer. Our Center seeks to incorporate all significant and network accessible computational chemistry services, belonging to the disciplines it covers, into its user interface while keeping the user as much as possible in one operating system environment. But network reliability and program and database diversity still make the current service less seamless than one might want. Most computational chemistry programs are not designed to work together or to share data. One feature chemists might hope to see in the future is a system where network modules perform functions for the chemists' computer directly, instead of turning the chemists' computer into a terminal emulator. Meanwhile appropriate user interfacing can eliminate many of the inconveniences of the current array of programs. The ASCII-based menu system for a VAX/VMS environment described in this paper[10] has successfully been used at our Center and has proven to be of great help in making new services available to our users and has significantly eased the tool maintenance and integration problems faced by the Center's staff. Developments currently in progress aim at a fully system independent user interface for computational chemistry network services.

REFERENCES AND NOTES

(1) VAXcluster is a Digital Equipment Corp. trademark implying that several computers share system resources in the same environment without regard for the machine originally used by the user to access the system.

(2) VAX and VMS are trademarks of Digital Equipment Corp., Maynard, MA.

(3) Unix is a trademark of AT&T, Murray Hill, NJ. Ultrix is a Trademark of Digital Equipment Corp., Maynard, MA.

(4) Convex is a trademark of the Convex Corp., Dallas, TX.

(5) Alternatively other network protocols or modem connections may be used.

(6) Thiers, A. H. M.; Noordik, J. H.; Boerhout, J. *J. Chem. Inf. Comput Sci.* **1990**, *30*, 19–22.

(7) Licenses to access the CAN/SND system are necessary to establish such a connection.

(8) Noordik, J. H. In *CAOS/CAMM Services. Synthesis Planning and Molecular Modeling Tools. Chemical Structures*; Warr, W. A., Ed.; Springer-Verlag: Berlin, 1988; pp 367–370, ISBN 3-540-50143-6.

(9) Noordik, J. H.; In *CAOS/CAMM Services. An Integrated System of Computer Assisted Chemistry Tools. Software Entwicklung in der Chemie 3*; Gauglitz, G., Ed.; Springer-Verlag: Berlin, 1989; pp 449–465, ISBN 3-540-50673-X.

(10) MENUXS (source) code, example MDF files and VMS command procedures to implement the system are available from the authors on request.