A public access interface to the OSI Directory

Paul Barker

Department of Computer Science
University College London
P.Barker@cs.ucl.ac.uk

ABSTRACT

This paper describes a user interface to the OSI Directory. Although there are a considerable number of user interfaces already available, system administrators have complained that none of these interfaces is specifically intended for use as a "public access" interface. Such an interface must be very simple to use. This requirement leads to a set of key design goals: when there is a conflict of aims, simplicity must be favoured over functionality; ergonomic issues are of vital importance; the on-line help system must be simple but comprehensive.

The search strategy employed by the interface is also discussed in some detail. The way that the strings provided by the user are mapped onto sets of X.500 operations and matched with Directory entries is described.

The development of this interface has been funded by the PARADISE project, which in turn is funded by COSINE. PARADISE has a number of goals, including the coordination of directory service pilots. In addition, PARADISE is providing a number of central services, one of which is a public access interface to the Directory.

1. Introduction

A considerable range of OSI Directory [ISO88a] user interfaces have been produced over the last two years during the life of the directory pilot. It seems reasonable to ask the question: do we really need yet another directory user interface? This paper attempts to convince the reader that there is a requirement for a user interface designed specifically for use as a public access interface. It is suggested that such an interface should possess a certain set of characteristics not found in many of the existing interfaces. A low level of users' computer literacy must be assumed, as must total unfamiliarity with the Directory. No assumptions should be made about the sophistication of the user's terminal, or indeed about the user's ability to be able to make use of its facilities. The interface should be geared to answering common queries, such as finding someone's telephone number or email address, rather than satisfying arcane requests.

This paper describes an interface which is intended to fill this niche. The interface will henceforth be referred to as *de*, which stands for Directory Enquiries. First, the main goals of the interface's design are established and then discussed in detail. Particular emphasis is given to the key area of ergonomic issues, where the design is a combination of the author's prejudices, improvements suggested by an HCI expert, and response to the advice and suggestions given by a number of *guinea-pigs* during the course of testing.

Another area that is also considered in some detail is the mapping of the strings that the user types onto X.500 operations. An approach where a single query can lead to a number of successive searches of increasing "fuzziness" is described at some length.

The development of the interface described herein has been under the aegis of the PARADISE project [Goodman91a] funded by the EEC's COSINE program. This project has a commitment to provide a mode of access to the OSI Directory to users in the research community who do not otherwise have directory software available.

2. Do we need yet another interface?

At the time of writing, there are at least 15 directory user interfaces (to the Quipu system alone) known to the author, and possibly more than double that in existence. Is it really possible to justify yet another interface? Some have argued that the plethora of interfaces presents an unconvincing picture to those new to directory services, indicating that the service providers don't seem sure themselves of the best tools for accessing the directory. However, there are a number of arguments in favour of having a variety of interfaces available.

First, there is the perennial trade-off between simplicity and complexity. Interfaces which offer complex facilities and a large measure of control over directory operations will inevitably be more demanding to use than interfaces which offer less control.

Second, the Directory is intended to support a wide range of queries. Following the principle of locality, a large number of queries will be for information about the local environment, which may be the organisation where the user works, or even the department in which the user works. Given that this type of query will predominate, it seems logical to offer users, *inter alia*, an interface which is tailored for these local queries.

Third, the computing environments in which people work vary greatly. Some directory users will have work-stations with bit-mapped screens, and run a windowing system with graphical user interfaces. These users may have a Directory System Agent (DSA) on their local area network. Other users may have personal computers, offering a sophisticated range of packages, but having rather restricted access to networks. A large group of users still only use character-mode terminals, and will have access to network services by use of a PAD or modem connection.

It is worth briefly considering some of the interfaces that are currently available. These interfaces are all available with the Quipu software [Kille88a], which is distributed as part of the ISODE package [Rose90a]. Almost all current directory piloting is based on the Quipu software.

dish This interface offers access to all aspects of the Directory Access Protocol. The interface is immensely versatile and can, in the style of the MH mail user interface, be used as a set of individual programs. This allows dish to be used to build other user interfaces using UNIX shell scripts. The interface supports modification of the DIT as well as querying. This is not an interface for the novice user!

sd This interface is designed for character addressable terminals. The interface is essentially navigational, and a user will probably use the interface more successfully if they grasp the concept that the data is hierarchically structured. This interface has been used with some success as a public access interface, but is not simple enough for the ultra-naive user.

pod This interface [Findlay90a] makes use of the X window system to handle the display. The interface is also navigational. Pod supports modification of entries. The use of X restricts this interface to workstation users, and those who run X on other personal computers

ufn Ufn [Kille91a] stands for user friendly naming, and is a style of directory querying and query resolution, rather than an interface *per se.* Several interfaces have this style of querying built-in. This style of querying is strongly favoured by many of those well-acquainted with the Directory. It requires knowledge of an input syntax, although this is very simple. This style of querying may well prevail as more and more users become aware of the Directory.

fred This interface is intended to be similar in style to the *whois* program, familiar to users of the Internet. This is an advantage and a disadvantage, depending on whether one has used the whois program before! Fred supports user-friendly naming.

osiw...Osiwotsits are a collection of very simple programs which are intended primarily for use for lookups within an organisation. The amount of typing required is minimal for simple, local queries, but the interface is more cumbersome for remote queries.

While some of the above interfaces could be used by novice users, none of them are specifically geared to the less sophisticated user of the Directory. This user may not be cognisant of the hierarchical nature of the Directory Information Tree (DIT); (s)he may be accessing the Directory remotely and not have access to on-line manual pages or other help with using the Directory; the user will probably be using a character mode terminal.

De has been designed principally as a public access user interface, to be used by those otherwise lacking a local access point to the directory. The interface could be configured for use in other circumstances, as it may be that some users prefer the style of querying and copious on-line help. At the time of writing it is too early too report on the success authoritatively as to the success of the design, and its suitability to the range of operational environments, although the initial feedback has been encouraging.

It is freely acknowledged that de has its limitations: it cannot, for example, be used for entry modification. However, it is hoped that some of de’s strength is derived directly from its focussed approach. The intention, nevertheless, is that de will suffice for the vast majority of directory enquiries about people and the organisations they work for. If it transpires that de fails in this regard, it will be modified!

The author also regards de as being an "interface for the moment", in that it is aimed at solving today’s problems. Following from this, de may well have a limited life. There are least three reasons for this. First, de is seen in part as an interface suitable for introducing new users to the OSI Directory. As users become familiar with the services provided and the potential of the Directory, it seems likely that some users will migrate to more powerful user interfaces which allow more control over access to the Directory. Second, as more and more computer users have bit-mapped screens on their desks, the arguments for avoiding designs which rely on these capabilities will diminish. Third, de makes some assumptions about the hierarchy of the DIT. As the Directory grows, the DIT will almost certainly deepen, although to what extent it is not possible to predict at this moment.

To conclude this justification for another interface, it is worth setting out the main design goals of de, and attributing emphasis between goals as and where they conflict. The goals are as follows:

- *Ease of use.* This is best summarised by what amounts to a definition: the interface must be sufficiently easy to use for the first time user, that they are not deterred from trying to use the Directory again. This is the principal goal.
- *Useful for most queries.* It is assumed here that the predominant query will be for communications information - telephone and facsimile numbers, electronic and paper mail addresses - for people working in organisations. The interface will offer some tailoring, but will be fundamentally geared towards queries about people.
- *Terminal independence.* The interface should work with full functionality from the lowest common denominator (virtual) terminal. It should also work with barely reduced functionality even when the user is unable to provide a terminal type, or the user’s terminal type is not recognised.
- *Good performance.* The interface must be able to deliver results reasonably quickly. Design decisions which prove to compromise responsiveness will be re-considered.

3. Design characteristics of the interface

This section considers the design of the interface. There is an underlying theme of ergonomics running through most of this section: the interface has to be simple to use. There is also a substantial discussion on de’s *query engine,* or how de makes use of X.500 services to provide the results the user requires.

3.1. Screen mode

A fundamental goal of this design is that de should be runnable from any type of terminal: it should even be runnable from a teletype! This clearly precludes the use of windowing interfaces. Full-screen designs are also considered inappropriate - experience has shown that a considerable number of users of an earlier public access service run at University College London (UCL) either found that their terminal type was not supported, or that the terminal emulation did not work correctly. One service provider told the author that many users are unfamiliar with the notion of a terminal type and, if asked to provide one, often type the name of the manufacturer of their monitor or keyboard. De tries to make it clear to such users that the service is perfectly usable even if this terminal information cannot be supplied. The more sophisticated network user, on the other hand, is often aware that they can use a variety of flavours of terminal emulation, but cannot find a type that the system they are connected to knows about. For such users, de provides the facility to list the supported types. Since so many terminal types are often supported (almost 400 under SUNOS, for example), a few well known ones are brought to the head of the list - many of the most familiar ones, such as vt100, come near the end of an alphabetically sorted list!

Given the requirement that de should function adequately without terminal type information, the design is for a scrolling interface, for both input and output. The basic style of querying is that the user is prompted for input by being asked a short series of questions. The prompts are verbose, so that it should be possible for the first timer user to query the Directory successfully. Results of queries are presented to the user a screen at a time. In the absence of any information about a user's terminal type, a terminal size of 24 rows of 80 columns is assumed. A special purpose pager has been written which supports a minimal set of commands. The pager's prompt includes information about how to get the next screen of information and how to exit the pager.

Apart from knowledge of screen dimensions, the only use which is currently made of the terminal type information is to allow the use of inverse video for prompts. It is noted that this feature has its supporters and its detractors: accordingly the use of inverse video is tailorable.

While the initial version of the interface is purely a scrolling, line-mode interface, it would be possible to produce a full-screen, character-mode interface or even a command line interface with many of de's characteristics. This is not regarded as a priority at the time of writing, but if a significant number of people request either of these styles, versions in these styles could be produced moderately quickly.

3.2. Specifying a query.

A user will specify a query by answering a short series of questions in response to some rather verbose prompts. As stated earlier the focus of the interface is on finding information about people. The user is asked to supply the following details:

1. The name of the person sought
2 Department name
3. Organisation name
4. Country name

The questions are asked in paper envelope order as this seems most natural to the author. Considerable use is made of default values: the default value is displayed as part of the prompt and accepted by entering <CR> at the prompt. Defaults help in two ways.

First, many queries made to an organisation's public access interface will be about people in the same organisation. It will thus be prudent for the system administrator to configure local values as defaults for the organisation and country names.

Second, while it is imagined that typical use of the directory will be to look up information about a single person or organisation, some support for extended querying is provided by retaining defaults from one query to the next. It should be noted that this feature is not always helpful. It is of benefit if a user wishes to make repeated queries within various departments of a single organisation. However, if people wish to query somewhere entirely different, the defaults can get in the way. This happens in two ways. It has been observed that users tend to get "default-happy", and press <CR> to accept defaults even when they don't wish to retain existing values! Another problem is that if a subsequent query requires a null entry for a particular field (we will see cases like this shortly), where a previous query had a value entered, syntax is required to allow the elimination of the previous value. de uses the '-' character to achieve this, and informs the user within the prompt of how to clear the field. However it is arguable that <CR> may be a more natural way of clearing a field. One possible solution which has been considered is to give the user the ability to destroy all the defaults with a simple instruction, but this requires more knowledge of syntax. Feedback from users is eagerly awaited on this issue of defaulting!

A couple of examples should demonstrate the style of the interface. The reader should note the different use of defaults in the two examples. The following input should suffice for a query about the author, if using de as configured at the author's organisation:

```
Person’s name, q to quit, * to list people, ? for help
:- barker
Dept name, * to list depts, <CR> to search all depts, ? for help
:- cs
Organisation name, <CR> to search ‘ucl’, * to list orgs, ? for help
:- <CR>
Country name, <CR> to search ‘gb’, * to list countries, ? for help
:- <CR>
```

The principal author of the Quipu software may be found by:

```
Person’s name, q to quit, * to list people, ? for help
:- robbins
Dept name, * to list depts, <CR> to search all depts, ? for help
:- <CR>
Organisation name, <CR> to search ‘ucl’, * to list orgs, ? for help
:- xtel
Country name, <CR> to search ‘gb’, * to list countries, ? for help
:- <CR>
```

The prompting for input is seen as central to the design. The prompts are verbose to the point that users should be in little doubt about what sort of information they should be entering: this comment is subject to the proviso that the prompt is actually readable. The prompt includes as much guidance as possible about how to use the interface. The prompts may appeared somewhat cluttered for the experienced user but, as de is not aimed at such users, this is a criticism which the author is prepared to live with. Apart from information on how to search for or list information in the Directory, two other key pieces of information are provided: how to quit the interface, and how to get help. De’s help system is discussed in more detail later: for the moment it should be noted that the on-line help system should be sufficient to obviate manual pages and make the interface readily usable by first time queriers.

It is also possible to search for information about organisations or their departments with the same set of questions. This is achieved by the (what it is hoped is intuitive) device of the user omitting to provide input for the questions for which (s)he is not expecting an answer. An example should show that this is not as complicated as it sounds! The following query will return information about the author’s department at UCL.

```
Person’s name, q to quit, * to list people, ? for help
:- <CR>
Dept name, * to list depts, ? for help
:- computer science
Organisation name, <CR> to search ‘ucl’, * to list orgs, ? for help
:- <CR>
Country name, <CR> to search ‘gb’, * to list countries, ? for help
:- <CR>
```

Note above that the first <CR> is interpreted as null input, there being no default offered, whereas the latter two <CR>s accept the defaults indicated.

3.3. Search strategy

This section considers the search strategy adopted by de to resolve a query. We will see that even simple queries may often map onto a complex set of X.500 search operations. This should not be surprising. The user will typically not provide strings which are exactly equivalent to the names in the Directory, but will often provide enough information that a unique match may be found given an intelligent search strategy. This strategy is now described.

A number of cases have to be considered: when there is a single match; when there are multiple matches; when there are no matches. For the moment we need not concern ourselves with what constitutes a match:

this is discussed in the next section.

When searching for a person, the search algorithm is broadly as below:

```
for (list of countries matching $co)
    for (list of organisations matching $org)
        for (list of departments matching $ou)
            search $name
```

Given the hierarchical nature of the DIT, the searching is of necessity for countries first, then for organisations within countries, and so on. If there is a single country which matches the entered country input, a single organisation matching the country input, etc, then the behaviour of the interface requires no explanation.

If there is more than a single match for any field other than the person's name, the user is presented with a list of matches and is invited to select one from the list. The matches are numbered so the user can select an entry with very little effort. The search then continues below the selected entry. The procedure is repeated if more multiple matches are discovered, except that the user is not forced to select a single person's entry and may be showed the details for a number of entries (up to a configurable limit). There are two key features of this approach. First, a query is progressed until no match can be found for a particular category of input. The user is then asked to re-enter a name only where no match was found: on input of a new name, the query continues. Second, the user is shown the progress of the search as the query is resolved. This feedback both provides some assurance that the system is working, and also provides a user with the opportunity to abandon an operation, if it is clear that an unanticipated and unintended subtree is being searched. The control that the user can exercise to interrupt operations is discussed later.

If no matches are found for a particular type of input, the user is informed that no matches can be found, and the user is prompted for further input. The user's failure to find an entry may mean one of three things. First, the user has simply mistyped the name and retyping will result in the appropriate entry being found. Second, the user may be specifying a name for an entry which exists in the DIT, but which does not have a name which can be matched with the name typed by the user. Third, the DIT is only sparsely populated at present, and it is quite likely that users will try to search for entries which are not yet in the Directory. For the last two reasons, it has been decided to give the user a "listing" option, to allow him/her to browse through some of the entries in the Directory. This option is invoked by typing a '*' character: it is hoped this choice is intuitive to most users, as '*' is widely used in command line shells to mean "all". It should be noted that a general listing capability will only be practicable while the DIT remains quite small. In particular, the ability to list all organisations in each country is not likely to be feasible, or even possible, for much longer. However it is felt that some sort of "listing" function will continue to be useful to allow users to examine the nature of the information in the DIT: this in itself may help users to specify their queries successfully. A few points should be noted:

- There is a fundamental assumption about the shape of the DIT. This assumption is in keeping with what currently prevails in the pilot. If practice changes, the design will need some enhancement. In particular, it is likely that an extra question asking for locality information will have to be asked before long.
- All references to *listing* above are in fact implemented by *search* operations for entries of the appropriate object class.
- Searching for a country will employ a single level search at the root of the DIT. Localities such as Europe and North America may also be searched for under the root. Searching for organisations will be by a single level search under country or locality entries. Searching for departments is initially by single level search underneath organisation entries. If single level searches fail, subtree searches are then attempted †. Searching for people uses subtree searches.

† The ability to specify an *n-level* search would help greatly here, and its omission from the X.500 standard is regrettable.

Given the design, there is no *prima facie* reason why the user should be restricted to searching within a single department within a single organisation within a single country. Indeed an early version of the interface offered this facility. However it has been excluded for two reasons. The first reason is technical. A search of this generality may well consume a vast amount of resources, and some sorts of limits will inevitably have to be imposed.

The second reason is political. The anxiety that many organisations and users have about making their data available will be heightened by interfaces which are able to trawl the global Directory. Since this interface is offered as the public face of the Directory, it has been decided initially at least to limit the scope of searching.

If users clamour for more powerful searching facilities, these can readily be provided by de.

3.4. Name matching

The X.500 search operation allows complex searches to be specified by combining filters using exact, substring, approximate and other types of matching with Boolean *ands, ors* and *nots.* The following is typical of the sort of filter that has been employed by user interfaces to find entries in the Directory. The filter is expressed in the syntax used by the Quipu system's dish interface.

> objectclass=person & (surname=$name | cn=*$name* | cn~=$name | userid=$name)

This can be expressed in english as "find all the people where one of the following conditions is met: the named entered exactly matches the surname, is a substring of a common name, approximately equals a common name, or exactly equals the login name attribute of a Directory entry".

However, the use of very complicated filters is not always helpful. In the above example search filter, exact matches will be returned mixed in with any substring and approximate matches. The failure to give preference to "good" matches over "looser" matches is counter-intuitive to most users. One user, mystified by the behaviour of an early interface, asked:

> *when searching for muller, why do I get 16 millers first?*

and

> *typing "search tuck" produced 12 names in list before his. Not very precise. None of them were tucks!*

There are several aspects to name matching which must be considered to make it more intuitive to the user. People will not usually type in strings of characters which exactly match names in the Directory. The interface must offer some sort of name matching support, and this must be available as a default. Most users will not be computer experts, deeply versed in the arcana of regular expressions, although many will know something of the use of wild-cards.

De offers both implicit and explicit wild-card support, although it is anticipated that explicit wild-carding will only be used by more expert users, and then only very occasionally.

Implicit wild-carding is provided using the following method. Rather than concocting complex search filters including various degrees of matching, de tries a sequence of searches with simpler filters. The initial searches specify exact, or very close, matching on the string entered, while subsequent searches specify increasingly looser matching. The looser matches are only tried if the closer matches fail to deliver any results. The proposed approach can entail up to four searches, although well specified searches will usually require less. This method of matching seems to provide the results expected, and reasonably quickly. An initial fear that this style of searching might be too slow has not been borne out by experience: if an initial search fails, subsequent searches of a DSA tend to be fairly fast as the DSA process and its database are paged into primary memory - this is particularly a feature of Quipu DSAs but other DSAs will probably share this characteristic to some extent.

One potential problem with this approach should be noted, although again experience has not as yet suggested that it is a practical problem. As a search is curtailed as soon as any matches are found, an unwanted exact match inhibits the search for less good matches, which might result in the required entry being found. This design decision will be reviewed in time, although it is difficult to see how to provide the user with more control over the searching in a comprehensible fashion.

An example of the search filters used by de should clarify the strategy used. First, let us consider the relatively simple filters used when searching for an organisation's entry. A sequence of up to four filters are tried in turn until some results are found.

objectclass=organization & organizationName=$org

objectclass=organization & organizationName=$org*

objectclass=organization & organizationName=*$org*

objectclass=organization & organizationName~=$org

An exact match on the organisation's name is preferred over a leading substring match, is preferred over an any substring match, is preferred over an approximate match.

A more complicated example sequence of filters is used by de to locate entries for people. The format of people's common names in the Directory varies considerably: some have one or more forenames and surname; others as little as a single initial and surname. To cope with this de examines the format of the name entered. If the name contains one or more spaces, the concept of exact match is extended. De deduces that the first letter in the string entered is the first initial and that the last part of the name is a surname. Using this technique "p barker" is considered to *exactly* match against an entry with a name of "paul barker", and an entry of "paul barker" is considered to exactly match "p barker" or "paul fred barker". This technique could be built upon substantially if required. It has been suggested that the current approach is not as helpful with chinese names as with western names. Enhancements to the matching algorithms will be incorporated providing that there is no significant impact on search speed in the general case.

Explicit wild-card support is also offered as this allows the slightly more sophisticated user more control over the matching that de attempts. If explicit wild-carding is specified, only a single search is attempted in accordance with the given filter - there is no recourse to approximate matching. This explicit wild-carding is, of necessity, relatively simple because of difficulties mapping such requests onto search filters. It is not possible to map regular expressions onto the filters provided by X.500. The following forms are supported: *xxx*, xxx*, *xxx and xx*xx.

3.5. Presentation of results

A key decision on presenting results is to display results to the user as the query progresses. Queries are resolved by first attempting to find a country with a name that matches the country name entered, then a matching organisation, etc. Queries will usually be typed as a set of strings which seem logical and convenient to the user, rather than as a set of relative distinguished names (RDNs). However, the user is shown the matched relative distinguished name. The following example shows what a user might reasonably type to find the author's entry, and the layout of the results.

```
Person’s name, q to quit, * to list people, ? for help
:- barker
Dept name, * to list depts, <CR> to search all depts, ? for help
:- cs
Organisation name, * to list orgs, ? for help
:- ucl
Country name, * to list countries, ? for help
:- uk
United Kingdom
  University College London
    Computer Science
      Adrian Barker
        electronic mail      A.Barker@uk.ac.ucl.cs
      Paul Barker
        telephoneNumber      071-380-7366
        electronic mail      P.Barker@uk.ac.ucl.cs
        favouriteDrink       16 year old lagavulin
                             guinness
        roomNumber           G21
```

Some points are worth noting. The name of the country is not the RDN of the country entry. Country entry RDN’s are the somewhat cryptic country codes specified in ISO 3166 [ISO81a]. The directory pilot provides an attribute "friendlyCountryName" which can be used for matching against user supplied names, but cannot be used meaningfully for display, as these names are often a set of multi-lingual alternatives. There is currently no way of selecting a name from this set of "friendly" names which can be guaranteed to be in the user’s native tongue: some sort of language tagging is required. De’s solution is to allow for a tailorable set of mappings between the 2-letter codes, and longer names meaningful for local users.

As stated above, names are shown to the user as the query is resolved. As soon as a match for "uk" is found, the string "United Kingdom" is displayed, and so on. It is hoped that the indentation makes the results easy to interpret. It is important to realise that the names presented do not necessarily constitute all the RDN parts of the entry sought, but merely the matches on the entered strings.

The attributes shown to the user are configurable, according to the object class of the entry. There is no option to return all attributes. This is deliberate as it prevents the returning of audio and photo attributes, which are inappropriate for this type of interface and have a severe impact on performance given limited network bandwidth. The attribute keywords are configurable, and so can be made comprehensible for humans, and even language independent. Some attributes are handled specially: for example, electronic mail addresses can be displayed according to local conventions; telephone numbers can be displayed according to a local format.

If a search results in a large number of entries being found, a restricted subset of attributes is shown in the following format:

```
United Kingdom
  University College London
    Computer Science

Got the following approximate matches.  Please select one from the list
by typing the number corresponding to the entry you want.

         1 Geraint Jones                              G.Jones@uk.ac.ucl.cs
         2 Hefin Jones                                H.Jones@uk.ac.ucl.cs
         3 Mark Jones          071-387-7050 x3673     M.Jones@uk.ac.ucl.cs
```

3.6. The help system

Since it is anticipated that users of de will often not have recourse to manual pages, either on-line or on paper, the help system is critical. Approximately 15 help screens have been provided at the time of writing, and can be enhanced and added to freely with no need to recompile the system. A user is initially shown a brief "welcome" screen which informs the user the questions which will be asked, how to get more help, and how to get out of the interface.

Requests for help are made by typing an initial '?' character in response to any of the prompts, optionally followed by a keyword indicating a help topic. If a single '?' is typed, the help is context sensitive: e.g. if the '?' was typed at the prompt for a person's name, help on entering a person's name is given. "??" gives "help about help", and lists all the help screens available. "?wildcards", or even "?wi", gives help on the use of wildcards in searches. Every help screen includes information on how to get further help.

Since learning by example is often an efficient way of getting started, some example queries are given and described in detail.

A convention is used throughout the help screens whereby any word which has an associated help screen appears in CAPITALS.

3.7. Resetting and escaping the interface

For a user to feel confident with an interface, it is essential that a number of important criteria are addressed. The following points were design goals for de.

- A user must be able to escape from de easily, at whatever point the user is currently at in the interface.
- It shouldn't be *too* easy to accidentally escape from the interface.
- A user shouldn't be faced with a plethora of questions of the sort *"Are you sure that you want to ...?"*
- A user must be able to correct erroneous input.
- A user must be able to abandon a query which has been initiated, but which has not yet returned results.

The above goals are achieved by a simple two-phase escape mechanism.

> *Phase one:* Control C resets the interface such that it redisplays the first prompt and waits for input for a person's name. If a search is in progress, the search is abandoned. If the user is part way through specifying a query, that input is abandoned and the prompt for a person's name is displayed.
>
> *Phase two:* A second interrupt character typed when awaiting the input of a person's name will cause the interface to exit. 'q' typed at this point will also cause de to exit.

Note that if the interface is awaiting the input of a person's name, then only a single interrupt character is required to cause the interface to exit.

3.8. Miscellany

This section discusses a number of other design issues which are considered important.

Experience has shown that the initial connection to the Directory can be rather slow relative to the time taken to perform actual operations. The paging in of processes (OSI processes can be quite large) and setting up of connections at all layers of the protocol stack can be sluggish. To circumvent this as much as possible, de binds asynchronously to the Directory, such that the binding is interleaved with the entering of the initial query. The approach seems helpful, although it seems that there is no substitute for keeping DUA and DSA processes paged into primary memory to obtain good performance.

However, the rest of de's operations are performed synchronously. There are two reasons for this. First, the synchronous model is easier to code! Second, it is not clear to the author what benefits are derived from asynchronous behaviour in most cases. If an exact match search, a substring search and an approximate match search are sent to a DSA "in parallel", the DSA has more scheduling to do, and may do much unnecessary work if a good match can be found. A measure of asynchrony may be provided by the DSA anyway as queries are passed on to other DSAs. It would certainly be useful and interesting though to thoroughly evaluate the possibilities in this area.

De does not set X.500 time or size limits. Past experience has shown that users are often frustrated by the intervention of limits. Administrative size limits imposed by DSA managers cannot, of course, be circumvented. If an administrative size limit is reached, the user is informed that there is more data than they have been shown, and told that this is a policy decision by a data administrator rather than a technical limitation. "List" operations will often be constrained in this way: the user is then invited to try and guess the name of the entry they want.

The absence of a time limit means that an operation may "hang" for some time, if a remote part of the network is unavailable. De allows the configuration of alarm times which trigger the display of a message informing the user that an operation is taking longer than expected. The user is told how to abandon the operation if they do not want to wait any longer. The delay before this warning message is shown to the user is configurable for local and remote queries.

4. Future work

While it is tempting to add more and more features to de, the principal goal of simplicity must remain. However, it will probably be possible to add in some additional features without complicating the model. The intention is that future work will primarily be guided by user feedback. However, some areas which are already under active consideration include:

- Incorporation of user-friendly naming - de could recognise the syntax and resolve one-line queries using the ufn algorithms.
- Display more information on the progress of a query. This is desirable, but it is difficult to achieve this without affecting the display of the results.
- Sets of results are presented to the user in the order they are received from the DSA. Quipu DSAs return results lexically ordered, although results are inherently sets of data and so can sequencing can be assumed. A more sophisticated approach than that used by de is probably desirable: for example, names of people would be best presented in surname order.
- Approximate matching can sometimes deliver results which mystify users. A possible solution is to indicate to the user if the results are derived from "fuzzy" matching.
- De doesn't search for localities beneath country entries in the DIT. Currently this is only a problem with a small amount of US data, but this will not remain the case for long. However, at the moment it is not clear what sort of objects will be localities in the DIT. The situation is under review!

5. Software availability

The software is available as part of the PARADISE package. Information on this package can be obtained from:

helpdesk@paradise.ulcc.ac.uk

De is also available as part of the ISODE software.

6. Acknowledgements

The author would like to thank Angela Sasse of the Computer Science department at University College London for her assistance with HCI aspects. In addition, Angela's early testing of the interface led to substantial and beneficial modifications of the design. She also made a valuable contribution to the structure of the help system.

Thanks are also due to Caroline Leary of the University of Sussex whose comments on de have led to changes whereby de is now more usable by ordinary mortals.

References

Findlay90a.
Andrew Findlay, Damanjit Mahl, and Stefan Nahajski, *Designing an X.500 User Interface: One Year In*, 1990. UKUUG, Cambridge

Goodman91a.
D. Goodman, *PARADISE International Report*, University College London, May 1991.

ISO81a.
ISO, ''Codes for the representation of names of countries,'' ISO 3166, 1981.

ISO88a.
ISO, *The Directory - Overview of Concepts, Models, and Service*, December 1988. International Standard 9594-1

Kille88a.
S.E. Kille, ''The QUIPU Directory Service,'' *IFIP WG 6.5 Conference on Message Handling Systems and Distributed Applications*, pp. 173-186, North Holland, October 1988.

Kille91a.
S.E. Kille, *Using the OSI Directory to achieve User Friendly Naming*, University College London, March 1991.

Rose90a.
M.T. Rose, *The ISO Development Environment:User's Manual (version 6.0)*, Jan 1990.