Designing an X.500 User Interface: The Early Stages

Andrew Findlay   Damanjit Mahl

Brunel University
Andrew.Findlay@brunel.ac.uk
Damanjit.Mahl@brunel.ac.uk

ABSTRACT

The X.500/ISO 9594 Directory is briefly described, and the early stages of the design of a user interface are detailed. Examples are included that give an idea of the appearance of the proposed interface under the X Window System.

1. Introduction

The idea of X.500 is ... *to facilitate the interconnection of information processing systems to provide directory services. The set of all such systems, together with the directory information which they hold, can be viewed as an integrated whole, called the* Directory. The information held by the Directory, collectively known as the Directory Information Base (DIB), is typically used to facilitate communication between, with, or about objects such as application entities, people, terminals, and distribution lists.[1]

The X.500 Directory can be thought of as an ‘electronic phonebook’. Like a phonebook, a geographic structure is imposed on the data; no phonebook lists all people in the world in a single alphabetic sequence. It is necessary to have some idea of where a person might be before starting to search for them.

The X.500 Directory is similar; information is held in a tree structure, called the Directory Information Tree (DIT). Information is distributed across a large number of co-operating Directory System Agents (DSAs), each holding data concerned with a relatively small area. A consequence of this is that any searches covering a wide geographic area tend to be slow and expensive. User interfaces must therefore encourage the user to narrow the field of search as rapidly as possible.

This paper describes the early stages of the design of a user interface for the X.500 Directory.

2. Background

At present few Directory User Agents (DUAs) exist for use with the X.500 directory. Those that are currently available with ISODE have a number of shortcomings:

- User-unfriendly in their assumption that the user has knowledge of the structure and terminology of the directory.
- None make use of the features available under windowing environments.
- Inflexible and provide no more functionality than the underlying directory service.

The current research aims to produce a high level design for an X.500 DUA that will fulfil these shortcomings. Implementations for X[2] and Microsoft windows will be built, subject to availability of supporting software.

The project is currently part way through the design phase. Completion of the design is expected around the end of January or the beginning of February. Bearing this in mind the purpose of this paper is three-fold:

(i) To introduce people to the services provided by the directory at the user level.

(ii) To give an idea of the kind of interface proposed.

(iii) To gain a response to the ideas put forward.

3. Directory Issues

Many directory issues have not yet been fully covered or clarified in the X.500 standard. This presents problems in designing and specifying the functionality of the DUA. Some of these issues will be taken care of by on-going improvements in the directory, but a few will have to be covered by the DUA itself.

3.1. The Organization of Information in the Directory

The structure and organization of the DIT is still an area of research. This prevents many assumptions being made about the relationships and interactions between various pieces of information. The problem described in the next section on attribute inheritance is one result of this. It is not just the organization of the directory that matters, but also the amount of data that is to be stored (although the two issues are related). For example, will two or more layers of organizational units be used by very large organizations? Will this make a subtree search at organizational level seem unreasonable? Questions like this will come to the fore when the Directory Information Base (DIB) begins to take in a lot more data.

3.2. Attribute Inheritance

Information is held at each level in the DIT. This avoids duplication of items like an organization's address in each employee's entry. Attributes in an entry must alway be inherited by it's descendants, *unless* any of the attributes contained in that entry are redefined at some point in the subtree. For example, if the address to contact a member of staff in some particular University is not to be found in his/her entry, then the address attribute contained in the entry's closest ancestor containing such an attribute will be displayed rather than forcing users to find this information for themselves. Attribute types to which inheritance can be applied are those that relate to physical contact methods, i.e. telephone numbers, fax numbers, post office box etc. There are problems with the notion of attribute inheritance. One such problem is highlighted in the diagram below.

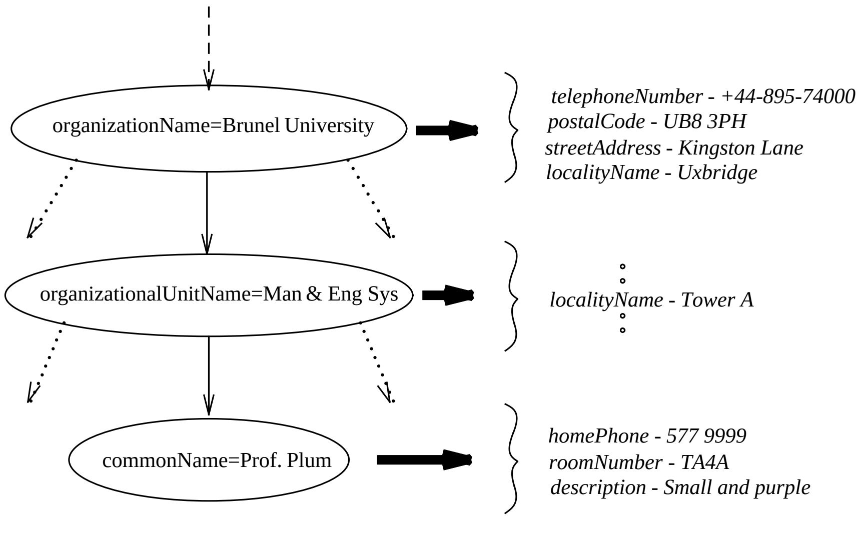

organizationName=Brunel University
telephoneNumber - +44-895-74000
postalCode - UB8 3PH
streetAddress - Kingston Lane
localityName - Uxbridge
organizationalUnitName=Man & Eng Sys
localityName - Tower A
commonName=Prof. Plum
homePhone - 577 9999
roomNumber - TA4A
description - Small and purple

Fig. 1 Inheritance of Attributes

If inheritance is assumed then the entry "commonName = Prof. Plum" would inherit the attributes:

- localityName - Tower A
- streetAddress - Kingston Lane
- telephoneNumber - +44-895-74000
- postalCode - UB8 3PH

Obviously ignoring the attribute "localityName - Uxbridge" because of the presence of "localityName - Tower A" is incorrect. This may show an incorrect use of "localityName," but until the contents of attributes are clearly specified such "misuses" are always likely to occur. Maybe the names given to attribute types need to be more specific. In this case it may be better if there was a separate attribute called "buildingName," in order to avoid confusion.

Looking at this example from the point of view of the DUA, one possible solution might be to inherit all such attributes up to, say, organizational level and make this fact explicit when displaying the attributes of an entry. Thus any conflicts can be left to the user for resolution.

3.3. Load Imposed on DSAs

The current version of the Quipu DSA (ver. 5.0) has a number of problems. Firstly, it is still quite slow for heavy searches, even with only one user. Secondly, it is unreliable, especially when a large search is requested. Although this situation will improve with the release of Quipu 6.0, it is unlikely that the improvement will be so great that a DSA will be able to cope with more than a few users at any one moment. Therefore the number of requests made to DSAs, in the form of direct requests, chaining or request referrals, must be kept as low as possible.

3.4. Charging for Use of DSAs

Commercial (i.e. British Telecom) DSAs are likely to charge. As a result, requests to such DSAs must be minimised.

4. Interface Issues

- The user interface must have a ‘look and feel’ that is as general as possible. The same interface should be recognizable from one environment to the next.
- The user interface must have a very flexible command structure. As well as having a general ‘look and feel,’ the interface’s command structures must be closely aligned to those of the WIMP environment being used.
- The user interface must be very simple to use. The aim is to shift as much work as possible from the user to the DUA. Users (especially those targeted by this project) should not have to formulate their own strategies in order to locate DIT entries on the basis of limited information.

4.1. The Basic Layout

The interface, as envisaged, is likely to consist of a main window and various ancilliary windows which are only on-screen when required. The main window will comprise of:

- A command sub-window, which allows access to the directory.
- A text window that will display the information returned by reading the attributes of an entry, error messages, warnings and reports.
- A text window which will display the "current position" held in the directory.

Windows will be created for:

- Display of lists returned by the directory.
- Compilation of mailing lists using output from the directory.
- User modification of parameters pertaining to directory access, e.g. timeouts, list size limits etc.
- User modification of the interface layout, size, visual mode and font type (see later sections for more detailed explanation).

Additional sub-windows may also be included.

4.2. Function Hiding

In order to keep the interface from becoming too complicated, only the more commonly used functions will be available in the command window. Functions that require knowledge of the DIT will be available in an activated sub-window or by explicitly telling the DUA to include these in the command window. This will also prevent the main window from becoming cluttered.

4.3. The Defaulting System

Our intention is to employ a rule based defaulting system that makes as many (reasonable) assumptions about a user's intentions as possible. For example if the position "countryName=<something>" is held in the DIT, then it is probable that searches performed on the children of this entry will be searches for an organization, thus the type parameter for such a search would default to "organizationName" upon moving to any entry whose name is of type "countryName".

Defaults for the scope parameter to the Search function can be obtained in a similar fashion. It is only reasonable to perform a subtree search when at lower levels of the DIT, thus one level searches will be the default unless the base object is of type organizationalUnitName or lower.

A table of DIT positions against defaults for search type and scope is shown below.

Default Search Parameters

Base Object Type	Default Type	Default Scope
root	countryName	One level
countryName	organizationName	One level
organizationName	organizationalUnitName	Subtree
organizationalUnitName	commonName	Subtree

Where a search guide attribute[3] is found in the directory, the information provided could be used to modify the defaults.

4.4. List versus Search

The List function returns all entries subordinate to an entry. It is, in fact, an instance of the Search function, i.e. search for all entries subordinate to the base object. It is reasonable to include List as a distinct function only when entries have, for the most part, a hundred or fewer subordinates. If entries are likely to have a greater number of subordinates, then use of the List function becomes impractical. Thus List will probably not be included as a distinct callable function.

4.5. Expanding on the Search Function

The Search facilities provided by X.500 are quite comprehensive and flexible. They provide:

- The ability to search on sets of attributes combined to give searches of the form: *match set A or set B but not set C.*
- A number of matching algorithms including: exact, sound alike (or approximate), less than or equal to, greater than or equal to and substring.
- The ability to limit the search to one entry, the immediate subordinates of an entry or the subtree below an entry.

This is fine for instances when the requestor has some information regarding an entry's attributes and *knows* that that entry will be below one specific entry, but if little is known about the location of an entry in the DIT, then the search facilities provided may well be insufficient. It is proposed that an additional search facility be provided that would perform a strategic search on the basis of limited information regarding an object's whereabouts within the DIT. This would take the same parameters as the normal Search function and in addition a list of attributes in order of decreasing scope. For example when looking for a person "commonName=<someone>," who is known to be in a computer science department in some university or college in London this list might be "localityName=London, organizationalUnitName=computer science." In the case of the search specified above and assuming that the base object has been set to "countryName=GB," the search function would take the following actions:

(i) Search for all objects immediately below the entry "countryName=GB" that contain the attribute "localityName=London."
(ii) Search below all objects returned by (i) for objects containing "organizationalUnitName=computer science."
(iii) Search below all objects returned by (ii) for objects matching "commonName=<someone>."

This description begs a number of questions. What form will the searches at each step take? What happens if steps like (i) or (ii) fail? The answer to both of these questions is that the search will probably be a one level search using approximate matching, then if this fails the user is warned and asked if he/she would like to continue the search. If yes then a one level search is performed below each child of the object previously searched under.

This kind of search has the potential for being highly time consuming, so the algorithm will take a kind of stop-go approach. If a step fails (either due to not being able to find anything or timing out) then the user will be informed and asked whether or not the search should continue.

4.6. Configurability

Facilities allowing configuration of certain aspects of the application (described in the following sub-sections) will be included in the final interface. Users will be able to save configurations made in this way to file.

The Visual Interface

Modifications to the following aspects of the visual interface will be allowed:

- The visual mode, either graphic or text oriented. This applies to user preference for an environment that employs graphic icons as opposed to text labels as the main visual source.
- The layout and sizes of windows and subwindows.
- Font type.

Other tailorable aspects may include:

- Choice of language.
- Enable or disable display of photographs.

Command Structure

The interface will permit control in a number of ways (this is dependent on what is allowed by window toolkits on the various systems being considered). Some likely approaches to control are:

- Menu based, with menus accessed by pulling down from a menu ‘bar.’
- Menu based, with menus being of a pop-up nature.
- Button based.

Directory Functions

Directory access and use will be tailorable in a fashion consistent with the Quipu configuration options using a per-user ".quipurc" file. Similarly a system-wide tailor file will be supported.

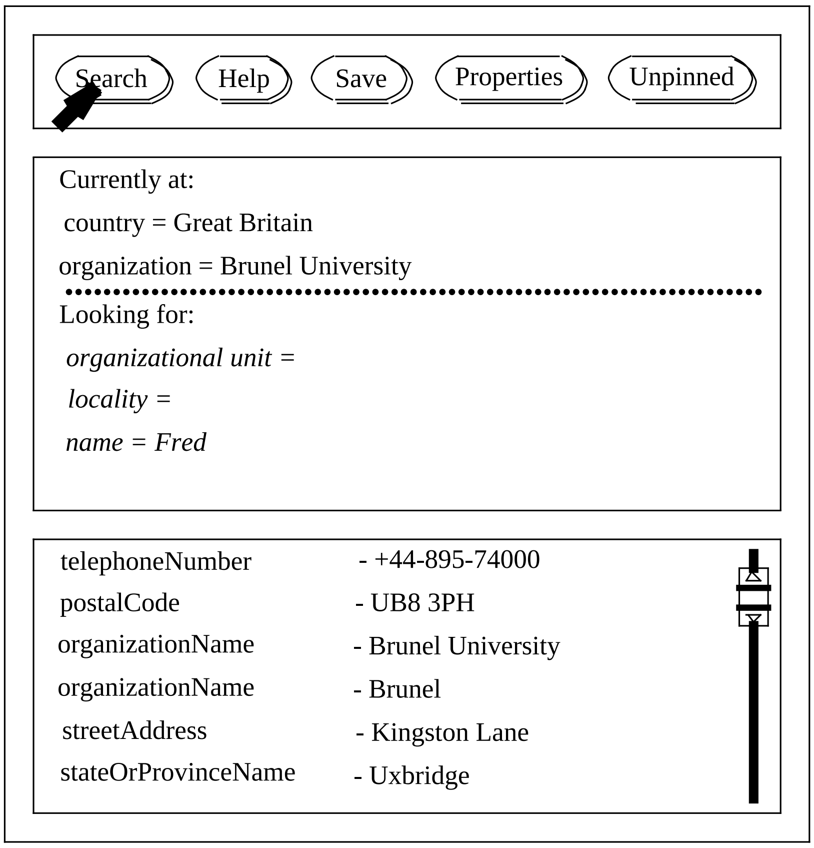
5. An Example DUA Session

In this section some of the ideas described above are illustrated in a set of examples representing a DUA session. For the sake of simplicity a ‘cut-down’ DUA is shown, containing only those functions that are DUA, as opposed to DSA, specific (e.g. Save) or that need some explanation (e.g. Search).

The control and visual aspects of the example DUA are based on those commonly available in the X environment.[2] Also the ‘look’ of the interface is based upon the visual style imposed by the OpenLook toolkit.[4] OpenLook is one of many toolkits vying to become an agreed standard. Other competitors are OSF Motif, Dec windows and Presentation Manager. As yet no winner has emerged, either under X or as a machine independent standard. Accordingly a final decision has not been made as to which style, or combination of styles, will be used in the final design. Thus, in this respect, the DUA shown in the examples is unlikely to be representative of the final DUA design.

5.1. *The Main Window*

Figure 2 shows the main window. It is composed of a command sub-window, a status sub-window and a data/message display sub-window. The functions called by each button in the command sub-window are described below.



Currently at:
country = Great Britain
organization = Brunel University

Looking for:
organizational unit =
locality =
name = Fred

telephoneNumber	- +44-895-74000
postalCode	- UB8 3PH
organizationName	- Brunel University
organizationName	- Brunel
streetAddress	- Kingston Lane
stateOrProvinceName	- Uxbridge

Fig 2. The Main Window

- Search - this will initiate a search specified by the parameters set in the status sub-window (see below for more details).
- Help - clicking on help then on another component of the interface will bring up a sub-window containing a specific help-text.
- Save - this will bring up a sub-window containing functions that allow saving of, say, mail addresses to a file.
- Properties - clicking on the Properties button then on another component of the interface will bring up a sub-window containing a set of relevant parameters that can be modified by the user (figure 3 shows a possible Properties window for the Search button).
- Pin - this allows the user to pin a window to the desktop. An unpinned window will automatically close if it becomes inactive.

The status sub-window is divided by a dotted line. The text above the dotted line represents the current position occupied in the directory, it is the base object used in any searches. The text below the dotted line is the set of search parameters. Initially this will be composed by a set of attribute types that are deemed ‘reasonable’ for a search based at the current position. These parameters are modified by a combination of keyboard and mouse. A more complex set of search parameters can be specified by invoking the Properties function on Search, then adding attribute value assertions (AVAs) using the And, Or and Not buttons.

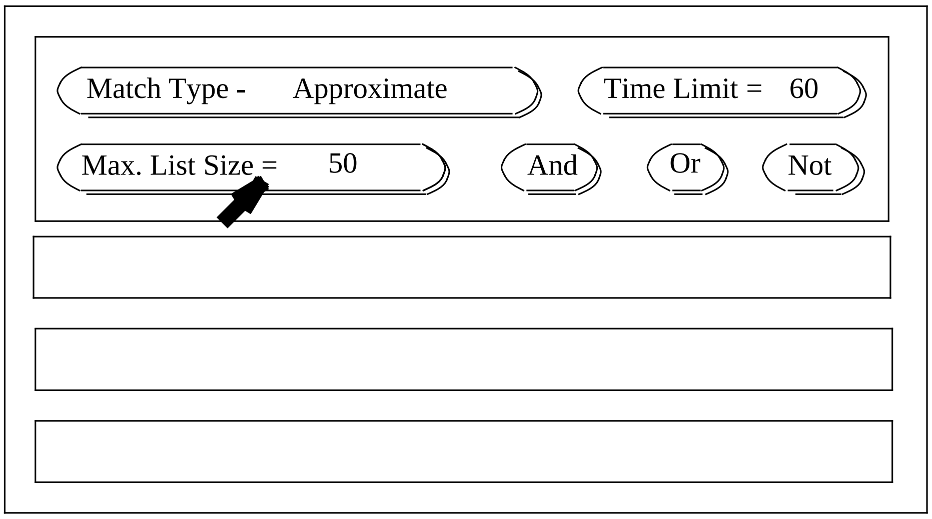
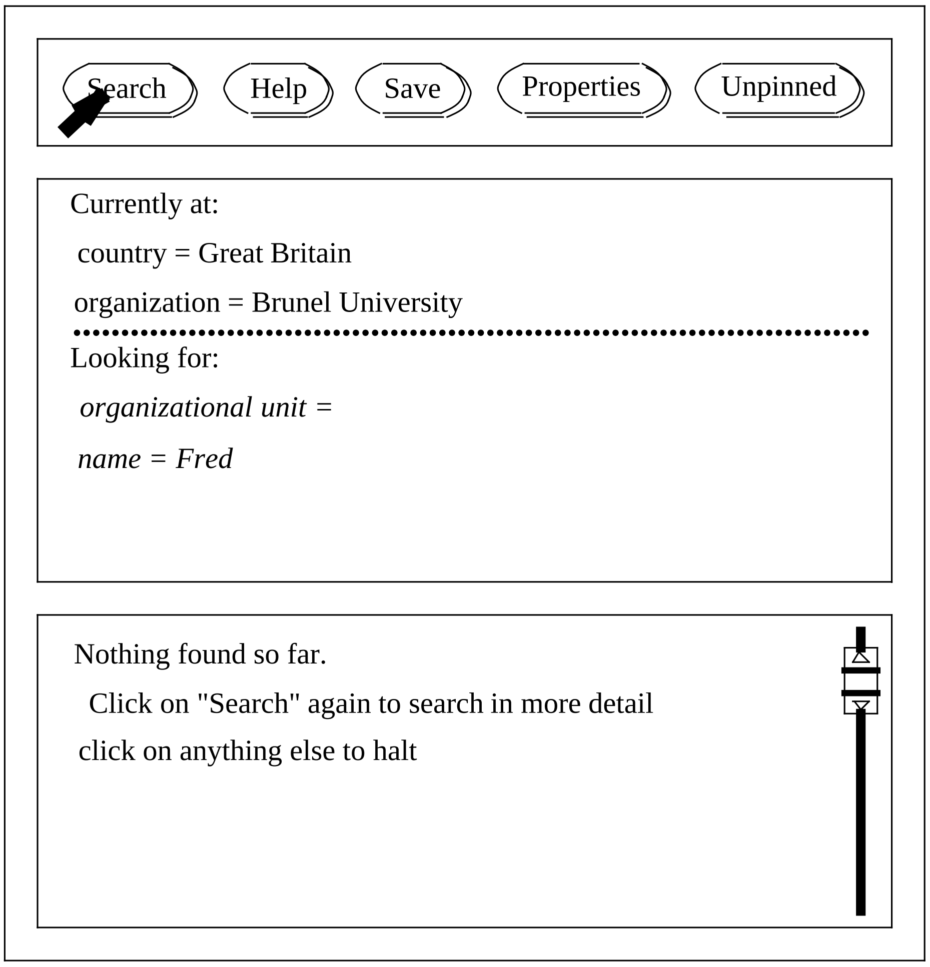


Fig. 3 Properties Window for the Search Function

5.2. *A Simple Search*

The following example show a search for "Fred Bloggs" in Brunel University. Figure 4 shows a snapshot of the DUA after a Search has been requested with the parameter "name = Fred".



Currently at:

country = Great Britain

organization = Brunel University

Looking for:

organizational unit =

name = Fred

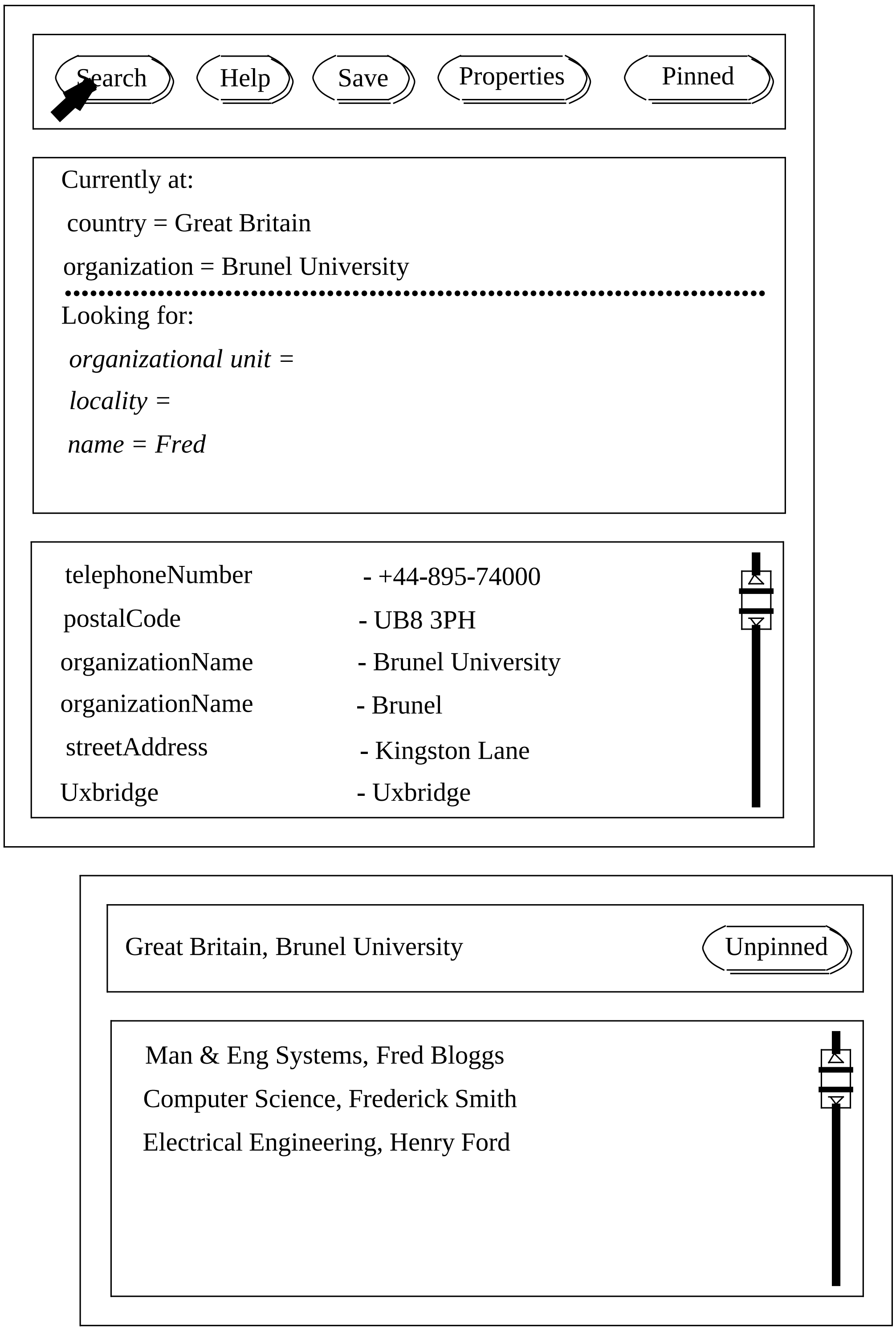
Nothing found so far.

Click on "Search" again to search in more detail

click on anything else to halt

Fig. 4

The Search algorithm used here is based on the Search function described in section 4.5. Nothing is found on the next level of the tree, i.e. at organizational unit level, so the user is asked whether or not the search should continue. Figure 5 shows the result of continuing the search. A new window appears containing a list of "Freds" found under Brunel University.



Currently at:

country = Great Britain

organization = Brunel University

Looking for:

organizational unit =

locality =

name = Fred

telephoneNumber	- +44-895-74000
postalCode	- UB8 3PH
organizationName	- Brunel University
organizationName	- Brunel
streetAddress	- Kingston Lane
Uxbridge	- Uxbridge

Great Britain, Brunel University

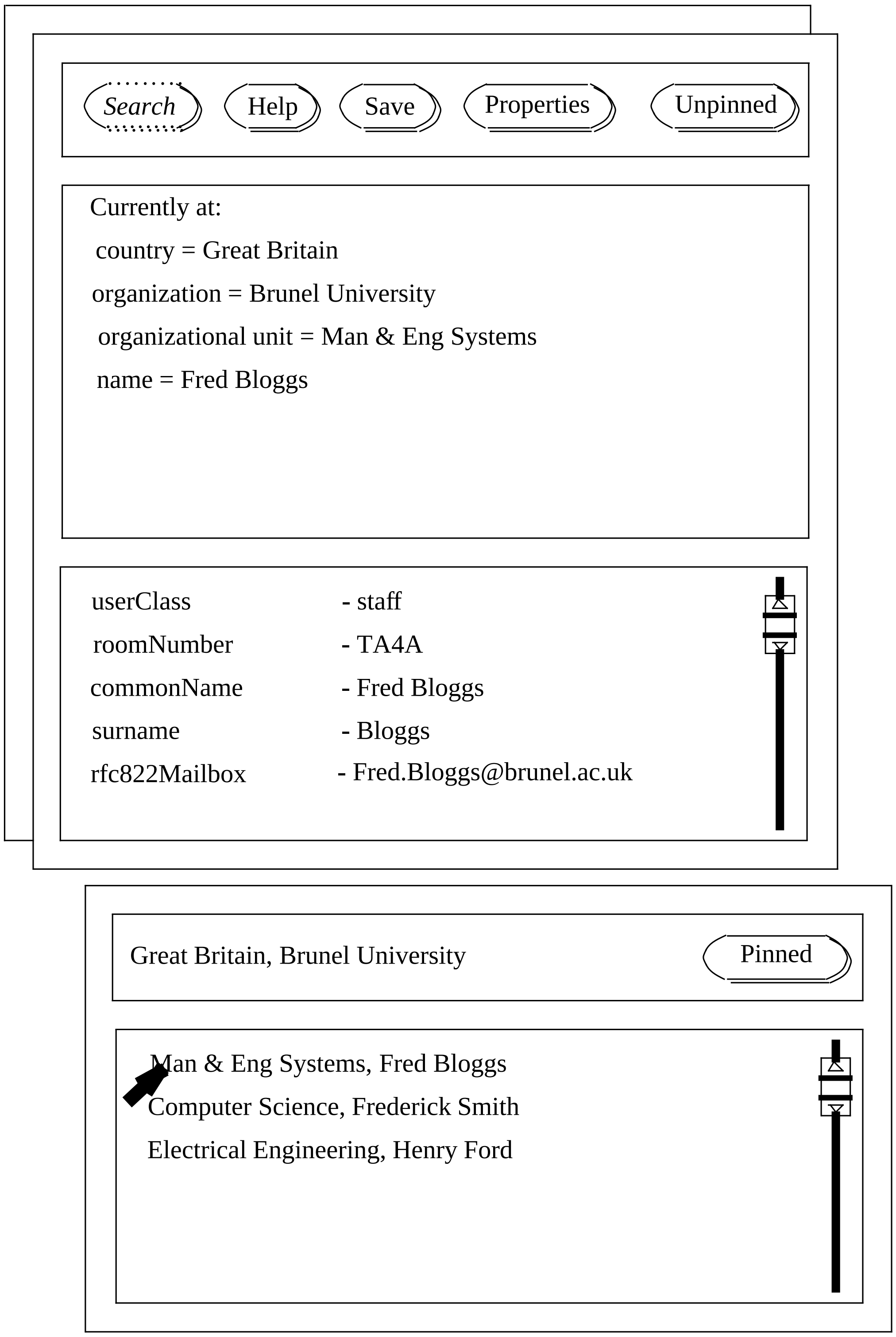
Man & Eng Systems, Fred Bloggs

Computer Science, Frederick Smith

Electrical Engineering, Henry Ford

Fig. 5

Figure 6 shows the user clicking on Fred Bloggs. The result of this is a new main window appearing and overlapping the old one (this overlapping occurs because the old main window has been pinned down, preventing the contents of that window from being overwritten). The new window shows the current position to be "Fred Bloggs" and the contents of his entry. Note that the list window remains on-screen because it has been pinned down, allowing the user to read another of the listed items.



Currently at:
country = Great Britain
organization = Brunel University
organizational unit = Man & Eng Systems
name = Fred Bloggs

```
userClass         - staff
roomNumber        - TA4A
commonName        - Fred Bloggs
surname           - Bloggs
rfc822Mailbox     - Fred.Bloggs@brunel.ac.uk
```

Great Britain, Brunel University

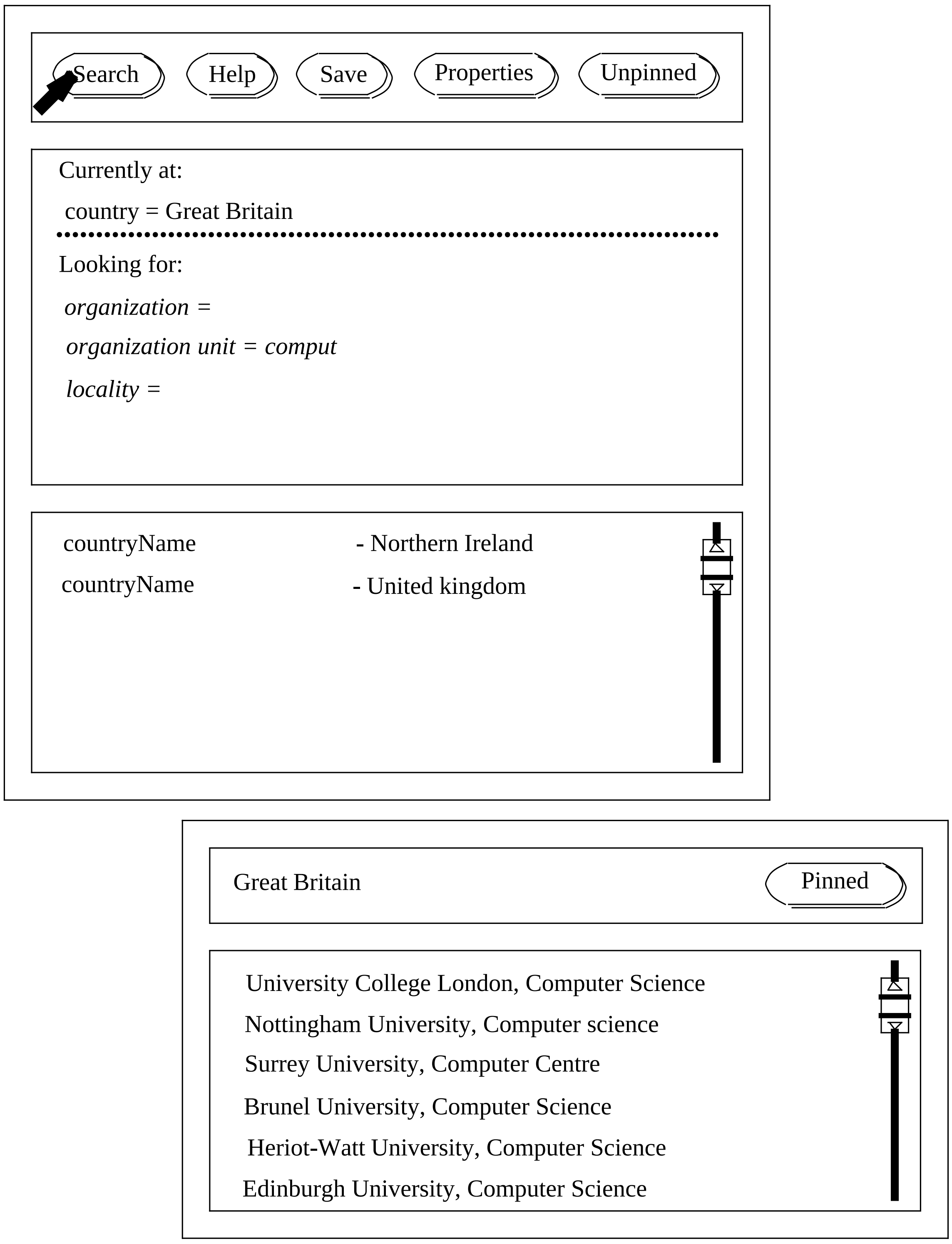
Man & Eng Systems, Fred Bloggs
Computer Science, Frederick Smith
Electrical Engineering, Henry Ford

Fig. 6

5.3. *A Two Stage Search*

This example illustrates the actions of a two stage search, first searching for a computer science department and then searching for someone within that department.

Figure 7 shows the results of a search for a computer science department in Great Britain.



Currently at:

country = Great Britain

Looking for:

organization =

organization unit = comput

locality =

countryName - Northern Ireland

countryName - United kingdom

Great Britain

University College London, Computer Science

Nottingham University, Computer science

Surrey University, Computer Centre

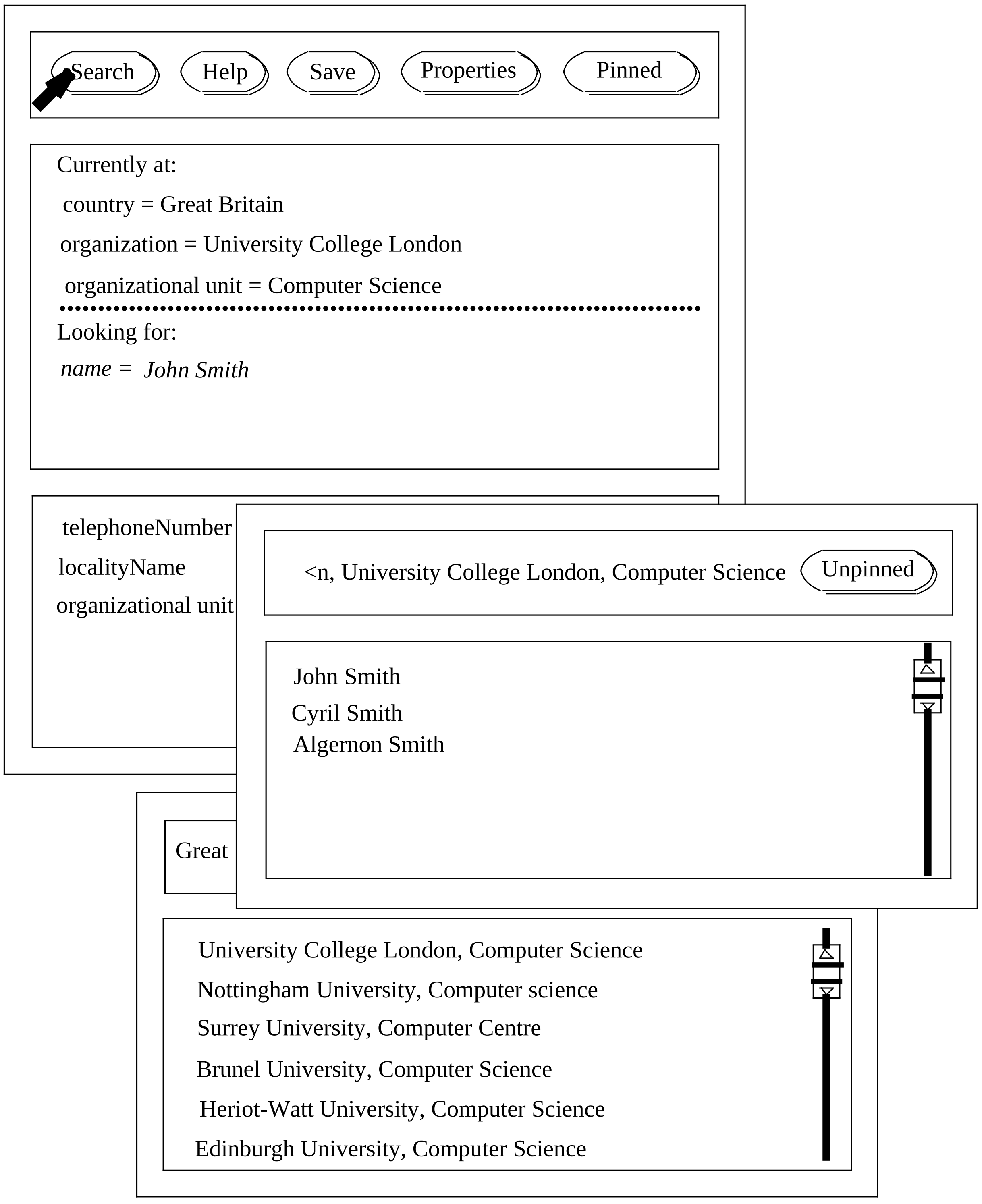
Brunel University, Computer Science

Heriot-Watt University, Computer Science

Edinburgh University, Computer Science

Fig. 7

The result of clicking on "University College, Computer Science", and then requesting a search for "John Smith", is shown in figure 8. The main window is unpinned so it’s previous contents are overwritten.



Currently at:
country = Great Britain
organization = University College London
organizational unit = Computer Science
Looking for:
name = John Smith

telephoneNumber
localityName
organizational unit

<n, University College London, Computer Science

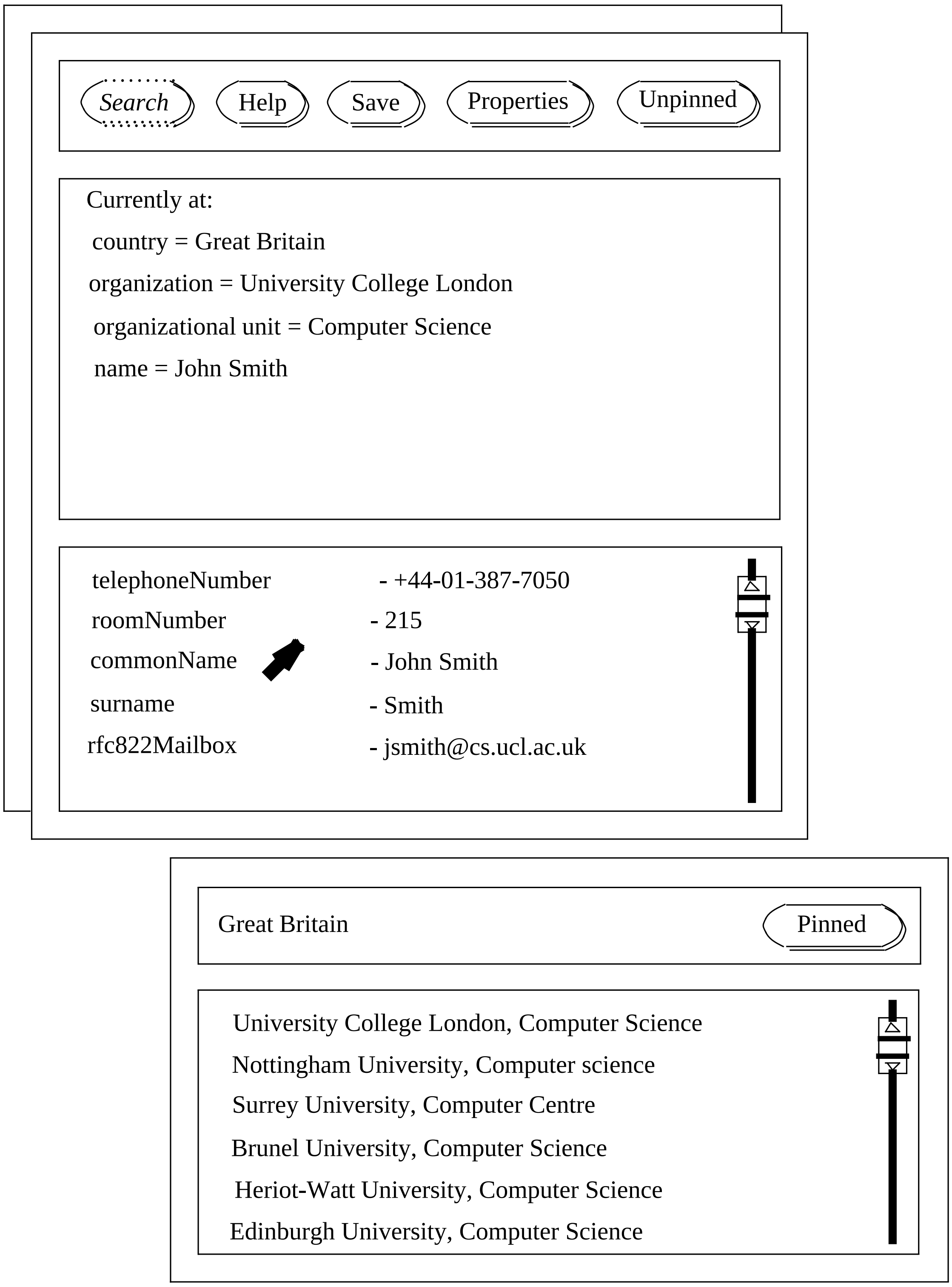
John Smith
Cyril Smith
Algernon Smith

Great

University College London, Computer Science
Nottingham University, Computer science
Surrey University, Computer Centre
Brunel University, Computer Science
Heriot-Watt University, Computer Science
Edinburgh University, Computer Science

Fig. 8

Finally the user clicks on John Smith, causing the (unpinned) list window to close and a new main window to open, as seen in figure 9.



Currently at:

country = Great Britain

organization = University College London

organizational unit = Computer Science

name = John Smith

telephoneNumber	- +44-01-387-7050
roomNumber	- 215
commonName	- John Smith
surname	- Smith
rfc822Mailbox	- jsmith@cs.ucl.ac.uk

Great Britain

University College London, Computer Science
Nottingham University, Computer science
Surrey University, Computer Centre
Brunel University, Computer Science
Heriot-Watt University, Computer Science
Edinburgh University, Computer Science

Fig. 9

6. Request for Comments

The authors would welcome feedback on the interface proposed in this paper. Comments should be addressed to X500@brunel.ac.uk.

To allow sites without ISODE to try an X.500 user interface, a public access service has been established at Brunel. The service is based on a modified version of the "widget" DUA released with ISODE 5.0, and can be accessed by calling uk.ac.brunel.dir on JANET. Again, any comments should be sent to the above address.

References

1. ISO/CCITT, *Recommendation X.500: The Directory - Overview of Concepts, Models and Services*, Geneva, March 1988. ISO 9594 is technically aligned with X.500
2. Valerie Quercia and Tim O’Reilly, *X Window System User’s Guide,* 3, O’Reilly & Associates, Inc., Sebastopol, California, July 1989. Second Edition
3. ISO/CCITT, *Recommendation X.520: The Directory - Selected Attribute Types*, Geneva, March 1988. ISO 9594-6 is technically aligned with X.520
4. Tony Hoeber, ‘‘Open Look design goals,’’ *Sun Technology*, pp. 63-75, Sun Microsystems, Mountain View, California, 1988.