Providing multiple external views on Directory User Interfaces

Rui J.P. José <rui@uminho.pt>
António Costa <costa@uminho.pt>
Joaquim Macedo <macedo@uminho.pt>
Vasco Freitas <vf@uminho.pt>

Abstract

Although there are many types of Directory User Interfaces they all tend to give to their users the same global view of the Directory. However, a typical user will only be interested in some portion of the total Directory database. Providing such specific view might be done for instance by restricting the default DUA display to a subset of object classes or attribute types meaningful to a particular application.

The work described in this paper is not about interface styles or user friendliness. It is about defining mechanisms that allow for users to easily setup several configurations based on the information they intend to access. They should be adaptable to a variety of interfaces and, to demonstrate that, as well as raising some practical implementation issues, two interfaces based on these concepts are presented: A windows LDAP DUA and a WWW to X.500 gateway.

Directory interfaces adopting this model are expected to help users make more advanced uses of the Directory and at the same time not need such a good knowledge of X.500 and its operations.

I. Introduction

The X.500 protocol is very powerful and in the opinion of many people, excessively powerful, due to the complexity that such a flexibility implies. Without entering into such discussions there is however one point that must be stressed. The users of the Directory rarely use the great majority of the functionalities that they have at their disposal.

A typical interaction, irrespective of the interface being used, consists of very simple operations usually performed without any parameters. Even an experienced DSA manager using a powerful interface like *dish*[1] will tend to work in such a rudimentary manner by using simple *list* and *show* operations. In so doing, the user is overwhelmed with more information, in terms of number of entries and attributes per entry, than he can handle or even need, which he could have avoided should he had used a correctly parametrized operation.

Superfluous information makes the user's task much more difficult. The alternative approach of writing one single well parametrized command that would return less but more significant information has important disadvantages:

- It requires a perfect knowledge of the interface commands and their parameters.
- It requires knowledge of the Directory schema in terms of attributes and object classes used, matching types or possible values for some attributes.
- It must be done every time the request is needed which may be tedious and a potential cause for mistakes.

Therefore, at least for the most common queries, some kind of automation that reliefs users from these burdens should be included in the functionality of the interface.

II. The external view

Although the Directory is not a general purpose database [2], it is, nevertheless, a Database and therefore it has many common aspects with traditional general purpose DBMSs[2]. These systems are usually based upon a 3-layer architecture as illustrated in Figure 1, known as ANSI/SPARC architecture [3] in which:

- The **internal level** - is the one closest to physical storage - i.e., it is the one concerned with the way the data is physically stored.
- The **conceptual level** - hides the details of physical storage structures and concentrates on describing entities, data types and relationships. The conceptual level is a representation of the entire information content of the database
- The **external level** - is concerned with the way the data is viewed by individual users. Each view typically presents in an adequate

[1] *dish* is part of QUIPU, the UCL X.500 implementation [1]

[2] Database Management Systems

way the part of the database that a particular user is interested in and hides the rest of the database from that user.

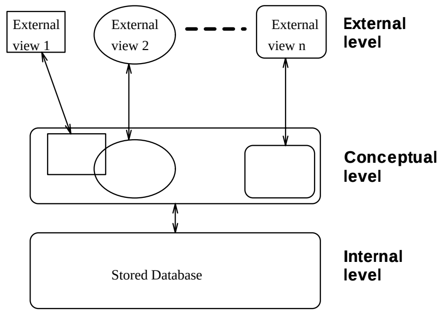


Figure 1: ANSI/SPARC architecture

If one tries to adapt this architecture to the Directory, one may easily associate the internal level with the data as is stored in DSAs[3] like for instance in an EDB[4] format.

As for the conceptual level it seems to match very well with the presentation of that data in a structure know as DIT[5].

Finally there is the external level which corresponds to the information being presented to a user by a DUI[6]. The ANSI/SPARC term for an individual user's view is an **external view**[5]. An external view, therefore, is the content of the database as seen by some particular user (that is, to that user the external view is the database). For example, a receptionist could see the Directory as an ordered list of people entries containing telephone and room numbers.

III. The conceptual interface

External views are important specially in a global service like the Directory and there are several ways of providing them on Directory interfaces. The most obvious is by creating specialized programs. This is not very common however since it requires a perfectly stable service. A X.500 capable mail reader with a Directory Interface for retrieving mail addresses could be mentioned as an example.

A more feasible alternative is to create generic interfaces which can be configured to work in many different ways. They exploit the fact that many applications tend to fall into a small number of rather stereotyped patterns. Therefore they can be tailored to suit some service specific requirements by simply providing values for a few configuration parameters.

In a more or less complex way, almost all DUIs have some kind of configuration which is usually set at installation time. This method of setup has very little flexibility because it does not take into account the information being consulted. Even when there is only one single user he may need to access many different types of information. If he only disposes of a possible configuration then it must be made generic enough to serve all those access patterns.

A much more useful method would allow for several more precise setups oriented at specific applications. In fact, the setup is much more influenced by what is being consulted than by who is consulting so the user should be given the possibility to personalize his setup in a number of ways. The conceptual interface will therefore allow for several choices of service oriented configurations.

In order to take advantage of the power of X.500 without running into the problems of complexity mentioned earlier on, the DUI will have the ability to store several configurations in a persistent way.

Thus the user may refer to previously prepared complex setups by simply choosing the appropriate configuration.

In order to make it easier to describe how to configure a display of information, it is considered that the conceptual interface has at least two types of display formats both of them appearing either simultaneously in different display areas or, alternatively, in the same display area. They are:

- The display of a group of entries in a short format. This display format will be called **list view**.
- The display of an single entry in a longer format or even the complete display. This display format will be called **entry view**.

The list view is often used to select an entry to be displayed in the entry view.

IV. The External Schema

Each external view is defined by means of an external schema, which basically consists of a set of meta-information elements or configuration variables. The defined meta-information elements

[3] Directory System Agent
[4] Entry Data Block, the format used by quipu DSAs to store information
[5] Directory Information Tree
[6] Directory User Interface. The term DUI will be used instead of DUA (Directory User Agent) since the later may imply an assumption that the agent is able to "talk" valid X.500 protocol [4]

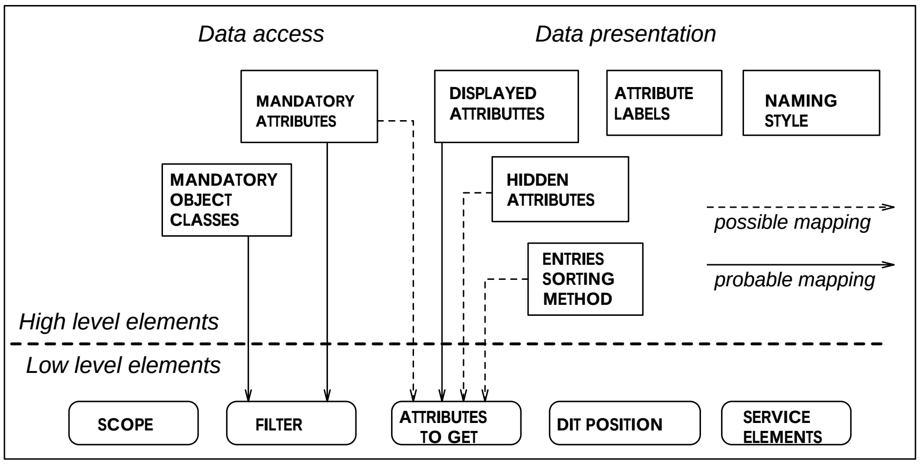


Figure 2: Mapping high level onto low level configuration variables

and their values will determine the view presented to the user. These elements however are all optional and the total absence of configuration will simply correspond to the conceptual view of the Global Directory Service.

Then, what should a representative set of meta-information elements be? An answer to this question has to account for both the actual user requirements and the protocol and interface capabilities. To simplify the inclusion of both inputs, the elements are classified into two layers. The first one will have a direct correspondance to the protocol operation parameters on top of which a second layer with more abstract elements may be implemented. The latter elements are more appropriate for users but in most cases must be mapped onto the elements of the lower level.

The choice between using only the elements of one level or a combination of both can only be made in the context of a particular implementation.

IV.A. Low level elements

Since most DUAs and X.500 gateways currently available are based on LDAP[6] and the functionalities of this protocol may also be found in the X.500, our model is also based on the LDAP protocol. As a consequence, the elements described match the parameters of the LDAP search operation.

Therefore, at the low level, an external schema will be defined by the configuration of the following variables or meta-information elements:

- Initial DIT position
- scope
- filter
- attributes (a list of the attributes to be returned from each entry found)
- Service elements (eg. sizelimit, timelimit or derefAliases).

IV.B. High level elements

Although very powerful, the low level elements are not very adequate for user manipulation and they provide no layout information. Therefore it is desirable to have the possibility of specifying a X.500 service in terms of higher level elements. During operation most of these elements will be mapped onto lower level elements as shown in figure 2.

High level elements are nearer the user and what is perception of a service definition might be. Instead of dealing with filters or other protocol related parameters, a user defines a service by providing answers to questions such as *What are the only object classes that should make part of my service universe?* or *What are the essential attributes that all the objects must have in order to be included in my service universe?*

A first set of high level elements has to do with **data access**. Its goal is to define the subset of the X.500 information that will constitute the service universe:

- **Mandatory object Classes** - this service element, when defined, determines that only objects of the indicated classes should be considered as belonging to the service universe. This element is mapped onto the low level element *filter* by adding a condition like *objectClass=ClasseType* for each mandatory class. For instance if *person* and *role* were defined as mandatory object classes then a condition *objectClass=person or objectClass=role* would be automatically included in the filter of every search operation. As a consequence, only entries of those object classes or sub-classes would be returned. Browse operations however could build a filter that also allowed for the return of Non-Leaf objects in order to maintain navigability.

- **Mandatory attributes** - if defined, establish that only entries containing at least one of these attributes will be considered as part of the service universe. Once again a browse operation may be the exception by including NonLeaf entries. If one intended to define an external schema for a telephone directory service, the attributes *telephoneNumber* and *facsimileTelephoneNumber* would be good candidates for mandatory attributes. The mapping onto the lower level elements would be achieved by adding to the filter the condition *telephoneNumber=* or facsimileTelephoneNumber=**. In this way, entries not containing any of these attributes would not be returned.

A second set of high level elements concerns **data presentation**. They configure the display of service data to the user in some appropriate fashion. The choices made here will affect the functioning of the list view and the entry view referred to in the description of the conceptual interface discussed in section III.:

- **Entries sorting method**, defines the order in which entries are listed in a list view by selecting the attributes used to determine that order. In principle, because there is no way of controlling the order in which entries are returned, this is not mapped into any lower level element and must be implemented by the client. In order that a sorting operation be possible, an implementation decision might choose to force the return of the attributes selected as sorting keys by including them in the low level element *attributes to get*.

- **Displayed attributes**, this variable gives an indication on the attributes that should be displayed when presenting entries. An ordered list may be used to give an additional indication of the appropriate order in which to present these attributes. This will be mapped into the lower level element *attributes to get*. There should be one for entry views and another for list views.

- **Hidden attributes**, in some cases it may be simpler to indicate which attributes should not be displayed. This is a very useful facility if one intends to specify a rather generic service but wants to hide attributes that are considered of no interest to the application, like, for instance, administrative attributes. In principle it applies only to entry views.

- **Naming Style**, when presenting an entry's name, different styles may be adopted. At least 3 styles should be allowed: **DN** for the full *Distinguished Name*, **RDN** for the entry's *Relative Distinguished Name* and **BaseDN** which is an imcomplete DN starting from the DN of the operation base object. The use of *User Friendly Naming* [7] may also be considered which would increase the possible values for this variable. Although name style configuration may be applied to entry views it may be more useful in list views.

- **Attribute labels**, the aim is to choose some schema to indicate what labels will be used to identify the names of the attributes to the user. The basic approach consists of just indicating whether attribute labels should or should not be presented. A more elaborated mechanism may include a choice among several languages or any other criteria. The external schema should allow different values for list and entry views.

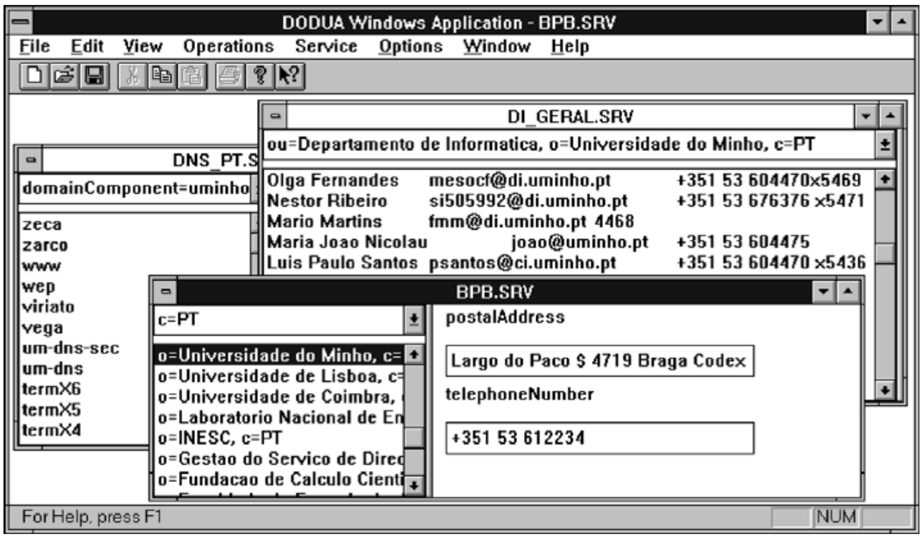
V. Implementation of the model

The elements referred to so far are only those considered to be more generic and which do not compromise the model with any particular implementation. It is important to notice however that when it comes to a particular implementation many other meta-information elements may be included using the same concepts as long as they make sense in that particular context.

Some of such elements were not included in the general model because they depend too much on the interface programs being used. An example of such an element would be that which concerns cache parameterization.

Other elements are not clearly application specific and therefore it should be an implementation decision whether to associate them with a certain external view or not.

A good example is a LDAP server. In principle it would be configured on a per installation basis but there might be some advantages in doing so on a service basis. For example, to choose a server that is nearer to the information to be con-


DODUA Windows Application - BPB.SRV
File Edit View Operations Service Options Window Help
DI_GERAL.SRV
DNS_PT.S
BPB.SRV
For Help, press F1
NUM

```
ou=Departamento de Informatica, o=Universidade do Minho, c=PT
Olga Fernandes     mesocf@di.uminho.pt        +351 53 604470x5469
Nestor Ribeiro     si505992@di.uminho.pt      +351 53 676376 x5471
Mario Martins      fmm@di.uminho.pt  4468
Maria Joao Nicolau          joao@uminho.pt    +351 53 604475
Luis Paulo Santos  psantos@ci.uminho.pt       +351 53 604470 x5436
```

```
domainComponent=uminho
zeca
zarco
www
wep
viriato
vega
um-dns-sec
um-dns
termX6
termX5
termX4
```

```
c=PT
o=Universidade do Minho, c=
o=Universidade de Lisboa, c=
o=Universidade de Coimbra,
o=Laboratorio Nacional de En
o=INESC, c=PT
o=Gestao do Servico de Direc
o=Fundacao de Calculo Cienti

postalAddress
Largo do Paco $ 4719 Braga Codex
telephoneNumber
+351 53 612234
```

Figure 3: A sample *dodua* screen with three open documents

sulted, and therefore faster, or to distribute complete configurations among beginner users.

V.A. Document Oriented DUA

The *dodua* implementation has been programmed in Windows. The service concept is implemented through the use of document files. Each document stores the meta-information associated with a certain service. Accessing a given service is achieved by opening the corresponding file. A user working with *dodua* can have several personal documents each one tailored to a specific information domain.
The document approach has two great advantages, namely:

- A user is completely free to create whatever documents he considers he may need.

- A manager can use his knowledge of Directory Services in general and of his organization in particular to create complex documents which may then be distributed among users. When DUAs are installed they may already include a set of documents enabling novice users to immediately access X.500 information in such an intuitive way as opening a file.

Another advantage of the use of documents as the basis for meta-information is the large amount of meta-information they allow to store. The current *dodua* implementation goes far beyond the elements defined in the previous section. In each document it includes meta-information like LDAP servers, a service name and description, user information, several hotlists, cache parameters and parameters of program behavior.

The document model might even go beyond meta-information and support downloading of information itself like in the *SWIX*[7] implementation.

Figure 3 shows a sample *dodua* screen.

V.B. Service Oriented Web gateway

SOW is an adaptation of the web500gw[8]. New operations were added to support the notion of multiple services. These new operations work with a special URL format in which the query string is used to carry meta-information variables. The lack of an accepted URL syntax for the X.500 protocol led us to define our own syntax. This syntax however is still under development and other related work like the one used in the SOLO [8] project is also being considered. The query string syntax is a list of configuration variables in the format *variableLabel=variableValue* separated by the / sign. Some variables allow for several values which must be separated by a comma like in the following example:

...MandOb=person/LiAttr=telephoneNumber,mail

The available variables allow the user to define the service information universe by setting mandatory objects or attributes and by specifying a generic filter and operation scope. They also allow to setup the display of information in

[7] *SWIX* is developed at UMDAC (umdac, Umea Universitet, se)
[8] web500gw was written by Frank Richter, Technical University of Chemnitz-Zwickau, Germany

terms of displayed attributes and respective labels for either entry view or list view. When the resulting page is returned, its DN links also come with the same meta-information appended in the query string. In so doing service parameters are kept for the time of a user session.

A possible scenario for this gateway is a CWIS[9] in which instead of the generic link to X.500 information several links could be set up to more specific services like the classical telephone directory or a list of host computers. More details about SOW are available at:

<URL:http://info.uminho.pt/˜rui/sow.html>

VI. Conclusions

Our experience with a user defined service interface has shown that, after all, it is nice to have the X.500 world on your desktop but it is a lot of weight to be there at the same time, specially when all one wants is to get hold of the telephone number of someone working in one's building. The provision of adequate views is therefore an essential element in the design of DUIs and can be specially useful to support directory applications that have reached a certain stability in terms of users and information content.

The model described seems to obviate to some of the problems which have been pointed out as the cause of the little use of network applications [9]:

- It gives the users a method of producing services that respond to their requirements.
- There is no conflict between complexity and flexibility in terms of protocol operation. Although they remain related, it is easy to set the right balance by defining more or less complex services.
- The X.500 protocol becomes more hidden from users.

The updating of information is not considered in this paper because not enough experimental work has been done yet in this area. However there are some obvious benefits that could be gained with the extension of this model to update operations like the inclusion of templates on the service definition.

The concepts described can also be extended to other types of interfaces like report generator applications similar to the ones that exist in most Database Systems.

[9]Campus Wide Information System

VII. References

[1] Robbins, C., and Kille, S., "The ISO Development Environment: User's Manual. Version 7.0", X-Tell Services Ltd, UCL, 1991.

[2] CCITT Blue Book, "Data Communications Networks - Directory - Recommendations X.500-X.521", ITU, 1988.

[3] Elmasri, R., "Fundamentals of Database Systems", The Benjamin /Cummings Pub Co, Inc.

[4] Barker, P., "DUA Metrics", RFC 1431, UCL, Feb 1993.

[5] Date, C.J., "Database Systems", Addison-Wesley Publishing Company.

[6] Kille, S., Yeong, W. and Howes, T., "X.500 LightWeight Directory Access Protocol", RFC 1487, Performance Systems International, University of Michigan, ISODE Consortium, July 1993.

[7] Hardcastle-Kille, S.E., "Using the OSI Directory to achieve User Friendly Naming", Draft Document OSI-DS 24, UCL, Jan 1992. Work in Progress from the OSI Directory Services (OSI-DS) Working Group of the IETF.

[8] Zahm, A., Huitema, C., Pays, P-A. and Woermann, A., "Simple Object Look-up protocol (SOLO)", Networking Working Group, October 1994. INTERNET-DRAFT, INRIA, TS E3X.

[9] Waugh, A., "Where are the Network Applications?", Technical report, CSIRO Division of Information Technology, 1994.

VIII. Author Information

Rui José graduated in Systems and Informatics Engineering in 1993 at the University of Minho, Portugal, and is currently finishing up his Masters Thesis with the Computer Communications Group of the Department of Informatics. He has been involved in several projects on X.500, WWW and Information Services.

António Costa is Assistant Lecturer in Computer Comunications at the University of Minho, Braga, Portugal. He graduated in Systems and Informatics Engineering in 1992 at this University and is currently pursuing post-graduate studies in network information services and protocols. He has been involved in several R&D projects under contract with the Computer Communications group in this area.

Joaquim Macedo is a Lecturer of Computer Communications at the University of Minho, Braga, Portugal. He graduated in Electronics and Telecommunications Engineering in 1983 at the University of Agostinho Neto, Angola, and received his Masters degree from the University of Minho in 1993. From 1990 until 1994 he participated in several technical working groups for the establishment of the Portuguese National R&D Network, the RCCN, and was particularly active in the development of X.500 Directory Services. He is currently doing research for a Doctoral degree where his interests concern networked information and directory services and protocols.

Vasco Freitas is Associate Professor of Computer Communications at the University of Minho, Braga, Portugal. He graduated in electronic and telecommunications engineering in 1972 at the University of Lourenco Marques and received his M.Sc. and Ph.D. degrees from the University of Manchester (UK) in 1977 and 1980 respectively. From 1989 until 1994, on a partial leave from his University, he was appointed a member of the Board of Directors of the Portuguese National Foundation for Scientific Computing, where he had the opportunity to foster the establishment of the Portuguese R&D Network (RCCN). He represented the national networking initiatives for R&D in the RARE Association, DANTE and Ebone from their beginnings until recently. Currently, his interests concern networked information services and protocols and the specification, modelling and prototyping of communication protocols.