Mapplet: a CORBA-based genome map viewer

Kim Jungfer and Patricia Rodriguez-Tomé

EMBL Outstation–Hinxton, The European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge CB10 1SD, UK

Received on April 28, 1998; revised on June 22, 1998; accepted on June 23, 1998

Abstract

Motivation: There are a large number of genetic and physical maps, distributed at many sites. Each site offers different kinds of access methods and viewers. CORBA, the de facto standard for distributed object-oriented computing, offers new opportunities to unify the view on these maps through standard interfaces. A collaboration of Infobiogen and the EBI proposes a common IDL for maps.

Results: A CORBA map viewer is presented which serves as a proof of concept for the proposed IDL. It demonstrates its usefulness in the context of map viewing and its ability to handle large maps with >1000 markers. The viewer gives access to the maps of the Radiation Hybrid Database at EBI. It gives a quick overview of several large maps side by side. The marker density at each map position is displayed and different marker types can be highlighted.

Availability: Demonstration and source code at: http://sun-nv.ebi.ac.uk/~jungfer/Mapplet

Contact: jungfer@ebi.ac.uk

Introduction

There are a large number of genetic and physical maps available (Hudson et al., 1995; Dib et al., 1996; Schuler et al., 1996). Maps are typically accessible through World Wide Web (WWW) interfaces, directly printable graphic files (e.g. Postscript) or simple text files. The disadvantage of these possibilities is that the user is restricted to predefined views, anticipated by the data provider. CORBA (Siegel, 1996; Ofali et al., 1997), the de facto standard for distributed object-oriented computing, overcomes such restrictions. Clients and servers interact through public interfaces, which hide implementation details. This separation allows flexible recombination of data sources with different viewers and applications, including WWW interfaces. Microsoft’s DCOM (Sessions, 1998) is currently the most important alternative to CORBA. However, DCOM is not considered here due to its proprietary nature and lack of cross-platform support.

The CORBA standard specifies the Interface Definition Language (IDL), which provides a language-independent way of describing public interfaces of objects. The Object Request Broker (ORB) transparently transmits request from clients to object implementations. Several research groups now provide public CORBA servers for map databases (Barril et al., 1998; Lijnzaad et al., 1998) and CORBA map viewers have been developed (Rodriguez-Tomé et al., 1997; Hu et al., 1998). A large number of other CORBA applications in Bioinformatics can be found at http://industry.ebi.ac.uk/~corba.

Map viewers and data sources from independent providers can only interoperate if they share a common set of IDL definitions. In collaboration with Infobiogen in France, EBI is developing a common IDL definition for maps (EU grant BIO4-CT96-0346). It captures those properties common to both the radiation hybrid maps at EBI and the genetic maps at Infobiogen. It is being proposed as a standard to the Life Sciences Research Special Interest Group of the OMG (http://lsr.ebi.ac.uk). The viewer builds on the results of this collaboration and provides a proof of concept for the common map IDL.

The Radiation Hybrid Database (RHdb) contains experimental radiation hybrid data and maps derived from these experiments (Lijnzaad et al., 1998). This includes STS data, scores, experimental conditions and extensive cross-references. Mapplet allows a quick overview of the maps currently in RHdb. It displays four maps side by side, can highlight different marker types, and gives access to associated experimental data. The basic functionality of the viewer depends only on the common IDL. However, additional data which are specific to RHdb can also be accessed and are described in a separate set of IDL definitions. Mapplet is a further development of a map viewer for the CORBA server (Rodriguez-Tomé et al., 1997). The main differences to the previous version are the adaptation to the new IDL, the possibility to highlight different marker types, and the possibility to display the marker density at a given map position.

Components and interfaces

CORBA objects are used to represent the main components of the application. These components are as follows.

- The maps: having an object reference to a map object, a client can request general information about the map (like name and type) as well as the markers on the map and their position.
- The map trader: a client can ask the trader which maps are available. Example criteria are species, chromosome or a specific marker on the map. If the trader knows about maps, which satisfy the query, it returns one or more object references for these maps. The map viewer can use the object reference to access the map directly and display it.
- The score database: if the map is a radiation hybrid map of RHdb, then additional information about the markers is available. Examples are marker types and score vectors.

The IDL definitions for the application consist of two parts. The first part is the common map IDL, which contains methods available for all types of maps. The second part extends the common IDL with methods specific to radiation hybrid maps. The common IDL is still under discussion and subject to changes. The map viewer will be adapted to future versions. The most recent version can be found at http://sunny.ebi.ac.uk/EBI/RHdb/EUCORBA. Here we present only those aspects of the common IDL necessary to understand the map viewer.

**The common map IDL**

The common IDL defines the interfaces for traders and maps. The most basic method of a trader is to return an object reference given an object identifier. Since this method is not specific to map traders, it was defined in a separate trader module.

```idl
module Traders {
    interface Trader {
        Object getByOid(in string oid) raises (ObjectNotFound);
    };
}
```

A specialized version of the trader is defined in the module Maps. Here it is possible to specify general search criteria like species and chromosome. The result type MapList represents a sequence of maps, since more than one map can match a query.

```idl
module Maps {
    …
    interface MapTrader : Traders::Trader {
        …
        MapList getMapList(
            in string mapType,
            in string species,
            in string chromosome,
            in Strings markers)
        raises (Traders::Trader::NotFound);
    };
}
```

The maps themselves are also defined in the module Maps. Their main attributes are oid, name, type, species, chromosome and elements. Through the attribute elements it is possible to get a list of all markers, which belong to the map. Because the MapElement (marker) is represented as ‘struct’ and not as interface, it is possible to get all marker data belonging to a map with one method call.

```idl
module Maps {
    …
    struct MapElement {
        MarkerData markerData;
        float position;
        …
    };
    typedef sequence < MapElement > MapElementList;
    …
    interface Map {
        readonly attribute string oid;
        readonly attribute string name;
        readonly attribute string type;
        readonly attribute string species;
        readonly attribute string chromosome;
        readonly attribute MapElementList elements;
        …
    };
}
```

**The IDL for radiation hybrid data**

The Scores database is represented by the interface DB in the module RHscores. It has two methods: one allows the access of additional information for an individual marker, and the second to get the marker types of all markers of a specified map. This second method is in theory redundant, but was added for performance reasons. The map viewer allows the highlighting of all markers of a specific type. If the viewer had to request the marker type individually for each marker, then network load and response time would be unacceptable.

```idl
module RHscores {
    typedef sequence<string> Strings;
    struct Reference {
        string database;
        string accession;
    };
    typedef sequence <Reference> References;
    struct Score {
        string rhid;
        string panel;
        string author;
        string vector;
        string sts;
        Strings stsTypes;
    };
}
```
typedef sequence <Strings> Types;
exception NotFound {};
interface DB {
  Score getScore(in string rhid) raises(NotFound);
  Types getTypes(in string mapid) raises(NotFound);
}; // module RHScores

Implementation
The map viewer was implemented in Java using the CORBA 2.0 compliant ORB OmniBroker 2.0.3 (http://www.ooc.com/ob.html). It can run as a stand-alone application as well as within a WWW Browser. Since no proprietary features are used, the same source code should work with any other recent, CORBA-compliant Java ORB. Therefore, the applet can use the built-in ORB of Netscape’s Communicator. The viewer displays four maps side by side, which can be selected by chromosome. Further selection possibilities are planned as soon as more maps are available. Each map can be scrolled and zoomed independently. Zooming can be done using the ‘In’, ‘Out’ and ‘Full’ buttons.

The maps are depicted in three parts.
- A scale.
- The markers: depending on the zoom level, they generally overlap.
- A marker density histogram: depicts how many markers are mapped in a specific region.

A marker can be selected with the mouse by clicking on it. Pressing the ‘show’ button of the corresponding map will display detailed information about the marker, like name,
position and cross-references. For radiation hybrid maps at EBI, the assay information together with the score vector is displayed. The ‘Next’ and ‘Prev’ buttons select the next or previous marker. Different marker types can be highlighted, e.g. all markers which belong to the framework or all genetic markers or all ESTs. The histogram will then show the density of the highlighted markers.

It would have been possible for the application to utilize the normal existing CORBA servers for the RHdb (Lijnzaad et al., 1998). However, in order to increase the performance, a special caching server was implemented, which uses these servers but avoids direct access to the underlying relational database at run-time. Since all servers use the same IDL, none of these changes affect the map viewer.

Discussion

The common IDL treats maps as CORBA objects. This has the advantage that map and trader can be separated. The trader can know about maps, which are not local to it. When a client asks for a map, then the trader does not return the map itself, but a reference to the map. The client can then use the reference to access the map directly without knowing where it resides (location transparency; Figure 2). We hope that the collaboration with Infobiogen will soon allow access to EBI’s radiation hybrid maps as well as to genetic maps at Infobiogen in France.

Unfortunately, this CORBA feature cannot be fully exploited if the client runs within a Web browser. Security restrictions require the map trader, the map and the HTTP server from which the applet is loaded to reside on the same machine. Some more advanced ORBs (e.g. Visibroker; http://www.inprise.com/visibroker) offer a solution to this problem using a demon which runs on the same machine as the HTTP server and which forwards requests to object implementations on different machines.

The map viewer has been designed for use over Wide Area Networks (WANs) where round-trip response times can be slow and where granularity of data access becomes the single most important factor in browsing performance. A straightforward translation of a conceptual data model into an interface-based IDL will normally lead to inefficient solutions. An example (Hu et al., 1998) is where the MapElement equivalent Locus is only represented as an interface. This means that in order to display a map, it is necessary to access each displayed Locus individually by remote method calls. Such an approach is therefore limited to small maps or local network applications. In this context, it is important to realize that IDL is not a data modelling language, but a specification language for an API. Ultimately, the concrete application determines the IDL—not an abstract data model. Using the common map IDL, the map viewer can download positions and names of all markers belonging to a map with only one method call. This is possible because the MapElement is modelled as a struct, which can be passed-by-value. Apart for this difference, the IDL (Hu et al., 1998) and the common map IDL show many similarities.

The Maplet is based on the common map IDL. However, it was necessary to extend this IDL to implement features specific to RHdb. This is probably a typical situation. Agreement on a common interface, while desirable, is often difficult to achieve. If achieved, it is often an unsatisfactory compromise. Since CORBA deals only with interfaces and not with implementations, this problem is less significant than, for example, with flat file formats. It is easy to build servers supporting different IDL definitions. Another possibility is the mechanism of inheritance for CORBA interfaces. For example, it would be possible to define a specialized map interface for radiation hybrid maps, which inherits from the common map interface. The specialized map object itself could then give the additional information. However, this approach was not chosen here in order to keep score data and map objects separate.

Conclusion

CORBA is an accepted standard for distributed computing. Interoperability between independent data providers and applications is only possible if agreement on standard interfaces is achieved. The common map IDL is an attempt to set such a standard. The Maplet demonstrates the usefulness of the common IDL for the purpose of a map viewer. The IDL is designed to deal with large maps of >1000 markers. The Maplet focuses on a quick overview of large maps. The main functionality is preserved when the Maplet is restricted to the common IDL. Specialized features are accessible using an additional set of IDL definitions.

Acknowledgements

We wish to thank Graham Cameron, Jeroen Coppieters, Tomas Flores, Carsten Helgesen, Philip Lijnzaad, Jeremy Parsons, Martin Senger, Anastassia Spiridou and Janet
Thornton for useful comments. The research into common CORBA interfaces on distributed data sources is supported by grant BIO4-CT96-0346 of the European Union (DG XII).

References


