Virtual Observatories of the Future
ASP Conference Series, Vol. 3×10^8*, 2000*
R.J. Brunner, S.G. Djorgovski, and A.S. Szalay, eds.

Approaches to Federation of Astronomical Data

Roy Williams

Center for Advanced Computing Research, California Institute of Technology, Pasadena, CA, 91125

Abstract.
We discuss some approaches to rich semantic interoperation of web-based services, so that a user of the Virtual Observatory can create a distributed network of services to read data from archive, calibrate it, compute with it, fuse it to their data, estimate query execution cost, and many others. Each of these services can be maintained by different people and connected by standard protocols. We point out the need for interoperating the FITS and XML ways of representing structured information, and the need for standard semantics and representation of generic data objects. We consider Capability documents as a way to interoperate remote services, as well as concrete implementations of these from the the geography and business communities.

1. Remote Data

This paper is about working with remote data. First let us define the idea of a *service*, by which we mean a program (perhaps a web server), that is listening on a socket for requests, then responding. The Yahoo search engine or an FTP server are both services. We will define a *remote service* to be one where the (human) data client has never met the curator of the archive, and they may speak no language in common. Remoteness is not about geographical distance, but about relying on standards: both client and server must adhere to standard protocols for request and response before the remote data service can be used.

Suppose an astronomer has been told about a catalog of astronomical objects that she finds interesting, and wishes to compare it with a catalog she already has. Let us consider three ways in which this can happen: the past, the present, and the (Virtual Observatory enabled) future.

Past: In the old days, this would mean a trip to the library to find out who has the data, writing a letter to him requesting a copy of a tape, followed by a wait of weeks, then hours of software installation, extraction of the required data, custom code for changing the coordinate system, and more custom code to convert to the required file format, then creating the "payload" code that does the client's actual scientific analysis.

Present: These days we imagine these difficulties to be almost gone. Those who own data are embarrassed not to have put it on the web; also we have email and 100 Megabit/second connections, so there is no tedious waiting for the mailman to bring the tape. But in fact, most of the stumbling blocks to data federation are still present. The data owner has put a big text file at an FTP site and told his friends where it is. The client uses a web search engine or email to find the FTP site, then downloads the text file. She looks at the top of the big table, and sees columns called "RA1" and "RA2". An hour later, she realizes that the 1 and 2 are footnotes, and at the bottom of the table is an explanation that they are different equinoxes. Still there is lots of custom code, or the use of tools like Matlab or Excel or IDL to bring about the cross-comparison.

Virtual Observatory Future: Let us suppose that the data owner has put the data not just "on the web", but done so with it Virtual Observatory Compliance. This means that the semantic meaning of the data is exposed, and not only to the sharp intelligence of a human, but also to the dim wit of a computer. If a pair of numbers represents a position in the sky, the computer knows that they can be converted to other coordinate systems, and can do this silently. VO Compliant data services will be registered with an information service – like Napster does for music files – although we would expect it to be distributed, more like the Gnutella music service. Thus it will be much easier to find relevant data. When a VO-compliant client connects to a VO-compliant server, there is a conversation about their capabilities that the humans need not worry about. Our astronomer client will use a shrink-wrapped catalog-comparison service, connecting the remote catalog and her own catalog, receiving a data object which is the result, complete with provenance data and a way to cite the result in publications.

Thus the VO will be an example of a "semantic web" (Berners-Lee, 1998), a web of not just data, but semantically meaningful content. Without being able to do this, much astronomy data will remain effectively inaccessible for meaningful research just because the science community will not be able to access, manage and manipulate all the available data.

Services could be connected together by a user with a "modules and pipes" model of component computing, each module representing a service that is remote, where the user has not met the curator. As soon as the module is brought on to the desktop, its capability document would be fetched, so that decisions can be made about how to connect it to other modules/services. The user could decide that an output from one service (for example "*g_magnitude*") should be used as the input to another service (perhaps "*mag*"). The computer checks that the data-types are compatible, and makes any necessary conversion of physical units, then the human decides that these quantities are semantically equivalent.

1.1. Data Federation

Data federation (Williams *et al.* 1999) is simply the use of multiple data sources to create knowledge, for example visual identification of radio objects is federation of different wavelengths; identifying variable stars is federation over different times. Given two catalogs, it is often interesting to find the set-wise intersection (find the same physical objects represented in both catalogs). In this new joined catalog, there is more data with each object, more discrimination from others, a better chance to find the rare objects and see the trends and clusters.

Federation of data is also a nonlinear effect: new knowledge can be created from the fusion of datasets that could not be seen in the isolated data. This new knowledge does not require a rocket launch or a telescope, indeed it is at very little cost. When the semantic web makes it easy to federate data sources, it can be done easily, checking an unlikely intuitive hunch, or just looking for interesting things.

2. Structured Information

2.1. XML for Structured Information

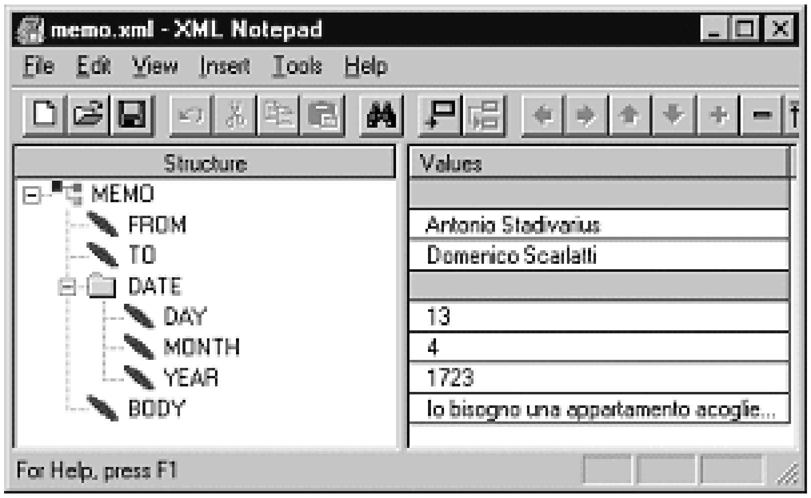
XML is a "file format for creating file formats", and is rapidly becoming an unassailable standard across the web. There is no doubt that it will become one of the cornerstone technologies of the Virtual Observatory. For a general introduction, go to any bookstore or visit xml.com or xml.org on the web.

XML looks superficially like HTML, in that it has both control elements and text. A date in the recent past might be represented in HTML as `<i>`April 12, 1997`</i>`, where the surrounding tag `<i>` means that the text should be in italic. An English-speaking human recognizes this as a date, but computers and non-English speakers may not. In an XML version of the same data, the date might be written:

```
<date>
<day>12</day>
<month>4</month>
<year>1997</year>
</date>
```

Now we have structured information. There are many tools that can display and edit such data. Such tools can be used to automatically check the schema of the document – for example, a memo can only have one sender, but can have many receivers, or that in a date, the day, month and year cannot be negative.

An XML representation of a date is more flexible than just a text string. A suitably informed computer can read and understand this date, doing such useful things as sorting documents in order, making histograms of the number of documents received in each of the last 12 months. In rendering the date for human consumption, it could write Month/Day/Year for Americans, and Day/Month/Year for the rest of the world, or substitute locale-specific month names for the numbers.



Structure	Values
MEMO	
FROM	Antonio Stadivarius
TO	Domenico Scarlatti
DATE	
DAY	13
MONTH	4
YEAR	1723
BODY	Io bisogno una appartamento acoglie...

Figure 1. A memo in XML, rendered with the XML notepad application. The tree structure of elements and text is clear.

XML is an excellent vehicle for expressing documents and metadata, and is now a powerful and universal web standard. However, it is not good at expressing bulk binary data; this is usually done with a link to a file or URL, or it can be converted to text with Base64 or similar.

Obviously if documents are exchanged, both sender and receiver should agree to the same standard. There are several examples of astronomical XML standards emerging, including Astores (Accomazzi *et al.* 1999) for catalog data and AML (Astronomical Markup Language; see, Guillaume 1998) for bibliographic and other information.

2.2. FITS and XML

The bulk of this paper is about ways of exchanging structured, semantically meaningful information, about how publishers of such information can advertise themselves, how clients can automatically configure an information transfer or initiate a remote procedure call. Such mechanisms will go a long way to creating the semantic web with will empower the Virtual Observatory.

But first we can discuss another, more human side to the story, a side that is less well-developed. Astronomers are further along than many fields of science in that they have a universal standard for structured information: FITS. This is a way to attach keyword-value pairs to binary information, originally for images, now extended to tables and other types of data. The development of XML followed a similar path: first it was for human-readable documents, the emphasis being on the separation of style from content in displaying such documents. However, XML has now become a generic way to represent structured information of many abstract kinds, and is becoming the universal language for everyone – with the possible exception of astronomers, who invented structured information first!

In many ways FITS and XML are equivalent. Users quickly realize that the challenge in using these for data exchange is not one of finding processing software, but in agreeing on the meaning of the data structure (the "FITS headers"). There has been a lot of progress in creating standards using FITS as a vehicle, and those who are mandarins of the Virtual Observatory should be very careful not to tyrannize or stifle such grass-roots efforts, but at the same time to choose and discriminate between competing emerging standards.

One project that will advance the VO considerably is a software toolbox to convert back and forth between FITS and XML. This will encourage cross-fertilization between the worlds of business and astronomy. In business, they are considering how to add bulk binary data to XML, and in astronomy, they are trying to get beyond the sometimes unintelligible 8-character names for FITS keywords.

2.3. XSIL: Extensible Scientific Interchange Language

XSIL (Williams 2000) is an XML dialect for common scientific datatypes. It defines a set of basic data objects – Parameter, Array, Table, data Stream, Time, and so on, and is designed for extensibility.

There are extension mechanisms so that people can build their own specialized data objects. For example, XSIL provides base objects Param (parameter) and Array. These might be combined to make TimeSeries (a one-dimensional array plus a parameter StartTime and EndTime). That object could then be extended in turn to make ObservationTimeSeries (*e.g.*, by adding more data about what instrument made the data).

XSIL comes with a Java parser and a browser to read it. If there is an element in the XML side, the Java code looks for certain Java classes of the same name to handle it, view it and edit it. Thus the browser is extensible in parallel with the XML.

XSIL can be used for a complete dataset (all data in XML), or it can serve as metadata, pointing to local or remote data, which can be URL or file, encoded or endian or plain text. Remote data is only read on demand,

and stored in a memory-efficient way. The browser uses Java Swing and the graphing and table components from KL Group (www.klgroup.com).

2.4. More Standards Needed

In addition to the standards mentioned above, one of the tasks of the Virtual Observatory is to agree on formats (XML or FITS) to represent a broader range of semantic objects. Many of these can come from other fields, the computer scientists and business are building these now.

- Document, Published article, Preprint, Person: we must borrow from the Digital Library community, as well as using existing standards such as AML.
- Table, Link, Parameter, Array, Image: if these basic objects are well-defined and implemented, we can build with them and reuse the software.
- Message, Exception report, Service capability, Program: we need to think sharply about the meaning of these things and the contexts in which they might be used, so that we can exchange them in a meaningful way.
- An astronomical object (*e.g.*, star) is distinct from an Observation of that object. Does every object have a position in the sky and a magnitude? What about large objects such as molecular clouds, and moving objects?
- Groups of objects, Extensions of object, Object handler. These "meta-objects" are the natural next thoughts. Once I have a table, I want a set of tables, the code to deal with tables, an so on.

The ISAIA project (Hanisch *et al.* 2000) is a wide collaboration to make a hierarchy of such semantic standards for astronomy.

3. The Semantic Web

There are many astronomical data services available today, but most of them assume that a human is using the service, not a computer. There is an idiosyncratic form to fill in, and the results come back in a nicely-colored HTML table. Such data cannot be read in any meaningful sense by a computer, making it difficult to federate data services.

What we envision in the Virtual Observatory is a network of services, each feeding data to another, perhaps with very large quantities of data. The services may be independently curated and managed, and only when a small, valuable piece of knowledge comes out is it presented to a human.

Alternatively, bulk data is delivered to a user's program, but it may be highly processed already, and fused with other data products.

For example, in the future VO we could combine several services like this:

- A yellow-pages service provides locations of necessary data and computing services.
- A monitoring agent can keep track of a long-running computation, allowing the human client to periodically check in, monitor, and steer the computations, while retaining diagnostic and logging information from the services being coordinated.
- A storage service can handle many Gigabytes for a long time for those with appropriate authentication.
- A crossmatch service can take multiple input catalogs, possibly unordered, and matching criteria, then create a "fuzzy join" on the catalogs, based on the criteria.
- Query estimation services can consider the bulk data transfers and computation that is suggested, and give estimates of resource consumption, both before and during the computation.
- A compute service can take a stream of data objects (*e.g.*, catalog entries), and route the stream to multiple slave processors that do pattern-matching, then collect back the results as an output stream.
- A raw data archive may have some data on tape, some on disk, some at remote locations, but it can respond to queries with a stream of data objects delivered at uniform fast rate.
- A compute service may calibrate the raw data on-the-fly, using a subsidiary database to get the calibration coefficients.

Services could then be connected together by a user with a "modules and pipes" model of component computing, each module representing a service that is remote, where the user has not met the curator. As soon as the module is brought on to the desktop, its capability document would be fetched, so that decisions can be made about how to connect it to other module/services. The user could decide that an output from one service (for example "*g_magnitude*") should be used as the input to another service (perhaps "*mag*"). The computer checks that the data-types are compatible, the human decides that these quantities are semantically equivalent.

In the following sections, we describe some ideas that will be used in the geographic community, and in the business world, to implement the semantic web in the next few years.

4. Capability Documents

There is a great deal of astronomical data already available on the web. Let us consider an imaginary catalog of some kind of interesting stars, together with a web-based service to find out what catalog members are close to a given point in the sky. The builder of the service might have decided to use HTTP GET protocol, so that a request might look like this:

http://www.blahblah.edu/getdata?
request=table&RA=185.0&Dec=23.0&Radius=0.5

and it produces a response like this:

```
183.22   22.6   17.1   16.8   17.3
186.13   22.9   16.3   15.9   16.4
```

A human could probably figure out that *RA* is right ascension and *Dec* is declination, and that *radius* is a search area. The human would be especially helped by seeing the form that came with the website, and reading the attached documentation. The response shown here is two objects from the catalog, each with RA and Dec and three magnitudes in different filters. However, none of this is clear without a human to read and understand. It would be tedious to manually configure the connection of a dozen or so services by examining the request and response semantics of each one.

In this section we discuss the idea of a **capability document**, where a service responds to a standard request with a document describing what the service can do. For example, to get the capabilities for the service above, we would submit this request:

http://www.blahblah.edu/getdata?request=capabilities

The capability document may contain these fields:

- The name of this service, with a title, abstract, contact information, and a link to a web page that describes it.
- The version number of the archive server software. When a request comes in with a different version number from that which the archive server supports, a negotiation could take place to determine a mutually acceptable version of the communication protocol.
- Acceptable distributed-computing protocols for requests and responses (*e.g.*, SOAP, Nexus, HTTP, RMI, Corba, *etc.*), and specific information about servers, port numbers, *etc.* While we assume that the

capability document itself is obtained by HTTP, there is no reason why the request and response should be transmitted this way.

- Available output formats, so that for example the keyword "format" can be used by the client. Then, for example, the request can contain "format=csv" if comma-separated values are wanted. A binary stream might be requested as "encoding=bigendian&timeout=900" for data stored on a slow medium like a tape robot. Ways of specifying binary streams are specified in the XSIL language (Williams 2000).

- Disposition of error, diagnostic and debugging information may be returned. The default is a plain text message in place of the expected response, but more sophisticated mechanisms may also be available.

- It may be that a service is only available to certain authenticated users, or that payment must be made. A section of the capability document defines the ways in which the server expects to be given this information.

Part of the capability document could be a way of building a user-interface, so that a human can frame a request. This would be an HTML form that can be used as part of another page. (Unfortunately, HTML is a much looser language than XML, so that its strict version, XHTML should be used to ensure that the capability document itself is well formed. See http://www.w3.org/TR/xhtml1/ for more details.)

4.1. Hierarchies of Capability: Subject Index

Once we have a way to describe a single data service, there is the possibility to describe collections of services and links between them. A simple way to do this is by presenting HTML web pages to a human user, who can navigate to a service that she wants to use. But there are advantages to having a machine parse a list of services. Therefore we have chosen to allow a capability document to describe lists of other capabilities documents as well as a data service. In this way we can organize and crosslink data services.

When a link is included in a capability document, it includes a "subject" element. Once we have the name of a subject we can append it to the name of the previous service, then ask for subject-specific information. For example, if we have been informed that subject-specific information is available on gamma ray astronomy, with the subject name *gamma_ray_astronomy*, then we can ask for specific capabilities as follows:

http://www.blahblah.edu/gamma_ray_astronomy?request=capabilities

Thus we can reflect a directory hierarchy of services on the server with a service hierarchy that is visible to the client.

4.2. Capability for the Request

In the example above, the capability document explains to a human that "RA" means "right ascension", a coordinate value on the celestial sphere. To a computer, an explanation has also been given, that this is a floating-point number between 0 and 360, and the text "degrees" should be written next to it. Similarly for "Dec".

The capability document might also explain that other coordinate systems are available, and that this server can respond to galactic coordinates as well as equatorial, that it can understand different notations. Thus there are alternate versions of the request, so that this would also be acceptable:

http://www.blahblah.edu/getdata?
request=table&Glon=124.3&Glat=19.3

Each service has associated a list of keywords that may be used in a request. Each keyword may have other information associated with it, including

- Information about the allowed values for this keyword (mathematically, it's domain),
- Default values if it is omitted,
- Short and long descriptions of the semantic meaning,
- Units (if any) implied for the value,

For the example above, the keyword "RA" has range 0 to 360 degrees, the title is "Right Ascension", and the explanation is "Right ascension in the J2000 coordinate system, see `http://...` for further details".

One further type of interaction we might imagine is making SQL queries to the service, or perhaps by sending some other script or code to be executed. Small pieces of such script can be sent as strings as part of the request, and the capability document can indicate that this is allowed.

Keywords may be arbitrarily grouped, so that we can specify which combinations of keywords can be used for a request. For example, "RA" and "Dec" might form a group, and "Glon" and "Glat" another. Then we can use either pair in a request, but we cannot mix keywords from different groups. Keyword groups may also have a domain specification: if a catalog covers a small domain of the sky, then this domain is naturally expressed on the joint object (RA, Dec) rather than on each coordinate independently.

4.3. Capability for the Response

We have explained how the capability document can define a service and how to make a request. Now we shall consider the explanation of what will be in the response, and what data formats are available. Let us think of the response as a table of data, with each row a data record, and here we consider how the capability document defines the headers for the table. Let us call each part of the data record a "column", to distinguish it from "keyword" that we have used while describing the service request.

Each column is defined by:

- The name of this column, its title, it abstract, its "further info" link,
- The data type to be used for storing it (int, float, complex, string, *etc.*),
- Default value to be used in case a value could not be read by the client,
- The minimum or maximum of the values in this column,
- Semantic meaning of the column: name, title, abstract, URL link.

For the column in the example above that is the chart number, it might be defined as an integer between 1 and 20 (the number of charts), and it has no units.

We would like to have the possibility of embedding links into the response data, so that we can draw catalog information on image data, then hyperlink to more information about that object. For example the response to a request might be a list of galaxies, each with some numerical fields (brightness, color, *etc.*), but also a link to an image, or to bibliographic records. There could be an individual link for each galaxy, or there could be a "template" link that came with the metadata, that is to be combined with some ID number of the galaxy. For example, the template might be

http://www.blahblah.edu/further_info?ID=$ID

so that the response column whose name is "ID" is to be substituted in the relevant place.

4.4. OpenGIS Web Mapping Testbed

The Geographic Information Systems community (GIS) has already defined much of the capability document infrastructure described above, and implemented many services that use it. In the figure below is an example of a web browser showing a composite map of the English Channel, with different map layers coming from different servers. Each server in the list (top of three pull-down menus) has provided a capability document, describing the nature of the different map layers that it can provide. The user has selected several of these (key, to the right), and chosen an order for the layer stack. Here we see evaporation, built areas, railways, population centers, and coastline.

The OpenGIS initiative (www.opengis.org) has already defined the XML format of the capability document for the case of map layers being returned by a server.

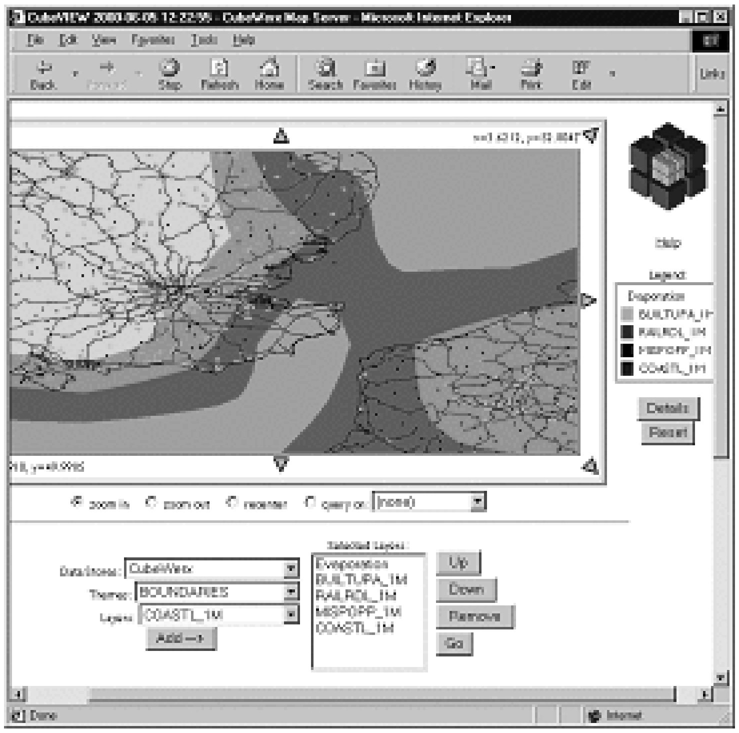


Figure 2. An example client in a web browser, showing London and France. Multiple layers are fused from multiple Open-GIS-compliant servers.

5. Business Initiatives

Business is also realizing the utility of interconnected web-based services. If a retailer has dispatched goods through a package service, he would like to show the customer the status of the package from his own web site, not to simply say "here is the tracking number, go look it up with the package company." Businesses that broker information have an obvious interest in a standard way for their computers to talk with those of their wholesalers. This "B2B" (Business to Business) market sector is growing strongly.

It is becoming clear that the universal vehicle for such communication is by XML-formatted messages delivered by the HTTP protocol. In this section, we summarize some of the initiatives from various companies and consortia that can create a business version of the semantic web.

Table 1 summarizes the protocol stack. We are becoming familiar with the idea of XML-formatted data objects carried on an HTTP protocol, itself carried reliably by TCP/IP.

UDDI	Registration and discovery
WSDL	Service description
SOAP	Remote objects and computing, who does what
XML	Structured information
HTTP	Identified file formats, *e.g.*, Text, JPEG
TCP/IP	Reliable transfer of byte streams

Table 1. A summary of the protocol stack.

In the following, we discuss the upper layers of the stack: SOAP to allow rich messages between web-client and web-server; WSDL to provide capability information for web services; and UDDI for publishing the existence of services. We should point out that the use of these protocols does not restrict data transfer to the HTTP protocol; however, we can use the flexibility of these protocols to decide how high-bandwidth communication can occur by some other mechanism, for example parallel FTP.

5.1. SOAP: Open Distributed Computing

SOAP (Simple Object Access Protocol; see, *e.g.*, Box *et al.* 2000) is an XML format that carries commands and objects between clients and servers, including who is being commanded by this message, and what reply is expected. While many web services today use a keyword-value combination as a request (such as the example above, with the *RA*, *Dec*, *radius* keywords), SOAP allows the use of a complex object as a request and response, both serialized in XML. There is also capacity for remote-procedure calls, leading to simple distributed computing, but without the complexity of CORBA or the language specificity of Java RMI.

Several prototype SOAP-based services are described at the Xmethods web site.

5.2. WSDL: Service Description

WSDL (Web Services Description Language) uses XML to describe network services or endpoints. It can describe services that use the HTTP GET and POST protocols (keyword-value set), and also services mediated by SOAP.

Requests and responses to a service are known collectively as messages. Each of these is a collection of types parts, for example "the variable alpha in the message is a floating-point number". An operation takes a message as input and produces one as output.

5.3. UDDI: Registration and Discovery

The UDDI initiative (Universal Description, Discovery and Integration of Business for the Web, www.uddi.org) was started by Microsoft, IBM, and Ariba, and many others. It provides a framework for the description and discovery of business services on the web. It does this by using distributed registries of services, and conventions for accessing that registry using SOAP.

There are three components to the UDDI specifications. White pages shows address and contact information for the service, and yellow pages categorizes the service within a hierarchy so it can be found easily. The green pages component defines the technical information about the service capabilities.

6. Conclusions

In addition to high-performance computing and high-performance networking, the Virtual Observatory will require a leap in semantic interoperability. While the web already provides interoperability between humans (mediated by computers), the VO will consist of services that interactively perform multiple steps on the user's behalf. These tasks may require one web service to call on other web services, coordinationg the steps like a traditional software program executes commands. The problem today is that integrating with other services remains difficult, because tools and common conventions for interoperation are lacking.

The astronomical community is well-versed in the art of making standard semantic data objects using FITS files, and they will benefit from further such standardization under the auspices of the VO. However, the range of information objects must be wider, taking such standards from the computer science community. Furthermore, astronomers must embrace XML in addition to FITS as a vehicle for structured information, thereby getting the best of both.

We are all expecting archive-based research to be a fourth arm of the scientific method, in addition to observation, theory, and simulation. For this to happen, it must be possible to connect apparently disparate ideas into a hypothesis and then make appropriate tests. Therefore, data services must be interoperable even when the the people involved have never met and nobody has ever thought of connecting these services. This is why standards are necessary. We do not know in advance what kinds of data will interoperate with what.

High bandwidth data exchange and high-performance computing come after the semantic web has been established. Once it is established that multiple servers can effectively interoperate and they have the necessary data, then the experimenter will be able to connect the services with high-performance data services. In the words of Kernighan, “First make it work, then make it fast”.

References

Berners-Lee, T. 1998, Semantic Web Road Map,
http://www.w3.org/DesignIssues/Semantic.html

Williams, R., Messina, P., Gagliardi, F., Darlington, J., Aloisio, G. 1999, Report of the European-United States Joint Workshop on Large Scientific Databases,
http://www.cacr.caltech.edu/euus/

Accomazzi, A., et. al. 1999, Describing Astronomical Catalogues and Query Results with XML,
http://vizier.u-strasbg.fr/doc/astrores.htx

Guillaume, D. 1998, Astronomical Markup Language,
http://monet.astro.uiuc.edu/ dguillau/these/

Williams, R. 2000, Extensible Scientific Interchange Language (XSIL),
http://www.cacr.caltech.edu/XSIL

Hanisch, R., McGlynn, T., Plante, R., King, J., White, R., Mazzarella, J. 2000, ISAIA: Interoperable Systems for Archival Information Access,
http://heasarc.gsfc.nasa.gov/isaia/

Box, D., et. al 2000, SOAP, Simple Object Access Protocol,
http://www.oasis-open.org/cover/soap.html,
http://xml.apache.org/soap,
http://msdn.microsoft.com/xml/general/soap_webserv.asp