Fig. 1

Fig. 2

Fig. 3

1. The GAVO Data Center

Markus Demleitner *(msdemlei@ari.uni-heidelberg.de)*

- A DC's role in the Virtual Observatory – what a DC is and how it communicates with the rest of the world.
- GAVO's approach – how we try to fill the DC's role and retain sanity.
- GAVO's data center and you – examples of data we serve and how you can contribute

(vgl. Fig. 1)

(vgl. Fig. 2)

(vgl. Fig. 3)

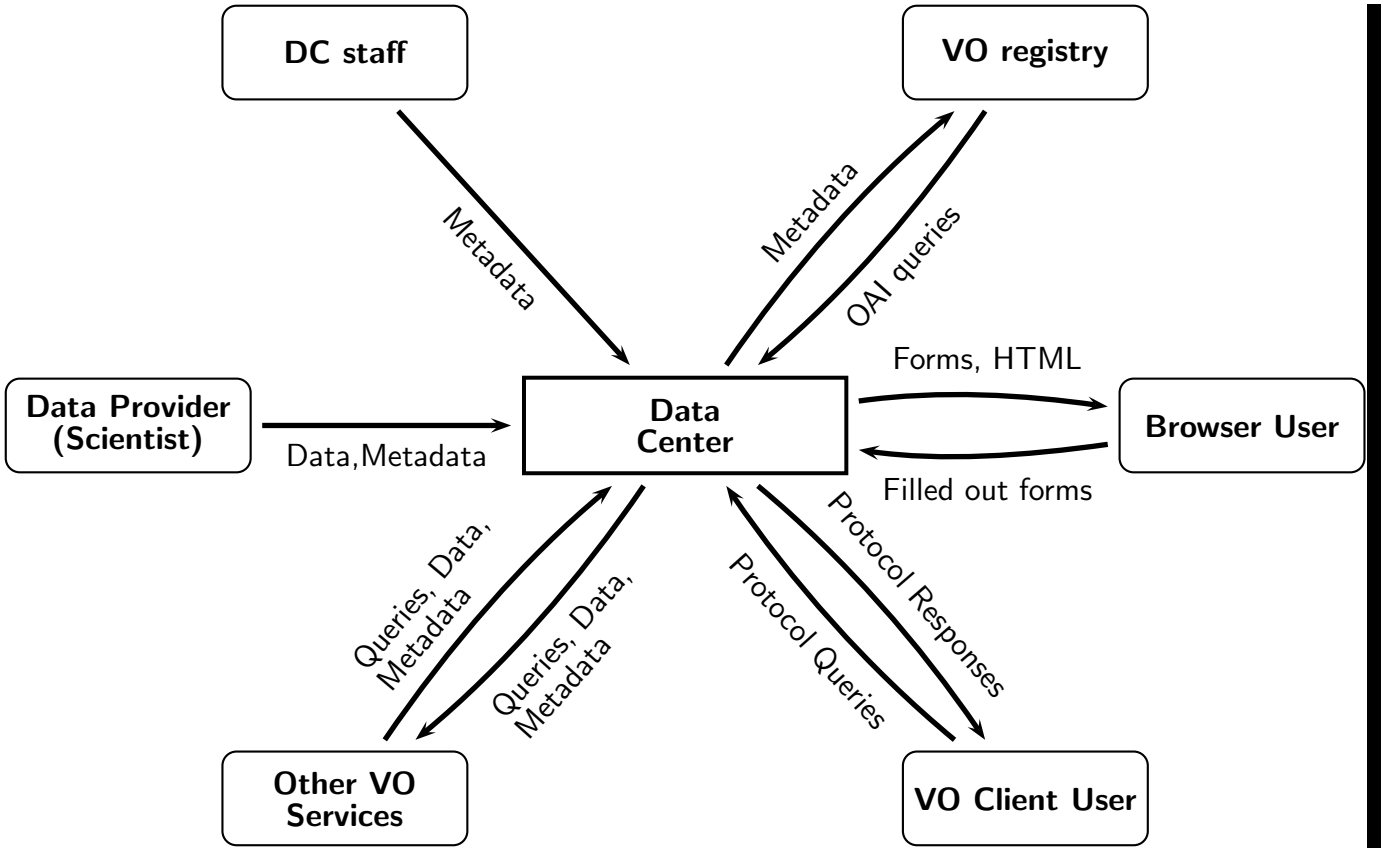

Fig. 4

2. A Data Center in the VO

(vgl. Fig. 4)

In this graph it looks like a DC is the spider in the VO's web. While one could draw similar pictures for all the other actors in the VO, a data center has to talk to quite a few parties:

1. The data providers will deliver their data in many forms and modes (text, FITS, fortran arrays, etc.; mail, ftp, HTTP uploads, etc.). They will also have to provide metadata.
2. The DC staff will provide additional metadata (e.g., on curation) and will make the DC software "ingest" the data coming in from the providers.
3. The registry queries a DC's "publishing registry" for what services it offers, and the DC has to respond with metadata in a well-defined format.
4. Most users still like to query services with their web browsers, and some services cannot meaningfully be published using defined VO protocols. So, the DC has to speak HTML.
5. Specialized VO clients use protocols like SCS, SIAP, or TAP to query data using IVOA-defined protocols. So, the DC will have to speak them, too.
6. Probably the most exciting operations of the VO are built on servers talking to each other on behalf of a user. Examples for this mode are cross matches using SCS URLs or VOSpace, and one can expect many more developments in this area as the VO becomes more established.

3. The GAVO DC's approach

In GAVO's DC, there are

- "resources", containing top-level metadata and prescriptions on how to use a particular data set,
- services belonging to resources, containing service-level metadata, a core and renderers,
- cores doing the actual work of querying or computing data and defining a low-level interface,
- renderers doing the protocol-level communication with the user.

Calling the top-level „thing" belonging to an incoming data set "resource" was a bad choice since too much is called that. Still, we want an entity containing common metadata (author, data description, source, etc) and holding the rest together.

Resource descriptors (RDs) contain table definitions. For the DC staff, there are ingestion recipes telling the software how to turn the data coming from the providers to the internal structured representation (i.e., as a rule a database table).

The purpose of having cores, renderers, and services is to be able to reuse service metadata for various output modes (e.g., form-based and SIAP). The renderers thus roughly correspond to capabilities in registry speak.

The separation between cores and services allows to abstract the raw core interfaces as necessary (e.g., for wrapping legacy non-networked applications).

4. Examples: Catalogues

To further illustrate what a DC's job is in GAVO's view, on this and the next few slides we mention a few examples of services we publish.

- PPMX – basically, as in VizieR, with both form-based and Simple Cone Search on-site. Of course, ADQL, cross-match, etc., are available as well, TAP soon to come.
- OH Masers – SCS, metadata and registry in the DC, custom form-based web service external.
- Magellanic Cloud extinction data – custom cone search (on fields).
- CARS – object catalogue with basic image and related data.

5. Examples: Image Collections

- Quasar lensing images – from various sources, partly with embargoes, partly with direct upload from La Silla, SIAP and custom.
- Plate scans from Heidelberg and Calar Alto – including image calibration, cutout processing to avoid having to deliver roughly a Gig a pop.
- Potsdam Carte du Ciel scans – external web service, SIAP and metadata on-site.

6. Examples: Funky Stuff

- APFS – wrapping legacy FORTRAN in fancy web and SOAP.
- Light pollution survey – upload and near real-time processing of distributed measurements.
- Inifinite lightcurve – client-side javascript to visualize lensing calculations.
- Dexter for your data – playing ADS for a data extraction applet.
- UCD resolver – wrapping a python library.

7. In Closing

If you are a data provider: Talk to us about getting your data into the VO.

If you are planning to offer VO services: Check out http://vo.uni-hd.de/odocs.

http://www.g-vo.org http://vo.uni-hd.de