User-focused evolution of VO service design

James Tocknell

data central

Why I’m giving this talk

- PhD student—background is fluid dynamics (i.e. not observational)
- Ran local “Coffee and Code” from ~2013-2020 (also Carpentries instructor)
  - Was first (and in many cases only) technical support person for Astro
- Went to local student events—friends with many local students/ECRs
- Meeting up again at ASA 2022 heard many issues related to “astronomy data services”
  - Usually resolvable by pointing in right direction
  - Some systems harder to use than need to be
- **Users don’t seem to understand what the VO is/what it’s for**

data central

What am I calling a “user”

- The human(s) interacting with the service
  - This makes it easier to distinguish between user and client (i.e. the tool that the user uses to interact with the service).
- There are different kinds of users in the IVOA ecosystem, with different skillsets:
  - User-Astronomers (will expand on later)
  - Small Observatories/Small Teams (mostly use existing, externally maintained codebases)
  - Large Observatories/Large Collaborations (mostly using internally developed codebases)
- NOTE: Above definitions imply that the large(r) groups dictate direction, so they need to make sure other group’s needs are actually met.

data central

Different kinds/skills of user-astronomer

- User-Astronomers vary from those using old ecosystems (e.g. IDL, PDL) to those pioneering the use of new languages (e.g. Julia, Rust?). We need to meet the needs of both!
- User-Astronomers use applications developed by Software Developers (e.g. TopCat, DS9, Aladin, DAS), but also write their own scripts
  - Aside: Many users would be better served by using applications rather than their own scripts, but there seems to be a culture against using them?
- User-Astronomers use different languages to Service Developers
  - e.g. use of R is common at some locations, I'm not aware of it being used in IVOA services
- **Takeaway: Coupling (User-Astronomer) recommendations to specific languages only reduces the available userbase, divides the ecosystem**

data central

Recommendations imply services targeted at different groups

- **"Data Access"** (e.g. TAP, SCS, SIA, SSA) are directly called by User-Astronomers—these recommendations need to be at least partially understandable by User-Astronomers, and design choices should reflect that
- **"IPC"** (e.g. SAMP, VOSpace, the Registry) may be used indirectly by User-Astronomers, but they do not need to understand the details—these recommendations have more freedom, but need to consider that clients must be able to run on user's devices (and people do not upgrade…)
- **"Inter-service communication"** (e.g. CDP, GMS, VOEvent Transport Protocol) are very unlikely to be directly used by User-Astronomers, and so should be guided by the needs of the developers (both Small Observatories/Small Teams and Large Observatories/Large Collaborations)

data central

Recommendation documents focus on server-implementers (not users or deployers)

- User-Astronomers are not going to read recommendations (lack technical background and ecosystem understanding), are interested in specific instances/datasets
- Small Observatories will use existing implementations, but don't know which versions are implemented (or how the implementation extends the recommendations)
- **Takeaway: We need a better way to introduce services to users, without requiring reading many recommendations.**

data central

Recent major user issues

- Astroquery limits sync TAP queries to 2000 rows (came up at busy week)
  - Done via modifying ADQL query (opaque to users!)
  - Suspected underlying reason to is avoid overwhelming services (or loading too much into memory)?
  - Better interface would be to handle async internally, and block at the user's end (e.g. library handles sensible polling strategy with jitter to avoid thundering herd)
  - **Takeaway: Error loudly—don't make users have to chase down unexpected results**
- “All services are standard, but some are more standard than others”
  - Origin of DAS (Data Aggregation Service) is from a PhD student trying to find FRBs
  - ‘DAS tries and adds distance information for “cone search” TAP queries. Distance function implementation is very heterogeneous and would confuse many user astronomers.’
  - **Takeaway: We need to be more explicit about what can be relied on, and what cannot (and what alternative options there are).**

data central

Authentication

- Switch away from certificates to tokens should make things easier for everyone
  - Less issues with TLS termination etc.
- Requiring User-Astronomers to implement OAuth is an issue.
  - Example: ESO Archive example code would result in client IDs/secrets appearing on GitHub due to hardcoding
- Allowing users to use API tokens which are limited in what they can do is much safer and easier for users
  - See PyPI newish token system for how they transitioned away from username/password pairs.

data central

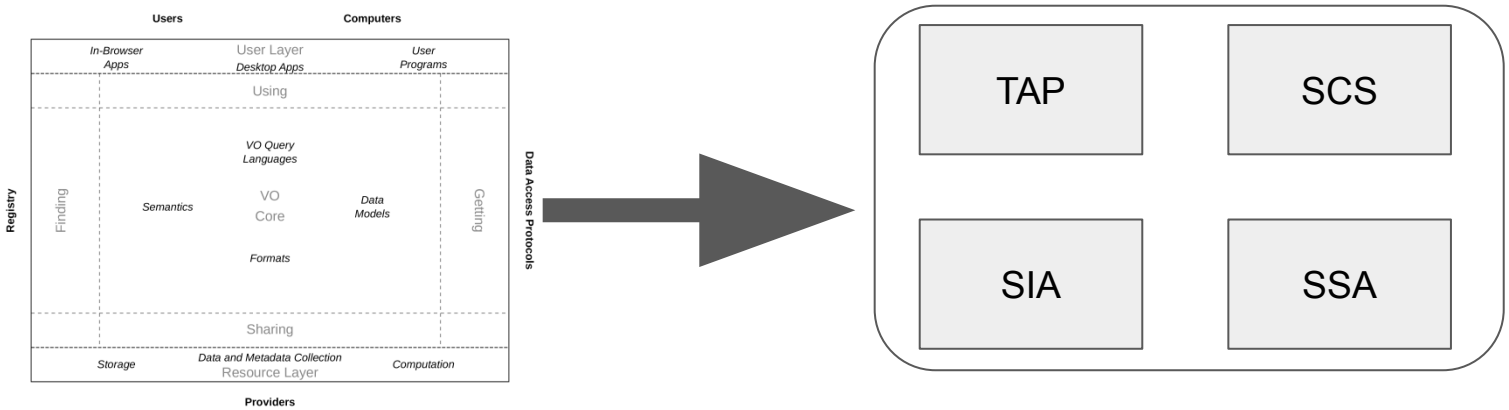
Some technical suggestions/ideas

- Don't require digging into the payload to work out that there are issues with request
  - Makes it easier to transition between formats—users will keep using old code when new code doesn’t meet need
  - Formats I’ve seen users want are CSV and FITS (so where are errors returned?)
  - Does not imply that having errors flagged in payload should not be done—more chances for user to spot error
- Assume users will mess up—try to reduce the blast radius
  - Users will paste passwords into code
- Remember target users:
  - Things like OpenAPI and GraphQL may help developers, but add barriers to User-Astronomer usage—curl+awk should be able to do most tasks.

data central

Some broad suggestions going forward

- Explicitly non-normative resource explaining the service ecosystem to semi-naive user-astronomers
  - Editable by anyone, e.g. on GitHub
  - Not tied to recommendation process
  - Step before architecture note



data central

Some broad suggestions going forward

- An IVOA "implementer's and operator's guide" covering the setup/operation of specific services, useful advice etc.
  - Include sections on both clients and servers
  - Include references to existing talks like Tom Donaldson's talk on ADQL and Firewall SQL-Injection Detection
  - Editable by anyone, e.g. on GitHub
  - Not tied to recommendation process
  - Could be in same space as ecosystem overview

data central

Some broad suggestions going forward

- Take ideas around service development process from the Carpentries' lesson development process
  - Create “user profiles” based on the learner profiles the Carpentries'—relate to different kinds of user
  - Explicitly refer to user profiles to build user stories in new recommendations (where there is a clear user)

data central

Summary

- Create “implementer's and operator's guide” and service overview
- Be explicit as to whether a service is/will be “Data Access” “IPC” or “Inter-service”
  - This should drive design decisions, not what developers prefer
  - “Inter-service” is where we should look at new technologies (e.g. for large scale data transport), not “Data Access”
- In migration plans, we’re more likely to be held up users than services
  - User-Astronomers will keep using unmaintained applications, libraries and languages

Questions? Ideas? Comments?

data central