Fig. 1

1. Capabilities' Roles

Markus Demleitner
msdemlei@ari.uni-heidelberg.de

- What's broken?
- How did it happen?
- Structuralism to the rescue
- DALIInterface? ComplexInterface?
- Volunteers wanted!

(cf. Fig. 1)

2. What's broken?

TAP 1.1 for a while wanted to register like:

```
capability standardID="ivo://ivoa.net/std/TAP#async-1.1">
  <interface xsi:type="vs:ParamHTTP" role="std" version="1.1"> ...
  <interface xsi:type="vs:ParamHTTP" role="std" version="1.1"> ...
    <securityMethod standardID="ivo://ivoa.net/sso#BasicAA"/>
  </interface>
  <interface xsi:type="vs:ParamHTTP" role="std" version="1.1"> ...
    <securityMethod standardID="ivo://ivoa.net/sso#tls-with-certificate"/>
  </interface>
</capability>

<capability standardID="ivo://ivoa.net/std/TAP#sync-1.1">
  <interface xsi:type="vs:ParamHTTP" role="std" version="1.1"> ...
  <interface xsi:type="vs:ParamHTTP" role="std" version="1.1"> ...
    <securityMethod standardID="ivo://ivoa.net/sso#BasicAA"/>
  </interface>
  <interface xsi:type="vs:ParamHTTP" role="std" version="1.1"> ...
    <securityMethod standardID="ivo://ivoa.net/sso#tls-with-certificate"/>
  </interface>
```

So, there were different capabilties for sync and async, each having several interfaces differing by security method.

A client wanting to operate such a service had to puzzle together interfaces from the different capabilties by securityMethod (and then again by mirrorURL).

3. Why broken?

- Assembly of the client interface ("bundle": sync, async, table, capabilities, example) becomes complex.
- Complex registry record, easy to get wrong.
- If there's both SSA and TAP (say): what's the tables endpoint about?
- If there's mirrors or different endpoints: what's the availability endpoint about?
- If there's both TAP and Datalink in a service: what's the examples endpoint about?

4. How did it happen?

As usual, each individual step made sense.

VOSI in the 2000s just wanted to associate its endpoints with services without changing VOResource.

And it worked fine when we had 1:1:1:1 services:

1 Resource
  :1 (interesting) capability
    :1 (standard) interface
      :1 access URL.

It fails now that we're in an n:m:s:p world. See the caproles note for what that world is made of.

5. Structuralism

At its root, structuralism is when you're trying to figure things out by swapping things in and out ("paradigmatic") and seeing what happens.

Classic example in phonetics:

fool $[\mathrm{fuː l}] \neq$ rule $[\mathrm{ruː l}]$, hence $[\mathrm{f}]$ and $[\mathrm{r}]$ belong to different phonemes in international English.

tomato $[\mathrm{təˈmɑːtəʊ}] =$ tomato $[\mathrm{təˈmeɪˌːtəʊ}]$, hence $[\mathrm{ɑː}]$ and $[\mathrm{eɪˌ}]$ aren't.

Here, basically, you look if a spoken word's meaning changes if you change a single sound.

There's limits to this, but it's a fine tool nonetheless.

6. VO Structuralism

To determine the level at which an endpoint E should be described,

Starting with capability, find the highest level in the hierarchy

at which swapping instances

will not change E's response.

The right level to attach the endpoint to is one level up then.

Let's try it for VOSI tables

Swap an SSAP with a TAP/Obscore capability: The table structure changes, so VOSI tables content has to change. So, go one level down.

Swap an authenticated and a non-authenticated interface: authenticated interfaces might show extra columns. So, go one level down.

Swap the main site with a mirror: both must return identical results, so table structure is preserved, so the endpoint content does not change.

Go one level up: VOSI tables should sit at the interface level.

Let's try it for VOSI availability

Swap an SSAP with a TAP/Obscore capability: TAP might be down while SSAP still works because the more complex TAP software stack might be under maintenance while simple SSAP churns on.

Swap auth vs. open interface: if the (federated, say) auth infrastructure is down, the auth interface may be down while the open interface churns on.

Swap main site with a mirror: main site may be down while mirror churns on: VOSI availability must vary at the access URL level.

For reference, here's the VOResource service stack again:

Resource

Capability

Interface

Access URL (Mirror URL)

7. Purity of Essence

Capabilties have been/are in use for:

1. Equivalent but different access modes (e.g., obscore vs. SSAP)
2. Different functions on a data collection (e.g., SSAP plus datalink)
3. Building blocks to be bundled to form workable client interfaces (e.g., sync and async, VOSpace)

I'd like to restrict this to (1). Reasons: (2) could equivalently be modelled through two resources and a relationship between them, which *probably* has better overall semantics; also, once (2)'s gone clients can assume all capabilities they've discovered give them approximately the same sort of access. (3) has the problems discussed on the "Why broken" slide.

8. DALIInterface

For DALI-type client interfaces (VOSI, sync, async, examples), we'd like to hang the endpoints off of interface. That needs a new interface type.

Plan: VODataService `vs:DALIInterface` like this:

```
<capability standardID="ivo://ivoa.net/TAP">
  <interface xsi:type="vs:DALIInterface">
    <accessURL>http://example.org/tap</accessURL>
    <mirrorURL>http://example.com/tap</mirrorURL>
    <endpoint>async</endpoint>
    <endpoint>sync</endpoint>
    <endpoint>tables</endpoint>
    <endpoint>capabilities</endpoint>
    <endpoint>examples</endpoint>
  </interface>
</capability>
```

. . . or would you prefer the name `vs:ComplexInterface`?

9. Cleaning up the mess

- VODataService – needs DALIInterface
- RegTAP – discovery on role=std (done)
- TAPRegExt – introduce DALIInterface, transition from ParamHTTP
- VOSI – no longer talk about capabilities
- VOResource – capabilitiesURL (this would stand in for the current VOSI capabilities capabilities [no error] which are structuralism-ok but aren't pretty because what's discussed on the Purity of Essence slide)? Have monitoredURL type for VOSI availability? Something like that we'd need if we want to have sane attachment of VOSI availability endpoints; but then perhaps we should just give recipes for how clients can figure out themselves what's up and what's not?
- DALI – rework sect. 2
- Datalink – strike VOSI caps
- SIAP2 – strike VOSI caps
- SimpleDALRegExt – have extra cap for WebBrowser

And then there's VOSpace that's using capabilities in a building-block way. I *think* the building blocks should migrate to an interface, too – but then I really don't understand enough of VOSpace to be sure.

10. The good news

1. Little or no existing (interoperable) code will be broken.
2. It's certainly less work than cube access.

Want to help out? Talk to me!