From the discussion with contributions from Stephen Burke and Akos Frohner

Use Cases

You might want a web tool which could present information for a given VO without having a credential for that VO, or a user might want to collect information to be used in a subsequent step where a different credential would be in use.

Another case would be using a service indirectly via another service which has more/ different rights to the user themselves. A current example would be that myproxies have an access control rule which specifies which WMSes (and indeed FTSes) can use them for proxy renewal - that right is keyed to the DN of the WMS and not related to the user.Hence having picked a WMS you need to find a myproxy which trusts it (potentially as well as trusting you, but at the moment that isn't restricted). With the new service providers you can do that with direct LDAP queries, but it would be nice if SD would support it. Similarly you could imagine a situation in which the FTS was authorised to write to an SE, but the user wasn't authorised to write directly, to allow datarates to be managed.

Authz might be time-related, e.g. you can only use a service out of office hours (condor pool?), but you might want to discover it anyway.

I quite often use lcg-infosites to query for resources for VOs I'm not a member of if I'm trying to debug a problem for someone else - it would be much less useful without the VO option.

Thoughts on authz

Thinking some more: you talk about alternative security models, but I think your conceptualisation is pretty much bound to the concept of a VOMS proxy. Firstly, it's not obvious that there is a local credential at all. A trivial example would be a service to which you type a password; a more complex one could be a myproxy in password mode, where you tell the service how to retrieve the credential but you don't have it locally yourself. Or there could be some indirect context, e.g. the service deduces your VO (or equivalent) from your ip address using rules you don't have access to.

Secondly, even if there is a credential there's no reason to assume that you can extract the relevant information from it. An obvious example is an ssh key; that contains no information at all about how it can be used. Or there may be information, but not in a form that would let you do the matching - a kerberos ticket is probably in that category. Or even in our standard grid world you may have a plain globus proxy which gets mapped to a VO via a grid map file (or gets rejected),
but from which you can't extract the VO.

I think the basic point I would make is that service discovery is not an authz *decision*, it's just a selection on something you hope will be fairly highly correlated with the decision the service will make if you ask it. What the service will publish are not the real internal authz rules (e.g. gridmap file, ban list, ...) but some kind of edited summary, constrained by the rules of GLUE (or other info model) syntax, privacy laws, security considerations and pragmatics - at the start of EDG we published every allowed user DN, but now we don't because a) it's illegal and b) it's impractical with 10,000 users. So it's perfectly possible that the rule you match against in the info system has no direct relation at all to your actual authorisation credential/token/whatever, but will correlate with the eventual authz decision in a probably simplified way via something the user or the software is likely to know - e.g. maybe the info system publishes "LCG" and people "just know" that that collectively means the atlas, cms, lhcb and alice VOs. The model where we have a local VOMS proxy from which you can easily extract the information (primary VO name, primary FQAN) which are directly used in the real authz decision, and which can be and are directly published in the info system, is in fact an extremely special case, albeit the one we're currently most familiar with!

Other points

There was a lot of discussion in the past in the OGF security groups to represent the authorization information in some simple way.

In our client tools we use gridsite to parse the VOMS FQANs, however I can imaging one using the VOMS libraries or even a SAML assertion from Sibboleth to gather the attributes.

SD may need to scan tens of thousands of services, and authz policy engines may well be rather slow

In the current EGEE service discovery API the authorization restriction is an explicit parameter, the service discovery does not pick it up from the security context. This simplifies dependencies and allows more flexibility on the actual security mechanism that you might want to use.

Having decided on something you should then find it easier to see what needs to go in the API to reproduce the same behaviour, although it will depend on how generic/ extensible you want to be. For example, for the current GLUE practice and likely extensions the best answer would probably be to pass a list of strings and an optional matching function, which would default to string equality - then you just match the list of strings against the list of ACBRs and keep the result if anything matches. A simple VO match would correspond to a string of "VO:atlas" and the default matching rule, which isn't especially taxing on the user.

Why not only one or at maximum a list of strings (i.e. VOMS FQANs)?