# The next generation of Virtual Organisations in UNICORE

*Krzysztof Benedyczak, Piotr Bała*
*ICM University of Warsow*

# Outline

- Virtual Organisations revisited
  - classification
- Current state of 'art'
  - VOMS/gLite
  - UNICORE
- Problems and goals
- Roadmap towards a real state of art

# Virtual Organizations revisited

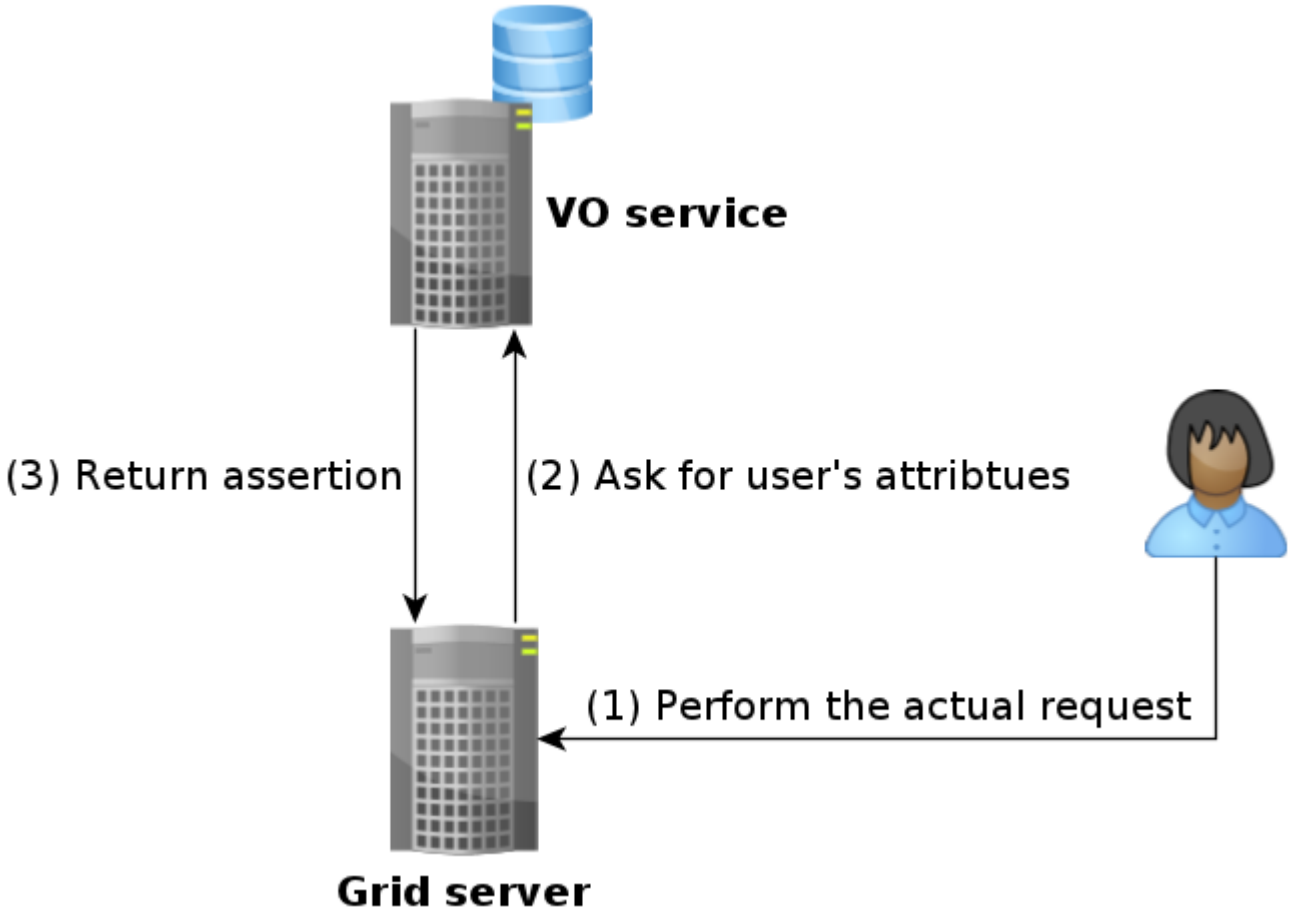
- An old (and boring?) concept...
- Many different meanings but mostly:

  *Grouping of users and resources from multiple real organizations, cooperating.*
- The biggest competitor: federations.
- VOs:
  - members maintained in a separate DB, centrally,
    - administration might be partially distributed,
  - member organizations assign resources to the VO,
  - the VO decides who gets what.

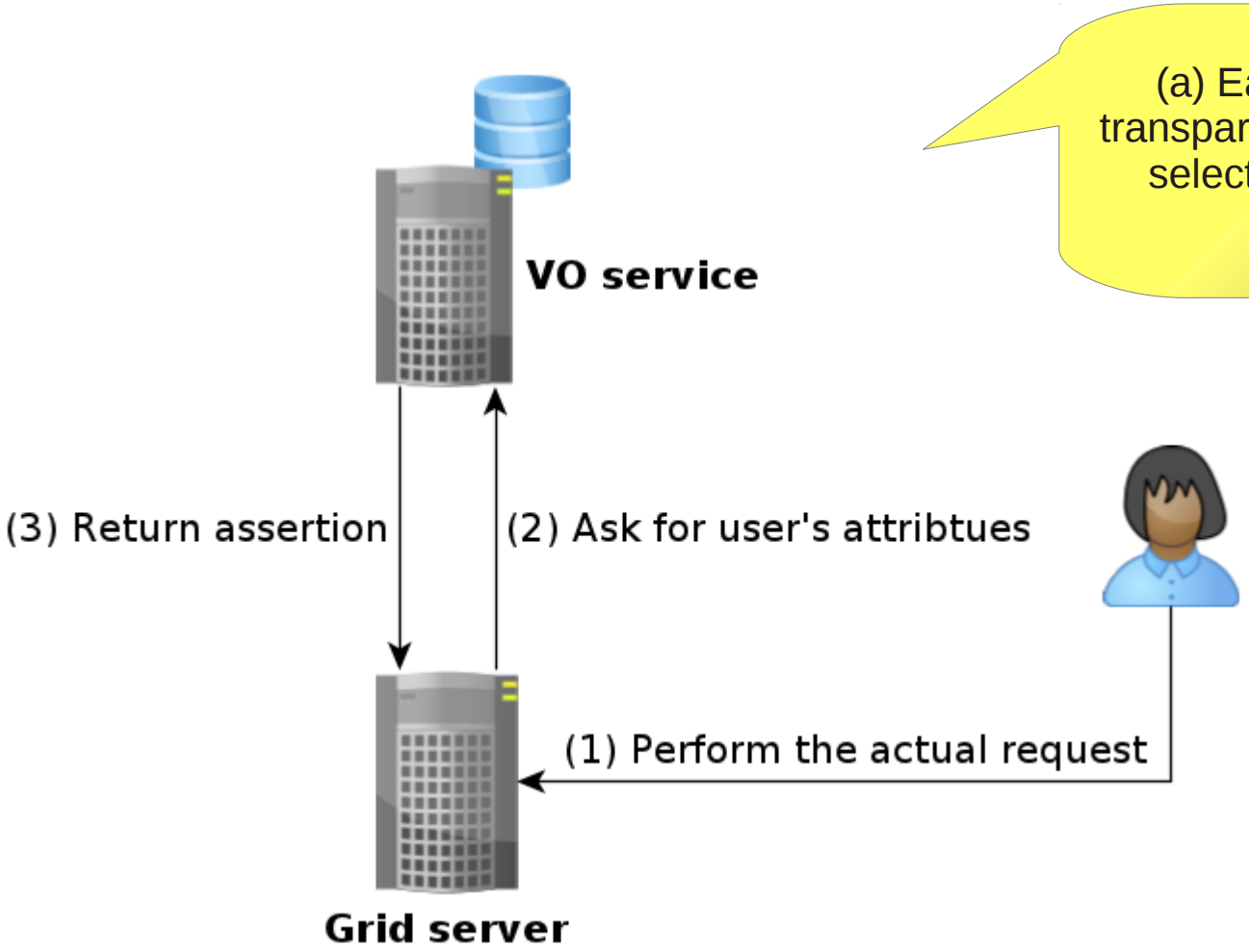
# What about federations?

- Federations:
  - members maintained in many DBs, by each (home) organization individually.
  - created as an agreement on common authorization language (e.g. used attributes and their meaning).
  - each home organization decides on rights of its users
  - resources access is defined using the common federation language.
- *In VOs world there is a strict control over who is the member, but there is identity duplication. Federations are opposite.*

# Obtaining VO information: the PULL mode

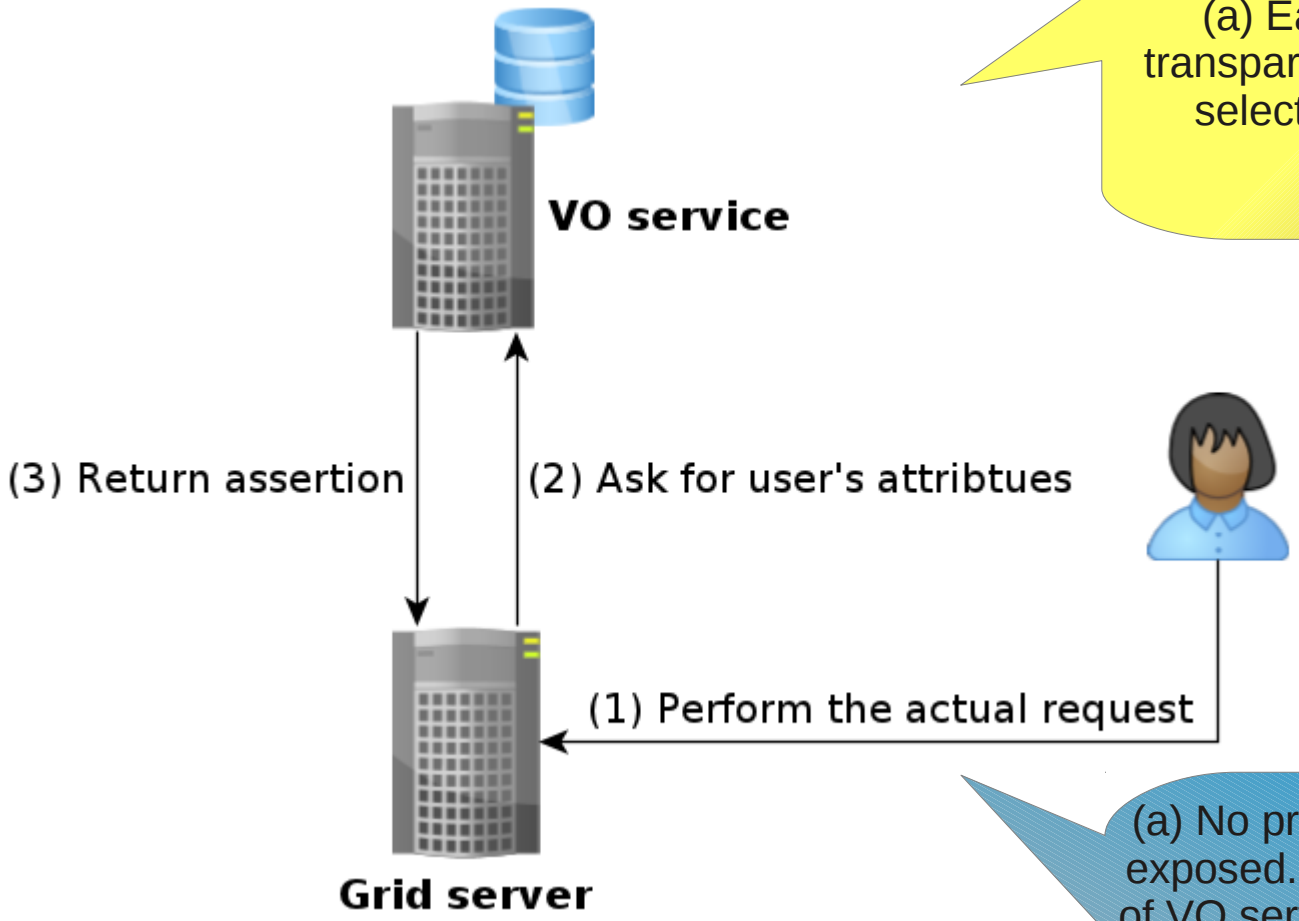


# Obtaining VO information: the PULL mode




(a) Easy for end-users, can be transparent. (b) Optionally users can select VO (and VO-options) via simple preferences.

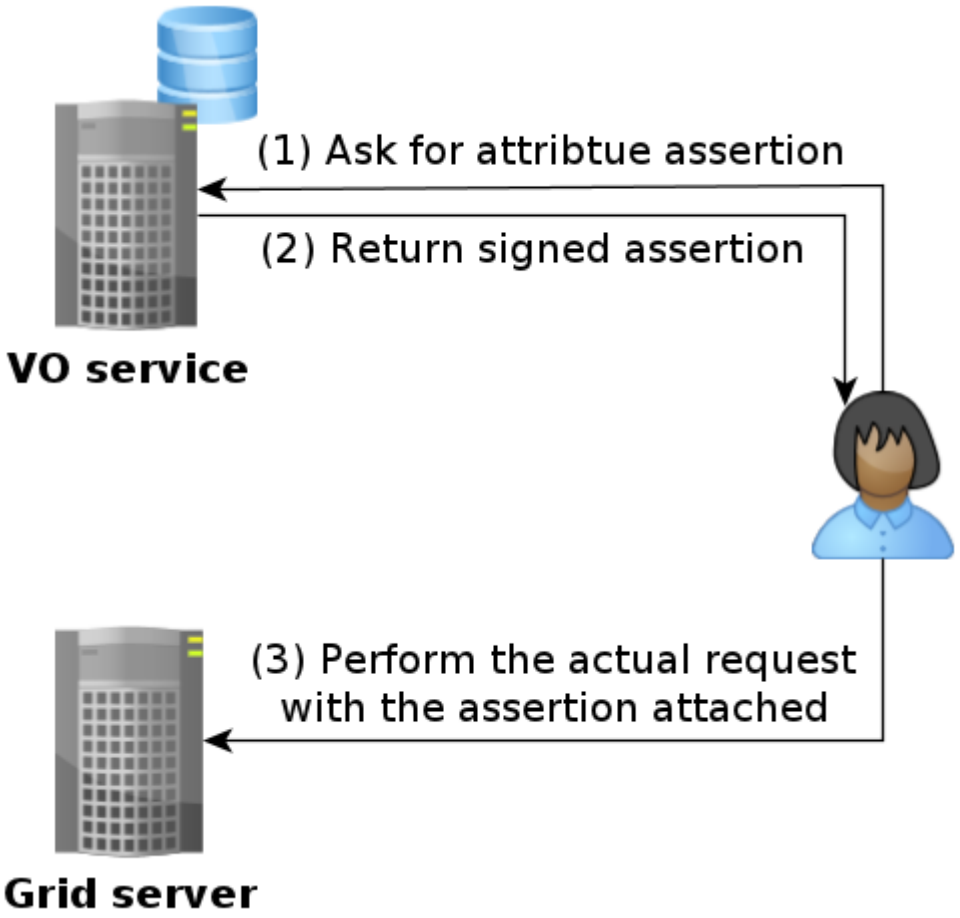
# Obtaining VO information: the PULL mode



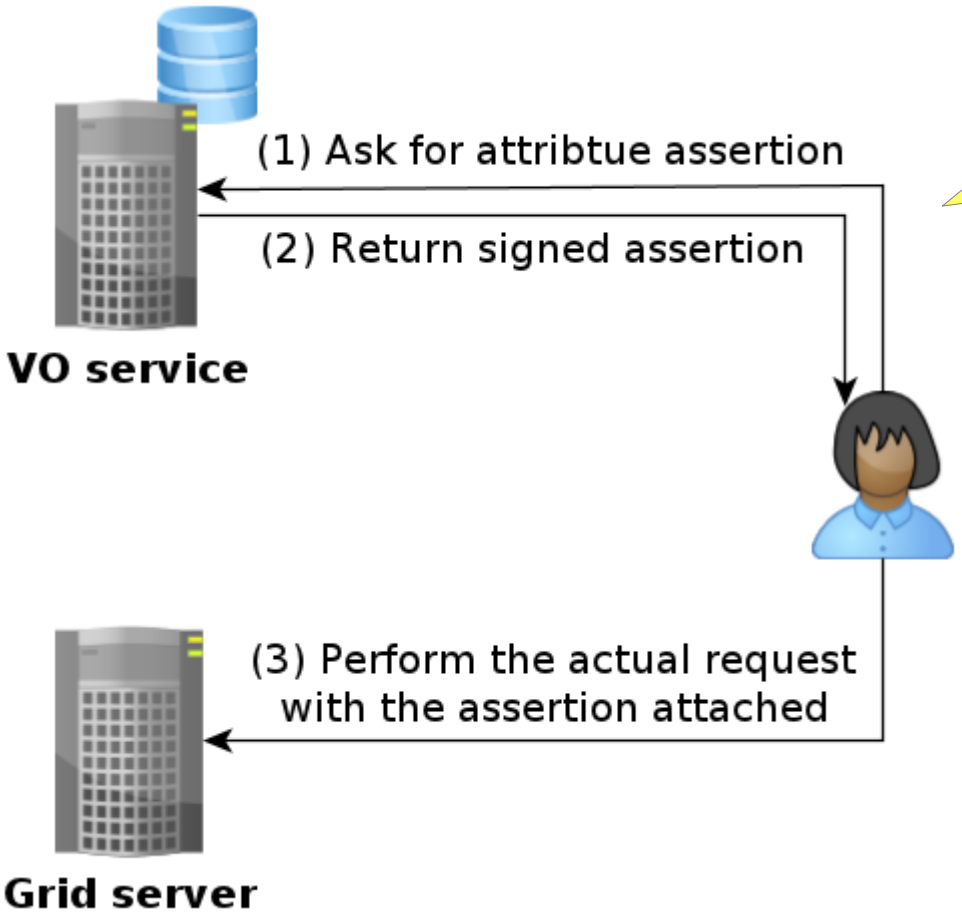

(a) Easy for end-users, can be transparent. (b) Optionally users can select VO (and VO-options) via simple preferences.

(a) No privacy - the whole VO contents exposed. (b) Not suitable when number of VO servers is large. (c) Even with few VOs it is difficult to provide sensible defaults. (d) Hard to configure permissions for all grid servers to access every VO service. Using delegation?

# Obtaining VO information: the PUSH mode

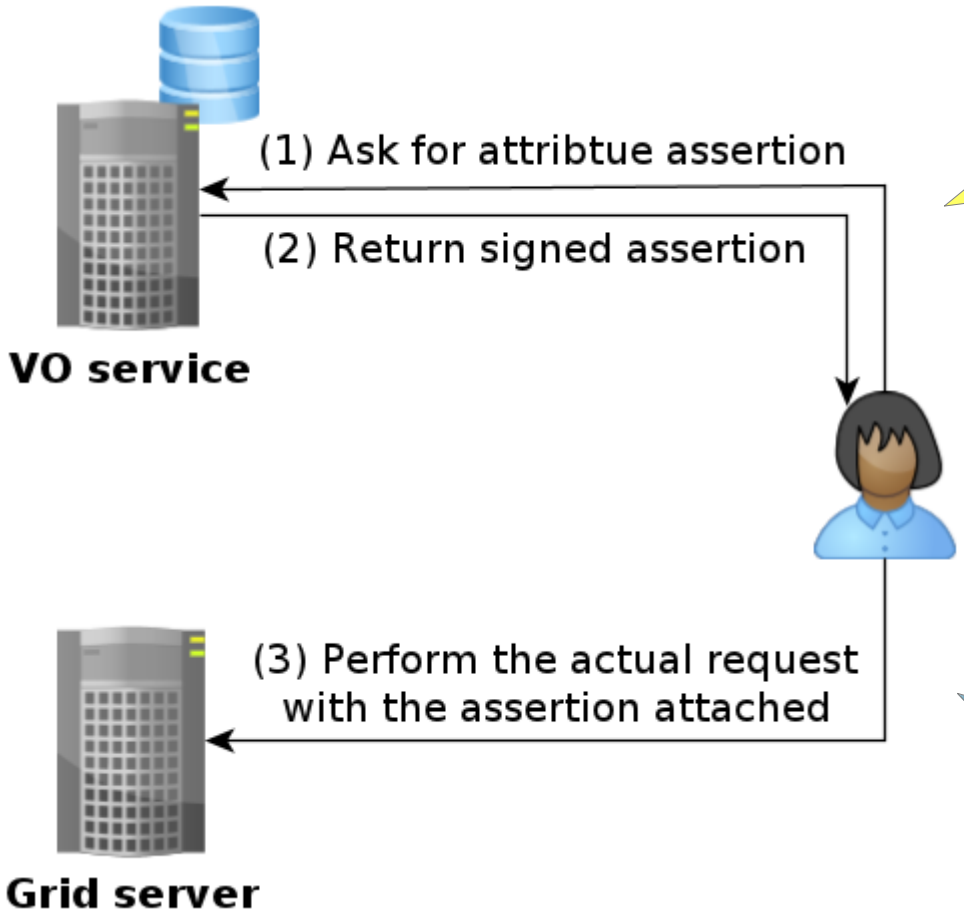


# Obtaining VO information: the PUSH mode




(a) End-users have full control over VO information exposed to the grid.
(b) Easily scalable in terms of VOs number and VO servers number.

# Obtaining VO information: the PUSH mode




(a) End-users have full control over VO information exposed to the grid.
(b) Easily scalable in terms of VOs number and VO servers number.

(a) Users are must handle the initial VO contact - select the VO and VO attributes that shall be exposed.
This is hard - very friendly UI needed.

# VO use cases in the Grid

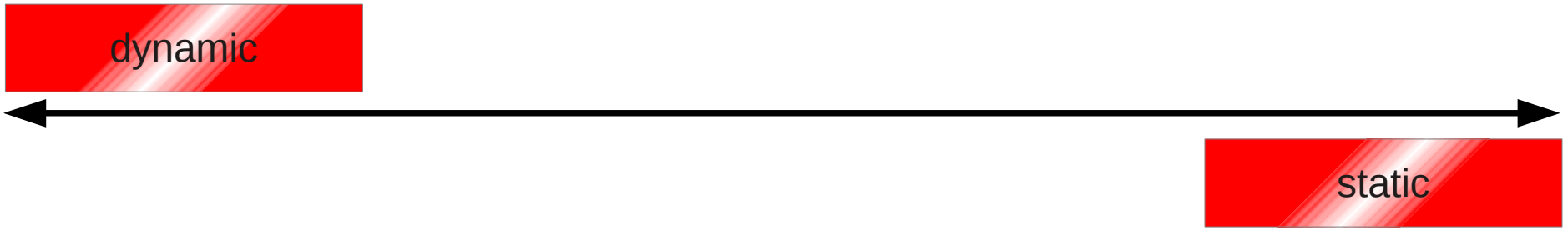
- VO software can provide an advanced user management system:
  - from user enrolment to removal.
- VO membership can be used for authorization
  - also other VO-defined attributes/roles/...
- Supporting a VO can automate users acceptance
  - no need for manual accounts set up etc.
- Jobs might be assigned to VOs
  - VO might be later charged (ranked, ...) for its users (accounting).
  - VO environment might be loaded (e.g. a special gid).
- VO members may collaborate
  - For instance can have access to a shared file space.

# Advanced VO use-cases

- VOs may be used to **automatically** set up specialized user's environment:
  - instantiate per-VO VMs images,
  - create reservations,
  - enable software licenses, ...
- VOs may coordinate inter-site collaboration
  - e.g. manage VO-wide clusters reservations with automatically negotiated reservation shares between the resource providers.
- VOs can be used to manage legal agreements that users have to sign.

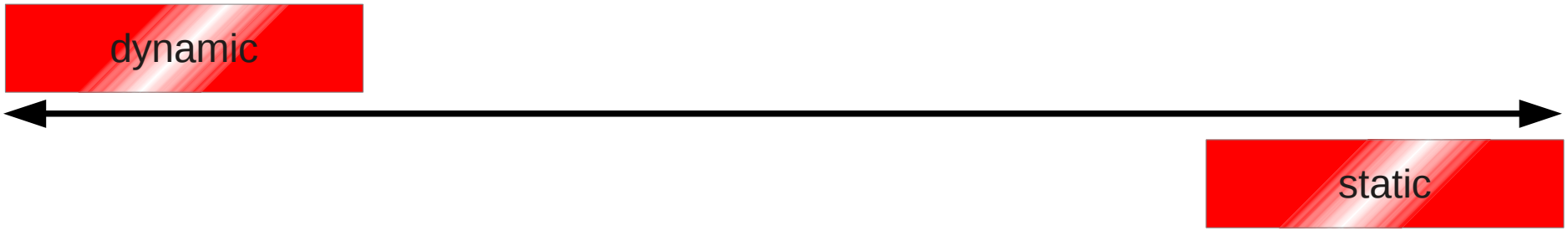
# Classification of VOs

# Classification of VOs



# Classification of VOs

VO is created ad hoc, between cooperating users. Typically medium or short term with few users. E.g. several colleges working on an experiment, who want to share their jobs' results and input.



# Classification of VOs

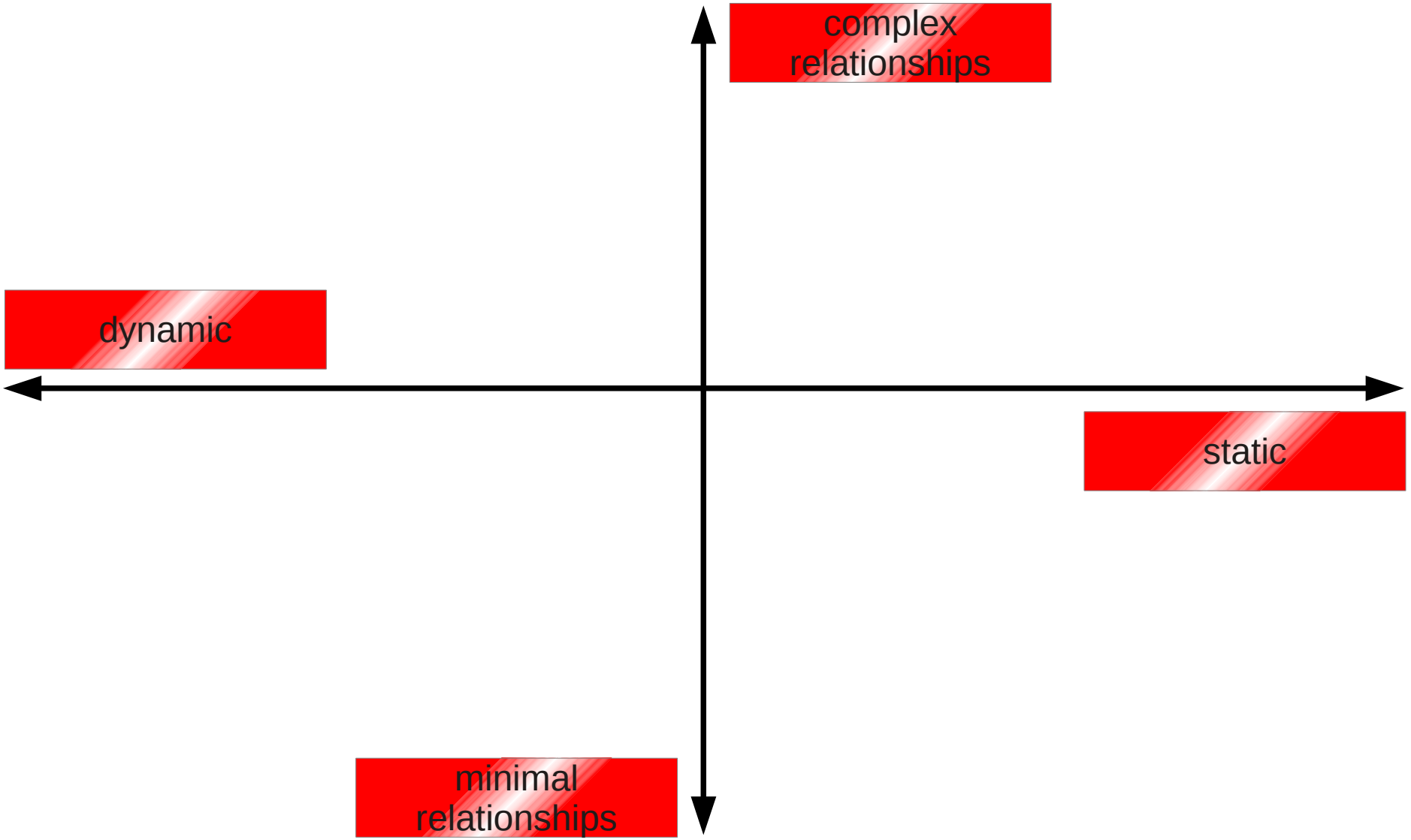
VO is created ad hoc, between cooperating users. Typically medium or short term with few users. E.g. several colleges working on an experiment, who want to share their jobs' results and input.

dynamic

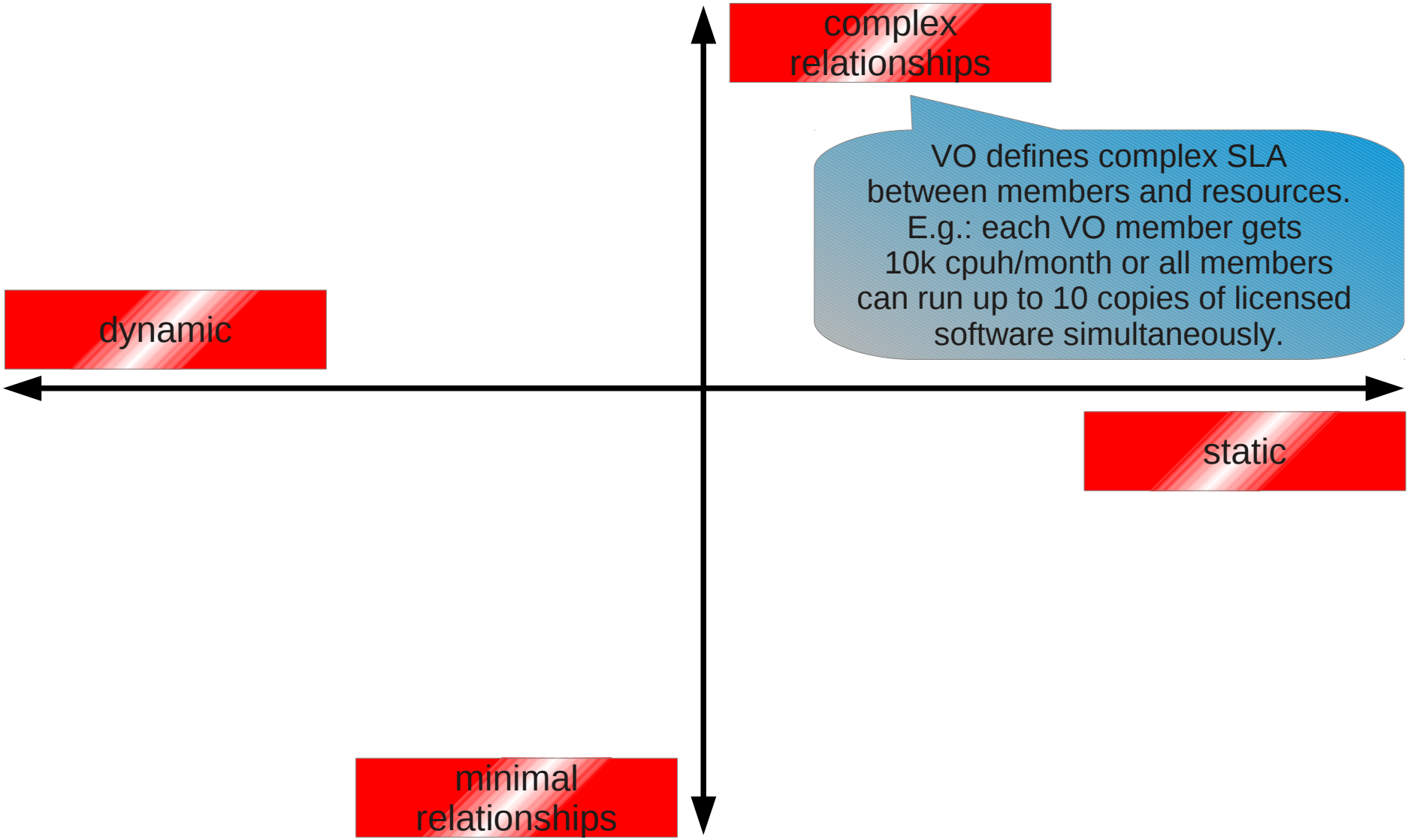
static

VO is rather big, created in effect of formal agreement between organizations, provides access to large resources. Set up and maintained by dedicated administrators. E.g. WLCG VOs.

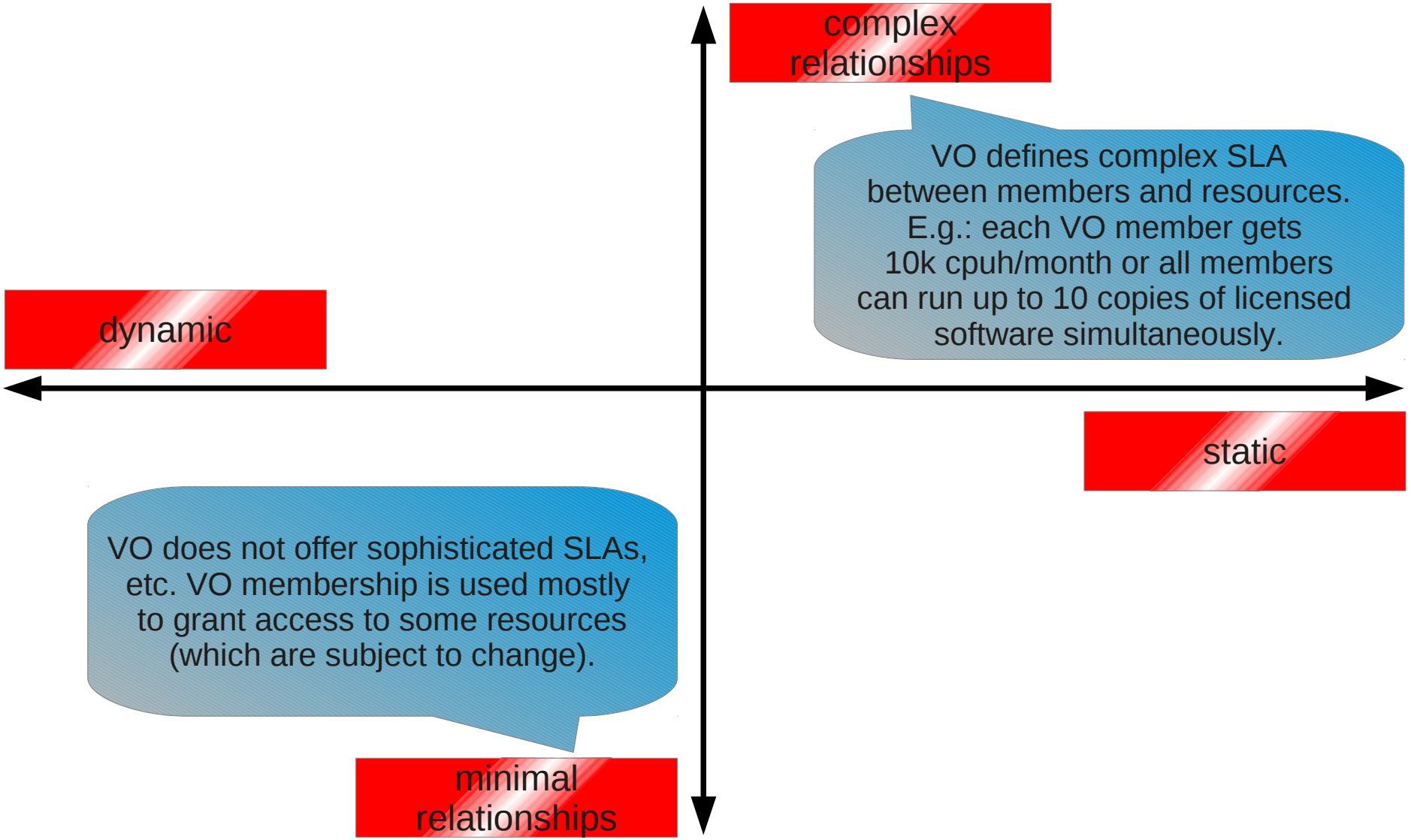
# Classification of VOs



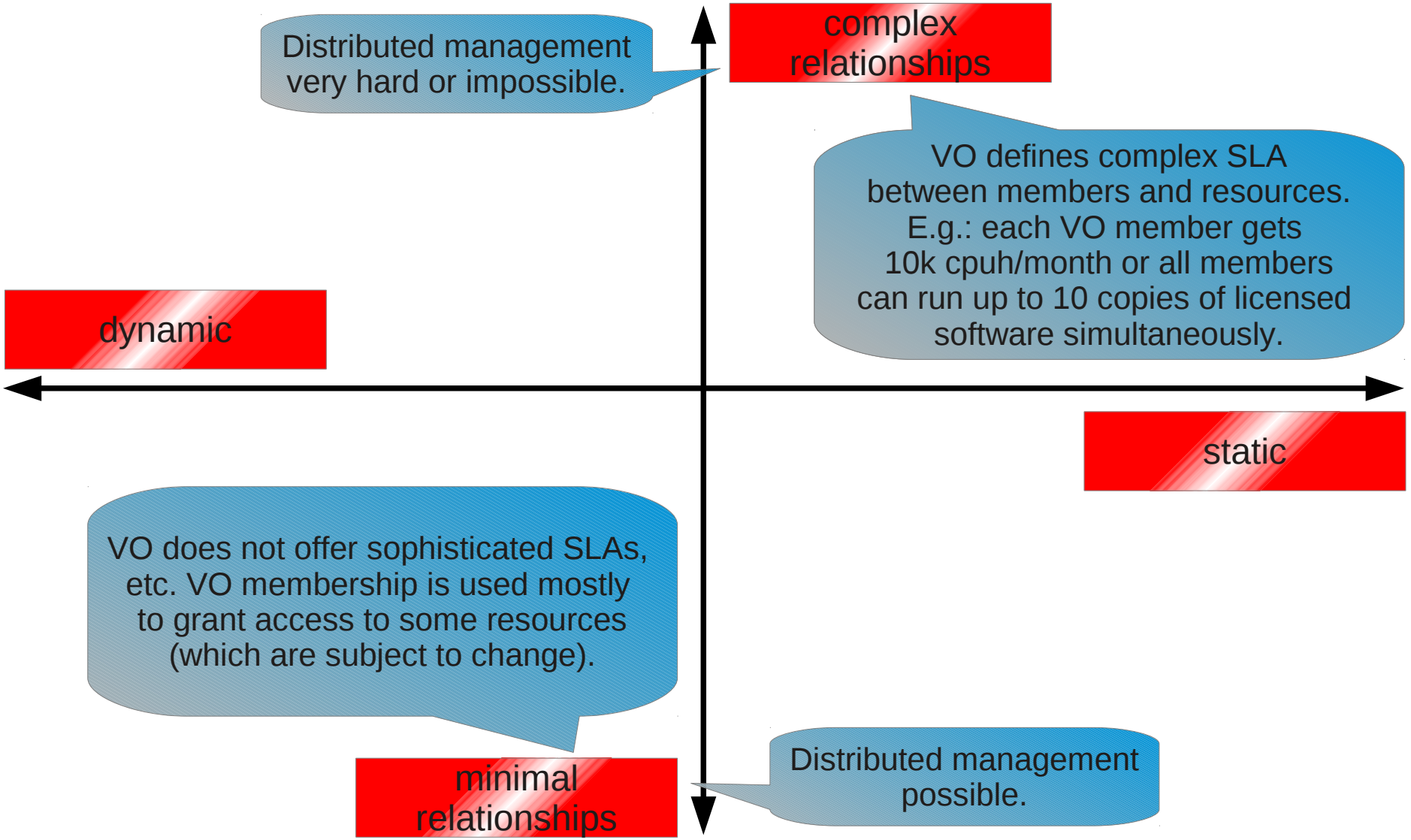
# Classification of VOs



# Classification of VOs



# Classification of VOs



# gLite and VOMS

- Virtual Organizations Membership Service.
  - INFN, used in EGI and WLCG.
- One VOMS instance maintains only a single VO.
  - *Difficult to set up new VOs.*
- VOMS exposes information on:
  - VO members, organized in hierarchical groups,
  - their roles, group scoped,
  - generic attributes, not scoped.

  in proxies with AC extension (*VOMS proxy*) or in SAML assertions.
- Only user can query for her attributes.
  - *Push mode supported only.*

# gLite and VOMS

- gLite LCMAPS allows for mapping of VOMS attributes to local uids and/or gids (fixed or pool).
- The client identity is VO-bound and therefore the VO information is tightly coupled with each request.
  - used for accounting
- Some statistics:
  - EGI maintains over 200 VOs, with over 21k members.
  - The biggest VO: atlas - nearly 3k members.
  - http://operations-portal.egi.eu/vo/usersSummary

# VOs in UNICORE up to 6.4.x
# *UVOS*

- UNICORE Virtual Organizations Service
  - Everybody mix this up with VOMS. New name in future?
- Server can handle arbitrary amount of VOs.
- Members can have multiple identities.
- Organized in hierarchical groups.
- With attributes - each can be group scoped.
  - *Possibility to store an arbitrary site-specific data as xlogins.*
- Only SAML supported as the assertion format.
- Both self and 3rd party queries possible.
  - *Push and pull modes possible.*

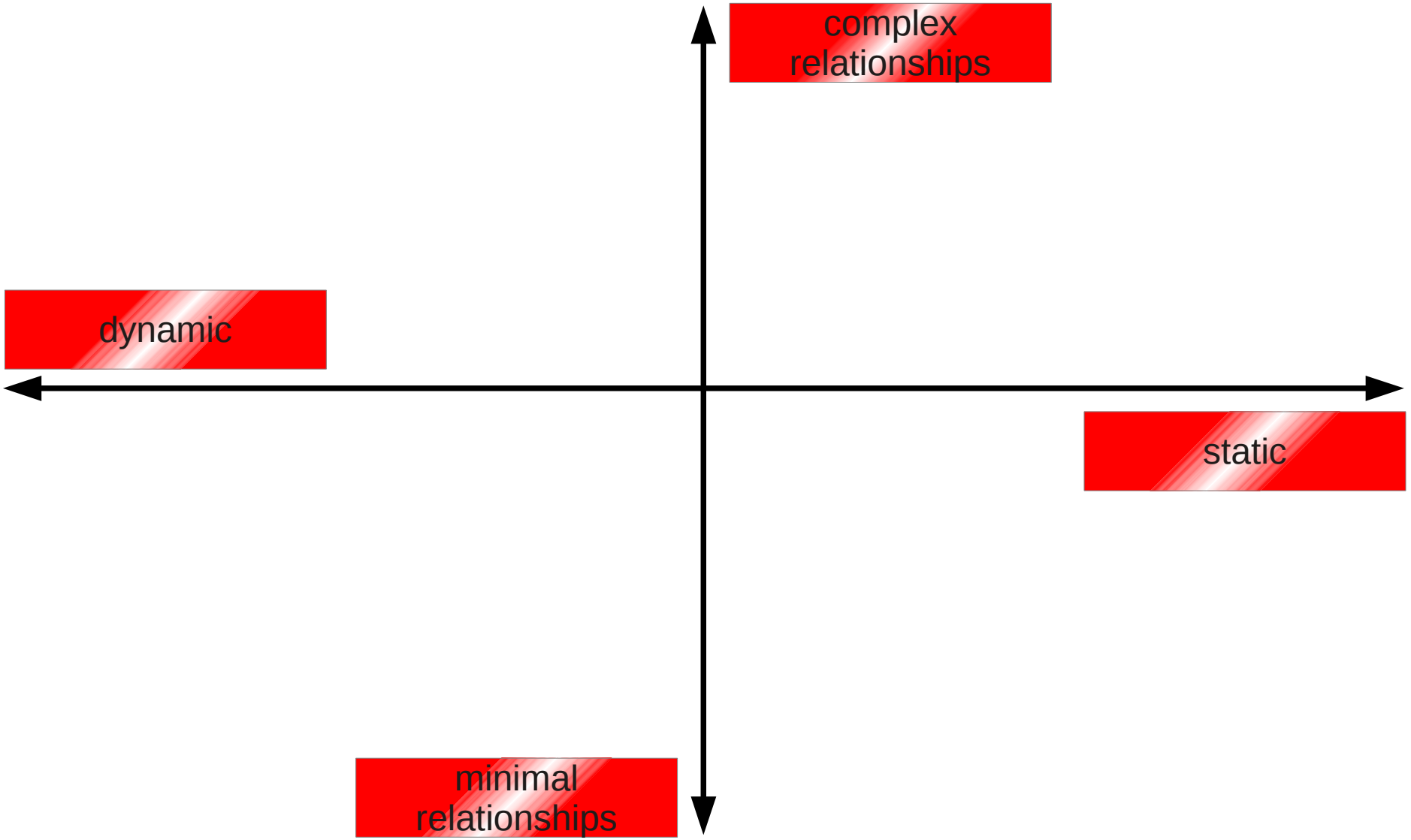
# VOs in UNICORE up to 6.4.x
# What do we have?

- Convenient user management UI provided by UVOS.
- UVOS can be used to **store site-local xlogins**
  - **distributed management** also **possible**.
- VO attributes are mapped to UNICORE standard ones (role, xlogin, ...) and in effect **VO membership is used in authorization only**
  - Implicitly - in fact only a role attribute from supported VOs.

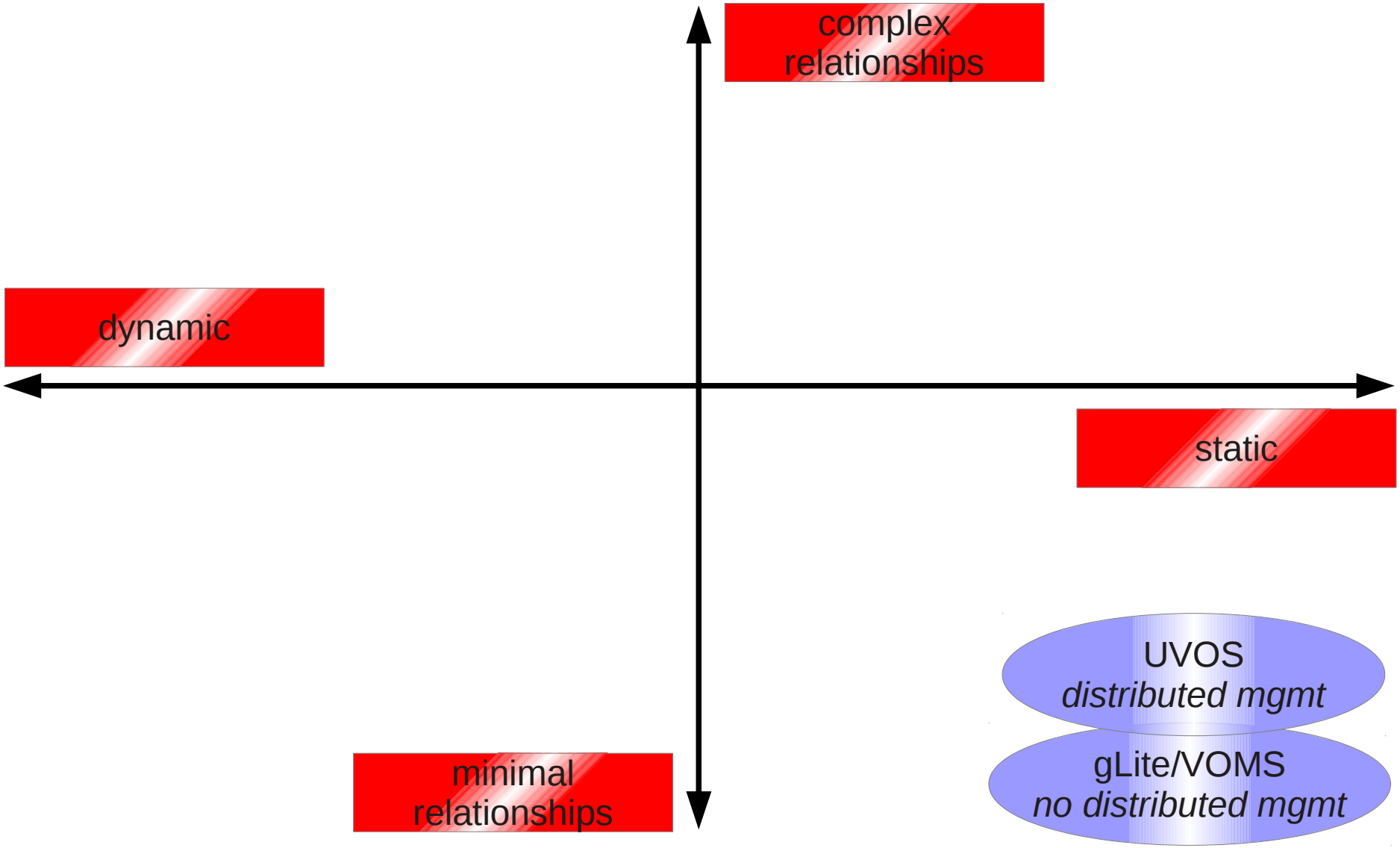
# VOs in UNICORE up to 6.4.x
# What are we missing?

- UNICORE server side supports both push and pull modes.
  - but UCC and URC doesn't support querying for and pushing of SAML attributes - **only pull mode possible**.
- Requests are not VO-bound.
  - and it is not possible to specify a preferred VO anyhow.
  - **no simple way to use VO-accounting, VO-gid etc**.
- UNICORE requires explicit DN->uid mapping for each user.
  - **Adding a site to a VO is very problematic**, all users need this mapping, typically site-specific.
  - Either coping of VO data to sites must be done or a central LDAP+UVOS deployed.
- **VO users can not cooperate in any way**.

# Where we are?


complex relationships
dynamic
static
minimal relationships

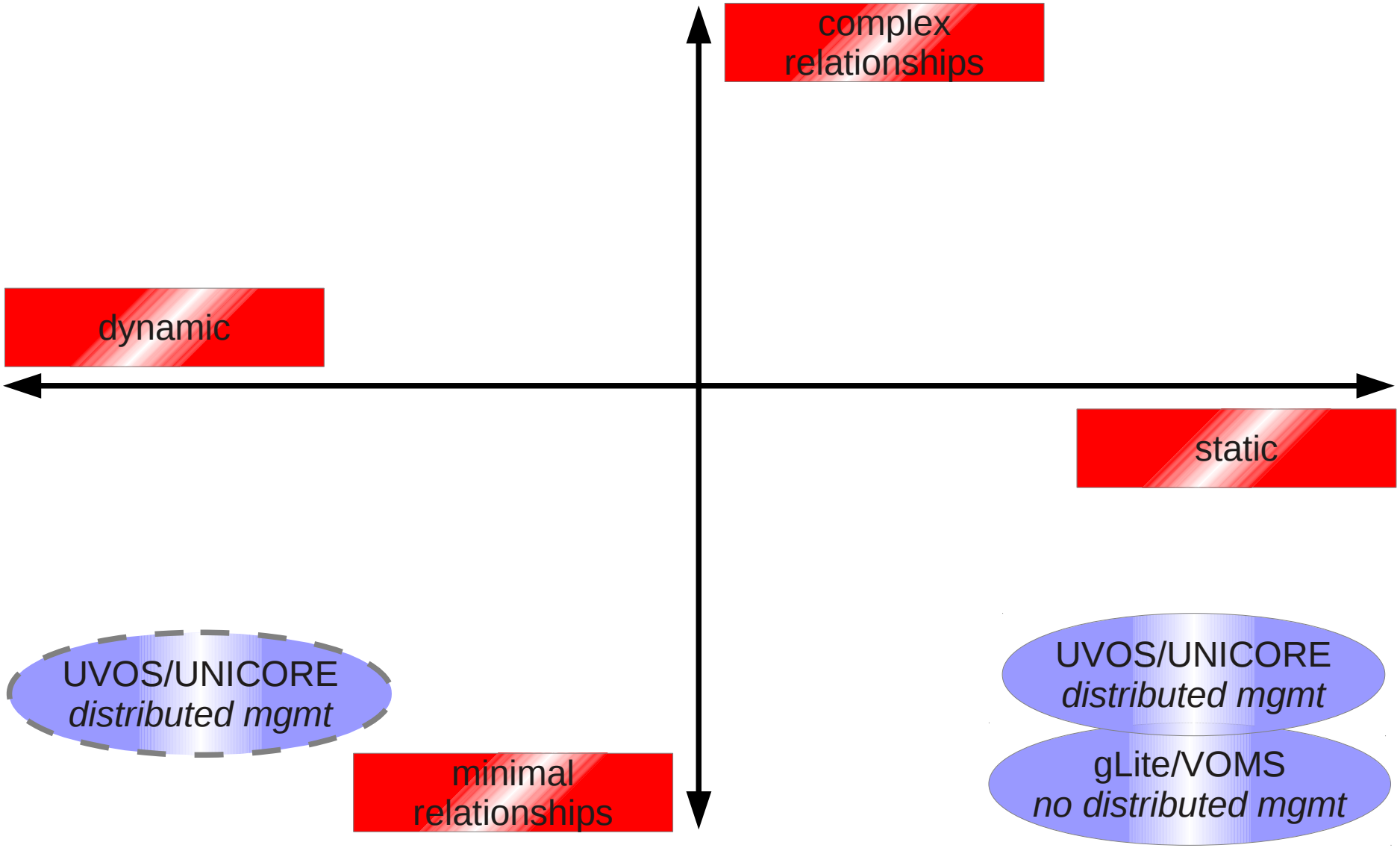
# Where we are?


complex relationships
dynamic
static
minimal relationships
UVOS
distributed mgmt
gLite/VOMS
no distributed mgmt

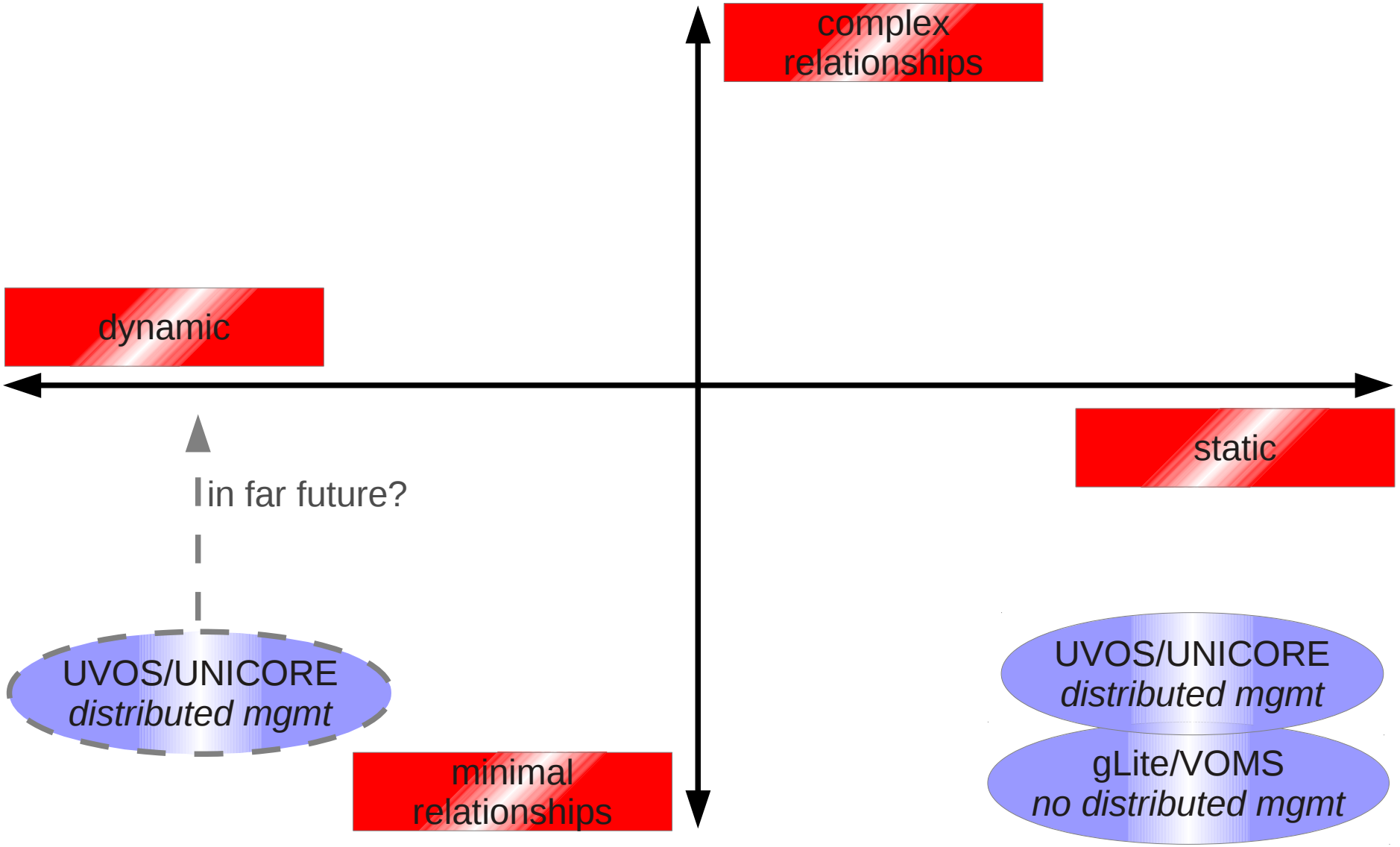
# Desired features
## *exterminate the limitations*

- Full support for push mode in UNICORE clients.
  - Scalability and others.
- Is should be possible to bound requests to VOs.
  - Accounting and others.
- Adding a site to a VO should be simple.
  - Hardcore requirements like grid-wide LDAP eliminated.
- **VO users should be able to cooperate in an easy way when using UNICORE**.

# Where do we want to be?



# Where do we want to be?



# The roadmap

- Support for specifying preferred VO for pull mode.
- Assigning of WSRF resources to VOs.
  - reflected in access control rules
  - reflected in backing OS artefacts
  - user controlled
- Support for dynamic incarnation attributes.
  - ability to assign uids & gids basing on static VO attributes
- VO service extended to support dynamic VOs, end-user-driven.
  - + client support

# Preferred VO

- In request preferences (together with preferred xlogin, gid, ...) a preferred VO can be specified.
- Already implemented in UCC 6.5.0.
- Server support in 6.5.0
  - preferred VO becomes a selected VO if user is its member,
  - settings from selected VO take preference over settings from other VOs,
  - selected VO is available during request processing.
  - Administrator can define a list of default VOs, used when there is no preference.

# Sharing WSRF resource with VOs

- Each WSRF resource (an object) will have two lists:
  - of VOs authorized to 'read' this resource
  - of VOs authorized to 'write' this resource
- Similar to UNIX FS owning group with rw perms.
- Only owner will be allowed to control the VO lists.
- Assigning to VO will be done in a recursive way.
  - parental relationships between WSRF resources need to be standardized.
- On client side:

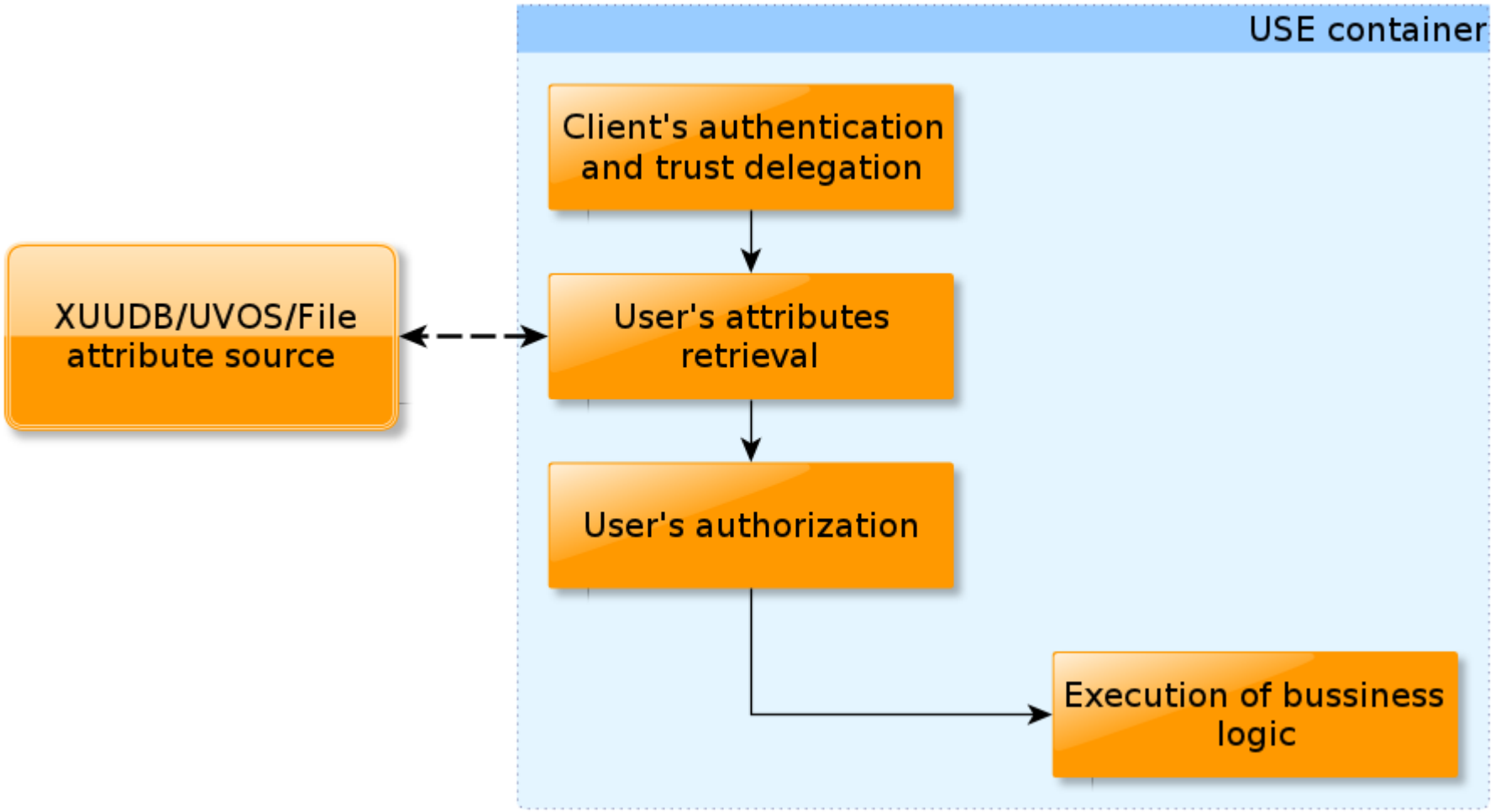
  *"share it with..."*

  in context menu of GridBrowser.

# Sharing OS resources with VOs

- For WSRF resources being backed by OS resources the situation is more complicated.
- We have:
  - SMS - files
  - JMS - processes
  - FTS - network transfers
- Sharing of transfers and processes control won't be possible - minimal usefulness & hardly doable.
- It will be possible to configure SMS files sharing mode:
  - portable but limited: chgrp
  - not so portable but unlimited: setfacl
- Open problem: sharing only selected SMS files (discovery).

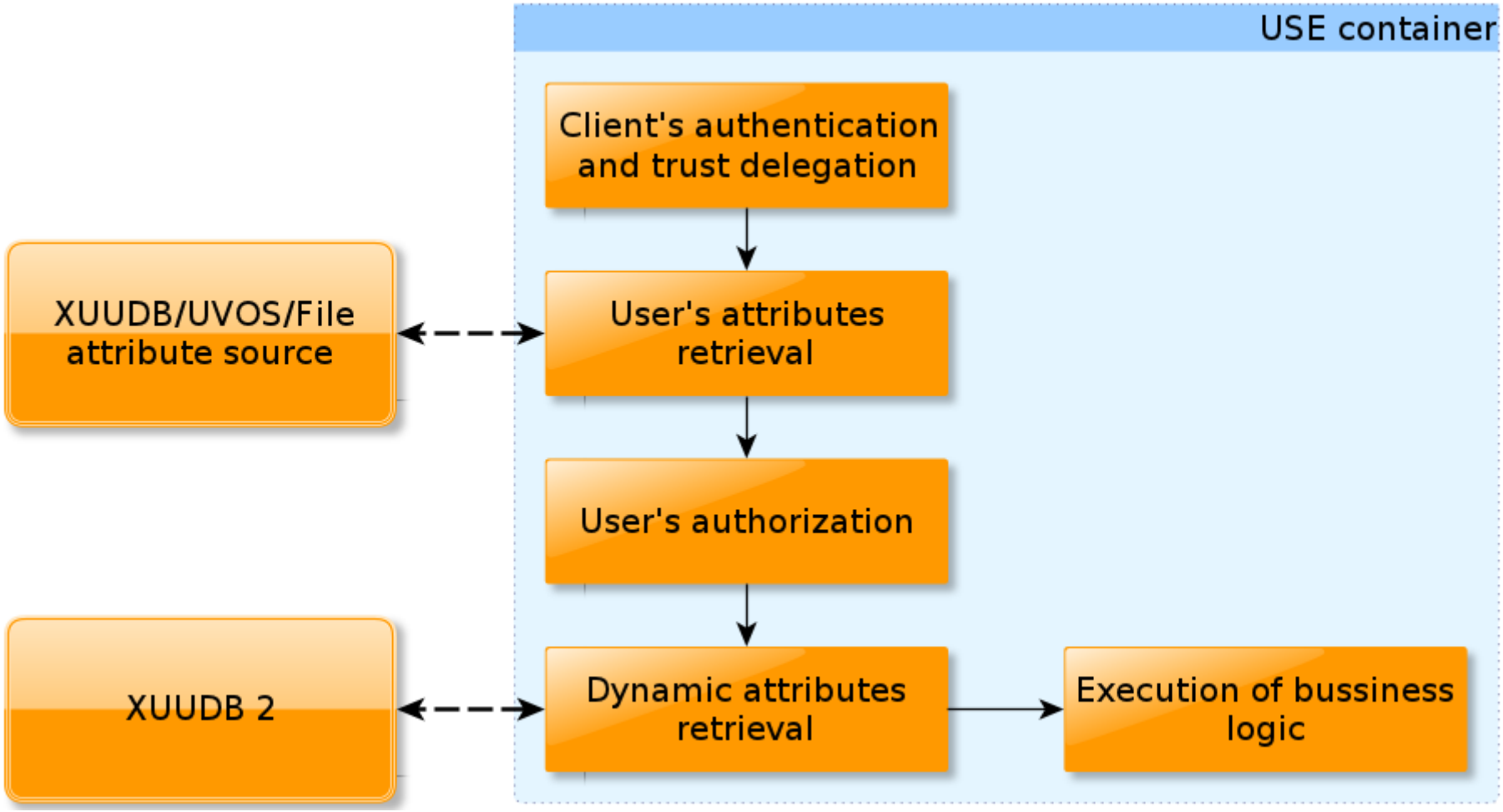
# Current security flow



# Dynamic attributes realisation

- XUUDB will be used as dynamic attributes site service.
- Thoroughly revised:
  - a new interface will be added,
  - with support for multiple RDBMes.
- It will be possible to:
  - assign a fixed uid/gid or add supplementary gid for VO member, role holder etc.,
  - assign an uid or **gid** from a pool,
  - assign via external application.
  - Use intuitive administrative utilities.
- There is no strong need to expire pool account assignments (what is the biggest problem).
- Hooked in USE *after* authorization.

# New security flow



# UVOS (?) extensions

- Dynamic VOs will be modelled as subgroups in static VOs.
- Most of the features and interfaces already in place. But:
  - a more flexible authorization mechanism required (users creating groups),
  - a more flexible attribute inheritance control,
  - administrative control over dynamic VOs (max per user, max total, ...)
- When a resource is shared it must be added to UVOS.
  - for easiest client's integration UVOS should offer a Registry-like interface to query VO contents.
- An easy to use support in client tools needed:
  - create a dynamic VO
  - invite users from the parent, static VO, accept invitations, ...
  - access VO registries.

# Summary

- A lot to do!
- But many things already advanced:
  - XUUDB 2.0, preferred VO, low level ACL and FS manipulation, push mode from UCC.
- In effect it will be possible to:
  - easily add a site to a VO (no need for static DN->uid mappings)
  - users will be able to share their data: workflows, jobs, SMSes.
- In future we can consider more advanced features
  - For instance site administrator can create specially configured TSF for VOs. And share them from URC :-)