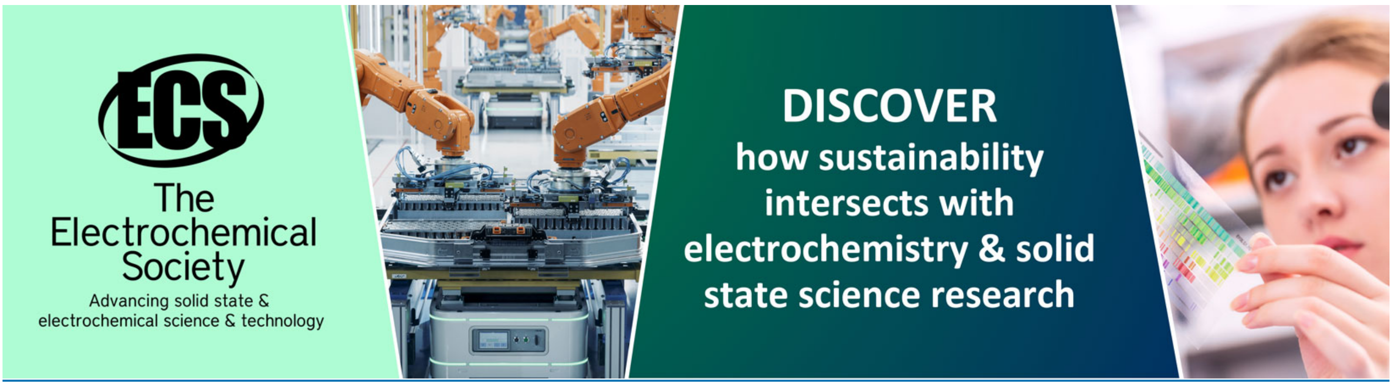
Experiences with the GLUE information schema in the LCG/EGEE production grid


ECS
The Electrochemical Society
Advancing solid state & electrochemical science & technology
DISCOVER how sustainability intersects with electrochemistry & solid state science research

Experiences with the GLUE Information Schema in the LCG/EGEE Production Grid

Stephen Burke[1], **Sergio Andreozzi**[2], **Laurence Field**[3]
[1] Rutherford Appleton Laboratory, Didcot, OX11 0QX, UK
[2] INFN-CNAF, Viale Berti Pichat 6/2, 40126 Bologna, Italy
[3] CERN, CH-1211 Genve 23, Switzerland

E-mail: `s.burke@rl.ac.uk, sergio.andreozzi@cnaf.infn.it, laurence.field@cern.ch`

Abstract. A common information schema for the description of Grid resources and services is an essential requirement for interoperating Grid infrastructures, and its implementation interacts with every Grid component. In this context, the GLUE information schema was originally defined in 2002 as a joint project between the European DataGrid and DataTAG projects and the US iVDGL. The schema has major components to describe Computing and Storage Elements, and also generic Service and Site information. It has been used extensively in the LCG/EGEE Grid, for job submission, data management, service discovery and monitoring. In this paper we present the experience gained over the last five years, highlighting both successes and problems. In particular, we consider the importance of having a clear definition of schema attributes; the construction of standard information providers and difficulties encountered in mapping an abstract schema to diverse real systems; the configuration of publication in a way which suits system managers and the varying characteristics of Grid sites; the validation of published information; the ways in which information can be used (and misused) by Grid services and users; and issues related to managing schema upgrades in a large distributed system.

1. Introduction

A Grid consists of a large number of sites, each providing many resources and services, with a wide variety of properties and configured in many different ways. In order to use a Grid infrastructure, there must be a way to discover the resources and services available, and determine their properties and current state. In addition, Grid operations staff and higher-level management need to be able to see an overall picture of the state of the entire Grid or a subset of it. Some typical use cases might be: (1) what Resource Brokers are available to CMS users? (2) find all Computing Elements running Scientific Linux 4 with a main memory greater than 2 GB; (3) find all Storage Elements with at least 20 TB of free space available to ATLAS; (4) how many jobs are running at the UK Grid sites? (5) how much disk space has ATLAS used in total?. These requirements imply that sites must provide information about themselves in a standardised format, defined by an information schema.

This paper discusses the development of and experience with the GLUE schema, as implemented in the LCG/EGEE production Grid, over the last five years. Section 2 explains the history of the GLUE schema and the way in which it has evolved. Section 3 gives some details about the structure of the schema. Section 4 discusses the experience gained with the schema in a variety of areas, and suggests some ways in which improvements can be made. Finally, section 5 contains some concluding remarks.

Note that in this paper LCG/EGEE refers specifically to the Grid operated jointly by those two projects [1],[2]; the worldwide LCG collaboration also uses other Grids, notably OSG [3] and NDGF [4]. OSG also uses the GLUE schema, but this is not specifically discussed here. NDGF currently uses a different schema, but is now involved in discussions about convergence with a future version of GLUE.

1.1. Designing a Schema

As discussed further below, some desirable properties of a schema are: *Not too big*, the object is not to model every detail of a computing system, just to provide sufficient information to facilitate the use cases mentioned above; *Not too small*, the schema needs to capture all the relevant information; *Flexibility*, the schema must be able to represent, at least to a reasonable approximation, the very wide range of different software and configurations in use in a production Grid; *Precision*, the semantics of the schema must be defined precisely and unambiguously; *Simplicity*, it should be as easy as possible to understand the meaning of the schema attributes; *Calculability*, the attributes must be capable of being calculated in real systems in a relatively short time - typically an information provider (the code which collects and publishes information according to the schema) needs to run in no more than a second or so; *Extensibility*, it is likely that the schema will need to evolve over time, and this needs to be possible with as little disruption to existing systems as possible. These properties are to some extent in tension with each other, so it will generally be necessary to find a reasonable compromise.

1.2. Related Concepts

The schema itself is an abstract description of information; actual Grids use one or more Information Systems to transport this information, using a variety of technologies. The primary use cases for the schema are for a snapshot of the current state, although for performance reasons there will typically be various caches so information propagation will have some latency. Monitoring use cases may also require historical information to be stored in databases. The information itself is collected by information providers, pieces of code which must be tailored to each implementation of the various Grid services. However, it is desirable to use common code where possible to preserve consistency.

2. The GLUE Project

There are a number of Grids in use, each with their own middleware. However, interoperability between Grids is highly desirable; for WLCG this particularly relates to the LCG/EGEE, OSG and NGDF Grids. To achieve this it must be possible to share resource information across Grids, and if they each have their own information schema this is likely to be difficult. If common concepts exist it may be possible to write translators from one schema to another, but this is likely to be error-prone and may be prevented by relatively small differences in the definition or interpretation of the attributes. It is therefore desirable to have a common, standard schema, at least for a core set of attributes necessary for interoperability. A common schema also allows different Grids to share their experience and avoid duplication of effort. However, standardization activities have the potential to be slow and can have difficulty reaching agreement between participants with divergent views, so it is important to have a focused process.

The GLUE (Grid Laboratory for a Uniform Environment) project was a collaboration between the EU-funded European DataGrid (EDG) [5] and DataTAG [6] projects and the US iVDGL [7], together with Globus [8]. The GLUE information schema was developed from the schema used by the EDG project from 2001. The initial specification for version 1.0 of the schema was published in September 2002 after several months of discussion in email and phone conferences. Some minor improvements resulted in version 1.1 [9] in April 2003, and this version was deployed in production by EDG and subsequently LCG/EGEE.

Experience with the 1.1 schema over the following year or so led to a number of changes being proposed by LCG in November 2004, and version 1.2 of the schema [10] was agreed in February 2005 and deployed by LCG/EGEE during 2006. A further evolution to version 1.3 [11] was agreed in October 2006, and is currently in the process of being deployed. At this time it was also agreed to create an OGF [12] GLUE working group to produce a major new version 2.0 of the schema; this project is described in a separate paper [13].

2.1. The Schema Evolution Process

The schema needs to evolve, both because the Grid itself is evolving and as experience is gained about missing or badly-designed features. However, the fact that the schema interacts with everything else - middleware, users, deployment - and must be agreed between many different Grid projects, makes evolution a non-trivial task. This is exacerbated by the fact that schema development has generally been regarded as a somewhat peripheral activity by the Grid projects, so only a fairly small number of people have been involved, and in most cases only with a fairly small fraction of their time.

The two major upgrades to date have involved a similar process. There has been a period of several months during which ideas for possible changes have been collected, and refined through email discussion and phone meetings. A single face-to-face meeting has then made decisions about what should and should not be accepted. This is partly due to time constraints which make face-to-face meetings difficult to arrange, but it has resulted in the meetings being effective as the participants knew that agreement must be reached. In both cases the meetings managed to cover everything on the agenda, and either accepted the proposed changes or agreed to defer them to a future version.

After the formal agreement there are a number of further tasks to put the updated schema into use. The documentation is updated, and reference implementations are produced, in particular for LDAP [4] which is currently the main technology used in the LCG/EGEE Grid. This is then deployed in a controlled way - the information system technology (BDII) [14] requires that the higher-level servers must have the schema updated first. This process typically takes a few months after the schema update is agreed.

However, putting the schema itself into production is only part of the process. Information providers must be written or upgraded to collect and publish the new information. Client code, including user applications, may also need to be modified. Inevitably this process is slow, and some sites and applications may take a very long time to follow the update. Thus far it has therefore been decided to make all schema updates entirely backward-compatible. However, this has the disadvantage that it limits the changes that can be made, and has resulted in a fairly large number of deprecated features. The proposed GLUE 2.0 will therefore not require backward-compatibility - in practice this will probably mean that it will be deployed in parallel with the existing schema.

3. The Structure of the Schema

The schema is defined in an abstract object-oriented style using a subset of UML, in terms of objects with attributes and relations to other objects. However, consideration must be given to the intended implementation technologies, currently including LDAP, relational (R-GMA [15]), XML and Condor [16] classads, as these are significantly different and impose constraints on the possible structure of the schema. In particular, for the LCG/EGEE Grid the primary technology is currently LDAP, which has various idiosyncrasies, most notably the inability to represent tables and the lack of ordering of attributes within an object. On the other hand,

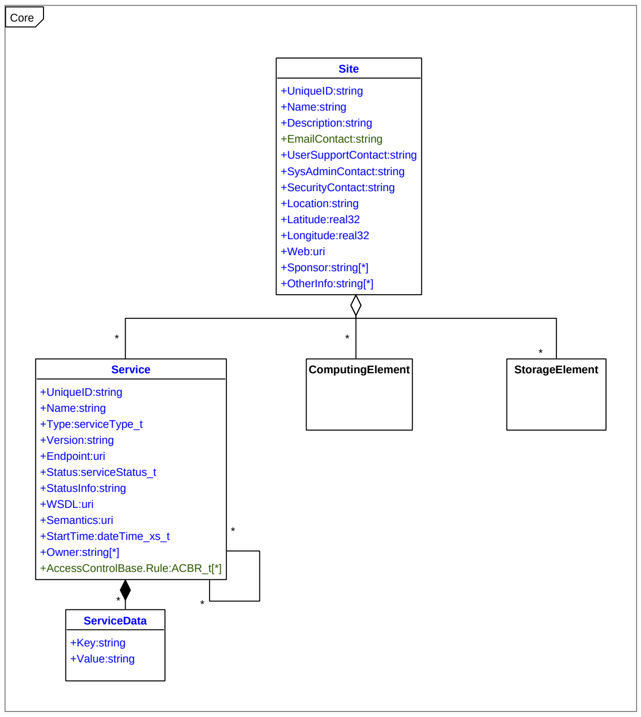


Figure 1. GLUE Schema 1.3 - Core

LDAP allows attributes to be missing or to be multivalued, i.e. to occur multiple times within one object, which are not natural in a relational implementation.

At a high level there are currently four parts to the schema. GlueSite represents some general attributes of a Grid site, e.g. the geographical location and contact email addresses for various purposes. GlueService is a general representation of a Grid (or other) service, i.e. a network service with a defined endpoint - this may or may not be a web service. This has attributes like the service type, endpoint and authorization. Arbitrary service-specific information can also be published as key-value pairs. These are shown in Figure 1.

3.1. The Computing Element

The GlueCE and related objects which describe a computing resource form the oldest part of the schema, originating in the original schema defined for the EDG project. The GlueCE itself is defined to map onto a queue in a batch system. The underlying hardware is represented by a Cluster/SubCluster concept, where the SubCluster was intended to capture the key properties of a homogeneous set of batch Worker Nodes, and the Cluster aggregates all SubClusters in a single computing system. Figure 2 shows the objects and their attributes and relations.

The original concept of a CE as a batch queue implicitly assumed that all jobs in a queue would be treated equally. However, in practice this is generally not the case; batch systems usually impose fair-share algorithms to classes of job within a queue, often in a complex way. To help to represent this situation, the 1.2 GLUE schema update introduced the concept of the VOView, which provides attributes for subgroups within a CE, with the grouping typically being by Virtual Organisation (VO) or subgroups within them.

The original GLUE schema also defined many attributes and objects to describe detailed properties of individual hosts, but this has not proven to be useful and is not implemented in the LCG/EGEE Grid.

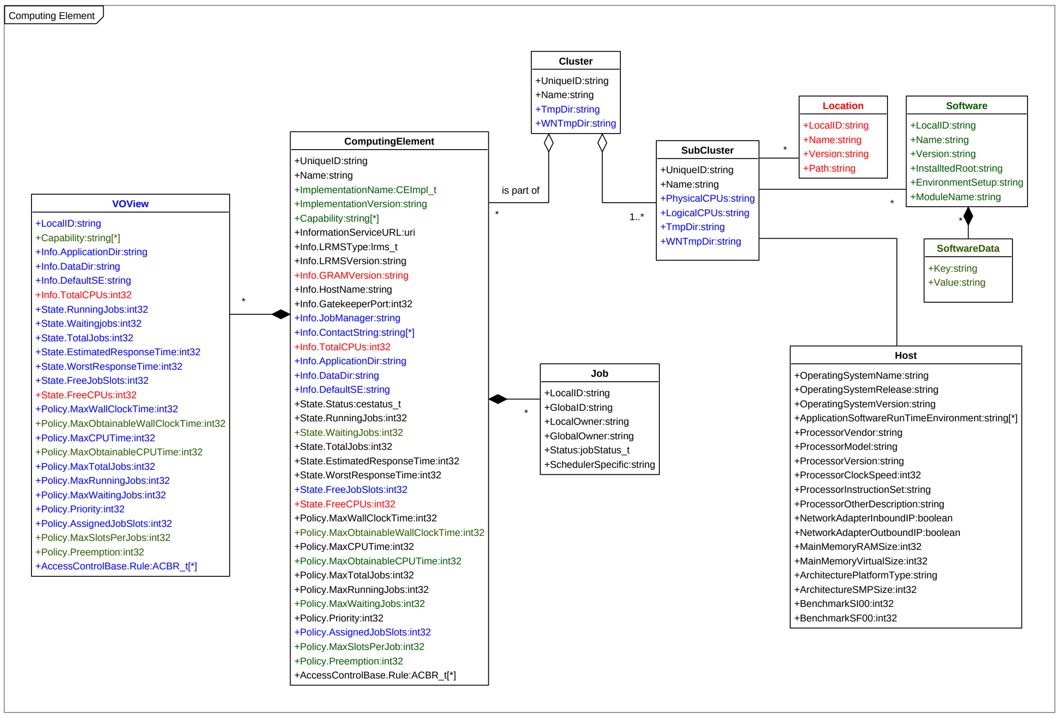


Figure 2. GLUE Schema 1.3 - Computing Resources

3.2. The Storage Element

The original GLUE schema defined a fairly small number of attributes and objects related to storage systems, as the only storage available at that time was the "classic SE" which essentially consisted of a Globus GridFTP server running over a simple flat filesystem, optionally with additional access protocols. The schema defined objects to represent storage spaces, essentially areas of disk space within the filesystem, and for the access protocols. It also had a concept of a storage library, intended to represent the storage hardware, but in practice this has not been used and is now deprecated.

In the intervening years the Storage Resource Manager (SRM) collaboration has defined a much more complex protocol [17] to manage large-scale Grid-enabled storage, and a number of different implementations exist. In addition the SRM protocol is currently undergoing a major change from version 1 to version 2, with the latter expected to be in production for the start of LCG data-taking. This therefore has major implications for the storage part of the GLUE schema. For the 1.3 schema revision there was a substantial change in the way the SE is described, which was developed in conjunction with the SRM developers; this is shown in Figure 3. However, at the time of writing this is not yet in production use in the LCG/EGEE Grid, so it remains to be seen how well it works in practice.

3.3. Service Discovery

The gLite middleware [18] used in the LCG/EGEE Grid has recently introduced a Service Discovery API using the GlueService part of the GLUE schema. The relative simplicity of the structure of this part of the schema allows a standardised API independent of the underlying information system technology. This is clearly an attractive concept and it would be desirable to extend it to the rest of the schema, but it is not yet clear if it will be feasible to do this for the more complex CE and SE structures.

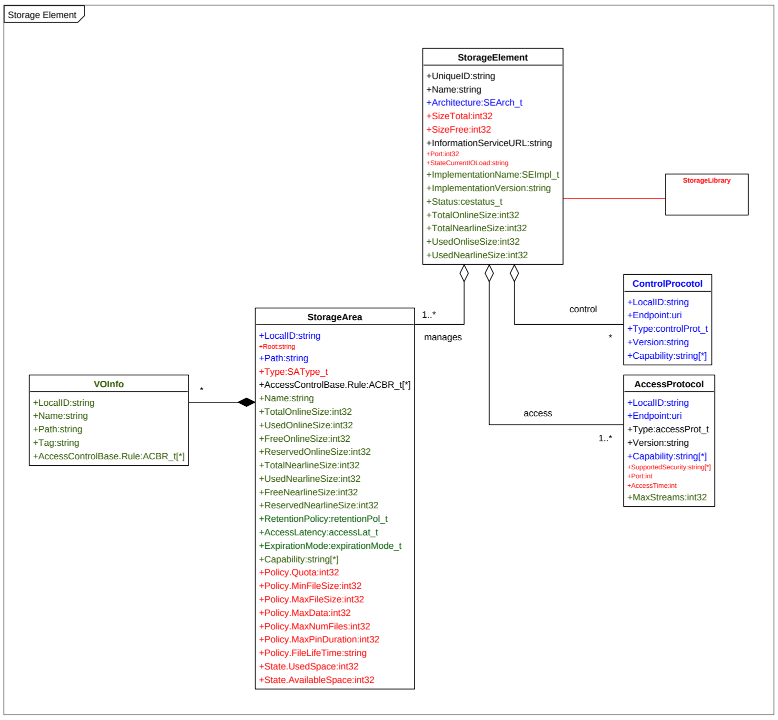


Figure 3. GLUE Schema 1.3 - Storage resources

3.4. Use of the Schema in the LCG/EGEE Grid

The Workload Management System (WMS) exposes a large part of the schema directly to users via the Job Description Language (JDL). This is based on Condor classads, and allows users to describe various attributes of their job. This includes arbitrary expressions involving GLUE schema attributes which can be used either to place requirements which must be satisfied by a CE to be considered for job submission, or to rank CEs to define an order of preference.

The data management middleware also makes substantial use of the schema, but this is less explicitly exposed to users. Queries are made to determine various properties of the SEs, and use is also made of the Service Discovery API to locate file catalogues, File Transfer System services and SRM endpoints. Service Discovery has otherwise had a fairly limited take-up so far, although it can also be used by the WMS client to find servers through which jobs can be submitted.

Some tools are also provided to allow users to query the information system directly. The schema is also used by various Grid monitoring tools like gstat [19] and GridICE[20]. The SE part of the schema is also currently used by a prototype storage accounting system [20], although this is not in general a target use case for the GLUE schema (the CPU accounting tool uses a customised schema and transport to transmit accounting records).

4. Experience and Lessons Learnt

We now have around five years of experience with the GLUE schema, and hence have acquired a good perspective on where the strengths and weaknesses lie. One general conclusion is that the schema attributes must be defined as clearly and unambiguously as possible, since they may be interpreted by many different people working in different areas (middleware developers, system managers, users etc) who may not share a common perspective on the Grid. This applies a fortiori when considering interoperability between different Grids, which may work in significantly different ways. The schema definition should therefore explain the attributes in some detail, together with intended use cases and examples, and give some guidance on how to proceed in cases where the schema assumptions differ from the situation in a real-world system.

4.1. The Computing Element

The biggest problem in the CE area is a mismatch between the schema model and the way the middleware works, and this illustrates that the schema cannot successfully be defined in isolation. The schema model assumed that the computing hardware can be described as a collection of homogeneous SubClusters. However, the WMS currently has no way to match the JDL against multiple SubClusters, or to tell the batch system to run a job on a specific SubCluster. In practice this has meant that we publish only a single SubCluster, and since computing systems are typically heterogeneous this means that the published attributes can only be "representative" in some poorly-defined way. Efforts are now being made to modify the middleware to allow jobs to be targetted to specific SubClusters.

The second major source of difficulty has been the publication of a single set of attributes for a CE, mapped to a single batch queue. In practice most sites define complex policies within a queue, and indeed some batch systems lack a queue concept at all. At a minimum it is usually desirable to have separate fairshares for each VO. This has led sites to typically configure a separate queue per VO, but this makes the system hard to manage and still does not address sharing within a VO, e.g. to separate production work from user jobs. The VOView was defined to try to improve this situation, but to date its use is still limited. This is partly because the version of the WMS which can interpret the VOView information is not yet fully in production, but is also due to the difficulty of matching the varying models used by the WMS, the schema, the various batch systems and by the authorization system (VOMS [21]) which is used to control the mapping to VOs and groups/roles within them. A recent review of this area has concluded that the GLUE model should be adopted by the rest of the middleware.

There are also some ambiguities in the mapping of the schema to real systems, for example whether CPUs are the same as job slots and whether the RAM size attributes are related to the host or the job. The introduction of multicore CPUs may make these questions even more complex. Another difficulty is that the schema has no information about scratch disk space available to running jobs. Such problems will be addressed in future versions of the schema.

4.2. Data Management

In the storage area the main issue is that the technology is comparatively immature. Version 1 of the SRM protocol has only been in production use since 2005, and version 2 is only expected to be deployed in production by the end of 2007. The revisions for version 1.3 of the GLUE schema were largely targeted at SRM version 2, in particular in relation to the concept of "space tokens", storage spaces used for different kinds of data with different storage properties. The changes were made in close co-operation with the SRM group, but experience will need to be gained on how well this works in practice. The ability of the schema to describe the amount of storage space in various ways has also been substantially enhanced, but there is an active debate about what can be feasibly published by the various SRM implementations.

A second area of difficulty is that the data management tools make use of service discovery as well as the main GlueSE information. However, in the current schema these are not well integrated, and there is no well-defined way to link the GlueSE with the corresponding GlueService entries. This will be a major focus for the future evolution of the GLUE schema.

4.3. Grid Monitoring

Monitoring has generally been seen as a somewhat secondary use case for the schema. However, in practice the gstat tool in particular has been one of the main ways to get an overview of the state of the Grid, as well as to provide alerts for anomalies. Monitoring is generally a less demanding use case as the output is intended to be human-readable and precise semantics are less critical.

The current storage accounting prototype also uses the GLUE schema and is in effect another monitoring application. However, it is unclear whether this is a good long-term solution; the provenance of the information is not guaranteed, the information system gives a current snapshot rather than a history so information must be archived continuously, and the granularity is limited (e.g. user-level accounting is not possible).

4.4. Information Providers

The schema is only as good as the information published into it, and hence good-quality information providers are essential. Mapping real-world, complex and varied systems on to a fixed schema inevitably involves compromises and simplifying assumptions, which need to be made in a consistent way to avoid mismatches, hence the development of code for information providers needs to be co-ordinated.

This area has evolved substantially over the years; the current system uses generic code where possible to ensure consistency, with plugins to support specific batch systems, storage systems etc. One exception to this is the lack of a generic information provider for GlueService, which is currently published using a static configuration for most services. However, a generic provider has now been written and is currently in the process of being certified.

The original schema was defined without close consideration of the ease with which the information could be calculated, which has caused problems in some cases. In particular the GlueCE includes an attribute called EstimatedResponseTime (ERT), intended to be an estimate of the time from queuing a new job to the start of execution, which is used as one of the primary ways to determine the best place to submit a job. However, it has proven very difficult to get a good estimate of this quantity, and some calculation methods have led to "black holes" - for example jobs on a CE may be failing right at the start, leading to a small ERT which makes the CE look very attractive. However, the current algorithm has been developed after substantial research and is much more robust.

One general conclusion from this experience is that consideration should be given to the likely use of the attributes, such that errors are fail-safe where possible, i.e. calculations should err in a direction which is likely to make a resource appear unattractive.

A final but critical issue for information providers is the speed of execution. To provide up-to-date information the providers should be run frequently, but this also implies that they should run quickly and should avoid placing a significant load on the system. One possibility is to cache information, but this reduces the currency of the information. A more sophisticated approach may be to cache slowly-changing information and refresh fast-changing information more frequently, but this makes the code more complex. This area is evolving continuously in the light of experience.

4.5. Configuration and Validation

It is essential to have the information providers configured correctly; sites which publish incorrect information will at best be misleading and at worst may act as black holes. Ideally the configuration should be as automated as possible; values set by system managers are vulnerable to typing errors, and also to misunderstandings about what should be published. Where attributes must be configured the documentation needs to be clear. It is also desirable to validate the published information as far as possible; the gstat tool performs a number of sanity checks and raises alarms if the information appears invalid. This implies that the schema should define allowed values as precisely as possible - for example, many textual attributes are defined as enumerated lists with a specific set of allowed values.

One sociological difficulty in this area is that if particular attributes are known to be largely unused, people tend to be less careful about publishing them correctly, and this in turn means that they cannot be used as they are unreliable. A general conclusion is that an effort needs to be made to publish all information correctly if it is published at all, so monitoring tools should validate all published information whether or not it is currently used, and any bugs in the configuration tools should be fixed even if they have no current effect.

4.6. Interpretation

A constant problem is the detailed interpretation of schema attributes, both in terms of what should be published and in the way that the information should be used. In some cases even schema experts can have difficulty in knowing how something should be published in a specific case, and non-experts (which means nearly everyone) may well form incorrect and conflicting views based on partial experience. For example, if an attribute is always observed to have a particular value in practice it may be incorrectly generalised to a belief that it will always have that value. Identifiers can also cause trouble; in most cases GLUE defines UniqueID attributes which are intended to be opaque and uninterpreted. However, in practice these are generally constructed from values like hostnames, and it becomes easy for these to be treated as embedded metadata. In all such cases the defences are clear definitions and comprehensive documentation. If erroneous usage becomes embedded it can be difficult to correct later, so it is important to get things right from the start.

It has also become clear that even apparently trivial questions can be difficult to resolve if the schema does not provide a precise definition. A good example is a long debate over a naming convention for Operating Systems, which took around two years to reach a generally-accepted consensus. Similar issues can arise over units, e.g. the difference between Gb and Gib.

Another interpretation issue relates to optional attributes. In principle almost all attributes defined in the GLUE schema are formally optional, i.e. they may not be published. However, it is not always obvious how to interpret the absence of a particular attribute. In general the schema does not define special values to mean "infinite" or "not applicable", so these may be signaled by an omitted attribute, but equally the omission may simply mean a choice not to publish the attribute at all. It would be desirable to have clearer semantics in this area in future schema versions.

The WMS exposes the GLUE schema attributes directly to users via the JDL, which requires them to have a good understanding of the schema. So far the main users of the LCG/EGEE Grid have been relatively expert, and in general they now have a lot of experience. However, less experienced users are now becoming more common, and it is likely that the schema attributes will have to be hidden by experiment frameworks or portals if mistakes are to be avoided. A particular problem is with attributes which are "usually" not relevant, e.g. limits on the number of queued jobs or on the wallclock time they consume. These are often ignored, which may cause problems with particular sites, or with future changes - for example the OS has always been Linux, but Windows systems are likely to appear in the near future and are likely to be unusable by many jobs!

4.7. Backward Compatibility

Deploying a new version of the schema is a slow and gradual process. Some Grid sites may take many months to deploy software updates, and client software, both middleware and user software, also changes slowly. To date it has therefore been decided to make all schema upgrades completely backward-compatible. However, this imposes quite severe restrictions on the changes which can be made, and also results in the accumulation of deprecated objects and attributes. Backward-compatibility may also reduce the pressure on sites and client code to follow upgrades in a timely way. The GLUE 2.0 upgrade is intended to be non-backward-compatible to allow a complete redesign. However, it will probably be necessary to run both schemas in parallel for some time, and it remains to be seen how well this works in practice.

One partial solution to this problem is to have provision within the schema for generic attributes which allow for prototyping new things without changing the schema itself. The simplest case is to have a multivalued free-text attribute, which has been used successfully with the CE to publish information about installed software and other local properties not otherwise catered for in the schema. This concept has now also been added to the storage schema. A somewhat more elaborate possibility is to have generic key-value pairs, which is currently available as part of the GlueService information. However, to date the use of this has been fairly limited.

5. Conclusions

The GLUE schema has been in use in the EDG and LCG/EGEE Grids since 2002, and has been upgraded twice in that period. It has been successfully used for job submission, data management and monitoring, and in general has been very successful; there have been no showstoppers. However, experience has also uncovered some problems, and evolution of the Grid technology itself also implies changes in the schema; these changes need to be carefully managed to avoid disruption of the existing service. Care needs to be taken to specify the schema attributes precisely, to configure and publish them correctly, and to use them with an understanding of their meaning and limitations. We now have enough experience to work towards a major upgrade of the GLUE schema in the OGF framework [13], which should unify all relevant modeling efforts in the Grid context and it should provide a firm foundation for Grid interoperability.

References

[1] LCG: LHC Computing Grid http://www.cern.ch/lcg
[2] EGEE: European Grid for E-sciencE http://www.eu-egee.org
[3] OSG: Open Science Grid http://www.osg.org
[4] Nordic Grid Facility http://www.ndgf.org
[5] EU DataGrid Project http://cern.ch/eu-datagrid
[6] EU DataTAG Project http://cern.ch/datatag
[7] iVDGL: International Virtual Data Grid Laboratory http://www.ivdgl.org
[8] The Globus Project http://www.globus.org
[9] 2003 GLUE Schema Specification - Version 1.1 http://glueschema.forge.cnaf.infn.it/Spec/V11
[10] Andreozzi S, Burke S, Field L, Fisher S, Kónya B, Mambelli M, Schopf J, Viljoen M and Wilson A 2005 GLUE Schema Specification - Version 1.2
[11] Andreozzi S, Burke S, Donno F, Field L, Fisher S, Jensen J, Kónya B, Litmaath M, Mambelli M, Schopf J, Viljoen M, Wilson A and Zappi R 2007 GLUE Schema Specification - Version 1.3 - draft 3
[12] Open Grid Forum http://ogf.org
[13] Andreozzi S, Burke S, Field L and Kónya B 2007 *Proceedings of the International Conference on Computing in High Energy and Nuclear Physics (CHEP2007), Victoria, Canada*
[14] Wahl W, Howes T and Kille S OpenLDAP http://www.openldap.org/
[15] Cooke A, Gray A, Ma L, Nutt W, Magowan J, Taylor P, Byrom R, Field L, Hicks S and Leak J e A 2003 *Proceedings of the Eleventh International Conference on Cooperative Information Systems*
[16] Condor Project http://www.cs.wisc.edu/condor/
[17] SRM: Storage Resource Manager http://sdm.lbl.gov/srm-wg/documents.html
[18] Laure E *et al.* 2006 Programming the Grid with gLite Tech. Rep. EGEE-TR-2006-001 CERN
[19] GSTAT http://goc.grid.sinica.edu.tw/gstat/
[20] Aiftimiei C, Andreozzi S, Cuscela G, De Bortoli N, Donvito G, Fattibene E, Misurelli G, Pierro A, Rubini G and Tortone G 2006 *In Proceedings of the International Conference on Computing in High Energy and Nuclear Physics (CHEP2006), Mumbai, India*
[21] Alfieri R, Cecchini R, Ciaschini V, dell'Agnello L, Frohner A, Gianoli A, Lörentey K and Spataro F 2004 *Proceedings of the 1st European Across Grids Conference, Santiago de Compostela, Spain, February 2003, LNCS* **2970** 33–40