# Storage Resource Manager Version 2.2: design, implementation, and testing experience

**Flavia Donno**[(1)]

**Co-authors: Lana Abadie[(1)],Paolo Badino[(1)], Jean-Philippe Baud[(1)], Ezio Corso[(2)], Shaun De Witt[(3)], Patrick Fuhrmann[(4)], Junmin Gu[(5)], Birger Koblitz[(1)], Sophie Lemaitre[(1)], Maarten Litmaath[(1)], Dimtry Litvintsev[(7)], Giuseppe Lo Presti[(1)], Luca Magnoni[(6)], Gavin McCance[(1)], Tigran Mkrtchan[(4)], Rémi Mollon[(1)], Vijaya Natarajan[(5)], Timur Perelmutov[(7)], Don Petravick[(7)], Arie Shoshani[(5)], Alex Sim[(5)], David Smith[(1)], Paolo Tedesco[(1)], Riccardo Zappi[(6)]**

[(1)]European Organization for Nuclear Research CERN G06910, CH-1211 Genève 23, Switzerland, [(2)]Abdus Salam International Centre for Theoretical Physics (ICTP), Strata Costiera 11, 34014 Trieste - Italy, [(3)]Rutherford Appleton Laboratory (RAL), Harwell Science and Innovation Campus Didcot OX11 0QX, United Kingdom, [(4)]Deutsches Elektronen-Synchrotron (DESY), Notkestraße 85, 22607 Hamburg, Germany, [(5)]Lawrence Berkeley National Laboratory (LBNL), 1 Cyclotron Road Mail Stop, Berkeley, CA, USA, [(6)]Centro Nazionale per la Ricerca e Sviluppo nelle Tecnologie Informatiche e Telematiche (CNAF) , Viale Berti Pichat 6/2, 40127 Bologna, Italy, [(7)]Fermi National Accelerator Laboratory (FNAL), P.O. Box 500 Batavia, IL 60510-5011, USA

Flavia.Donno@cern.ch

**Abstract**. Storage Services are crucial components of the Worldwide LHC Computing Grid Infrastructure spanning more than 200 sites and serving computing and storage resources to the High Energy Physics LHC communities. Up to tens of Petabytes of data are collected every year by the four LHC experiments at CERN. To process these large data volumes it is important to establish a protocol and a very efficient interface to the various storage solutions adopted by the WLCG sites. In this work we report on the experience acquired during the definition of the Storage Resource Manager v2.2 protocol. In particular, we focus on the study performed to enhance the interface and make it suitable for use by the WLCG communities. At the moment 5 different storage solutions implement the SRM v2.2 interface: BeStMan (LBNL), CASTOR (CERN and RAL), dCache (DESY and FNAL), DPM (CERN), and StoRM (INFN and ICTP). After a detailed inside review of the protocol, various test suites have been written identifying the most effective set of tests: the S2 test suite from CERN and the SRM-Tester test suite from LBNL. Such test suites have helped verifying the consistency and coherence of the proposed protocol and validating existing implementations. We conclude our work describing the results achieved.

## 1. Introduction

The Worldwide LHC Computing Grid (WLCG) [1] Infrastructure is the largest Grid in the world, including about 230 sites worldwide [2]. It has been mainly established to support the 4 Large Hadron Collider (LHC) experiments at CERN. The LHC is the world's biggest machine to study the fundamental properties of sub-atomic particles and is due to start operating in 2008.

The goal of the WLCG project is to establish a world-wide Grid infrastructure of computing centers to provide sufficient computational, storage and network resources to fully exploit the scientific potential of the four major experiments operating on LHC data: Alice, ATLAS, CMS and LHCb. These experiments will generate enormous amounts of data (10-15Petabytes per year). Computing and storage services to analyze them would be implemented by a geographically distributed Data Grid.

Given the variety of the storage solutions adopted by the sites collaborating in the WLCG infrastructure, it was considered important to provide an efficient and uniform Grid interface to storage and allow experiments to transparently access the data, independently of the storage implementation available at a site. This effort has given rise to the Grid Storage Management Working Group (GSM-WG) at the Open Grid Forum (OGF) [3].

In what follows we report on the experience acquired during the definition of the Storage Resource Manager (SRM) v2.2 protocol. In particular, we focus on the study performed to enhance the interface and make it suitable for use by the WLCG communities.

In Section 2 we elaborate on the protocol definition process and on the collection of the requirements as described by the LHC experiments. In Section 3 we talk about version 2.2 of the SRM protocol as it is defined today and implemented by multiple storage solution providers. In Section 4 we describe the storage solutions that provide SRM v2.2 compliant interfaces today. Section 5 summarizes the activities that led to the design of the static and dynamic models. To validate the protocol against the user requirements and verify the compliance of the several implementations against the specification, the need of test suites was recognized. In Section 6 we report on the study of the protocol and the design of test suites. In order to achieve concrete results it was important to establish a testing framework: the S2 testing framework is described in Section 7. Exhaustive tests can demonstrate the correctness of the system under test (SUT). However, exhaustive tests are impracticable. Black box techniques have therefore been applied to reduce the number of tests. Section 8 elaborates on that. In Section 9 we give a summary of the CERN S2 families of tests available today and show the results obtained. In Section 10 we report about other testing efforts performed in order to check the usability of the high-level tools and APIs available in WLCG. In the conclusion we report on related work and outline the main achievements.

## 2. Storage Requirements

In the second quarter of 2005 the WLCG Baseline Service working group [4] was established in order to understand the experiment requirements for the first data challenges. A data challenge is a set of tests to establish the extent of readiness and functionality of the computing frameworks of the LHC experiments. The report of the working group based on early experience was published [5]. In the report the definition of a "Storage Element" is given and the main functionalities of a storage service and the needed file access protocols are listed together with target dates for their availability.

However, during the Mumbai workshop preceding the CHEP conference in 2006, it was clear that the LHC experiments had learned more about what was needed in terms of storage and changed their requirements. The WLCG Data Management Coordination group was therefore established to discuss the last open points and converge to a final agreement. It was only in May 2006 during the Grid Storage Workshop at Fermilab that the final WLCG Memorandum of Understanding for a Grid Storage Service [6] was agreed to by experiments and Grid Storage Service providers.

At the end of 2006, a second working group was established in WLCG, the Storage Class Working Group. The mandate of this working group was to understand definition and implementation issues of the qualities of storage demanded by the experiments. It has evolved into the Grid Storage System

Deployment (GSSD) working group [7], in charge of coordinating the deployment into production of the new Grid Storage Services.

### *2.1. The Storage Element*

A Storage Element (SE) is a logical entity that includes:

- A mass storage system (MSS) which is defined by disk-based hardware, or some disk cache front-end backed by a tape system.
- A storage interface to provide a common way to access the specific MSS.
- A GridFTP service to provide data transfer in and out of the SE, to and from the Grid. The implementation of this service must scale to the bandwidth required.
- Local POSIX-like input/output calls providing application access to the data on the SE.
- Authentication, authorization and audit/accounting facilities, based on ACLs and Virtual Organization Management System (VOMS) roles and groups [8].

A site may provide multiple SEs with different qualities of storage. For example, it may be considered convenient to provide an SE for data intended to remain for extended periods and a separate SE for data that is transient – needed only for the lifetime of a job or set of jobs. Since most applications will not communicate with the storage system directly, but use higher-level applications, it is clear that these applications must also be enabled to work with storage interfaces.

### *2.2. The Storage Service Interface*

The WLCG Baseline Service Working Group has defined a set of required functionalities that all SE services must implement before the start of LHC operations. In what follows we describe the main characteristics of the WLCG Grid Storage Interface.

*File types*

The need to introduce the concept of several file types has been recognized:

- **Volatile**: it is a temporary and often sharable copy of an MSS resident file. If space is needed, an unused temporary file can be removed by the MSS garbage collector daemon. Applications using the file can impose or extend the lifetime of the file ("pinning"). If the lifetime is still valid a file normally would not be automatically removed by the system.
- **Permanent**: it is a file that can be removed only by the owner or other authorized users. Files in WLCG are permanent (but a file may have a number of volatile copies as well).

*Space types*

A need was expressed to be able to operate on a set of files at once. This can be achieved introducing the concept of space. A logical space is a container that hosts a set of files. Two categories of space have been defined corresponding to the two file types. The meaning of each type being similar to the one defined for files. Spaces in WLCG are mostly permanent.

*Space reservation*

Experiments require the ability to dynamically reserve space to ensure that a store operation does not fail. Space can be statically reserved by site administrators explicitly allocating storage hardware resources to a Virtual Organization (VO), or in advance by VO managers (via some administrative interface), or at run time by generic VO users. The reservation has a lifetime associated with it. The user is given back a space token that he/she will provide in the following requests to retrieve a file from tape or from another remote storage system, or to write a new file. The space can be released either by the user/VO manager or by the system when the lifetime associated with the space expires.

*Permission and Directory functions*
Similar to POSIX Access Control Lists (ACLs), permissions may be associated with directories or files. LHC VOs desire storage systems to respect permissions based on VOMS groups and roles. ACLs are inherited from the parent directory, by default.
For administrative reasons, functions to create/remove directories, delete files, and rename directories or files are needed. File listing functions are also needed. However, it was felt that there is no need for "mv" operations between two different SEs.

*Data transfer control functions*
These functions do not normally move any data but prepare the access to data. These are the functions that deal with the MSS stagers. The only exception is the copy function that moves data between two Storage Elements. All these functions can operate on a set of files and they report a status or error condition for each file.

*Other requirements*
Experiments have expressed the need to refer to directory paths with respect to the VO base directory. This is to avoid finding out per site the root assigned by the site to a specific VO. The storage interface should discover the correct namespace for a given VO. This also allows catalogue entries to be shorter. The need to be able to send multiple requests to the MSS and to allow the MSS to handle priorities, and to optimize tape access was deemed as essential by the storage system managers, and recognized by the experiments. The use of the storage interface methods guarantees this ability, and would be the preferred method of initiating file transfers.
It is vital that all the storage interface implementations interoperate seamlessly with each other and appear the same to applications and grid services. To this end a test-suite has to be used to validate implementations against the WLCG agreed set of functionality and behaviour.

### *2.3. The Storage Classes*

In WLCG the Storage Class Working Group was in charge to understand the requirements of the LHC experiments in terms of quality of storage (Storage Classes) and the implementation implications for the various storage solutions available. For instance, this implied the study of how to assign disk pools for LAN or WAN access and to devise common configurations for VOs and per site.

A **Storage Class** *determines the properties that a storage system needs to provide in order to store data.*

The LHC experiments have asked for the availability of combinations of the following storage devices: Tapes (or reliable storage system always referred to as tape in what follows) and Disks. If a file has at least a copy on Tape then we say that the file is in Tape1. If a file resides on an experiment-managed disk, we say that the file is in Disk1. Tape0 means that the file does not have a copy stored on a reliable storage system. Disk0 means that the disk where the copy of the file resides is managed by the system: if such a copy is not pinned or it is not being used, the system can delete it.

Following what has been decided in various WLCG Storage Class Working Group meetings and discussions only the following combinations (or Storage Classes) are needed and therefore supported:

- In the **Custodial-Nearline** storage class, data is stored on some reliable secondary storage system (such as a robotic tape or DVD library). Access to data may imply certain latency. When a user accesses a file, the file is recalled in a cache that is managed by the system. The file can be "*pinned*" for the time the application needs the file.
- In the **Custodial-Online** storage class data is always available on disk. A copy of the data resides permanently on tape, DVD or on a high-quality RAID system as well. The space owner (the virtual organization) manages the space available on disk. If no space is available

in the disk area for a new file, the file creation operation fails. This storage class guarantees that a file is never removed by the system.

- The **Replica-Online** storage class is implemented through the use of disk-based solutions not necessarily of high quality. The data resides on disk space managed by the virtual organization.

## 3. The Storage Resource Manager protocol version 2.2

A **Storage Resource Manager** (SRM) is a middleware component whose function is to provide dynamic space allocation and file management on shared storage components on the Grid [9].

An SRM is a standardized interface offered by a Storage Element that satisfies most of the requirements explained in Section 2. It allows authorized clients to initiate requests for allocating and managing a given amount of space with a given quality. The space can be used to create new files or access existing files stored in the underlying storage system.

The v2.2 SRM interface functions or methods are described in detail in [10] and can be categorized in five families: space management functions, permission functions, directory functions, data transfer functions, and discovery functions.

### *3.1. Space Management Functions*

**Space management functions** allow the client to reserve, release, and manage spaces, their types and lifetime. Once the space is assigned, it can be referred to with a space token. The main space management functions are: *srmReserveSpace, srmUpdateSpace, srmReleaseSpace, srmChangeSpaceForFiles, srmExtendFileLifeTimeInSpace.*

### *3.2. Data Transfer Functions*

**Data transfer functions** have the purpose of retrieving files from and getting files into SRM spaces, to/from the client's computer or other remote storage systems on the Grid. Data transfer functions support requests for multiple files within a single transaction. These functions constitute the core of the SRM specification. The main methods are the following ones: *srmPrepareToGet, srmBringOnline, srmPrepareToPut, srmPutDone, srmCopy, srmReleaseFiles srmAbortRequest/srmAbortFiles, srmExtendFileLifeTime.*

### 3.3. *Other Functions*

Other SRM functions include:

- **Directory functions** which are similar to their equivalent UNIX functions. The SRM interface offers a namespace, which is similar to that of a filesystem. In particular, files have a path and a filename. The SRM directory functions allow users to put files with different storage requirements in a single directory. Changing the space characteristics of a file does not change its position in the directory but only its assignment to the corresponding space.
- **Permission functions** which allow a user to assign read and write privileges on a specific file to other users. Such functions allow client applications to specify ACLs as well, wherever supported by the underlying storage service.
- **Discovery functions** which allow applications to query the characteristics of the storage system behind the SRM interface and the SRM implementation itself.

## 4. Storage Solutions

In order to define a successful Grid Storage Protocol and Interface, it is important that the developers of Grid Storage Services get involved in the definition process and provide prototype implementations of the proposed protocol early enough to allow for the discovery of inefficiencies in the protocol and incoherence in the interface. In what follows we give a brief overview of the Storage solutions adopted in WLCG that provide an SRM v2.2 implementation as of today.

### *4.1. dCache*

dCache [11] is a Grid storage service jointly developed by DESY and Fermilab. It can manage very large numbers of disks and supports various tape back-ends.

### *4.2. DPM*

The LCG lightweight Disk Pool Manager (DPM) [12] is a complementary solution to the dCache system. It focuses on manageability at smaller sites and therefore has been made easy to install and configure. It only supports disk storage at this time.

### *4.3. BeStMan*

The Berkeley Storage Manager (BeStMan) [13] has been developed by LBNL primarily to manage disk storage, while also supporting HPSS as tape back-end.

### *4.4. StoRM*

The StoRM [14] system implements a Grid storage solution based on parallel or distributed file systems. StoRM is the result of a research collaboration between INFN and ICTP (International Centre for Theoretical Physics, in Trieste, Italy) to build a disk-based SRM service on top of high-performance parallel file systems and at the same time provide a pilot national Grid facility for research in economics and finance.

### *4.5. CASTOR*

CASTOR [15], the Cern Advanced STORage system, is a scalable, high throughput storage management system that offers support for several tape back-ends.

## 5. The SRM Static and dynamic model

The definition process of SRM v2.2 has been quite long, involving many discussions around the concepts of spaces, file copy management, and name space handling. Confusion arose from the absence of a clear data model for SRM. The SRM was in fact specified in terms of its application programming interface that reflected the requirements of storage system users. The need for a domain analysis model was recognized later. The model described in [16] supplements the API and other specifications with an explicit, clear and concise definition of its underlying structural and behavioural concepts; it makes it easier to define the semantics; it helps service developers and providers for a more rigorous validation of implementations; it helps identifying unanticipated behaviors and interactions.

## 6. The SRM testing suites

Another important aspect in the definition of a protocol and in checking its efficiency is the verification against various implementations. The verification process helps to understand if foreseen transactions make sense in the real world. It shows as well if the protocol adapts naturally and efficiently to existing storage solutions. These reasons lead to setting up a test bed where real use cases can be tested. In what follows we describe the design of a functional test suite to verify the compliance of the implementations with the defined interface and their interoperability. In particular, an analysis of the complexity of the proposed SRM interface shows that a large number of tests need to be executed in order to fully check the compliance of the implementations to the specifications. An appropriate testing strategy has to be adopted to allow a finite set of tests to provide a sufficiently large coverage.

## 7. The S2 testing framework

In order to speed up the development of the testing suite, we looked for a testing framework with the following characteristics:

- Support for quick development of single test cases

- Minimization of human coding errors.
- Plug-in support for external libraries such as an SRM client implementation.
- A powerful engine for parsing the output of a test program.
- Support for both parallel and sequential execution of testing units; support for timers at instruction and test unit level.
- "self-describing" logging facility.

We examined several products available on the market based on the TTCN-3 standard for telecommunication protocols. Such products often had the problem of being protected by license or being too complex and very much oriented to the telecommunication world. In the end we focused on S2 [17], a tree-based language with SRM 2.2 protocol support developed initially at RAL specifically for executing tests.

An S2 tree consists of command **branches**. The S2 interpreter starts the evaluation of an S2 tree at its root: the first branch without indentation. Branches are in relationship with other branches (parents, children or disconnected). Together with a set of optional parameters, a specific *action* defines the branch.

A fundamental action is the execution of an SRM command. The S2 interpreter recognizes a set of expressions with a command-line syntax that exercise all the implemented SRM calls. The output of an SRM command is stored in variables. By use of regular expressions, the occurrence of given patterns in the output can be searched for.

S2 has allowed us to build a testing framework that supports the parallel execution of tests where the interactions among concurrent method invocations can be tested easily. The S2 test suite has allowed for the early discovery of memory corruption problems and race conditions. The coding of such test cases required very little time (a few minutes).

## 8. The black box techniques

In order to verify the compliance of a specific implementation to a protocol a test-case-design methodology known as **Black Box** or **functional testing** is often adopted. Functional testing is used for:

- ***Validation***: of the system against the explicit and implicit user requirements
- ***Consistency***: checking for inconsistency, incompleteness, or inefficiency
- ***Verification***: correctness of implementations with respect to the specification
- ***Performance*** and reliability

Exhaustive testing can demonstrate the correctness of the System Under Test (SUT). However, such an approach is impracticable given the infinite number of test cases. Therefore, the Black Box testing technique focuses on identifying the subset of all possible test cases with the highest probability of detecting the most errors. The specifications are examined to find the input domain of the system, the operating conditions that affect the system behavior and input-output relationships that define how the system reacts to inputs under given operating conditions.

The most popular black box testing approaches are **Equivalence partitioning**, **Boundary-value analysis**, **Cause-effect graphing** and **Error guessing** [18]. Each of these approaches covers certain cases and conditions but they do not ensure the identification of an exhaustive testing suite. However, one of them can be more effective than the other ones depending on the nature of the SUT.

## 9. The S2 test families and results

The S2 testing framework and the described black box testing approaches have been used to design a test suite to validate the correctness of the implementations with respect to the protocol established. We proceeded dividing the test suite in families that probed the SUTs with an increasing level of detail. In particular, we have designed and developed six families of tests:

- **Availability**: the **srmPing** function and a full put cycle for a file are exercised.

- **Basic**: the test suite checks the basic SRM v2.2 functionality, testing that the expected status codes are returned when passing simple input parameters. Here the equivalence partitioning and boundary condition analysis is applied.
- **Use cases**: boundary conditions, exceptions, real use cases extracted from the middleware clients and experiment applications are tested. The cause-effect graphing technique is used to reduce the number of test cases.
- **Interoperability**: remote operations (servers acting as clients) and cross copy operations among several implementations are exercised.
- **Exhaustive**: These tests check the reaction of the system when long strings or strange characters are passed as input arguments or when mandatory arguments are missing or optional arguments are improperly used.
- **Stress**: Parallel tests are used for stressing the systems, sending multiple requests, concurrent colliding requests, space exhaustion, etc.

The S2 tests have been run typically *5 times per day* against up to *21 SRM v2.2 endpoints* running 5 different storage services with *different configurations*. The S2 test suite has been integrated in the *WLCG certification process. S2* has been used as well by the developers to check their implementations and distributed to the USA Open Science Grid project for Tier-2 validation. The Test families have been expanded continuously to cover new scenarios, new decisions taken at the protocol level, and to check for discovered bugs or race conditions.

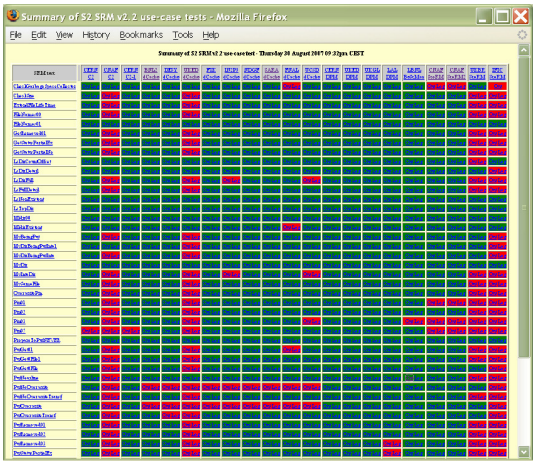


Figure 1. Web pages showing the results of use case tests performed against storage implementations installed at several sites

Figure 2 plots the results of basic tests for all implementations. The number of failures over the number of total tests executed is reported over time. After a long period of instability, the implementations converged on a large set of methods correctly implemented. We note that new features and bug fixes introduced by the developers together with the addition of new tests produced oscillations in the plot. Toward the end of the testing period the number of failures is almost zero for all implementations. That is when large-scale testing can stop.

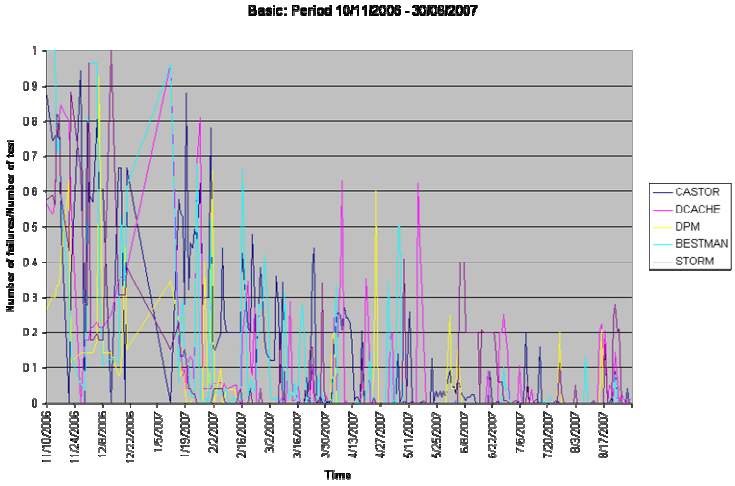


Figure2. Plot showing the number of failures/number of basic tests performed over a period of time that goes from November 2006 to August 2007

## 10. Other tests

Together with the S2 families of test suites, other testing efforts contributed to the success of the SRM v2.2 protocol. We explicitly mention the LBNL SRM-Tester written in Java, the only test suite available since the first version of the SRM v2.2 specification.

Other testing efforts have contributed to:

- Check the status of the high-level WLCG SRM v2.2 clients such as GFAL, lcg-utils, FTS
- Verify the transparent access to storage independent of the protocol in use (SRM v1.1 vs SRM v2.2).
- Experiment data access patterns.
- Verify the correct handling of VO specific proxies with VOMS groups/roles.
- Verify the compliance to user requirements and functionalities needed
- Check the availability of required bindings (C++, Python, Perl, etc.) and support for mandatory platforms (Scientific Linux 4 in 32-bit mode, 64-bit mode later).

The main outcomes of these activities are:

- User documentation, education and training.
- Preparation for the integration of experiment frameworks with the SRM v2.2 environment.

## 11. Conclusions and related work

Storage management and access in a Grid environment is a rather new topic that is attracting the attention of many scientists. Given the heterogeneity of the existing storage solutions with their constant evolution and the complexity of the functionality required by Grid applications, a completely satisfactory solution to storage in the Grid has not yet been proposed.

There are many attempts made in this direction. For instance, the “Grid storage” initiative promoted by IBM and proposed in the Open Grid Forum's File System Working Group aims at creating a topology for scaling the capacity of NAS in response to application requirements, and enabling a single file system. However, the issue of offering a transparent interface to data stored on different media with different access latencies and management technologies is not tackled.

The Internet Backplane Protocol (IBP) [19] aims at providing a virtual publicly sharable storage all over the world. The main idea is to make an online storage available for public usage. Therefore, quality of storage and transparent access to any kind of MSS is not normally a concern of IBP.

We have introduced the SRM proposed by the OGF SRM-WG that aims at proposing a control protocol for storage management and access in the Grid. The SRM offers important functionalities

such as the possibility of dynamically reserving space of a certain quality, dynamic space and file management, access to system optimization patterns, the negotiation of file access and transfer protocols between client and server, optimized copy operations, uniform namespace management functions, file permission functions, transparent data access, etc.
Designing a protocol and achieving a wide acceptance is a long and time-consuming effort that has evolved in a strong international collaboration. The experience acquired with the design and implementation of SRM v2.2 has allowed us to acquire fundamental knowledge in terms of protocol design and specification techniques.

**References**


[1] Worldwide LHC Computing Grid Project Web site: http://www.cern.ch/lcg
[2] WLCG Site Functional Tests Web Page: https://lcg-sft.cern.ch/sft/lastreport.cgi
[3] OGF web site: http://www.ggf.org/
[4] The WLCG Baseline Service Working Group http://cern.ch/lcg/PEB/BS
[5] I. Bird, J. Andreeva, L. Betev, M. Branco, P. Charpentier, A. De Salvo, F. Donno, D. Duellmann, P. Elmer, S. Lacaprara, E. Laure, R. Popescu, R. Pordes, M. Schulz, S. Traylen, A. Tsaregorodtsev, A. Wannanen *The WLCG Baseline Service Working Group Report* v1.0, 24 June 2005 http://lcg.web.cern.ch/LCG/peb/bs/BSReport-v1.0.pdf
[6] The WLCG Memorandum of Understanding for a Grid Storage Service, FNAL, May 2006 http://cd-docdb.fnal.gov/0015/001583/001/SRMLCG-MoU-day2%5B1%5D.pdf
[7] https://twiki.cern.ch/twiki/bin/view/LCG/GSSD
[8] V. Ciaschini, A. Frohner *Voms Credential Format* http://edgwp2.web.cern.ch/edg-wp2/security/voms/edg-voms-credential.pdf
[9] A. Shoshani, P. Kunszt, H. Stockinger, K. Stockinger, E. Laure, J.-P. Baud, J. Jensen, E. Knezo, S. Occhetti, O. Wynge, O. Barring, B. Hess, A. Kowalski, C. Watson, D. Petravick, T. Perelmutov, R. Wellner, J. Gu, A. Sim *Storage Resource Management: Concepts, Functionality, and Interface Specification*, GGF 10, The Future of Grid Data Environment, 9-13 March 2004, Humboldt University, Berlin Germany
[10] A. Sim et al., *SRM v2.2 Specification*, 2 April 2006, http://sdm.lbl.gov/srm-wg/doc/SRM.v2.2.html
[11] M. Ernst, P. Fuhrmann, T. Mkrtchyan, J. Bakken, I. Fisk, T. Perelmutov, D. Petravick, *Managed data storage and data access services for Data Grids* CHEP, La Jolla, California, March 2004
[12] LCG Disk Pool Manager (DPM): https://twiki.cern.ch/twiki/bin/view/LCG/DpmAdminGuide
[13] A. Shoshani, A. Sim, J. Gu *Storage Resource Managers: Middleware Components for Grid Storage*, 9th IEEE Symposium on Mass Storage Systems, 2002
[14] E. Corso, S. Cozzini, F. Donno, A. Ghiselli, L. Magnoni, M. Mazzucato, R. Murri, P. Ricci, H. Stockinger, A. Terpin, V. Vagnoni, R. Zappi. *StoRM, an SRM Implementation for LHC Analysis Farms*, Computing in High Energy Physics *(CHEP 2006)*, Mumbai, India, Feb. 13-17, 2006.
[15] CASTOR: CERN Advanced STORage manager - http://castor.web.cern.ch/castor
[16] F. Donno, A. Domenici, *A Model for the Storage Resource Manager,* ISGC 2007, International Symposium on Grid Computing, Taipei, March 26 . 29, 2007
[17] F. Donno, J. Mençak *The S2 testing suite* 15 September 2006, http://s-2.sourceforge.net
[18] G. J. Myers, C. Sandler (Revised by), T. Badgett (Revised by), T. M. Thomas (Revised by) *The ART of SOFTWARE TESTING 2 edition*, December 2004, ISBN 0-471-46912-2 nd
[19] J. Plank, M. Beck, W. Elwasif, T. Moore, M. Swany, and R. Wolski The Internet Backplane Protocol: Storage in the network in Network Storage Symposium, 1999.