

DELIVERABLE D3.1
OPTIMIZATION OF DATA ACCESS
TASK 3.4
SRS

WP 3, Task 3.4 New Grid Services and Tools

Document Filename:	CG-3.4-SRS-0012.doc
Work package:	WP 3, Task 4
Partner(s):	CYFRONET
Lead Partner:	CYFRONET
Config ID:	CG-3.4-SRS-0012-1-2
Document classification:	PUBLIC

Abstract: This document specifies preliminary requirements specification for task 3.4 – ‘Optimization of Data-Access’. Task 3.4 provides services to help users in their interaction with large data manipulation; we propose a kind of expert system for sophisticated data-handlers selection, allows building flexible environments for data storing and serving. Additionally, within this task a common middleware for fast tape file access and a system for data access time estimation will also be developed.



Delivery Slip

	Name	Partner	Date	Signature
From	WP3, task 4	CYFRONET	6 May 2002	
Verified by				
Approved by				

Document Log

Version	Date	Summary of changes	Author
1-0-DRAFT-C	5 Apr 2002	Draft version	Jacek Kitowski, Łukasz Dutka, Renata Słota, Darin Nikolow
1-1-DRAFT-A	5 May 2002	Implementation of remarks	Jacek Kitowski, Łukasz Dutka, Renata Słota, Darin Nikolow
1-1-FINAL-B	6 May 2002	Corrections	Jacek Kitowski, Łukasz Dutka, Renata Słota, Darin Nikolow
1-2	25 May 2002	After Internl Review Corrections	Jacek Kitowski, Łukasz Dutka, Renata Słota, Darin Nikolow

CONTENTS

1. INTRODUCTION	5
1.1. PURPOSE	5
1.2. SCOPE	5
1.3. DEFINITIONS, ACRONYMS, AND ABBREVIATIONS	5
1.4. REFERENCES	6
1.5. OVERVIEW	7
2. OVERALL DESCRIPTION	8
2.1. PRODUCT PERSPECTIVE	8
2.1.1. <i>Component-Expert Subsystem</i>	11
2.1.2. <i>Tape Resident Large Files Middleware - TRLFM</i>	15
2.1.3. <i>Data Access Estimation</i>	17
2.1.4. <i>System interfaces</i>	18
2.1.5. <i>User interfaces</i>	18
2.1.6. <i>Hardware interfaces</i>	18
2.1.7. <i>Software interfaces</i>	18
2.1.8. <i>Communications interfaces</i>	19
2.1.9. <i>Memory constraints</i>	19
2.1.10. <i>Operations</i>	19
2.1.11. <i>Site adaptation requirements</i>	19
2.2. PRODUCT FUNCTIONS	19
2.3. USER CHARACTERISTICS	20
2.4. CONSTRAINTS	20
2.5. ASSUMPTIONS AND DEPENDENCIES	21
2.6. APPORTIONING OF REQUIREMENTS	21
2.7. DEVIATION BETWEEN TECHNICAL ANNEX AND CURRENT STATE OF TASK 3.4	21
2.8. TASK 3.4 ADDED VALUE	21
3. SPECIFIC REQUIREMENTS	23
3.1. EXTERNAL INTERFACES	24
3.1.1. <i>Component - Storage Element (3.4)</i>	25
3.1.2. <i>Component - Replica Manager (DataGrid / 3.4)</i>	25
3.1.3. <i>Component - Data Access Estimator (3.4)</i>	26
3.1.4. <i>Component - GridFTP Plugin (3.4)</i>	26
3.2. INTERNAL INTERFACES	26
3.2.1. <i>Component - Component-Expert Subsystem (3.4)</i>	26
3.2.2. <i>Component - Storage Configuration (3.4)</i>	27
3.2.3. <i>Component - TRLFM (3.4)</i>	27
3.3. FUNCTIONS – USE CASES	28
3.4. PERFORMANCE REQUIREMENTS	29
3.5. LOGICAL DATABASE REQUIREMENTS	29
3.6. DESIGN CONSTRAINTS	29
3.7. STANDARDS COMPLIANCE	29
3.8. SOFTWARE SYSTEM ATTRIBUTES	29
3.9. REQUIREMENTS ANALYSIS	29
4. APPENDIX	37
4.1. EXTERNAL COMPONENTS SPECIFICATION (NOT INCLUDED IN TASK 3.4)	37
4.1.1. <i>ComponentName - GridFTP</i>	37
4.1.2. <i>ComponentName - Portal</i>	37
4.1.3. <i>ComponentName - Replica Catalog</i>	37
4.1.4. <i>ComponentName - HSM</i>	39
4.1.5. <i>ComponentName - Secondary Storage System</i>	40

TASK 3.4 SRS
Optimization of Data Access

4.1.6. ComponentName - Tape Storage	40
4.1.7. ComponentName - Disk Cache	40
4.1.8. ComponentName - MO Disk storage	40
4.1.9. ComponentName - Metadata Catalog	40
4.1.10. ComponentName - Application	40
5. INDEX	41

1. INTRODUCTION

1.1. PURPOSE

This document is aimed to describe requirements, which inspire and influence form of the Data Access Package. The document is intended to be a base for the final software design for the task 3.4 and its cooperation with other tasks. It should be helpful to formulate coherent vision of the CrossGrid environment.

1.2. SCOPE

In the document we present the proposed architecture for data-access and optimization problems for the CrossGrid Project. Dependencies between system components and connections between them are shown, with some more detailed specifications of interfaces. The architecture relies mainly on SOAP and GridFTP protocols with possible extensions to OGSA when it becomes available. Programming languages and libraries are mentioned as well.

1.3. DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

AML	Automated Media Library
actor	Someone or something, outside the system that interacts with the system.
artifact	A piece of information that is used or produced by a software development process. An artifact can be a model, a description, or software. Synonym: product.
CE	Component Expert
CEA	Component-Expert Architecture
CES	Component Expert Subsystem
CG	CG - CrossGrid
DAP	Data-Access Package
DFS	DFS - Distributed File System
DS	Data Storage
EDG	European Data Grid Project
ETA	Estimated Time of Arrival
feature	An externally observable service provided by the system which directly fulfills a stakeholder need.
GRM	Globus Replica Manager
HDF5	HDF5 - Hierarchical Data Format version 5
HSM	Hierarchical Storage Management
LDAP	Lightweight Directory Access Protocol
MMSRS	Multimedia Storage and Retrieval Systems
NetCDF	NetCDF - Network Common Data Format
OGSA	Open Grid Service Architecture
PDO	Physical Data Object

RA	RA - Roaming Access is the module of task 3.1
RC	Replica Catalogue
SE	Storage Element, each node in the Grid keeping data.
SOAP	Simple Object Access Protocol
T34	Task 3.4
TBD	To Be Defined
TRLFM	Tape Resident Large Files - middleware placed on top of an existing HSM system
TSS	Tertiary Storage System
UCFM	UniTree Central File Manager
UML	UML - Unified Modelling Language. The industrial standard in modelling and application design.
user	A person who will use the system that is developed.
VD	Virtual Directory
VDO	Virtual Data Object
VSL	Virtual Symbolic Link
VTSS	Video Tertiary Storage System
WSDL	Web Services Definition Language

1.4. REFERENCES

1. DataGrid, "Data Access and File Systems – The State of The Art Report" http://grid-data-management.web.cern.ch/grid-data-management/docs/DataGrid-02-D2.1-0105-2_0.pdf
2. DataGrid, "WP5 Mass Storage Management – Review of Current Technologies" <http://edms.cern.ch/document/336677/>
3. Dutka, Ł., and Kitowski, J., „Implementation of expert technologies in information systems based on a component methodology”, MSK 2001 Conf., Nov. 19-21,2001 Cracow (in Polish).
4. Dutka, Ł., and Kitowski, J., „Component-expert technology in mass-storage grid applications”, ICCS 2002 Conf., April 2002, Amsterdam.
5. *Globus* <http://www.globus.org/>
6. Nikolow, D., Slota, R., Kitowski, J., Nyczyk, P., Otfinowski, J., "Tertiary Storage System for Index-Based Retrieving of Video Sequences",in: Hertberger, B., Hoekstra, B., Williams, R. (Eds.), Proc. Int. Conf. High Performance Computing and Networking, Amsterdam, June 25-27, 2001, Lecture Notes in Computer Science 2110, pp. 62-71, Springer, 2001.
7. Nikolow, D., Slota, R., Kitowski, J., "Benchmarking Tertiary Storage Systems with File Fragmentation", PPAM2001 Conf., Naęczów, Lect.Notes in Comp.Sci., in press.
8. Slota, R., Kosch, H., Nikolow, D., Pogoda, M., Bredler, K., Podlipnig, S., "MMSRS - Multimedia Storage and Retrieval System for a Distributed Mediacal Information System" in: Bubak, M., Afsarmanesh, H., Williams, R., Hertberger, B., (Eds.), Proc. Int. Conf. High Performance Computing and Networking, Amsterdam, May 8-10, 2000, Lecture Notes in Computer Science 1823, pp. 517-524, Springer, 2000

9. *WSDL* <http://www.w3.org/TR/wsdl>
10. Nikolow, D., Slota, R., Dziewierz, M., Kitowski, J., “Access Time Estimation for Tertiary Storage Systems”, to be presented at EuroPar’2002, Paderborn, Germany.
11. DataGrid, “Architecutre and Design WP 5 Mass Storage Management”
<http://edms.cern.ch/document/336679/>

1.5. OVERVIEW

The document is divided into three sections: ‘Overall Description’, ‘Specific Requirements’ and ‘Appendices’.

The ‘Overall Description’ section illustrates the package and its functionality on the basic level as a result from the more detailed analysis. The next chapter - ‘Specific Requirements’ is the section in which package requirements and functionality are described on the technical level. The section ‘Appendixes’ contains additional information, which could be helpful as detailed features analysis and description of external (not included in task 3.4) components.

2. OVERALL DESCRIPTION

2.1. PRODUCT PERSPECTIVE

The optimization of data-access is a package of tools and services making the data access problem easier. The structure of the package follows Task 3.4 description presented in the document ‘Annex 1’ of the Project.

This package is an extension of the available tools developed within Globus and DataGrid projects mainly, but other ideas will be incorporated as well.

The aim of this package is to:

- a) answer the question which replica is the best (this will be based on Globus /DataGrid outputs);
- b) support for local-site optimization for data storage strategy, performed by sophisticated components (data-handler) selection based on Component-Expert Architecture – detailed description is formulated in section 2.1.1;
- c) make some extension of GridFTP to allow flexible management of the data handlers;
- d) design and implement a mass-storage-system-independent middleware to provide faster access to large tape resident files (TRLFM) – detailed description is shown in section 2.1.2;
- e) make some data access estimation – detailed description is presented in section 2.1.3.

CrossGrid is a heterogeneous environment combining different kinds of applications, users, and hardware/software platforms. This heterogeneity reflects also the data-access package. We have many: kinds of file systems, kinds of storage devices that differ radically in the access strategy (e.g. RAID subsystems, HSM storages, etc.), as well as sophisticated requirements from applications, like: the required throughput, critical limitations of data latency, quasi-real time response and so on.

DAP is embedded in the CrossGrid structure presented in Fig. 2. The applications, prepared by WP1 and the Portal prepared by Task 3.1, are the main clients for DAP.

In general the aim of the package fulfils the scope of the Task 3.4 as presented in the Technical Annex, however there is a deviation in the item a). Instead of the expert system (proposed originally in the Task 3.4) proposed for the data replication, we will reuse the Replica Manager from the EDG project. Further details are given in section 3.9.

Expert system together with the proposed component technology (CES) will be used locally for optimization of data access locally in each of CG storage centers (see Fig. 1 for outline of the task 3.4).

TASK 3.4 SRS
Optimization of Data Access
Section 2 Overall Description

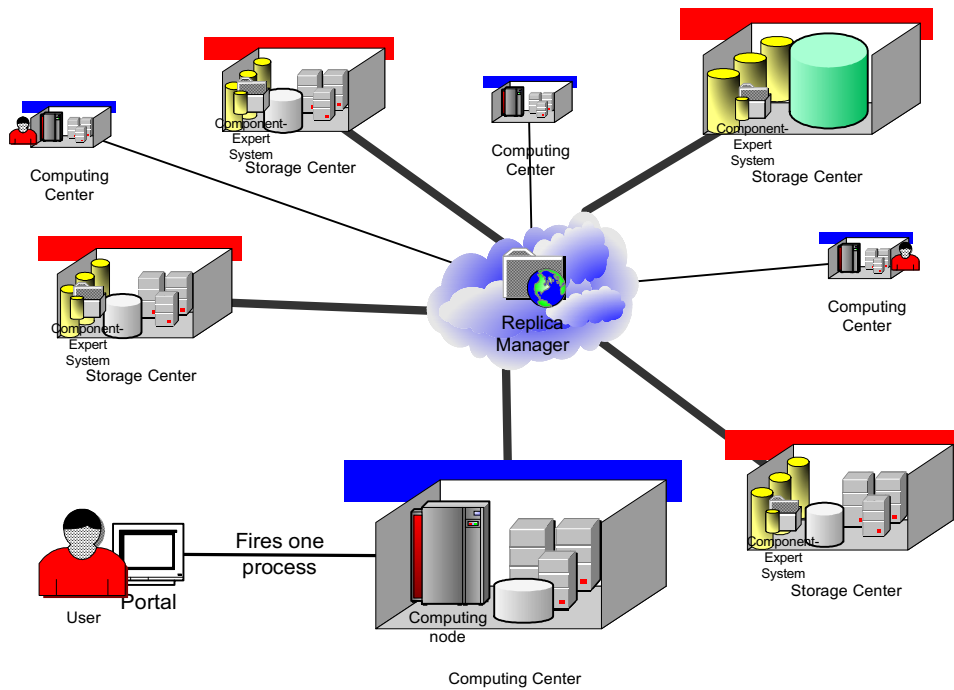


Fig. 1 Outline of task 3.4

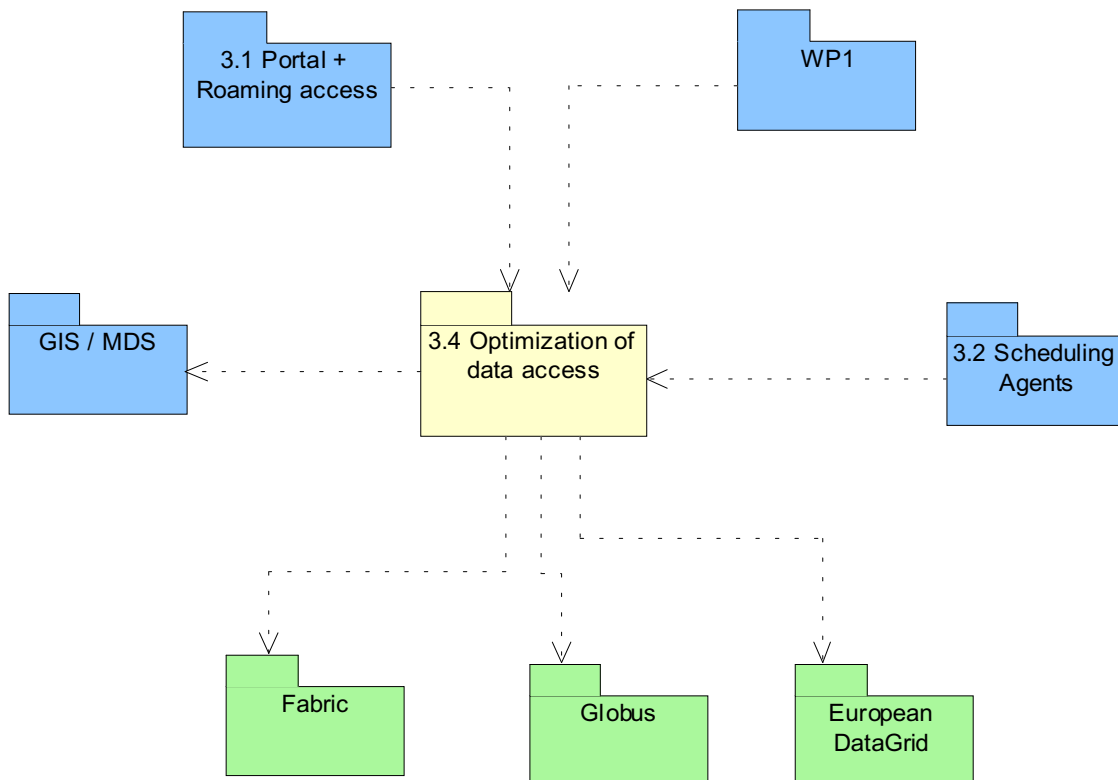


Fig. 2 Dependencies between DAP and other packages

DAP making replica-selection decisions uses information stored in GIS / MDS and this fact is presented in the figure above by the dependency between the '3.4 Optimization of data access'

package and the ‘GIS/MDS’ package. Additionally, the ‘3.2 Scheduling Agents’ package making deploying decisions will use DAP, thus the ‘3.2 Scheduling Agents’ package depends on the ‘3.4 Optimization of data access’ package.

As was mentioned, at the beginning DAP will be an extension of Globus facilities but also implementing some of them. This is depicted by the dependency relation between ‘3.4. Optimization of data access’ and Globus package. DAP will use GSI as a security infrastructure and Metadata Catalog to obtain attributes of data objects.

Since DAP should solve many different issues, its architecture is divided into separate components. The boundary components, visible for other packages, are depicted in Fig. 3. There are the following DAP components to be developed within Task 3.4 (mentioned by 3.4 in the figure):

- Replica Manager – the component selecting the best replica of a data object. The data could be stored in a file or in a database.
- Data Access Estimator – the component estimating latency of data access and bandwidth of the data transfer inside Storage Element. This component is especially dedicated for Replica Manger for replica selection process.
- Storage Element – the component virtualizing storage center, often consisting of many storage-devices.
- GridFTP Plugin – the component expanding capabilities of GridFTP server, allowing the usage of Component-Expert Architecture.

To realize services of the above external components, the following internal components have to be built:

- Component Expert Subsystem – a sophisticated architecture of selection of components (data handlers), which allow building a flexible solution of data-access in local environment.
- Tape Resident Large Files middleware

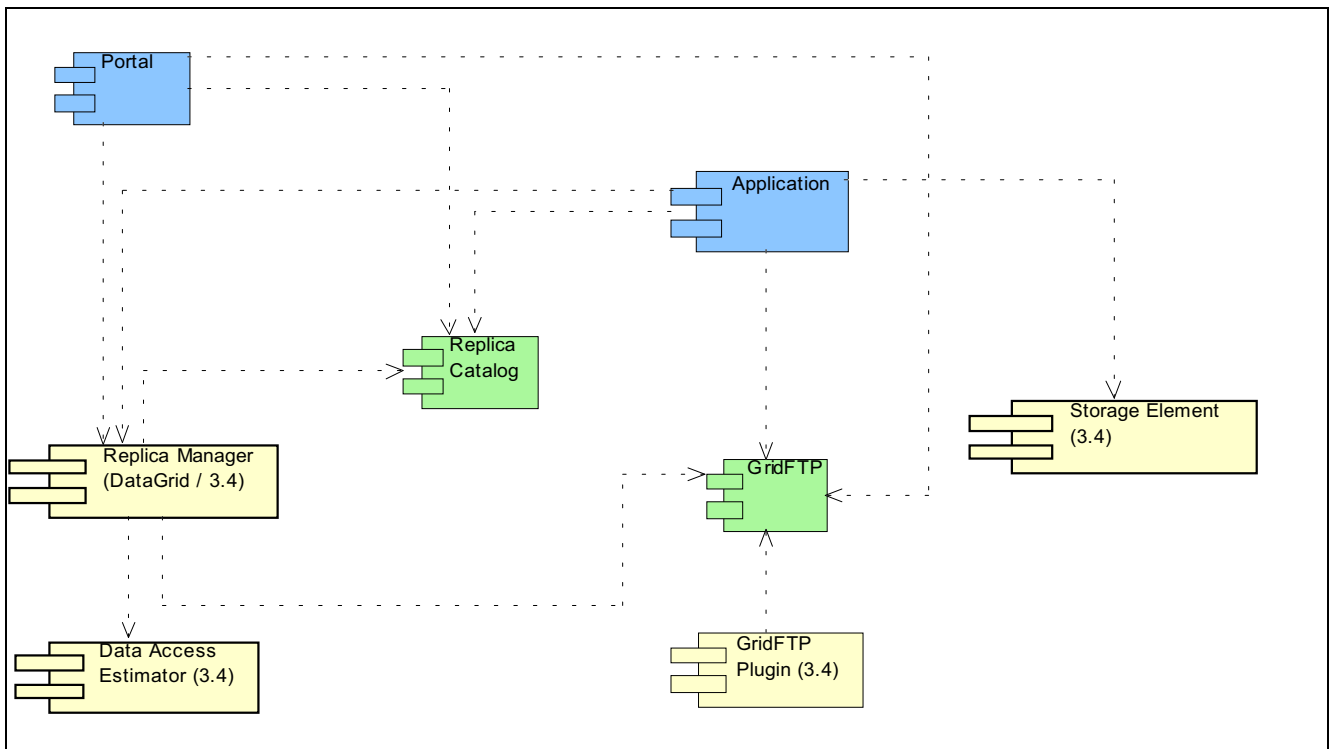


Fig. 3 Dependencies between Task 3.4 components and sample external components

The sections (2.1.1, 2.1.2 and 2.1.3) describe in detail the main goals of this task.

2.1.1. Component-Expert Subsystem

The Component-Expert Subsystem should be installed in each CrossGrid site, which operates data storage systems [1,4]. Its purpose is to optimise usage of the storage systems by proper selection of storage strategy due to different applications requirements and existing storage abilities. Probably it will be the plug-in for GridFTP, which will operate on the data. CES in such a case will be responsible for choice of the best component (i.e. strategy) for data operation (read, write etc.) and this decision will be taken using knowledge rules stored in the expert system. The human expert should prepare these rules. If necessary, the programmers will build specialized components for some sophisticated data-access methods. Thus, the programmers obtain a flexible environment, which can be easy extended. Since the decision which component should be used will be taken by the expert system and not by the programmer himself, stability of the whole Data-Storage system is maintained. Otherwise, the programmers should know internal structure of DS in detail.

CEA transfer of the responsibility for proper components choice from the programmer to the system, which has some kind of 'intelligence'. For this reason elements of the expert systems are introduced.

TASK 3.4 SRS
Optimization of Data Access
Section 2 Overall Description

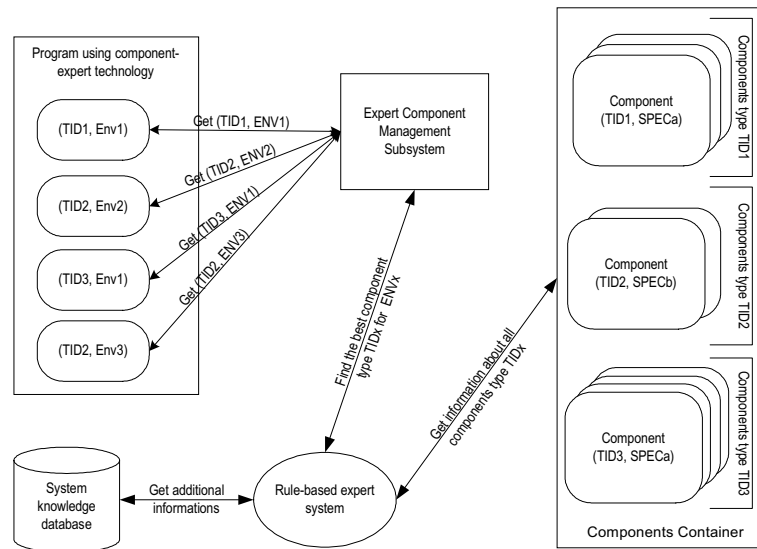


Fig. 4 Connection Diagram of Component-Expert Application

To realize this strategy the system consists of the following elements (Fig. 4), like: Components, Components Container, Expert Component Management Subsystem, Expert System and an optional Knowledge Base. The Components Container keeps up many components of the same type, TID, but with different specializations, SPEC. The flexibility in component invocation enables different combinations of the component type, TID, and the call-environment, ENV. CE that represents context of invocation, facilitates proper selection of components (Fig. 4 and Fig. 8).

The operation of CES is summarized as follows:

The program requires a service of the TID_x component type for the ENV_y CE . It requests from the Expert Component Management Subsystem a handle to the component of a certain type, which is most suitable for ENV_y.

The Expert Component Management Subsystem parses obtained information and, using the Expert System, it finds the most suitable component for ENV_y.

The Expert System gets all TID_x components type from the Component Container.

Based on the rules-based knowledge of the expert system and on the basis of knowledge which is incorporated in the System Knowledge Database and especially on specialization of all components required type, the Expert System takes a decision which component is most suitable for a given ENV_y.

If the Expert System finds an appropriate component then the Expert Component Management Subsystem returns a handle to it, if not then the program gets an error message.

The program calls the service of component.

In the reported approach, the component consists of a header, a code, a stream of input parameters, a stream of output parameters and a data stream (Fig. 5). The header consists of a component type and of an attributes list (Fig. 6).

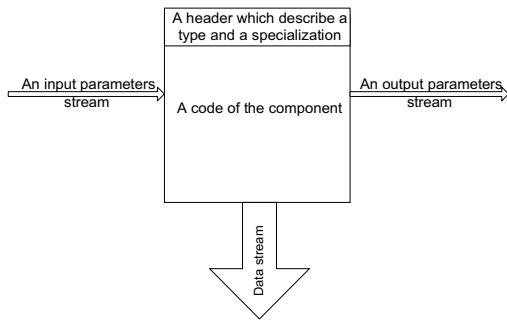


Fig. 5 Logical scheme of the component structure

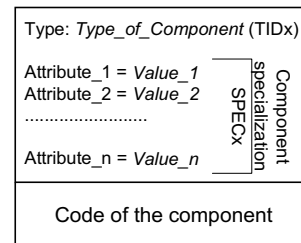


Fig. 6 Internal structure of the header

The component type describes the general purpose of the component, while the attributes list specifies a specialization of the component. The specialization of component represents the detailed purpose for which the component is best suited. The specialization is used by the rule-based expert system for deduction, i.e., for selection of the component from the component collection according to CE (the context). Different types of the attribute values may be implemented, e.g., numbers, strings, enumerators, etc.

An important element of the whole architecture is the Call-Environment, which describes the actual state of the environment of component invocation and defines requirements and expectations that should be fulfilled by the most suitable component for the invocation context. CE consists of the attributes list with their values (Fig. 7). It is connected with the required component type and it arises at the program side only. The pair component-type and call-environment is passed to the Expert Component Management Subsystem, which parses and processes it. Every call of the component in the whole application may have different call-environments. At the component invocation the most suitable component of a given type is selected from all components accumulated in the system. The precision of the component fitting is depended both on precision of the CE description and specialization of the components.

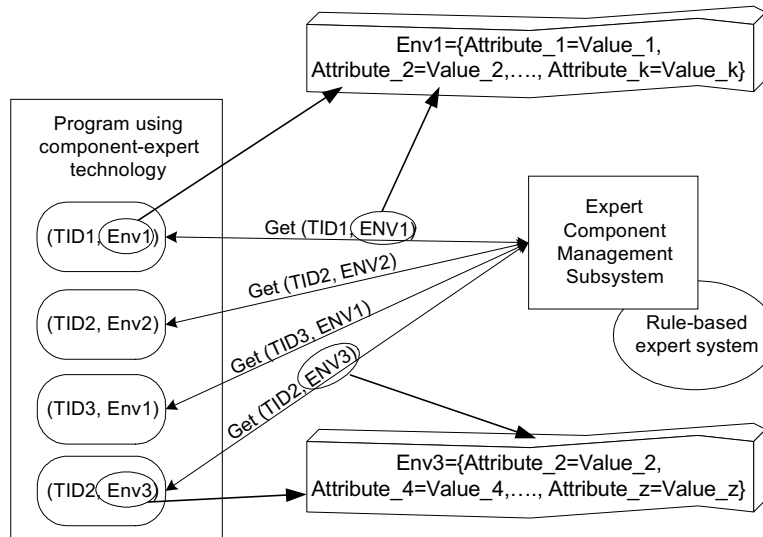


Fig. 7 Structure of Call-Environment

For the data access problem examples of CE are the following:

```
ENV1={User ID = Tim; Data ID = file1.dat;
      Data Type = Indexed Mpeg;
      Application Purpose = Medicine Simulation}
```

```
ENV2={User ID = Tester; Data ID = file1.dat;  
      Application Purpose = Medicine Simulation}  
ENV3={User ID = Tester; Data ID = file2.dat}  
ENV4={User ID = Tester; Data ID = file3.dat;  
      Required throughput = 768kb; Critical level = Very high}  
ENV5={User ID = Jim; Data ID = file3.dat;  
      Critical level = Very high}  
ENV6={User ID = Jim; Data ID = file4.dat}
```

A set of attributes for the component specialization and a set of attributes for the call-environment are unified. However, in both cases subsets of the whole set could be used; in extreme case those subsets may be empty.

2.1.1.1. CEA for Data-Access Problem for Grid-enabled Storage

The CrossGrid Project deals with a new category of applications that implement interactions with a person in a processing loop. As stated in Technical Annex, examples of these applications are: interactive simulation and visualization for surgical procedures, flooding crisis team decision support systems, distributed data analysis in high-energy physics and air pollution combined with weather forecasting.

There can be a variety of components types (i.e., components of different purpose), for example the components to read, to write, to test throughput and to replicate data, usually with different specializations (i.e., suitable for specific tasks). Deduction provided by the rule-based expert system is used for selection of the component, which is best suited for the service required by the application.

Assuming the reading process, there can be an universal component, which reads everything, another which is specific for small files reading, and others specialized for huge files, for remote reading using ftp protocol, for tape-resident data using RAID technology, etc. To describe specialization a list of attribute values is applied. A sample set of attributes is presented below:

1. User ID,
2. Computing Node ID,
3. Data ID,
4. Required throughput,
5. File type,
6. Application purpose,
7. Critical level.

They have been selected according to the kinds of applications implemented in the Project, their distributed nature and diversity of equipment. The important thing is that in a component specialization and in a call-environment there is no need to use all of this attributes and in most cases a subset of them is enough (in special case the empty subset is also possible). To show an example of components specialization the simplest *Read* type was chosen (see Fig. 8). Additionally, an implementation of CES for the CrossGrid environment must be maximal transparent for the user. Thus, all needed extensions will be done in lower layers. The CEA allows extension of the component set by new components in any time of the system operation, according to the gathered experience.

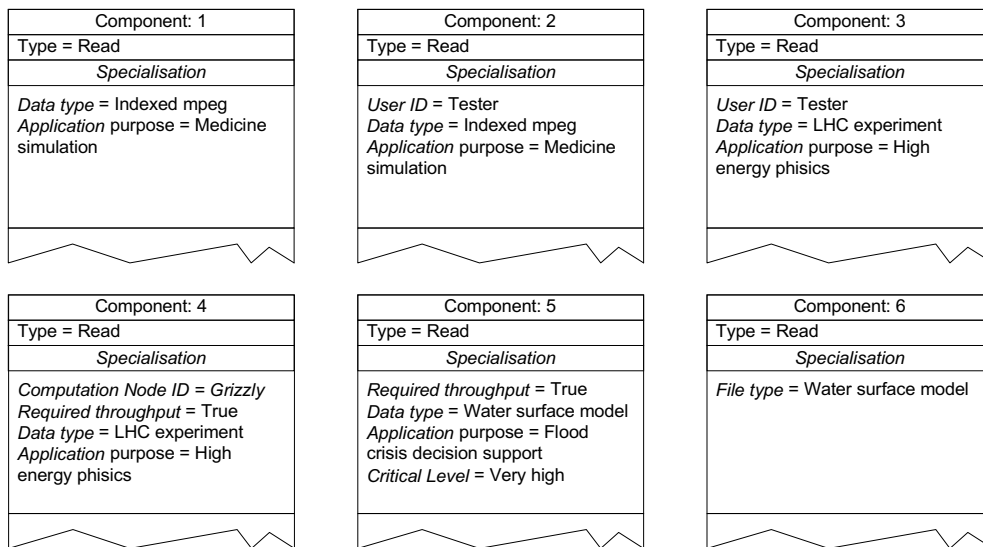


Fig. 8 Example subset of components type read with their specializations

2.1.2. Tape Resident Large Files Middleware - TRLFM

The purpose of this section is to describe Common Middleware for Faster Access to Tape Resident Large Files (TRLFM). In detail, this section will provide the description of elaborated TSS middleware, which is optimized for fast access to tape resident data.

TSS systems are suitable for storing and retrieving huge amount of data. The purpose of developing the TSS middleware is to shorten the access time to large files. This is done by using a middleware for the existing HSM systems. The second solution is to develop an original TSS kernel. The benefit of using them is to provide more efficient access to tape resident files in comparison with the existing HSM systems (e.g. UniTree CFM).

MMSRS [8] is an example of usage of the developed middleware for the existing UCFM . The middleware is to be autonomous. The VTSS makes use of the originally developed TSS kernel. Both are intended for supporting storage requirements for applications running on Grid .

The developed middleware could be used for HSM already used in CG sites to improve access to data.

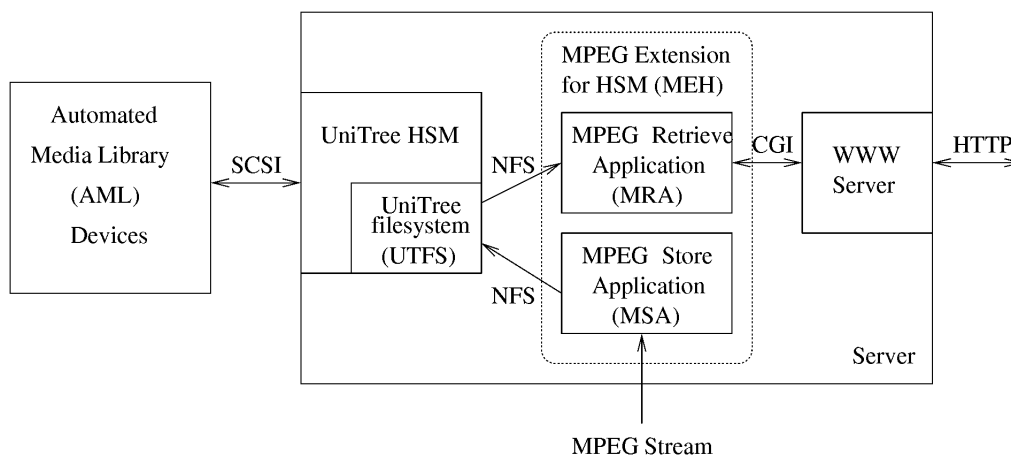


Fig. 9 MMSRS implementation with MEH extension

The architecture of MMSRS is shown in Fig. 9. It is based upon the UniTree HSM software. The system consists of the Automated Media Library (AML) and of the following software components: the UCFM managing system, the MEH middleware and a WWW server. MEH has been added on the top of the UniTree HSM to achieve efficient access to file fragments.

Because of file granularity imposed by the UCFM software and our requirements for low startup latency MEH task is to cut the files into pieces of similar size and store them as different files into the UCFM file system. The cutting substantially reduces the start-up delay. Tests have been performed for video files. MEH receives the name of the video and the frame range (start frame and end frame) from the client. According to the range, it computes which video pieces (subfiles) will be needed to produce the output MPEG stream. The video fragment is requested and received by the client using HTTP.

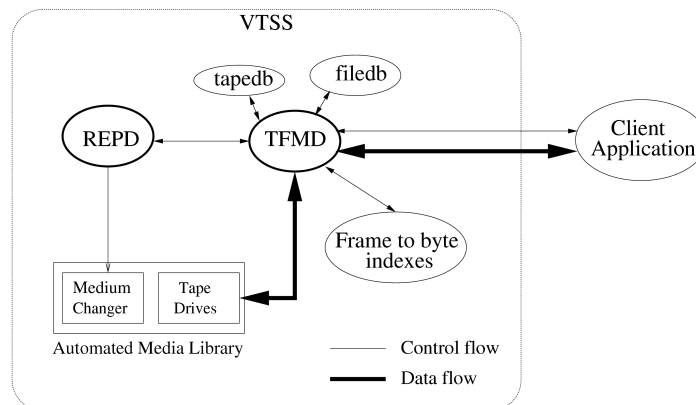


Fig. 10 VTSS architecture

The architecture of VTSS [6, 7] is shown in Fig. 10. The system consists of two main daemons: the Repository Daemon (REPD) and the Tertiary File Manager Daemon (TFMD). REPD keeps repository information in its internal data structures. When a tape mount request is received REPD issues appropriate SCSI commands to the robot arm of automated media library (AML). Tapes in AML are identified by unique labels written at the beginning of the tapes. In order to locate and retrieve a requested file fragment, the tape identifier and start-up position of the fragment on the tape are needed.

The cutting strategy depends on datatypes and will be selected on the base of rules from CES. For this purpose a set of components will be developed.

2.1.3. Data Access Estimation

The purpose of this section is to describe the Access Time Estimation Subsystem for tape resident files. In detail, this section will provide the description of two elaborated models developed for this purpose. It will provide some requirements and problems for other CG WPs.

HSM systems are suitable for storing and retrieving huge amount of data. The access time to data stored on TSS could vary a lot – from milliseconds to minutes or hours. The ‘a priori’ knowledge of the access time could make it possible to increase the efficiency of applications by better planning its I/O operations.

Two approaches at estimating the TSS access time have been considered:

- Open TSS approach in which case the source code of the TSS is available and well known, so changes in the code could be made by adding event reporting functions, and
- Gray box TSS approach in which case the essential system information is accessible via its native tools only.

Both approaches will use TSS simulation to estimate the access time.

In Fig. 11 the open TSS [10] approach for estimating data access for tape resident files is shown. This approach is based on simulating the TSS in order to obtain an ETA for a given request processed by the TSS. Here the real TSS have to be changed in such a way that it reports essential events to the TSS simulator. These events are: arrival of new request, moving a tape from a slot to a drive, moving a tape from a drive to a slot, unloading a tape, loading a tape, positioning a tape, tape being in use, tape being idle.

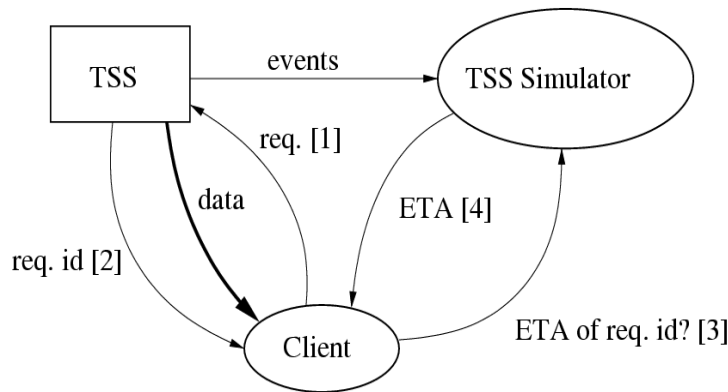


Fig. 11 Open TSS approach

In this approach, the Client first issues its request to the TSS and receives an identifier for that request. Then while waiting for the data to come the Client can ask the TSS Simulator for the ETA of its request. Next, the simulator sends the ETA back to the Client.

Second approach will concern estimating the access time for tertiary storage systems in which no changes of the source could be made. Our first target will be UniTree HSM system. A gray-box approach will be made at estimating its access time which will be based on some knowledge about how it works and miscellany data gathered from the available utilities delivered by the vendor.

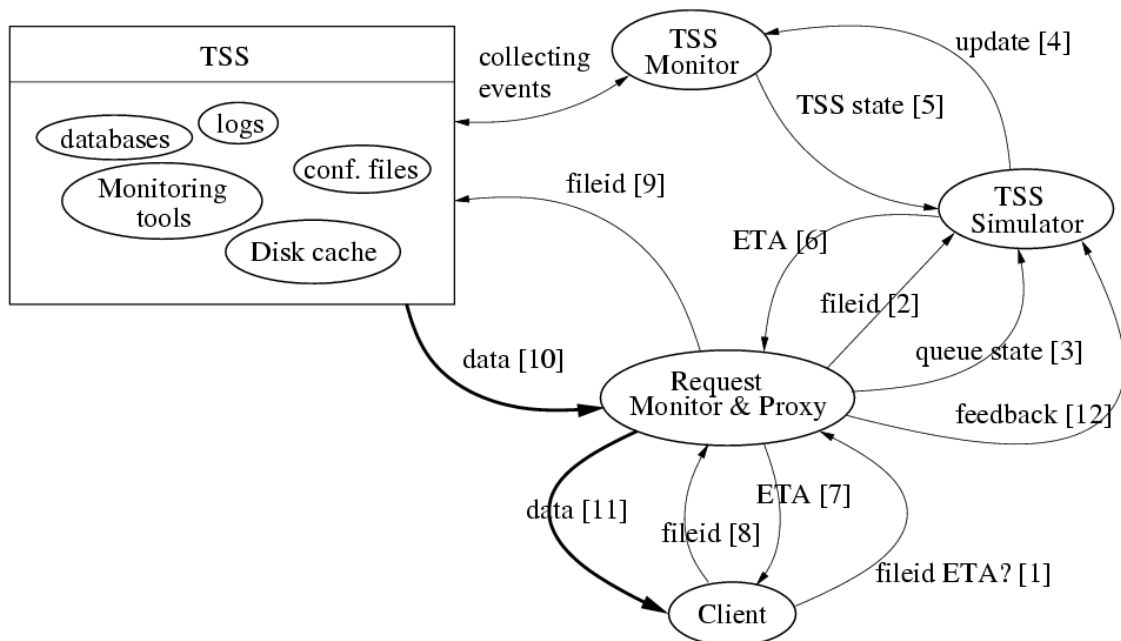


Fig. 12 Gray box approach

Fig. 12 presents the mentioned approach. The core of the system will be the TSS Simulator which will be quite similar to the one presented in the open TSS approach. TSS Monitor will collect the

necessary (available) data about the state of UniTree. Request Monitor & Proxy will catch all client requests in order to have more detailed information (queue order, file identifiers, time statistics) about the requests being processed by UniTree.

2.1.4. System interfaces

This package will use system I/O interfaces, system transport layer, system synchronization mechanism – semaphores, and Globus Security Infrastructure.

2.1.5. User interfaces

The data-access package is classified as a middleware and in this case no user interface is planned.

2.1.6. Hardware interfaces

In this task, two kinds of nodes could be picked out, first for Replica Manager service and second for each Storage Element and Data Access Estimator.

The Replica Manager service should work on some Unix machine (at the beginning Linux operating system is only considered). It should be connected to normal TCP/IP network without any special requirements. The volume of the traffic between Replica Manager and its clients will be very low.

The second kind of nodes is the Storage Element. Since SE could be every node in Cross Grid working keeping a data, thus there are no special hardware requirements for nodes sustain SE software.

2.1.7. Software interfaces

Globus version 2.0 is the underlying package providing basic services, from authentication and security to basic data transport (GridFTP); Metadata Catalog as an information provider on special data attributes. In the future, the replacement from Globus version 2.0 to version 3.0 is planned.

The usage of Replica Manager for European DataGrid project is planned.

2.1.8. Communications interfaces

The connections between, Data Access Estimator, Storage Element and their clients will be made using SOAP. The TCP/IP protocol, with some security extensions existing in Globus, as a transport protocol for these components is sufficient.

Because sometimes there is a need to transport data between nodes inside SE with high throughput, thus the named pipe or the shared memory are intended in such cases.

2.1.9. Memory constraints

There are no special memory constraints, which could not be fulfilled by contemporary machines.

The predicted Replica Manager, requisition for memory is measured in megabytes.

The Data Access Estimator and Storage Element have no memory constraints.

2.1.10. Operations

DAP recognized three kinds of package's actors: user, administrator and application. These actors directly or indirectly influence the functionality of the package.

Users send jobs thru the portal to the Grid environment and specify directly or indirectly, which Replica Manager, Replica Catalog and Metadata Catalog has to be used.

Administrators of storage systems install and configure CES for a selection of the data handlers (components). Moreover, administrators of storage systems may install TRLFM allowing faster access to large data files stored on HSMs. The whole configuration of SE implies on the decision of the data handler selection.

Applications seek the best replica of data using Replica Manager, transfer data by GridFTP protocol from SE to Computation Node. The data transferring implies arising of data handler at the SE side. The selection problem of optimal data handler will be solved by CES. To make optimal decision Replica Manager among other things asks Data Access Estimator for the latency and bandwidth inside the Storage Element.

2.1.11. Site adaptation requirements

Each node related with DAP should have at least one public IP address, it affects nodes holding components: SE, Replica Catalog, Metadata Catalog, Data Access Estimator, Replica Manager, GridFTP.

Each node, which must store data should have at least installed following components:

- GridFTP
- Storage Element

Additionally the larger sites should have installed component 'Component-Expert Subsystem' and package of components for CES.

2.2. PRODUCT FUNCTIONS

The Data Access Package should support following functions (each item on the below list of functions has a unique identifier in the form 'FEATxx', a number after 'FEAT' expression is a unique identifier and not a counter):

- FEAT91 *Optimal replica selection* – Task 3.4 will adopt Replica Manager component from Globus / Datagrid. The main goal of this component is to answer to the question which replica of a data-object is the best for some context.
 - o FEAT91.1 *All replica-selection features available from code* – The replica-selection interface of the Replica Manager component has to be accessible from code of any application. It's important to note that the mentioned applications could be written in any programming language.
 - o FEAT91.2 *Optimal database selection* – In the grid may be plenty replicated databases. System should give some facilities to choose an optimal database. There is an assumption that these databases are well replicated and there are no important differences.
- FEAT94 *Data access estimation* – The estimation of the latency in the data access or the bandwidth of the data transfer inside the Storage Element is an important issue, especially for Replica Manager.
- FEAT102 *Optimal data handler selection* – The strategy of the data access depends on many factors as type of storage device, type of the data, access limitations etc. Each strategy has own data-handlers. There is a necessity of a flexible mechanism of the data handler's selection.
 - o FEAT102.1 *Component Expert Subsystem* – CES was introduced to realize optimal data handler selection. CES is based on the Component Expert Architecture.

- FEAT102.2 *Process of data-access requirements* – The user sets some data access requirements, which affects on low-level data-access strategies (algorithms). The system should choose the best access strategy.
- FEAT105 *WSDL description of interfaces* – The interfaces should be described using WSDL language.
- FEAT106 *SOAP the essential communication protocol* – All interfaces should be accessible using SOAP communication protocol. Obviously, this protocol is much slower and might be used only when time restrictions allow this.
- FEAT107 *GridFTP as file-data transfer protocol* – GridFTP is currently the most popular file-data transfer protocol in the Grid environment, and CrossGrid is going to use it. Thus, every large volume data transfer should use GridFTP protocol.
- FEAT111 *Tape Resident Large Files middleware (TRLFM)* – This middleware is placed on top of an existing HSM system. It is intended to speedup the read access to large files (>100MB) stored on tapes, by means of reducing the latency time.

The detailed analysis of features is stated in section 3.9.

2.3. USER CHARACTERISTICS

As was mentioned in section 2.1.5 this package belongs to middleware, thus these components essentially do not have direct contact with the user.

If to treat the application programmer as a user of DAP components then he/she should have experience with Unix systems, and understand Web Services techniques.

2.4. CONSTRAINTS

- a) Regulatory policies: none.
- b) Hardware limitations (e.g., signal timing requirements): none special constraints.
- c) Interfaces to other applications: connection with Metadata Catalog is wanted.
- d) Parallel operation: none special constraints.
- e) Audit functions: none special constraints.
- f) Control functions: TBD
- g) Higher-order language requirements: C, C++, and Java.
- h) Signal handshake protocols (e.g., XON-XOFF, ACK-NACK): none on such low level.
- i) Reliability requirements: Very High
- j) Criticality of the application: Very High
- k) Safety and security considerations: realization of architecture stated in GSI.

2.5. ASSUMPTIONS AND DEPENDENCIES

The package is developed with the assumption that Globus version 3.0 and OGSA architecture will be future standards in such solutions. Thus, all architecture efforts are verified against future compatibility with OGSA and Globus 3.0.

2.6. APPORTIONING OF REQUIREMENTS

All requirements are prioritized. Features with the highest priority will be developed first. The detailed table with features and their priorities is attached in section 3.9.

2.7. DEVIATION BETWEEN TECHNICAL ANNEX AND CURRENT STATE OF TASK 3.4

The technical Annex, which amongst other things specifies the objectives of the task 3.4, introduced the expert system for data management. This expert system was intended to operate as an advisor for selection of migration/replication policy in the defined user states. The final analysis of a current Replica Manager development, provided by EDG, shows that the similar functionality is provided by the EDG and basing on the tenet of maximal reuse of the existing solutions, the decision of reusing this work has been undertaken. However, our requirements analysis revealed that the selection of the best replica of data stored in databases was an important task, which had not taken into account during preparation of the Annex. Therefore the task 3.4 is going to extend the functionality of the EDG Replica Manager in domain of the best replica selection of the data stored in databases.

2.8. TASK 3.4 ADDED VALUE

The component-expert subsystem, data time access estimator, optimization of access to large tape resident files and the extension of EDG Replica Manager are the added value of the task 3.4. The first three items represent low-level elements, not exactly grid services, however they are new tools that can be useful in the grid environment. The last item is the extension of the EDG activity in data management domain. The component-expert subsystem is the proposition of an extension of the Request Manager introduced by the EDG project [11]. This subsystem introduces the rule-based expert system (the replacement of Request Manager), which selects the best components (the replacement of data-handlers). The outline of this architecture is presented in Fig. 13.

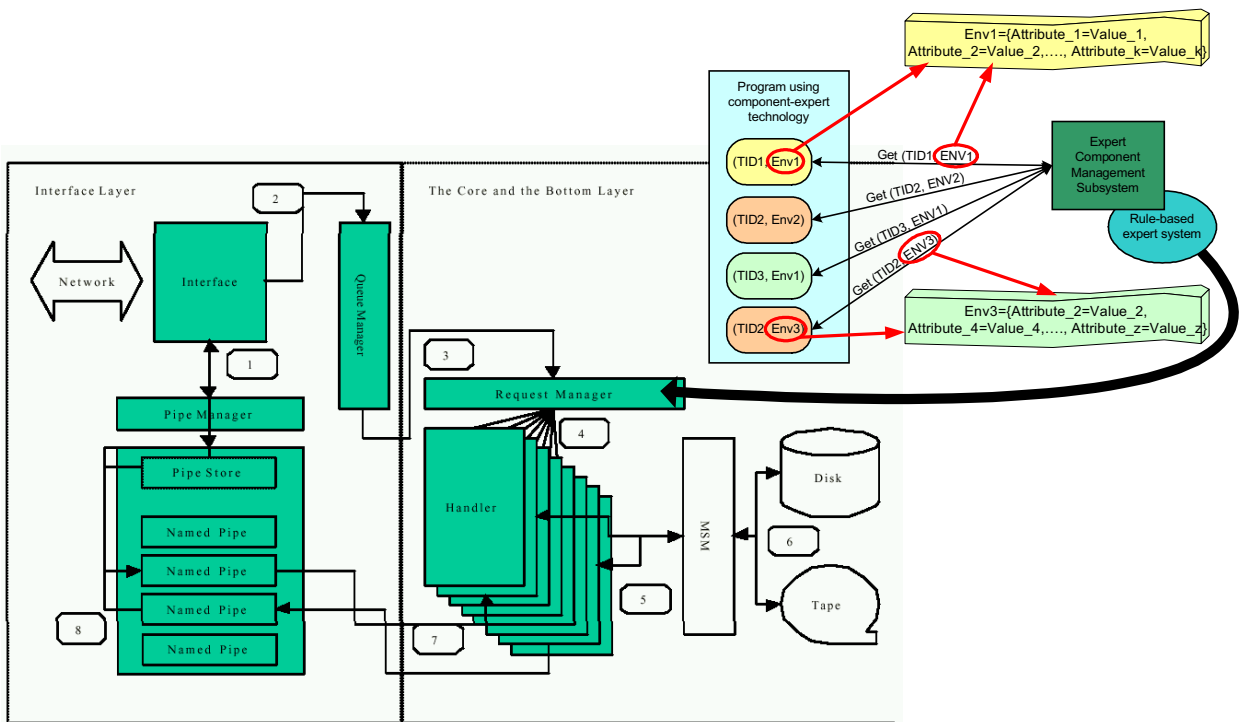


Fig. 13 Data flow diagram through the EDG SE and application CES as a Request Manager

3. SPECIFIC REQUIREMENTS

This chapter states some diagrams for better overview of the whole system. These diagrams present relations between components realized by the task 3.4 and external components. Components and interfaces included in the task 3.4 are depicted by light items (yellow on color printer) and external ones are represented by gray items (green or blue on color printer). The detailed description of requirements is presented in the section 3.9.

The description of components realized by the Task 3.4 are stated in sections 3.1 and 3.2. Information about components not provided by T34 could be founded in the section 4.1.

The 'External Interfaces' (sect. 3.1) and the 'Internal interfaces' (sect. 3.2) subsections specify components and their interfaces included in the Task 3.4 and shown in Fig. 14 and Fig. 15.

This diagram shows dependencies between components important for DAP . Components, which will be realized by task 3.4, are marked additionally by '(3.4)' indication and are filed by the light color (yellow). All other components will be not provided by task 3.4 and resused from other tasks or projects. The dotted directed line between two entities means that the first one depends on second.

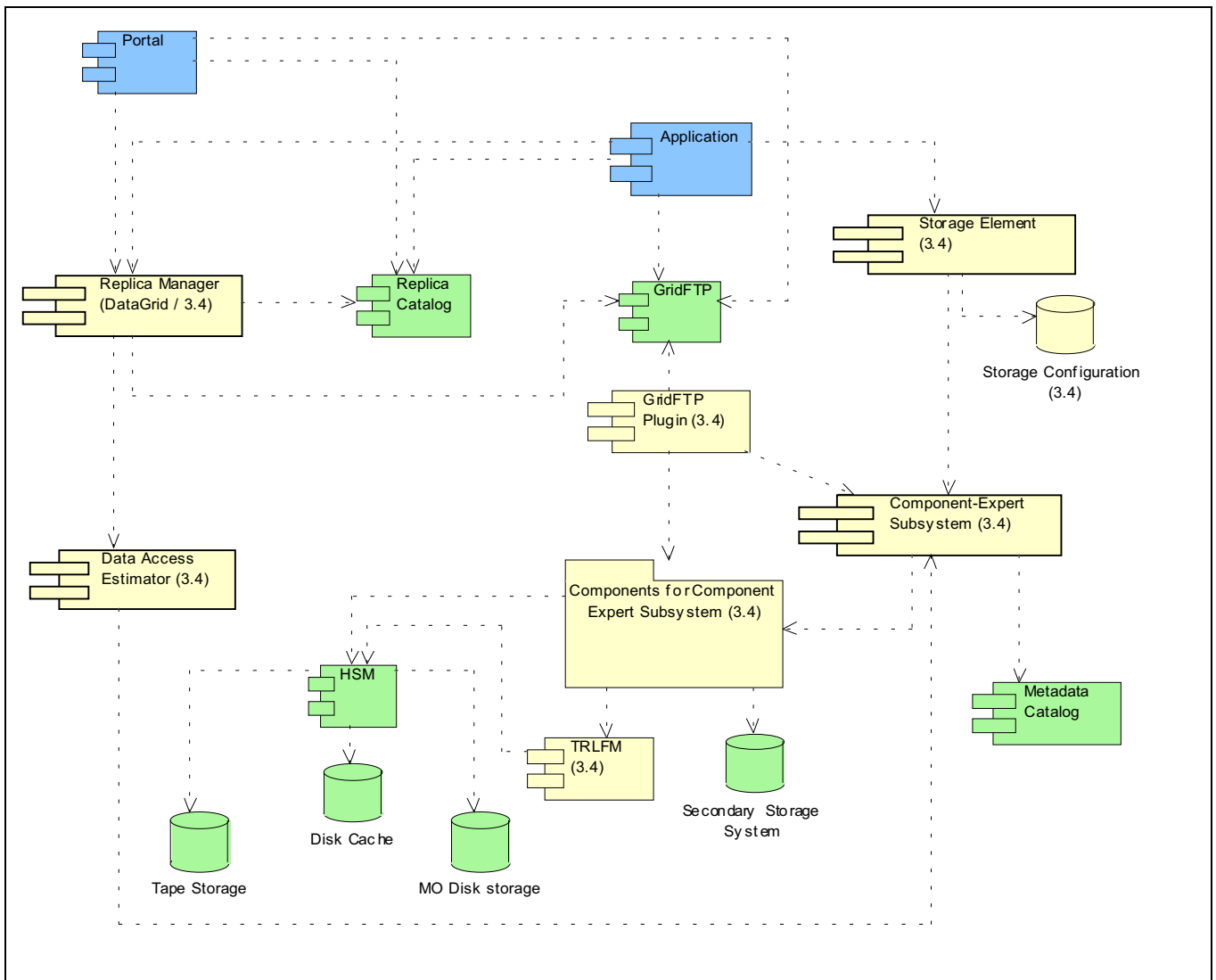


Fig. 14 Dependencies between all components

TASK 3.4 SRS
Optimization of Data Access
Section 3 Specific Requirements

In Fig. 15 the connections between components and interfaces to other components are shown. All interfaces have the name beginning with the 'I' and the abbreviation of the component which realize the interface. Similarly, to the previous diagram the internal components are light (yellow) and the external (to Task 3.4) components are gray (green). To make this diagram easier to read interfaces for the external components were introduced. These interfaces do not exist in physical components but they are used here to group similar operations supported by external components.

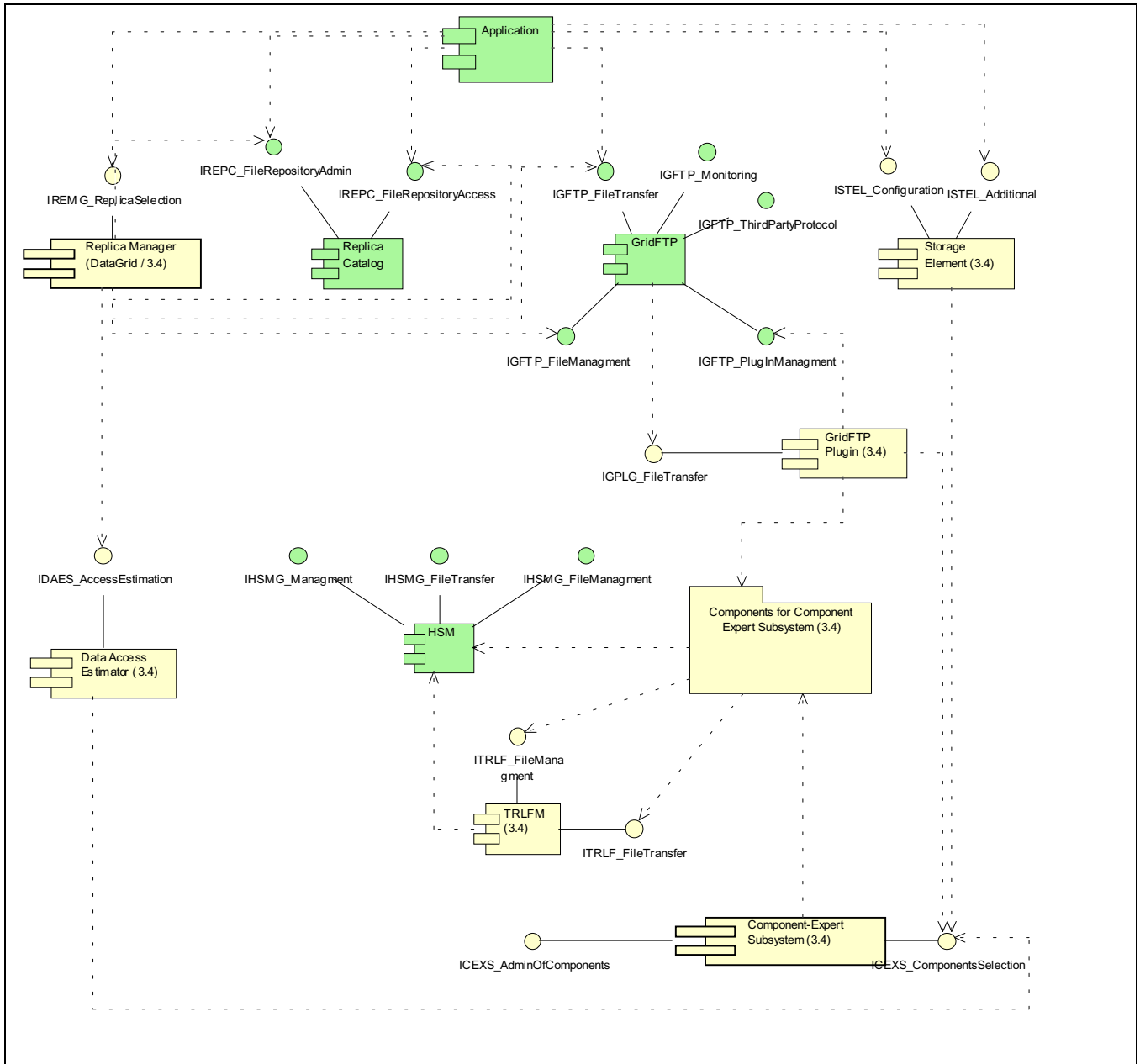


Fig. 15 Connections between all important components

3.1. EXTERNAL INTERFACES

Interfaces are assigned to components. Thus, this description has to be grouped by components. Each component has an own section in this chapter, and each assigned interface has an own subsection. This chapter specifies only external interfaces (i.e. external to Task 3.4 interfaces used by other CG modules), however internal (to Task 3.4) interfaces are specified in section 3.2.

3.1.1. Component - Storage Element (3.4)

Storage Element is responsible for publishing current configuration and answering on general questions about current configuration, e.g. when administrator of a storage element register storage somewhere then he don't need to specify any value of attributes of his storage but he just enter address of the registered storage and the rest will be obtained automatically. Additionally Storage Element will be used as a temporary tunnelling platform for SOAP, which may be used for additional control of the data transfer when Component-Expert Subsystem is used. This component should be deployed on each Storage Element. The abbreviation of this component name is STEL.

3.1.1.1. *ISTEL_Additional*

Interface name: ISTELE_Additional

Documentation: This interface combines all additional operations, which are not matching for other interfaces, e.g. temporary operations for a control of component-expert subsystem.

Operation name: prepareBestComponent4CallEnv

Documentation: It using Component-Expert Subsystem selects the best component given in parameters type for given also in parameters call environment (context).

Protocol: SOAP

Returns: Handle to prepared component.

Parameter name: required component type

Documentation: Specialized components for Component-Expert Subsystem are divided onto types. This parameter specifies required type of a component.

Parameter name: call environment

Documentation: The context for a taking decision which component is best.

3.1.1.2. *ISTEL_Configuration*

Interface name: ISTELE_Configuration

Documentation: This interface combines all operations related to obtaining information about current Storage Element configuration.

Operation name: isComponentExpertSubsystemWorking

Documentation: Returns true if on storage-element node works component-expert subsystem.

Protocol: SOAP

Returns: bool

3.1.2. Component - Replica Manager (DataGrid / 3.4)

Replica Manager is an advisor for selection of migration/replication policy in defined user states. Its main goal will be to suggest whether and when a chosen data file should be replicated into the local environment of the remote user in order to shorten the waiting time for data availability. A sophisticated method will be used to take right decision. The EDG Replica Manager looks to be the best for CrossGrid project, however some small extension of this component is planned. The abbreviation of this component name is REMG.

3.1.2.1. *IREMG_ReplicaSelection*

Interface name: IREMG_ReplicaSelection

Documentation: This interface combines all operations related to replica selections, e.g. choose the best replica for some context.

Operation name: chooseBestReplica

Documentation: The replica manager takes a decision which replica is the best in given context. This operation returns the best-matched physical file instance.

Protocol: SOAP

Returns: physical file path

Parameter name: repository

Documentation: In Cross Grid is acceptable existing of many Global Virtual File Repositories. Thus, user must specify which repository is using.

Parameter name: virtual file path

Documentation: The whole virtual file path.

Parameter name: allow new replica

Documentation: Sometimes it is possible to make new replica, which will be more suitable than all existing ones. If this parameter is true then it is a permit to make new replica.

Parameter name: context

Documentation: Context describes a user or application requirements.

3.1.3. Component - Data Access Estimator (3.4)

Data Access Estimator estimates data access cost in for a given data storage system. It can return many measures of cost e.g. latency, bandwidth etc. This component should be located on each Storage Element. The abbreviation of this component name is DAES.

3.1.4. Component - GridFTP Plugin (3.4)

GridFTP allows to expanding itself by plugins, which can take control on any Grid command. This feature is used to connect GridFTP to Component-Expert Subsystem and to use chosen components as a local data serving modules. This component should be deployed on each Storage Element . To the direct connection between GridFTP and TRLFM another kind of GridFTP Plug-in is used. The abbreviation of this component name is GPLG.

3.1.4.1. IGPLG_FileTransfer

Interface name: IGPLG_FileTransfer

Documentation: This interface will transmit data from/to CEComponent.

3.2. INTERNAL INTERFACES

Similarly to the previous section, the interfaces of internal components are grouped by components, which realize them.

3.2.1. Component - Component-Expert Subsystem (3.4)

This component realizes the proposed Component-Expert (CE) strategy. It has many responsibilities. First of all, it registers new CE components, which can be chosen as the best matching CE component in some context. It returns a handle to the best matching CE component in some context. This component should be located on each Storage Element . The abbreviation of this component name is CEXS.

3.2.1.1. ICEXS_AdminOfComponents

Interface name: ICEXS_AdminOfComponents

Documentation: Administrator of storage element registers or unregisters components for Component-Expert (CE) Subsystem uses this interface. All operations concerning administration of CE components will be passed by this interface.

Operation name: getAllComponents4Type

Documentation: It returns the list of all components, which have type as specified in parameter list for this operation.

Protocol: Some internal. Not decided yet.

Returns: List of components

Operation name: registerNewComponent

Documentation: This operation registers a new CE component and makes its available for the selection process. Used by a developer or a system administrator.

Protocol: Some internal. Not decided yet.

Returns: Succeed/fail code

Operation name: unregisterComponent

Documentation: This operation unregisters a CE component and makes its unavailable for the selection process. Used by a developer or a system administrator.

Protocol: Some internal. Not decided yet.

Returns: Succeed/fail code

3.2.1.2. ICEXS_ComponentsSelection

Interface name: ICEXS_ComponentsSelection

Documentation: Component Expert Subsystem selects the best matching CE component for a given call-environment. This interface is used to run such a process and to get handle to the best CE component.

Operation name: getBestComponent4CallEnv

Documentation: This operation starts the best component selection process, which is made by Component-Expert Subsystem and as a result of this operation a caller obtains handle to the best matching component type as a parameter and for a given call-environment (context). It returns success or error code of the operation.

Protocol: Some internal. Not decided yet.

Returns: Success/fail code

Parameter name: componentType

Documentation: The type of component.

Parameter name: callEnvironment

Documentation: The encoded call environment for a deduction process.

3.2.2. Component - Storage Configuration (3.4)

Storage Configuration artifact represents actual settings of Storage Element. It is used by component Storage Element. This component should be deployed one each Storage Element.

3.2.3. Component - TRLFM (3.4)

The TRLFM (Tape Resident Large Files) middleware is placed on top of an existing HSM system. It is intended to speedup the read access to large files (>100MB) stored on tapes, by means of reducing the latency time. This is done by splitting the large files into smaller sub-files during the file writing via TRLFM. The splitting reduces the latency time of the read file operation since smaller files are staged faster than the big ones into the disk cache, where they can be accessed. Additionally the

TRLFM middleware gives to applications the ability to access fragments of files (only the necessary sub-files are then staged, thus making more efficient use of the disk cache space).

The cutting strategy depends on datatypes and will be selected on the base of rules from CES. For this purpose a set of components will be developed.

3.2.3.1. ITRLF_FileManagment

Interface name: ITRLF_FileManagment

Documentation: This interface is used to manage files stored via the TReLaF middleware.

3.2.3.2. ITRLF_FileTransfer

Interface name: ITRLF_FileTransfer

Documentation: This interface is used to transfer files stored via the TRLFM middleware.

3.3. FUNCTIONS – USE CASES

DAP will provide three main functions: applying CEA for component (data-handler) selection (see sect. 2.1.1), implementation of the mass-storage middleware - TRLFM (see sect. 2.1.2) and designing the data-access estimator (see sect. 2.1.3).

The selection of the best data-handler (component) will be done by Component-Expert Subsystem, which will be controlled by a special plug-in attached to GridFTP server. This plug-in will take control of the FTP command supported by the GridFTP sever and will require the best component from CES. To make this decision CES has to obtain all useful information for the deduction process from the component 'Storage Element' and form the component 'Metadata Catalog'. Such a session could look as follows:

1. User's application requires from the Replica Manager the best copy of a data-object. Let this data object be a file e.g. 'file1.dat'. The Replica Manager returns the best localization of required data-object.
2. Since this is a file, the application opens a GridFTP session, connects to the node selected by Replica Manager and opens the file for data-transmission. However, the GridFTP Plug-in is attached to the GridFTP server and it creates the call-environment consisted of Data ID and User ID attributes and connects to the Component Expert Subsystem and asks for the best component for reading the data.
3. CES obtains from the GridFTP Plug-in data identifier, which allows CES to find additional information about the data, for example: data type, storage type, data size etc. This information is necessary for selection of the best data-handler. Finally CES selects the best component for the given call-environment and a specified type.
4. Since CES has found the best data-handler then GridFTP can use it and transfers data to the application.

TRLFM is a component, which optimizes the way the data are stored on HSM and makes the read access to it more efficient, especially when the size of a file is greater than 100 MB. This is mainly useful for large multimedia files. Such files are divided on fragments and they are stored on HSM.

Data-Access Estimator is used for the estimation of the bandwidth and the latency of data availability inside the Storage Element. This estimation does not include network transmission aspects. This anticipation is useful for making the right decision by the Replica Manager. Since, the data-access estimation problem depends strongly, amongst other things, on the storage type, thus the way of the estimation of the data-arrival latency is different for HSM, TRLFM, Secondary Storage and relational database etc. This implies, that the CES could be used to simplification the architecture of this component. The session of data estimation could look as following:

1. Replica Manager asks the 'Data Access Estimator' component for the estimation of the data latency.
2. Data Access Estimator asks the CES for the best component for the data-access estimation of the data-arrival latency.
3. CES obtains additional information about the considered data-object; especially it obtains the type of storage, which contains currently the data-object. On the basis of the obtained information about the taking into account data-object, CES finds the best component for the estimation of the data-access latency, and returns the handler to the Data Access Estimator .
4. Data Access Estimator runs the obtained component and returns estimated values to the Replica Manger.

3.4. PERFORMANCE REQUIREMENTS

Performance requirements strongly depend on a type of operations. However, most of operations should be done in time less then 1-5 seconds. Obviously, most replica creation procedure could take much more time, but it is depended on the data volume and the efficiency of network and storage facilities.

3.5. LOGICAL DATABASE REQUIREMENTS

N/A

3.6. DESIGN CONSTRAINTS

The package is intended to work in the future with Globus version 3.0. Thus, it should be compatible with OGSA specification and Web Services.

Furthermore, UML has been established as a standard modeling language in WP3.

3.7. STANDARDS COMPLIANCE

The package should be compliance with following standards:

- UML
- SOAP
- WSDL
- OGSA

3.8. SOFTWARE SYSTEM ATTRIBUTES

The following attributes imply on the package:

- Reliability
- Availability
- Portability

3.9. REQUIREMENTS ANALYSIS

The current shape of proposed architecture is a result of preliminary requirements analysis. As a basis for this requirements-analysis have been used, software requirements stated in early draft version of SRS and in kick-off presentation materials.

TASK 3.4 SRS
Optimization of Data Access
Section 3 Specific Requirements

The below requirements are divided on the three parts: software requirements – SR; stakeholder requirements STRQ and product features FEAT. The last one (FEAT) is in mainly part an implication of SRs and STRQs. Software Requirements (SRs) were articulated in documents and presentations of all potential clients of DAP.

ID	Requirements	Priority	Traced-to
SR1	Task 1.1 Biomedical application	Medium	
SR1.1	Person in the loop New kind of grid applications - a person in the computing loop.	Medium	
SR1.2	Select on demand Data should be selected on demand.	High	
SR1.3	Databases in use Databases in use	Medium	FEAT91.2;
SR1.4	Distributed locations Data gathering, processing and interpretation in geographically distributed locations.	Medium	
SR1.5	Issue of data throughput Issue of data throughput is still a challenge. A scanner produces 2,56 GBytes/sec.	Medium	
SR1.6	Database with scanned medical images and meta data Bio-medical task plans to make an interface to CGfor own database with scanned medical images and meta data	Medium	
SR1.7	History DB Bio-medical task plans to make an interface to CGfor History DB database that is under development.	Medium	FEAT91.2;
SR1.8	Geographically distributed Data generators and databases are geographically distributed.	Medium	
SR1.9	Transfer time should be less then 2 minutes. Access to medical image database by simulator will be used only a few times (possibly once only) during a session to obtain the initial data and later to store simulation result. The amount of data transferred will be in the order of 100 MB. Transfer time should be less then 2 minutes.	Medium	
SR1.10	Higher-order language requirements Higher-order language requirements- C, C++, Prolog, possibly other languages.	Medium	FEAT91.1; FEAT106;
SR1.11	Criticality of the application The application will be uses in a medical domain; correctness is critical.	High	
SR1.12	Optimization of data access Not available optimization of data access.	Medium	
SR1.13	Duration Duration - 1 hour.	Medium	
SR1.14	Latency Latency - 2 sec.	Medium	
SR1.15	Temp storage Temp storage - from 20 MB to 10 GB	Medium	
SR1.16	Permanent storage Permanent storage - about 1 GB	Medium	
SR1.17	Response Response - from 2 sec. - 2 min.	Medium	
SR1.18	Data transfer Data transfer - over 20 MB	Medium	
SR1.19	Manual using of the portal should be not necessary There are biomedical applications, which are not going to use portal and builds in all grid operation in self.	High	FEAT91.1;
SR2	Task 1.2 Flooding Crisis Team Support	Medium	
SR2.1	Many sources The application uses many sources of data and probably many kinds of storage technique (row files, databases etc.)	Medium	FEAT91.1;

TASK 3.4 SRS
Optimization of Data Access
Section 3 Specific Requirements

SR2.2	External sources of information There are external sources of information used by Flooding applications: - Global and regional centers GTS - EUMETSAT and NOAA - Hydrological services from other countries.	Medium	
SR2.3	Databases There are databases in use.	Medium	FEAT91.2;
SR2.4	High availability of resources There is a need of high availability of resources (24/365)	Medium	
SR2.5	Large input/output data sets Large input/output data sets = 50MB - 150 MB /event	Medium	
SR2.6	Duration Duration - 2 days.	Medium	
SR2.7	Response Response - 1/2 hour.	Medium	
SR2.8	Data transfer Data transfer - about 100 MB	Medium	
SR2.9	Latency Latency - minutes.	Medium	
SR2.10	Temporary storage Temporary storage - 100 GB on disk.	Medium	
SR2.11	Acquisition systems There are systems for acquisition and processing of satellite information.	Medium	
SR2.12	Automatic stations There are in use surface automatic metrological and hydrological stations.	Medium	
SR2.13	Radars There are in usage metrological radars.	Medium	
SR2.14	Roles of Participants in VO Flooding Team Support Task specifies data providers and storage providers in Virtual Organization.	Medium	
SR2.15	Kinds of storage Task 1.2 predicts the usage of Permanent storage and temporary storage.	Medium	
SR2.16	Precipitation forecast The output form metrological simulation is stored as a precipitation forecast and is used by other applications. Thus, exchange between metrological, hydrological simulation will be made by exchange data in files.	Medium	
SR2.17	Model repositories Model repositories	Medium	
SR2.18	Topographical data repositories Topographical data repositories	Medium	
SR2.19	Hydro-meteorological data repositories Hydro-meteorological data repositories	Medium	
SR2.20	Optimization of data access Task 1.2 requires an optimization of the data access.	Medium	FEAT91; FEAT91.2;
SR3	Task 1.3 High energy physics	Medium	
SR3.1	Optimize the use of distributed databases Apart from the file-level services offered by DataGrid CrossGrid will offer an object-level service to optimize the use of distributed databases.	High	FEAT91; FEAT91.2; FEAT102;
SR3.2	Distributed databases One of the challenging points is an access to large distributed databases in the Grid.	Medium	FEAT91.2;
SR3.3	Duration Duration - hours	Medium	
SR3.4	Response	Medium	

TASK 3.4 SRS
Optimization of Data Access
Section 3 Specific Requirements

	Response - 10 min.		
SR3.5	Data transfer Data transfer - N * 50 MB	Medium	
SR3.6	Latency Latency - 10 sec.	Medium	
SR3.7	Temp storage Temp storage - over 1 TB	Medium	
SR3.8	Permanent Storage Permanent Storage - over 1 TB	Medium	
SR3.9	Using in grid existing application without modification There are a lot of application, which should be used in grid and modification them is often impossible.	Medium	
SR4	Task 1.4 Weather forecast & air pollution	Medium	
SR4.1	Integration of distributed databases	Medium	FEAT91.2;
SR4.2	Migration of data mining algorithms to Gird	Medium	
SR4.3	Duration Duration - 1 hour	Medium	
SR4.4	Data transfer Data transfer - GBs	Medium	
SR4.5	Data mining and high computing applications. The task includes both data mining and high computing applications.	Medium	
SR4.6	Sharing of the data Massive data sets must be shared by a large community of researches. These researches need efficient transfer of large data sets to perform analyses in their local sites.	High	
SR4.7	Data access control The data management environment must provide security services such tools as authentication of users and control over who is allowed to access the data.	Medium	
SR4.8	Replica selection service In many cases, different participating institutions create local copies and replicas of the simulated and observed data sets to overcome long wide-area data transfer latencies. Thus, system must provide tools able to determine where all existing copy or create a new one to meet performance needs of their applications.	High	FEAT91;
SR4.9	The usage of NetCDF and HDF5 data format Daily output from atmospheric weather prediction simulations is comparatively large (in order of gigabytes). Data are not maintained in databases but rather as flat files, typically stored in a structured data format such NetCDF or HDF5 with associated metadata.	Medium	FEAT102;
SR4.10	Daily-weather-prediction files are not updated once released. Daily-weather-prediction files are not updated once released.	Medium	
SR4.11	Secure and efficient data transport mechanism Application requires a secure and efficient data transport mechanism between storage systems and access to comparatively large amount of data by many geographically distributed users for analysis and visualization of the data.	Medium	FEAT91;
SR4.12	DFS is required Task 1.4 needs something similar to DFS(distributed file system) supported high-volume usage, data replication and local caching.	Medium	
SR4.13	Structured data access Task 1.4 requires an access to structured data from a variety of underlying storage systems, exploring for example HDF5 facilities.	High	FEAT102.1;

TASK 3.4 SRS
Optimization of Data Access
Section 3 Specific Requirements

SR4.14	GridFTP Task 1.4 deems the GridFTP protocol to appropriate for data transfer and access protocol in Grid environment, which realizes weather prediction tasks.	Medium	FEAT107;
SR4.15	Retrieve job output Task 1.4 requires a service or tools allowing users to retrieve job output.	Medium	
SR5	Task 2.3 Benchmarks & Metrics	Medium	
SR5.1	Interactions with Optimization of Data-Access Task 2.3 wants to interact with task 3.4.	Medium	
SR5.2	Access a source of row data In order to obtain performance information, task 2.3 needs to access a source of raw data, as well as the techniques enabling to obtain performance metrics based on the row data.	Medium	
SR5.3	The raw data is processed to produce a derived performance data The raw data is processed to produce a derived performance data (e.g. the time spent by a process in receive over an internal, communication/computation ratio, computation time as a function of application features, like memory access patterns or problem size.)	Medium	
SR5.4	Performance data will concern on Data transfer The grid application related performance data would concern on data transfer.	High	
SR6	Task 3.1.1	Medium	
SR6.1	Retrieve A 'Retrieve' service allowing users to retrieve the job output.	Medium	
SR7	Task 3.1.2	Medium	
SR7.1	Application to manage data files The migrating Desktop will allow the user to access grid resources and his local resources from remote computers. It will allow running applications, to manage data files, store personal settings (configuration definitions that characterize e.g. links to the user data files, links to applications, access to portals and HPC infrastructure, as well as windows settings), independently of the localization or the terminal type.	Medium	FEAT107;
SR7.2	Mechanism for designing and managing the workflow They are going to build some mechanism for designing and managing the workflow.	Medium	
SR7.3	Migration mechanism They are going to build a sharing mechanism that allows the user to make files stored on his machine available from other locations/machines.	Medium	
SR7.4	Optimization of Data Access The uploading procedure is a little bit different from downloading. Two things are taken into consideration: the update of the LDAP base and proper placement of file on the file server chosen by task 3.4. The RA should update mapping of the file name in LDAP according to the answer from the task 3.4 module.	Medium	
SR7.5	Uploading files In general, we assume that the local computer could be switched off every time, so the required files should be copied to a much safer place. The high availability file server must be open for any temporal files waiting for the placement of the job. Therefore, the first step is 'Upload' and the second - the migration of the input file from a temporal GridFile Server to the execution host. Only at that, time could be run the application.	Medium	
SR7.6	Grid desktop The Grid Desktop consists of objects called icons that are links	Medium	

TASK 3.4 SRS
Optimization of Data Access
Section 3 Specific Requirements

	to applications or to any kind of data. An icon has properties that are stored in the LDAPbased server.		
SR7.7	Grid desktop and grid files On the grid, desktop there will be links to grid files and applications. Links are stored within the grid. A double click on a local or grid icon will call an appropriate action.	Medium	
SR7.8	Moving icons between desktop Adding icons to the local desktop will initialize the procedure of storing their properties in the LDAPbased server. Adding an icon to the grid desktop will additionally cause a physical move of file into the grid (into GridFTP Server or other storage).	Medium	
SR7.9	Communications interfaces Proposed protocols are: SOAP, SSL, SHTTP, LDAP.	Medium	
SR7.10	Communication with Task 3.4 via SOAP One of the propositions is to establish communication between the Roaming Access module and the expert system for data management using the SOAPprotocol, which is an XML based protocol and can be used in combination with the HTTPS protocol.	Medium	FEAT106;
SR8	Task 4.2 Integration with DataGrid	Medium	
SR8.1	Linux Red Hat 7.2 The recommended future platform for CrossGridis Linux Red Hat 7.2.	Medium	
SR8.2	RPM v.4 The recommended software distribution tool is RPM v4.	Medium	
SR8.3	GNUMake GNUMake as a make tool.	Medium	
SR8.4	CVS CVS as a code versioning system.	Medium	
SR8.5	UML The UML as a modeling language.	Medium	
SR8.6	CMT, SCRAM CMT, SCRAM as a configuration management.	Medium	
SR8.7	Tools for managing platform dependencies, packaging, exporting Tools for managing platform dependencies, packaging, exporting are: autoconf, CMT, SCRAM.DAR, rpm, GRID install and export tool.	Medium	
SR9	Task 5 Architecture Team	Medium	
SR9.1	Heterogeneous computer and storage systems We are working with heterogeneous computer and storage systems	High	
SR9.2	Web Services The architecture team expects that all new services will be compatible with Web Service concept.	High	FEAT105; FEAT106;
SR9.3	OGSA Whole architecture must be compatible with OGSA specification.	High	FEAT105; FEAT106;
SR9.4	UML as a modeling language As a standard to define designs in the project, we propose UML	High	
SR9.5	Compatibility with Globus 3.0 The third version of Globus is emerged. The release date is predicted on the late of 2002. The CrossGridproject will be finished on spring of the 2005. Thus, Globus v. 3.0 will be probably a standard then, and all todays solutions must be compatible with version 3.0 of Globus.	High	

The stakeholder requests.

ID	Requirements	Priority	Traced-to
STRQ1	Data intensive application	High	FEAT91;

TASK 3.4 SRS
Optimization of Data Access
Section 3 Specific Requirements

	New category of Grid enabled applications computing and data intensive.		FEAT102.2;
STRQ3	Near to real time response There are applications, which need near to real time response.	High	FEAT91; FEAT102; FEAT102.2;
STRQ4	Easy to use for users of the Unix All solutions must be easy to use for user which currently know Unix	High	
STRQ5	The essential programming language is C/C++ As an essential programming language is chosen C/C++	High	
STRQ6	Possibility of usage from any language The grid community is C oriented. However, there are many problems, which require other programming languages. It implies the possibility of usage all essential features from any programming language.	Medium	FEAT105; FEAT106;
STRQ7	The person in the loop It should be kept in mind that in the loop is a human.	High	
STRQ8	The work in a distributed environment The work in a distributed environment	High	FEAT91; FEAT107;

Proposed features realized by DAP .

ID	Requirements	Priority	Difficulty	Traced-from	Traced-to
FEAT91	Optimal replica selection Task 3.4 will adopt Replica Manager component from Globus/ Datagrid. The main goal of this component is to answer to the question which replica of a data-object is the best for some context.	High	High	SR2.20; SR3.1; SR4.8; SR4.11; STRQ1; STRQ3; STRQ8;	FEAT94;
FEAT91.1	All replica-selection features available from code The replica-selection interface of the Replica Manager component has to be accessible from code of any application. It's important to note that the mentioned applications could be written in any programming language.	Medium	Medium	SR1.10; SR1.19; SR2.1;	FEAT105; FEAT106;
FEAT91.2	Optimal database selection In the grid may be plenty replicated databases. System should give some facilities to choose an optimal database. There is an assumption that these databases are well replicated and there are no important differences.	Medium	Medium	SR1.3; SR1.7; SR2.3; SR2.20; SR3.1; SR3.2; SR4.1;	
FEAT94	Data access estimation The estimation of the latency in the data access or the bandwidth of the data transfer inside the Storage Element is and important issue, especially for Replica Manager.	Medium	Medium	FEAT91;	
FEAT102	Optimal data handler selection The strategy of the data access depends on many factors as type of storage device, type of the data, access limitations etc. Each strategy has own data-handlers. There is a necessity of a flexible mechanism of the data handler's selection.	High	High	SR3.1; SR4.9; STRQ3;	
FEAT102.1	Component Expert Subsystem	High	Medium	FEAT102.2;	

TASK 3.4 SRS
Optimization of Data Access
Section 3 Specific Requirements

	CES was introduced to realize optimal data handler selection. CES is based on the Component Expert Architecture.			SR4.13;	
FEAT102.2	Process of data-access requirements The user sets some data access requirements, which affects on low-level data-access strategies (algorithms). The system should choose the best access strategy.	High	Medium	STRQ1; STRQ3;	FEAT102.1;
FEAT105	WSDL description of interfaces The interfaces should be described using WSDL language.	Medium	Medium	FEAT91.1; SR9.2; SR9.3; STRQ6;	
FEAT106	SOAP the essential communication protocol All interfaces should be accessible using SOAP communication protocol. Obviously, this protocol is much slower and might be used only when time restrictions allow this.	High	Medium	FEAT91.1; SR1.10; SR7.10; SR9.2; SR9.3; STRQ6;	
FEAT107	GridFTP as file-data transfer protocol GridFTP is currently the most popular file-data transfer protocol in the Grid environment, and CrossGrid is going to use it. Thus, every large volume data transfer should use GridFTP protocol.	High	Medium	SR4.14; SR7.1; STRQ8;	
FEAT111	Tape Resident Large Files middleware (TRLFM) This middleware is placed on top of an existing HSM system. It is intended to speedup the read access to large files (>100MB) stored on tapes, by means of reducing the latency time.	Medium	Medium		

4. APPENDIX

4.1. EXTERNAL COMPONENTS SPECIFICATION (NOT INCLUDED IN TASK 3.4)

This section specifies components connected to the DAP but not realized by Task 3.4. To unify whole documentation and to make diagrams easier for reading, interfaces to the external components were introduced. These interfaces do not exist in physical implementation of these components but represent logical artifacts, which bring together some similar operations and make external and internal components unified.

4.1.1. ComponentName - GridFTP

GridFTP is a Globus component. This component is still under development and there are plans for many new features. For consistency, this component was depicted similar to others and there are modeled interfaces. Some interfaces are not available yet, e.g. there is no interface to tunnelling SOAP control messages but such an interface is planning and is under construction now. This component should be deployed on each Storage Element. The abbreviation of this component name is GFTP.

4.1.1.1. IGFTP_FileManagment

Interface name: IGFTP_FileManagment

Documentation: This interface groups all GridFTP commands associated with managing physical file structure, e.g. make directory, remove one, delete file etc.

4.1.1.2. IGFTP_FileTransfer

Interface name: IGFTP_FileTransfer

Documentation: This interface groups all GridFTP commands associated with data transfer, e.g. get, put etc.

4.1.1.3. IGFTP_Monitoring

Interface name: IGFTP_Monitoring

Documentation: This interface groups all GridFTP commands associated with monitoring transfer.

4.1.1.4. IGFTP_PluginManagment

Interface name: IGFTP_PluginManagment

Documentation: This interface groups all GridFTP commands associated with managing plugins.

4.1.1.5. IGFTP_ThirdPartyProtocol

Interface name: IGFTP_ThirdPartyProtocol

Documentation: GridFTP allows building own transfer protocols, as an extension for dedicated solutions in some cases.

4.1.2. ComponentName - Portal

Portal is the Cross Grid component, which is developed by another task and is meant for easy access and usage of the Cross Grid. This component should be installed on only one node in virtual organization, and a best solution is to assign one dedicated node to serve this one.

4.1.3. ComponentName - Replica Catalog

Replica Catalog is a component keeping information about all data-objects and theirs' replicas. Important is the possibility of registering different kinds of data-objects as raw files, hierarchical files

and tables in relational databases or collections of objects in objects databases. Since there are several similar components (e.g. Globus Replica Catalog, EDG Replica Catalog etc.), thus it should be selected one and assigned interfaces are just an example of elementary operations required form such component. This component will not be implemented by task 3.4. The abbreviation of this component name is REPC.

4.1.3.1. IREPC_FileRepositoryAccess

Interface name: IREPC_FileRepositoryAccess

Documentation: This interface allows obtaining information about virtual file structure. E.g. it returns all physical files connected to the virtual one etc.

Operation name: getCountOfPhysicalFileInstances

Documentation: Global Virtual File Repository is also a replica catalog, which must often answers to the questions where physical copies of replicas are located. This is such a method.

Protocol: SOAP

Returns: List of physical file addresses (LFNs). It can be empty.

Parameter name: virtualFilePath

Documentation: Logical file path which unambiguously point virtual file.

Operation name: getFileType

Documentation: Because Global Virtual Files Repository keeps as well files as database tables; thus, it must be a way to obtain an answer on a question what physical type is some virtual file.

Protocol: SOAP

Returns: Type of the physical organization of the file.

Parameter name: virtualFilePath

Documentation: Logical file path which unambiguously points a virtual file.

Operation name: getPhysicalInstances

Documentation: Returns a list of physical instances for a given virtual file if any exists, else it returns an empty list.

Protocol: SOAP

Returns:

Parameter name: virtualFilePath

Documentation: Logical file path which unambiguously points a virtual file.

Operation name: getVirtualFile4Physical

Documentation: Returns a list of virtual files, which are attached to a given physical file if any exists, else returns an empty list.

Protocol: SOAP

Returns:

Parameter name: PFN

Documentation: This is physical file name.

4.1.3.2. IREPC_FileRepositoryAdmin

Interface name: IREPC_FileRepositoryAdmin

Documentation: This interface allows managing information in a virtual file structure. Users register physical files, modifies virtual structure etc.

Operation name: attachPhysical2LogicalFile

Documentation: User or administrator registers replicas using this operation. He must specify a virtual file path and a physical file path at least.

Protocol: SOAP

Returns: Succeed/fail code

Parameter name: virtual file path

Documentation: The whole virtual file path.

Parameter name: physical file path

Documentation: The whole path to the physical file

Operation name: makeVirtualDirectory

Documentation: This operation makes a virtual directory in Global Virtual File Repository.

Protocol: SOAP

Returns: Succeed/fail code

Parameter name: dir name

Documentation: Directory which should be made.

Operation name: makeVirtualFile

Documentation: This operation makes a virtual file in Global Virtual File Repository.

Protocol: SOAP

Returns: Succeed/fail code

Parameter name: file name

Documentation: Name of file, which should be created.

Operation name: registerStorageElement

Documentation: Physical files are stored in Storage Elements. Thus, it should be registered in advance. This function registers the storage element.

Protocol: SOAP

Returns: Succeed/fail code

Parameter name: Storage Element Address

Documentation: Address to storage element.

4.1.4. ComponentName - HSM

The abbreviation of this component name is HSMG.

4.1.4.1. IHSMG_FileManagment

Interface name: IHSMG_FileManagment

Documentation: HSMs support often some special interfaces for file management, and this is a model of such interfaces, but they are strongly depended on HSM .

4.1.4.2. IHSMG_FileTransfer

Interface name: IHSMG_FileTransfer

Documentation: HSMs support often some special interfaces for file transfer, and this is a model of such interfaces, but they are strongly depended on HSM . Often, there is no special interface but data are available via standard I/O. In such cases this interface could represent standard I/O library.

4.1.4.3. IHSMG_Managment

Interface name: IHSMG_Managment

Documentation: This is additional, depended on concrete HSM interface. It represents all HSM management functions.

4.1.5. ComponentName - Secondary Storage System

Secondary Storage System represents a disk storage system, which is under control by operating system.

4.1.6. ComponentName - Tape Storage

Tape Storage belongs to the layer of Tertiary Storage System. It represents all solutions using tapes.

4.1.7. ComponentName - Disk Cache

Disk cache is an external component, which is usually out of our control and is used by HSM storage systems to cache files that are stored or will be stored on tapes. The abbreviation of this component name is DICA.

4.1.8. ComponentName - MO Disk storage

This is a kind of physical storage and is introduced only as an illustration.

4.1.9. ComponentName - Metadata Catalog

Metadata Catalog is a Globus component keeping detail attributes of files stored on the grid. This component should be installed on only one node in virtual organization, and the best solution is to assign one dedicated node to serve this one. The abbreviation of this component name is MTCA.

4.1.9.1. IMTCA_Attributes2Files

Interface name: IMTCA_Attributes2Files

Documentation: Metadata Catalog keeps attributes for files. So, this interface allows obtaining files matching to some attributes.

4.1.9.2. IMTCA_File2Attributes

Interface name: IMTCA_File2Attributes

Documentation: Metadata Catalog keeps attributes for files. So, this interface allows obtaining attributes for some file.

4.1.10. ComponentName - Application

The Application component is an example of the component, which is possible to use the component 'File Access Transparent Library' and it doesn't need anything else. This component will be deployed on many nodes in virtual organization. The abbreviation of this component name is APPL.

5. INDEX

A

actor, 5
AML, 5, 16
Application, 12, 13, 14, 31, 32, 39
artifact, 5, 26

C

CE, 5, 12, 13, 25, 26
CEA, 5, 11, 14, 27
CES, 5, 11, 12, 19, 27, 28, 35
CG, 5, 15, 16, 24, 29
Component-Expert Subsystem, 11, 19, 24, 25, 26,
27
CrossGrid, 5, 8, 11, 14, 20, 24, 30, 33, 35

D

DAP, 5, 8, 9, 10, 18, 19, 20, 22, 27, 29, 34, 36
Data Access Estimator, 10, 18, 19, 25, 28
DataGrid, 8, 18, 24, 30, 33
DFS, 5, 31
Disk Cache, 39
DS, 5, 11

E

EDG, 5, 24, 37
ETA, 5, 17

F

feature, 5, 25

G

GIS, 9
Globus, 5, 6, 8, 10, 18, 19, 20, 28, 33, 34, 36, 37, 39
Grid, 5, 6, 14, 15, 18, 20, 25, 31, 32, 33, 34, 35, 36
GridFTP, 5, 8, 10, 11, 18, 19, 20, 25, 27, 32, 35, 36
GRM, 5

H

HDF5, 5, 31, 32
HSM, 5, 6, 8, 15, 16, 17, 20, 26, 27, 35, 38, 39

I

ICEXS_AdminOfComponents, 26
ICEXS_ComponentsSelection, 26
IGFTP_FileManagment, 36
IGFTP_FileTransfer, 36
IGFTP_Monitoring, 36
IGFTP_PluginManagment, 36
IGFTP_ThirdPartyProtocol, 36
IGPLG_FileTransfer, 25

IHSMG_FileManagment, 38
IHSMG_FileTransfer, 38
IHSMG_Managment, 38
IMTCA_Attributes2Files, 39
IMTCA_File2Attributes, 39
IREMG_ReplicaSelection, 24
IREPC_FileRepositoryAccess, 37
IREPC_FileRepositoryAdmin, 37
ISTEL_Additional, 24
ISTEL_Configuration, 24
ITRLF_FileManagment, 27
ITRLF_FileTransfer, 27

L

LDAP, 5, 32, 33

M

MDS, 9
Metadata Catalog, 10, 18, 19, 20, 27, 39
MMSRS, 5, 6, 15, 16
MO Disk storage, 39

N

NetCDF, 5, 31

O

OGSA, 5, 20, 28, 33

P

PDO, 5
Portal, 8, 36

R

RA, 6, 32
RC, 6
Replica Catalog, 18, 19, 36
Replica Manager, 5, 10, 18, 19, 24, 27, 28, 34

S

SE, 6, 18, 19
Secondary Storage System, 39
SOAP, 5, 6, 18, 20, 24, 25, 28, 33, 35, 36, 37, 38
Storage Configuration, 26
Storage Element, 6, 10, 18, 19, 24, 25, 26, 27, 34,
36, 38

T

T34, 6, 22
Tape Storage, 39
TRLFM, 6, 8, 15, 19, 20, 25, 26, 27, 35
TSS, 6, 15, 16, 17

TASK 3.4 SRS
Optimization of Data Access
Section 5 Index

U

UCFM, 6, 15, 16
UML, 6, 28, 33

W

WP1, 8
WSDL, 6, 7, 20, 28, 35

V

VTSS, 6, 15, 16