

SDSC Technical Report 2007-2

**Constraint-based Knowledge Systems for Grids,
Digital Libraries, and Persistent Archives:**

**Final Report
May 2007**

**Reagan W. Moore (SDSC)
Arcot Rajasekar (SDSC)
Michael Wan (SDSC)
Wayne Schroeder (SDSC)
Yannis Katsis (UCSD)
Dayou Zhou (UCSD)
Richard Liu (UCSD)
Alin Deutsch (UCSD)
Yannis Papakonstantinou (UCSD)**

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the National Science Foundation, the National Archives and Records Administration, or the U.S. Government.

Table of Contents

1. Introduction	1
2. Basic concepts in distributed data management	2
3. Implementation Status	7
4. Rules in iRODS.....	9
5. Validation of iRODS Rules	10
6. User Administration Interface for Managing Rules	11
7. RLG/NARA Assessment Criteria for a Trusted Digital Repository	12
8. Electronic Records Archives Capabilities List	14
9. Future Development	15
10. Software Distribution	16
11. Acknowledgement:	17
12. References	17

1. Introduction

The need for data management systems that assemble distributed files into shared collections continues to grow. The shared collections enable collaborative research, the publication and discovery of data and resources, the federation of real-time data from sensor systems and integration with static and virtual files and databases, and the management of technology evolution for preservation environments. Each of these environments inherently deals with combinations of distributed data sources, distributed data curators, and distributed users. Thus the ability to assemble a shared collection whose records reside in multiple types of storage systems, at multiple institutions, located in multiple nations now is an essential requirement for collaborative research projects.

The NSF funded project on “Constraint-based Knowledge Systems for Grids, Digital Libraries, and Persistent Archives” [1] is exploring the development of constraint and rule-based systems through the creation of next-generation data grid technology called the integrated Rule-Oriented Data System (iRODS). The new data management technology provides for the management of distributed data based on the model of the Storage Resource Broker data grid system [2,3]. The goal is to be able to characterize the management policies that are needed to enforce authenticity, integrity, access restrictions, data placement, and data presentation, and to automate the application of the policies for services such as administration, authentication, authorization, auditing and accounting as well as data management policies for replication, pre- and post-processing and metadata extraction and assignment. The management policies are mapped onto rules that are applied on the execution of all data management operations. The rules are controlled by parameters (rule metadata attributes). The results of applying the rules are saved as persistent state information (file metadata attributes) that is associated with each record in the collection. By examining the persistent state information, the effect of applying any rule can be tracked and evaluated.

Rule sets can be implemented to perform multiple tasks. Rule sets can control the properties of a specified collection for a defined set of users to meet a specific management goal. An example of such a rule set is the automation of the evaluation of trusted digital repository assessment criteria [4]. This example is very interesting because it explicitly characterizes the management policies as well as provides a mechanism to track the results of applying the management policies. An example of this mapping is given in Section 7. Applications of this rule set are appropriate for the NSF DIGARCH [5] project being implemented in collaboration with the Library of Congress and for digital library projects that manage data curation policies. Examples of the latter include the NSF National Science Digital Library (NSDL) [6], the DSpace [7] and Fedora [8] digital libraries, the NARA Producer Archive Workflow Network [9], etc.

A second type of application occurs in persistent archives [10,11,12,13,14]. It is possible to define a mapping of the set of capabilities required by the NARA Electronic Records Archives [15] to rules that control execution of fundamental services [16]. This enables an understanding of the basic services required to implement all 854 listed capabilities, as well as providing a representation of the preservation metadata that is managed in a persistent information store. A total of 174 rules applying 212 metadata attributes can be defined for the implementation of the ERA capabilities. An example of this mapping is given in Section 8.

A third and important application type occurs in cyber infrastructure management of shared collections within data grids [17,18]. Management polices are required at the storage resource

(node) level, at the shared collection (data grid zone) level, and at the level of federation of zones. Data management rule sets are planned for distributed sensor real-time data systems such as the NSF ORION project [19] and the NSF NEON project [20] (and their eventual federation). Other applications are in domain-specific data grids, for virtual organizations such as the NSF National Virtual Observatory (NVO) [21], the National Optical Astronomy Observatory (NOAO) [22], the NSF Southern California Earthquake Center (SCEC) [23], the NSF National Earthquake Engineering Simulation (NEES) [24], the NSF Science Environment for Ecological Knowledge (SEEK) [25], etc. Internationally shared collections are being created in projects such as the eScience data grid [26], Worldwide Universities Network data grid [27], PRAGMA – Pacific Rim Applications and Grid Middleware Assembly [28], APAC – Australian Partnership for Advanced Computation [29], KeK High Energy Accelerator Research Organization [30], BaBar high energy physics data grid [31], and Taiwan’s National Digital Archives [32]. Current research, design and development in the iRODS project are geared towards these three types of applications with possible extension to other applications.

As part of the iRODS project, we are also conducting research into theoretical aspects of micro-service oriented and rule-based data management system. Topics include consistency of rule bases, consistency of a changing rule system, abstraction of metadata into relational schemas, automatic presentation and accession of data and metadata and new models in developing user interfaces (see Sections 5 and 6). Further areas of theoretical interest include notification semantics, distributed error management, scalable rule-based systems, federation of trusted rule-based data management systems, and concepts in data grids, persistent archives and digital libraries applied to the micro-service oriented and rule-based data management paradigm.

The integrated Rule-Oriented Data System (iRODS) represents the next generation of distributed data management technology. The development of iRODS has been driven by the lessons learned from the deployment and use in production of the Storage Resource Broker data grid technology (SRB) and by adapting well-known paradigms from other fields such as active databases, transaction systems, logic programming, workflows and service-oriented architecture. A beta release of the iRODS software was made in December 2006. A second release is slated for May 2007. The software is available for download from the wiki <http://irods.sdsc.edu/>.

2. Basic concepts in distributed data management

The fundamental principals implemented within the SRB are also planned for implementation within iRODS. The first release of the iRODS technology implemented a core subset of the functionalities provided by SRB that are needed for distributed data management. Both the SRB and current iRODS system provide mechanisms for [13]:

- Data virtualization, the ability to manage the properties of the shared collection independently of the storage systems where the data are located. These properties include the name spaces used for files (logical file name), users (distinguished user name), and resources (logical resource name).
- Trust virtualization, the ability to authenticate users and manage access controls independently of the storage systems where the data are located.
- Latency management, the provision of bulk operations needed to minimize latency when accessing remote storage systems. Bulk operations are used for aggregation of files before movement and storage, aggregation of I/O commands through use of remote procedures, and aggregation of metadata to enable bulk loading of attributes into a metadata catalog.

- Standard operations that support the functions required by user-requested access methods. These operations include the ability to administrate the files, manage the metadata catalog, manipulate state and descriptive information associated with each file, manage users, and manage addition of new resources.
- Standard operations that are executed at each remote storage system. These operations include the Posix I/O operations (create, open, close, unlink, read, write, seek, sync, stat, fstat, mkdir, rmdir, chmod, opendir, closedir, and readdir); bulk operations for aggregating files, I/O commands, and metadata; integrity operations (checksum, synchronization); creation of replicas, versions, and backups; support for network devices (client-initiated parallel I/O, server-initiated parallel I/O, network load levelers, private virtual networks), file registration, and remote procedures.
- Standard operations for managing a catalog within a database (SQL generation, bulk metadata load and unload, schema extension, user-defined metadata attributes, import and export of XML files).
- Federation protocols for access across independent data grids. SRB has a core set of cross-grid APIs that can become a standard for federation of data grids.
- Community-specific clients. Currently 24 different APIs are used with the SRB: C library calls, C++ library calls, Unix shell commands, Java class library, Windows load library, Perl load library, Python load library, Python browser, Perl browser, Windows browser, Web cgi interface, DSpace digital library, Fedora digital library, Open Archives Initiative Protocol for Metadata Harvesting, Kepler workflow actors, Real-time sensor packet stream access API, Cheshire analysis environment, GridFTP transport, I/O redirection libraries, Semplar MPI I/O library, HDF5 library, GridSphere portal, WSDL, OpenDAP data access protocol. For the iRODS system, the C library calls and Unix shell commands are provided. Ports are underway for the Java class library, Python browser, PHP-based browser, and Windows browser. Once the Java class library interacts with the iRODS system, other clients based on Java will work automatically.

iRODS represents the continued evolution and abstraction of the basic control mechanisms required to manage distributed environments. The control mechanisms that have been implemented by current technologies include:

- Storage virtualization. This is the approach of commercial software and is represented by SAN and NAS technology that allow multiple operating systems to control portions of the same disk storage infrastructure.
- Resource virtualization. The grid community provides standard services to enable jobs to be executed across independently managed compute and storage resources.
- Data virtualization. Data grids such as the SRB and iRODS manage the properties of a shared collection that is distributed across multiple sites.
- Trust virtualization. Both Grids and Data Grids manage authentication, authorization, auditing and accounting independently of the remote resource.

The additional virtualization mechanisms that are needed to improve cyberinfrastructure are:

- Workflow virtualization. This is the ability to manage the execution of a distributed workflow independently of the compute resources where the workflow components are executed. This requires the ability to manage the properties of the executing jobs independently of the choice of grid technology. Most grid workflow systems implement client-side workflows in which the data are moved to the compute resource for manipulation. The iRODS system implements the concept of server-side workflows through chaining of micro-services within nested rule sets, and the execution of the workflows at the remote storage system.

- Management policy virtualization. This is the expression of management policies as rules that can be implemented independently of the remote storage system. We characterize management policies in terms of policy attributes that control desired outcomes. For each desired outcome, rules are defined that control the execution of the standard micro-services (remote operations). For each rule application, persistent state information is maintained on the result of executing the micro-service. Consistency rules can be implemented that verify that the micro-service outcomes comply with the policy attributes. Rule-based data management infrastructure makes it possible to express management policies as rules and define the outcome of the application of each management policy in terms of updates to the persistent state information. iRODS applies the concept of transactional rules (ACID properties at data management level) using datalog-type Event-Condition-Action rules working with persistent shared metadata.
- Micro-service virtualization. The operations that are performed by the rule-based data management systems can be encapsulated in micro-services. A logical name space can be constructed for the micro-services that makes it possible to name, organize, and upgrade micro-services without having to change the management policies. This is one of the key capabilities needed to manage versions of micro-services, and enable a system to execute correctly while the micro-services are being upgraded. iRODS micro-services are constructed on the concepts of well-defined input-output properties, consistency verification, and roll-back properties for error recovery. The iRODS micro-services provide a compositional framework realized at run-time.
- Rule virtualization. This is a logical name space that allows the rules to be named, organized in sets, and versioned. A logical name space for rules enables the evolution of the rules themselves.

One way to characterize the impact of the iRODS system is that it adds management virtualization to the data and trust virtualization mechanisms supported by the Storage Resource Broker. This is displayed in Table 1.

Table 1. Mapping of Community Policies to Rules and Micro-services

<i>Data Management Environment</i>	Conserved Properties	Control Mechanisms	Remote Operations
Management Functions	Assessment Criteria	Management Policies	Capabilities
	Data grid – Management virtualization		
Data Management Infrastructure	Persistent State	Rules	Micro-services
	Data grid – Data and trust virtualization		
Physical Infrastructure	Database	Rule Engine	Storage System

Each community has objectives that must be met consisting of assertions they want to make about their digital holdings. The assertions are equivalent to conserved properties that they want to be able to verify at any point in time through validation of assessment criteria. They define control mechanisms to control how the digital holdings are managed, and express these mechanisms through community-specific management policies. Each community has specified services (or capabilities) that they provide to the users of their digital holdings. We observe that a major challenge is that the commercial storage systems provide low-level byte operations rather than the

operations needed to implement the desired capabilities. To simplify the development of a data management application, higher-level services are needed that simplify the creation of desired capabilities. In the iRODS environment, micro-services are used to aggregate the operations provided by remote storage systems into desired operation sets that simplify the creation of data management capabilities. The management policies are mapped onto rules that control the execution of the micro-services. Assessment criteria are evaluated by queries on the persistent state information created whenever a micro-service is executed.

The mapping from the management virtualization layer (iRODS) to the physical infrastructure such as databases, rule engines, and storage systems, is managed by traditional data grid data and trust virtualization mechanisms. The result is that the iRODS system manages six logical name spaces:

- Logical name space for users
- Logical name space for digital entities (files, URLs, SQL command strings)
- Logical name space for storage systems
- Logical name space for rules
- Logical name space for micro-services
- Logical name space for persistent state information

The iRODS system can be extended by adding sets of {rules, micro-services, persistent state}. The application of a new micro-service under the control of a new rule generates new state information. This means the system can evolve, with both old rule sets and new rule sets running in parallel on different collections. A rule can be written that manages the migration of data from an old-rule base to a new rule-base.

The iRODS system is being implemented as open source software. The mechanisms needed to create production quality software are being used in the iRODS development. These include:

- Licensing under the BSD open source license
- CVS system to manage the source code
- Daily build of the software
- Daily function testing of the software
- Independent software builds and testing on the NMI testbed at the University of Wisconsin
- Wiki for collaborative creation of documentation
- API generator to ensure uniform command line options across all operations
- Consistent error numbers for all observed failures, and recovery procedures for each action
- Peer-to-peer server architecture to enable dynamic expansion of the system

Many of these attributes enable the assembly of a modular system that allows alternate implementations of specified services. This is essential for managing upgrades to the system, with individual components replaced as new features are added. The architecture is shown in Figure 1.

At a high-level, the components shown in Figure 1 are used as follows:

- User executes a task which translates to a set of server calls from the client interface, for accessing the shared collection
- The user is authenticated and the desired collection is identified
- An appropriate rule set is identified in the rule base that specifies what the user is allowed to do with the collection
- The rule set is loaded into a rule engine

- The metadata attributes for the records that will be manipulated are loaded into a temporary state repository (“Current state”) from the persistent metadata store (“Meta Data Base”)
- The rule engine executes the micro services
- On successful completion, the persistent metadata store is updated.

The administrator (and privileged users of the system) may modify, delete or generate new rules and micro services and add new resources. This requires an additional set of interfaces that support:

- Organization of the micro services – Service Manager
- Organization of the rules – Rule Modifier Module
- Validation of the consistency of the rules and services across versions – Consistency Check Module
- Organization of the resources – Configuration Modifier Module
- Management of the metadata needed for rule execution and persistent state – Metadata Modifier Module
- Validation of the consistency of the metadata across updates – Consistency Check Module

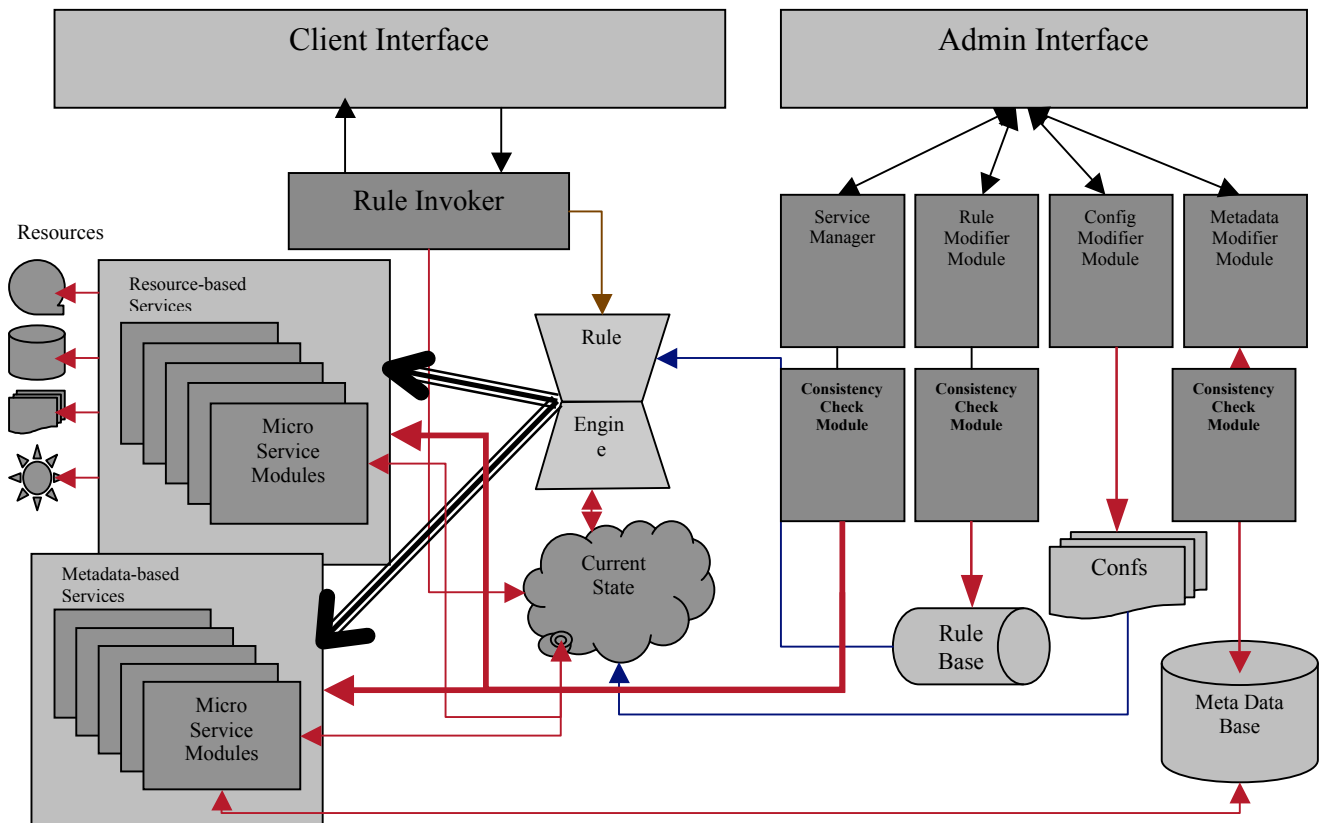


Figure 1. Architecture Design for a Rule-based Data Management System

3. Implementation Status

The original plan was to use portions of the SRB software to speed the implementation. However the redesign of the software to support micro-services required a rewrite of the entire code. A desired consequence is that iRODS is being implemented as open source software, and is being published under a BSD license. SDSC is actively soliciting collaborative development of the software. A Wiki has been established that lists the possible areas of collaborative development and describes the status of the project. Current collaborators include the UK e-science data grid, the IN2P3 institute in France, and researchers in the US and Australia.

To date, over 100,000 lines of “C” code have been written. This should be compared with the size of the Storage Resource Broker software that is about 300,000-500,000 lines of “C”, depending upon whether clients are included. Implemented components include:

- Logical name space for labeling resources
- Logical name space for labeling files
- Logical name space for users
- Basic client/server model with a more efficient protocol than was used in the SRB. The protocol serializes control data structures for movement between servers. Small files can be transferred with a single message, improving performance by up to a factor of 2.
- Federation of peer-to-peer servers at remote storage resources.
- Packing/unpacking modules so that very complex data structures can be transferred between clients and servers, and between servers. These routines are used to implement the low level client/server protocol and the API framework.
- Data movement including parallel I/O, and server to server third party transfer.
- Authentication environment supporting single sign-on. Authentication is currently implemented through a challenge-response mechanism. GSI grid security infrastructure will be integrated next.
- Standard operations performed at the remote storage system. Fifty operations have been implemented including Posix I/O operations, replication, and MD5 checksum.
- Standard API level commands. Examples of iCommands that have been implemented include:
 - iinit (connect to iRODS and authenticate the user),
 - iput (put a file into a shared collection),
 - iget (get a file from a shared collection),
 - icp (copy a file within the shared collection),
 - ils (list files in the shared collection),
 - irm (remove a file from a shared collection),
 - imkdir (make a directory in the shared collection),
 - icd (change directory within the shared collection),
 - imv (move or rename files in a collection),
 - iphymv (physically move files between resources),
 - ipwd (print out the name of the current working directory within the shared collection),
 - imeta (add, remove, list, or query user-defined Attribute-Value-Unit triplets metadata),
 - ilresc (list resources),
 - ichksum (checksum files and collections),

- irmtrash (manage removal of files in a trash collection),
 - irepl (replicate a file),
 - itrim (trim the number of replicas),
 - irsync (synchronize files and collections of files),
 - ireg (register files and directories from the local system into an iRODS collection),
 - iexit (terminate interactions with the shared collection),
 - iadmin (support administrative functions). Administration functions include:
 - lg [name] – list group information
 - lu [name] – list token information
 - lr [name] – list resource information
 - ls [name] – list directory, subdirectory and files
 - lz [name] – list zone information
 - lg [name] – list group information and group members
 - lf DataId – list file details where DataId is output from the “ls” command
 - lgd Name - list details of the group “Name”
 - mkuser Name Type Zone DN – make user
 - moduser Name [name | type | zone | ND | comment | infor | password] newValue – modify user attribute
 - rmuser Name Zone – remove user
 - mkgroup Name - make group with the group “Name”
 - rmgroup Name - remove group with the group “Name”
 - atg groupName userName - add a user “username” to group “groupName”
 - rfg groupName userName - remove user “username” from group “groupName”
 - mkdir Name – make directory
 - mkresc Name Type Class netAddr defPath Zone – make resource
 - rmresc Name – remove resource
- New persistent metadata catalog called RCAT. This automatically generates the SQL required to interact with persistent metadata and provides a standard API for interacting with databases.
 - Rule engine. This executes rule sets that are downloaded from the RCAT. A number of rules for data access, registry, and resource registration have been developed, with more being added.
 - User authentication system. A challenge response authentication system has been written that uses an MD5 one-way hash algorithm to confirm the password without transmitting it over the network.
 - Client code generation tools. These simplify the creation of new C library calls and shell commands, ensuring a common set of options are used across all commands.
 - Installation script “install.pl”

We have successfully built iRODS on the NMI Build and Test facility on a Linux host. This included installation of a PostgreSQL database, an odbc interface to PostgreSQL, and the iRODS system. iRODS is near production level software that includes support for recursive operations on directory structures, and support for both Unix and Windows file naming conventions.

4. Rules in iRODS

iRODS is an exemplar implementation of an "Adaptive Middleware Architecture". AMA provides a way to develop middleware systems that can be customized easily to adapt to the differing needs of the end user community. The foundation for an AMA system is based on the following key concepts:

1. A Persistent Database [#] that shares data (facts) across time and users.
2. A Transient memory [\$] that holds data during a session.
3. A set of Actions [T] that name and define the tasks that need to be performed
4. A set of internal well-defined callable microServices [P] made of procedures and functions that provide the methods for executing the sub-tasks that need to be performed,
5. A set of external Attributes [A] that is used as a logical namespace to externally refer to data and metadata.
6. A set of external micro Services [M] (or methods) that is used as a logical namespace to externally refer to methods in the rules.
7. A set of mappings [DVM] that defines a relationship from external attributes in A to internal elements in # and \$.
8. A set of mappings [FNM] that defines a relationship from external micro services in M and Actions T to procedures and functions in P and other action names in T. In a sense FNM can be seen as providing aliases. One use will be to map different versions of the functions/procedures in P at run time to the actual execution process.
9. A set of rules [R] which defines what needs to be done for each action [T] and is based on A and M.

There can be one or more rules defined for each action. Actions provide decision-based transactional workflows during a session and provide consistency management across sessions. Actions invoke micro-services and other actions to provide the functionality. The rules are in clusters called 'rule pods'. The rule pods are combined to form the rule base that is used during the execution of the requested operation. Different rule pod combinations can give different results! When more than one rule is defined for an action, the rules are sorted by priority. The conditions on the rule with the highest priority are checked for firing the rule. If the conditions are satisfied, then the body of the rule is executed. If there are any failures in the body of the rule, a recovery action is initiated to bring the state of the transient memory and persistent database to the state before the firing of the rule and the next rule in priority order is tried. Hence each rule is transactionally consistent, with either the body being completely executed or no changes being performed. A rule body can be made of micro-services as well as actions and hence the execution can be complex and possibly recursive. A rule consistency checker is invoked when a new rule is added to verify that the rules do not have un-ending loops in execution (cf. Section 5).

Some of the concepts that are used in the iRODS system are:

1. Declarative Programming - through a Rule-based Approach along with rule-consistency checks performed to verify rule execution for cycles and other consistency checks.
2. Transparent Processing & Agile Programming - similar to Business Rules Logic.
3. Event Condition Action (ECA) Paradigm - similar to active databases.
4. Transactional & Atomic Operations - Similar to ACID properties of RDBMS. Each rule either succeeds completely or does not change the operational data (both transient and persistent metadata).

5. WorkFlow Paradigm for defining a sequence of tasks.
6. Service oriented paradigm based on micro-services and rules.
7. New Programming paradigms - based on coding micro services and developing workflows (rules) and stitching the microservices at runtime to the requested operation.
8. Abstraction and logical naming at multiple levels: data, collections, resources, users, metadata, methods, attributes, rules and micro-services
9. Novel managemnt of version control in the execution architecture. All versions can coexist. Users can apply their versions and rules at the same time to achieve their tasks.
10. Data grid paradigm providing standard distributed data management functions: collection management; federation; descriptive metadata; third party authentication; authorization; auditing and accounting (quota system); bulk, parallel and third-party transfers; heterogeneous resource access including file systems, archival systems, databases, and sensor streams; autonomous administration.
11. Digital library paradigm providing standard digital library functions: metadata based discovery; metadata extraction; extensible metadata; accession workflows; curation support; access roles.
12. Persistent archive paradigm providing standard preservation functions: containerization; metadata packaging and Archival Information Packages; infrastructure independence; semantic replication support; format migration policy support; rcord replication; federation across dim and dark archives; validation checks; lineage tracking; auditing; provenance metadata.

The relationship between the logical names, the excutable micro services and the rules is shown in Figure 2.

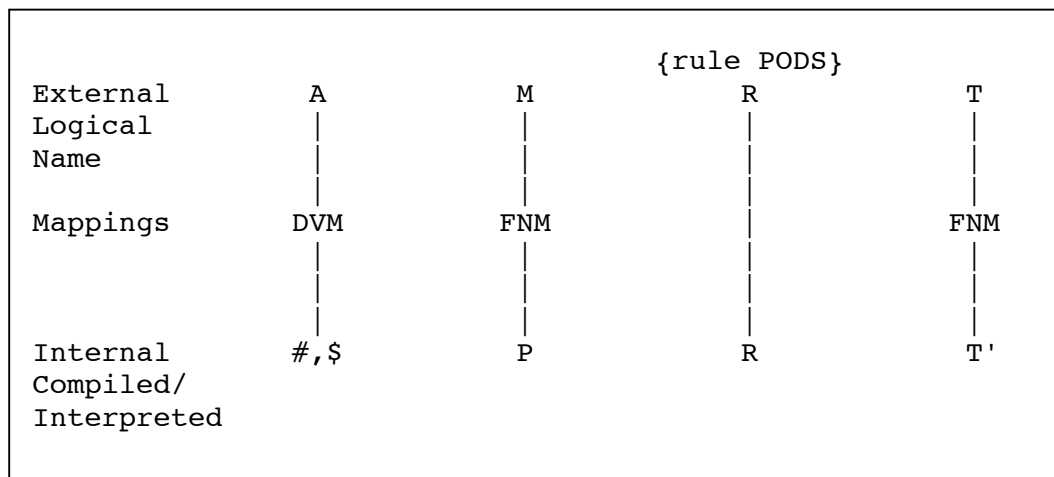


Figure 2. Mapping of extrnal logical names to the executable rules and micro services

5. Validation of iRODS Rules

A rule-oriented data system has the flexibility to support dynamic changes to both the rules and micro services. This means new versions of the software can be updated and incrementally installed. This is feasible if the consistency of the rules can be validated.

A formal verification process is being researched by Dayou Zhou under the direction of Professor Alin Deutsch. A specification language called WAVE has been developed on which the reasoning can be applied. This does require translation of iRODS rules into the WAVE syntax.

Currently, a set of rules that describe operations in the SRB data grid have been translated into the WAVE specification language. The language specifies states, input parameters, actions, and a rule schema. Input rules are defined for initiating operations. State rules are defined that govern actions, and Target Rules specify changes to the persistent metadata.

The WAVE program automates verification of temporal properties, such as the causal order of multiple micro services that a rule might apply. The system is designed to verify Prolog-style rules. Two examples are listed below:

- "Permission checks must be performed for D_CREATE before insertion into ADREPL, which in turn must occur before insertion into ADACCS."
(performPermissionChecksForD_CREATE() B ! ADREPL("tuple"))
&&
(ADREPL("tuple") B ! ADACCS("tuple"))
- "Insertion into ADREPL implies that data modify timestamp must have been 'null' at some point."
F ADREPL("tuple") -> F DMTS("NULL")

Verification time and performance seems to be very good. This is the first time that automated formal verification has been performed on any part of the iRODS system. It is our belief that, with the implementation of an automated translator (see "next steps" below), the use of WAVE, with its high expressivity and strong theoretical guarantees, will greatly contribute to checking correctness and robustness of the iRODS system. Two new publications [33,34] describe the WAVE environment.

The research on mapping iRODS rules into the WAVE syntax forms the basis for a Masters thesis that is being written by Richard Liu, a graduate student in the CSE department at UCSD. The title of the thesis is "A Verification Framework for a Rule-Oriented Data Grid management System".

Current work includes the development of a 'service discovery'. Here, a 'service' can have a broad meaning, in particular, it can be an iRODS rule base. Service discovery is a problem in which we find desirable services without having to know all details about them. We believe this is useful and applicable in the case of iRODS since developers may just want to locate rule base(s) which can perform certain tasks without much knowledge about their details. We describe services by the properties they are known to satisfy (which follows nicely from our first stage, where we worked with SDSC on specifying and verifying properties) and index the properties. This way, we maintain a 'broker' for all the available services (rule bases). When a user issues a 'query' (which is also expressed as a property), it is evaluated against the broker to efficiently locate the services which are known to satisfy the query property.

6. User Administration Interface for Managing Rules

The ability to manage dynamic changes to rule sets requires the design of a system that allows scientists to easily create hosted web applications that capture the data and process needs of their

communities. A tool to support dynamic construction of rules is being developed by Yannis Katsis and under the direction of Professor Yannis Papakonstantinou. In order to facilitate this need, we are working on a system, called *app2you* that allows every scientist, even if she has no web application programming and database design experience and knowledge, to create and own a *customized hosted Web application* and *underlying database* where her community exchanges data and collaborates. We call such a user the *community application owner* (or just *owner*).

App2you allows a community owner to build a customized Web application simply by drawing the web pages of the application using an appropriate WYSIWYG interface. No knowledge of web application programming or database design is needed in order to use the described interface. While drawing the web pages of the application the user implicitly (i) customizes the series of steps (i.e., the process) followed by the community members during their collaboration, (ii) specifies the types of exchanged data and (iii) assigns access rights for the users of the application (i.e., the members of the community), thereby specifying the role of each user or group of users in the collaboration. Essentially by drawing web pages she implicitly specifies rules about the (i) workflow used by community members to enter and retrieve data, (ii) the data that are of interest to the community and (iii) the access control scheme that models the functioning of the community. In order to make bootstrapping new applications easier, the system also provides Web application templates, which the user can modify through an appropriate WYSIWYG interface to implement her community's data and process needs.

While the community owner designs the web pages, app2you analyzes the HTML structure of the drawn pages, the hyperlinks connecting the pages and other aspects of the web application and infers the structure (schema) of the underlying database and a web application that appropriately queries the database in order to retrieve the data that are needed by certain pages and updates the database with the data submitted by the user. The relevant algorithmic challenge is how to automatically derive from the pages and links drawn by the owner a database schema (i.e., the structure of the underlying database) and the web application programs that interact with the database and produce the pages.

The applications of this technology go beyond the generation of a web interface. Current research efforts are examining the ability to work with a pre-defined database schema, and support modifications to the schema. This will make it possible to create an interface that handles updates to the persistent metadata catalog as new rules and new micro services are added.

7. RLG/NARA Assessment Criteria for a Trusted Digital Repository

The iRODS system integrates the capabilities of data grids, digital libraries, preservation environments, and digital repositories. One of the concerns about the design is the verification that the types of rules that are supported by iRODS will be complete, that is, able to manage the types of assertions required by each of the data management applications.

To evaluate the iRODS design, the RLG/NARA assessment criteria for a trusted digital repository were mapped to iRODS style rules. A description of the mapping was published in the proceedings of the Joint Conference on Digital Libraries [35]:

The assessment raised the following observations:

1. The assessment criteria can be mapped to management policies.
2. The management policies can be mapped to a set of rules whose execution can be automated.
3. The rules require definition of input parameters that define the assertion being implemented.

4. The execution of the rules generates state information that can be evaluated to verify the assertion result
5. The types of rules that are needed include:
 - Specification of assertions (setting rule parameters - flags and descriptive metadata)
 - Deferred consistency constraints that may be applied at any time
 - Periodic rules that execute defined procedures
 - Atomic rules applied on each operation (access controls, audit trails)
6. The rules can be applied to multiple levels of management granularity. These can correspond to the enterprise level, the archives level, the collection (record-series) level, and the item level as expressed in the DSpace digital library. A specification of the multiple levels of management granularity is needed to understand how to apply the assessment criteria. It is possible to specify assessment criteria that are specific to a single collection.

The rule that is applied at each level of granularity may differ, even though the same assessment criteria are being applied. This is one of the most important observations, that each management policy may require the definition of multiple rules that are applied at different levels of granularity. We observed that the types of rules that were defined tended to be correlated with the level of management granularity. Thus the rules used at the enterprise level are typically assertions that define the state information required by rules executed at finer levels of granularity. The deferred rules are typically applied at the collection level to enforce assertions made on the collection. Examples are checking compliance with consistency constraints. The periodic rules are applied at the collection (record series) level, and are driven by mandates for periodic validation of integrity. An example would be the validation of checksums every 6 months. The atomic rules are evaluated at the item level on each execution of a related operation. The standard example is the checking of access controls before an operation is performed upon a file.

7. The actual implementation of the assessment criteria is dependent upon the persistence of the name spaces on which the management policies are applied. The management policies need to differentiate between unique identifiers for users, files, metadata and constraints; the name space for defining management state information; and the name space for physical resources (storage systems, databases).
8. The trusted preservation repository should implement multiple levels of virtualization to enable migration onto new technology without impacting the ability of the system to meet the assessment criteria. In practice this includes both persistent name spaces and standard operations for interacting with storage systems (data virtualization), authentication environments (trust virtualization), and rule expression engines (constraint virtualization).
9. The input parameters needed for execution of the rules and the state information that results from application of the rules determines the metadata required for authenticity and integrity. This means that the specification of a valid rule set for a community can define the metadata attributes that should be associated with each record in the shared collection. This is the opposite of current archival practice, which specifies authenticity and integrity metadata, and then attempts to define the rules required to maintain the authenticity and integrity metadata.

A major advantage of a rule-based data management system is that it can identify the metadata required to implement the desired management policies, as well as the rules that enforce the management policies. A trustworthy digital repository needs to demonstrate that the rules needed to manage authenticity and integrity have been implemented, and that the metadata required to execute the rules is being preserved.

8. Electronic Records Archives Capabilities List

The National Archives and Records Administration has published a list of 854 capabilities required for a production preservation facility. This list includes the set of operations that need to be performed by the data management system. To verify that the iRODS design is consistent with the desired capabilities, a mapping was derived of the listed capabilities onto iRODS style rules. This process identified not only the rule set needed to implement the ERA, but also the metadata that needs to be managed. The capabilities were quite generic and would apply for the most part equally well to other data management applications. They included:

- Management of disposition agreements describing how record retention and disposal actions
- Accession, the formal acceptance of records into the data management system
- Arrangement, the organization of the records to preserve a required structure (implemented as a collection/sub-collection hierarchy)
- Description, the management of descriptive metadata as well as text indexing
- Preservation, the generation of Archival Information Packages
- Access, the generation of Dissemination Information Packages
- Subscription, the specification of services that a user picks for execution
- Notification, the delivery of notices on service execution results
- Queuing of large scale tasks through interaction with workflow systems
- System performance and failure reports. Of particular interest is the identification of all failures within the data management system and the recovery procedures that were invoked.
- Transformative migration, the ability to convert specified data formats to new standards. In this case, each new encoding format is managed as a version of the original record.
- Display transformation, the ability to reformat a file for presentation.
- Automated client specification, the ability to pick the appropriate client for each user.

The analysis identified five types of metadata attributes:

- Collection metadata 11 attributes
- File metadata 123 attributes
- User metadata 38 attributes
- Resource metadata 9 attributes
- Rule metadata 32 attributes

In order to support all of the capabilities that were desired, multiple systems needed to be integrated:

- PAWN submission pipeline 34 operations
- Cheshire indexing system 13 operations
- Kepler workflow 53 operations
- iRODS data management 597 operations
- Operations facility the remaining capabilities

Of the 597 operations that would be controlled by the data management system, 174 generic rules were identified. This number is substantially smaller because we were able to take advantage of the iRODS design that allows the specification of multiple rule sets or PODS. Thus many of the rules corresponded to the selection of the appropriate rule set (or template in NARA terminology), and the application of that rule set to the specified record or collection. The types of rule sets that were needed included:

- Parsing templates for extracting information from a file

- Formatting templates for creating reports or transformative migration
- Display templates for presentation of metadata, system performance, and failures
- Redaction templates for applying access restrictions on portions of records
- Notification templates for composing messages

Each of these templates required the use of multiple iRODS micro services for accessing metadata, accessing remote files, assemblage of the metadata and data into an appropriate form, and delivery of the result to the user.

An interesting evaluation is the extent that the ERA environment could be implemented using a workflow environment. While explicit services could be defined for each capability, the assertion that the services have been performed requires the management of state information, and the ability to control the operations performed upon each record. As soon as state information and guaranteed control of operations is implemented, you have created a data management system. Hence the iRODS environment is developing the correct set of capabilities for the generic management of distributed data. By building upon the concepts demonstrated and implemented in production in the Storage Resource Broker data grid, we are assured that iRODS will provide the correct set of generic services for building shared collections that are distributed across multiple sites, institutions, and storage systems.

9. Future Development

iRODS is an ongoing project that is implementing a new paradigm for cyberinfrastructure data management [36]. It is based on sound principles based on experiences with the Storage Resource Broker as well as through applications of theories and concepts from a wide range of computer science areas such as transactional databases, active databases, workflows, service-oriented architectures and program verification. The demand for rule-oriented data management occurs across multiple scientific disciplines, as well as within multiple data management communities. Groups that are willing to be first appliers of the technology include:

- Cognitive Science – management of Institutional Research Board approval forms, and sharing of access controlled data
- Orion oceanography cyberinfrastructure – management of real time sensor data, including distribution across multiple research facilities
- NARA research prototype persistent archive – demonstration of controlled federation of active, dim, and deep archives
- UK e-Science data grid – specification of collection dependent disposition and versioning options
- DSpace digital library – rights management and trusted digital repository assessment criteria

In order to support these applications, significant development is needed of the iRODS system. The continued research and development is estimated to take two more years. The components that are needed include:

- Completion of fundamental data management micro services and the development of corresponding rules
- Development of client APIs, command line interfaces and GUIs and web-based services
- Development of export mechanisms for iRODS rule sets and test mechanisms for consistent merging of rule sets,

- Queuing system that interconnects data management micro services with remote workflow execution systems such as Kepler as well as micro service schedulers for performing asynchronous, delayed and queued actions.
- Scalable rule engine validation.
- Development of testing techniques and theories for rule-based systems, administration toolkits for iRODS, and user manuals and help desk utilities,
- Demonstration of consistency validation at scale for rules and micro services.
- Demonstration of large-scale applications of iRODS technology.

An interesting development is an extension to the server-side workflow environment to support loops and conditional execution. A simple rule-based language has been implemented that allows quite sophisticated rules to be created. The new rule constructs include “if; elseif; else”, “while”, “for”, “foreach”, and “parallel”.

10. Software Distribution

The iRODS technology is available as open-source software from the wiki <http://irods.sdsc.edu/>. To promote the technology and extend the set of collaborators interested in joint development, a series of papers and presentations have been made. These include tutorials that will be conducted at conferences during 2007:

1. Moore, R., A. Rajasekar, “Tutorial on Rule-Oriented Data Management”, Supercomputing Conference, Reno, Nevada, November 2007.
2. Moore, R., W. Schroeder, “Tutorial on Distributed Data Management using the SRB and iRODS Data Grids”, Australian Partnership for Advanced Computing, Perth, Australia, October 2007.
3. Moore, R., A. Rajasekar, “Rule-Based Distributed Data Management”, tutorial proposal submitted to Grid 2007 conference, Austin, Texas, September, 2007.
4. Moore, R., “Large-scale Data Management”, white paper submitted to Archival Storage Life Cycle Management workshop at the Twenty-Fourth IEEE Conference on Mass Storage Systems and Technologies, San Diego, California, September 2007.
5. Moore, R., A. Rajasekar, M. Wan, and W. Schroeder, “Rule-Oriented Data Management”, tutorial proposal submitted to Teragrid Conference, Madison, Wisconsin, June 2007.
6. Moore, R., “Rule-based Preservation Systems”, white paper submitted to National Aeronautics and Space Administration workshop on Science Archives in the 21st Century, College Park, Maryland, April 2007.
7. Moore, R., A. Rajasekar, R. Marciano, “Implementing Trusted Digital Repositories”, DigCCurr2007 international symposium in digital curation, Chapel Hill, North Carolina, April 18, 2007.
8. Moore, R., “Managing Large Distributed Data Sets using the Storage Resource Broker”, ITEA Journal of Test and Evaluation, March 2007.
9. Moore, R., M. Smith, “Automated Validation of Trusted Digital Repository Assessment Criteria”, submitted to Journal of Digital Information, March 2007.
10. Moore, R., “Managing Large Distributed Data Sets Using the Storage Resource Broker”, submitted to The ITEA Journal of Test and Evaluation, January 2007.
11. Smith, M., R. Moore, “Digital Archive Policies and Trusted Digital Repositories”, proceedings of The 2nd International Digital Curation Conference: Digital Data Curation in Practice, November 2006, Glasgow, Scotland.
12. Moore, R., A. Rajasekar, M. Wan, W. Schroeder, R. Marciano, A. Jagatheesan, “On Building Trusted Digital Preservation Repositories,” 5th e-Science All Hands Meeting, Sept. 2006, Nottingham, UK.

13. Moore, R., "Building Preservation Environments with Data Grid Technology", American Archivist, vol. 69, no. 1, pp. 139-158, July 2006.
14. Moore, R., M. Smith, "Assessment of RLG Trusted Digital Repository Requirements," JCDL on "Digital Curation & Trusted Repositories: Seeking Success", June 2006, Chapel Hill, North Carolina.
15. Rajasekar, A., M. Wan, R. Moore, W. Schroeder, "A Prototype Rule-based Distributed Data Management System", HPDC workshop on "Next Generation Distributed Data Management", May 2006, Paris, France.

11. Acknowledgement:

This software development was supported by NSF ITR: (ASE+NHS) - (int+dmc): "Constraint-based Knowledge Systems for Grids, Digital Libraries, and Persistent Archives" award #0427196 and by the NSF SCI0438741 (NARA supplement).

12. References

1. iRODS-An integrated Rule Oriented Data System, <http://www.sdsc.edu/srb/future/index.php>
2. SRB-Storage Resource Broker, <http://www.sdsc.edu/srb>
3. Baru, C., R. Moore, A. Rajasekar, M. Wan, "The SDSC Storage Resource Broker," Proc. CASCON'98 Conference, Nov.30-Dec.3, 1998, Toronto, Canada, p. 5.
4. Audit Checklist for Certifying Digital Repositories, http://www.rlg.org/en/page.php?Page_ID=2076
5. DIGARCH: The Digital Archive Initiative, <http://gdc.ucsd.edu:8080/digarch/about-project/DIGARCH-Initiative/>
6. NSDL: National Science Digital Library. <http://www.nsdl.org>
7. DSpace: <http://www.dspace.org>
8. Fedora: <http://www.fedora.info>
9. PAWN: Producer Archive Workflow Network, <http://narawiki.umiaccs.umd.edu/twiki/bin/view/Main/PAWN>
10. Moore, R., A. Rajasekar, M. Wan, W. Schroeder, R. Marciano, "On Building Trusted Digital Preservation Repositories," 5th e-Science All Hands Meeting, Sept. 2006, Nottingham, UK.
11. Moore, R., "Building Preservation Environments with Data Grid Technology", American Archivist, vol. 69, no. 1, pp. 139-158, July 2006.
12. Moore, R., A. Rajasekar, M. Wan, "Data Grids, Digital Libraries and Persistent Archives: An Integrated Approach to Publishing, Sharing and Archiving Data", Special Issue of the Proceedings of the IEEE on Grid Computing, Vol. 93, No.3, pp. 578-588, March 2005.
13. Moore, R., "Building Preservation Environments with Data Grid Technology", American Archivist, vol. 69, no. 1, pp. 139-158, July 2006.
14. Moore, R., R. Marciano, "Technologies for Preservation", chapter 6 in "Managing Electronic Records", edited by Julie McLeod and Catherine Hare, Facet Publishing, UK, October 2005.
15. NARA Electronic Records Administration, <http://www.archives.gov/era/>
16. ERA capability requirements as of 8/11/2006, <http://www.crl.edu/content/DigArc/DigArc2/ERArequirements.xls>
17. Moore, R., A. Rajasekar, M. Wan, "Storage Resource Broker Global Data Grids", NASA / IEEE MSST2006, Fourteenth NASA Goddard / Twenty-third IEEE Conference on Mass Storage Systems and Technologies, April 2006.

18. Moore, R., M. Wan, A. Rajasekar, "Storage Resource Broker: Generic Software Infrastructure for Managing Globally Distributed Data", Proceedings of IEEE Conference on Globally Distributed Data, Sardinia, Italy, June 28, 2005.
19. Ocean Research Interactive Observatory Networks (ORION) <http://www.orionprogram.org/>
20. NEON: National Ecological Observatory Network, <http://www.neoninc.org/>
21. NVO: National Virtual Observatory, <http://www.us-vo.org/>
22. NOAO: National Optical Astronomy Observatory <http://www.noao.edu/>
23. SCEC: Southern California Earthquake Consortium, <http://www.scec.org/>
24. NEES: Network for Earthquake Engineering Simulation, <http://www.nees.org/>
25. SEEK: Science Environment for Ecological Knowledge <http://seek.ecoinformatics.org/>
26. UK eScience <http://www.rcuk.ac.uk/escience/>
27. WUNGrid: Worldwide Universities Network, <http://wungrid.org/>
28. PRAGMA: Pacific Rim Application and Grid Middleware Assembly, <http://www.pragma-grid.net/>
29. Australian Partnership for Advanced Computation, <http://nf.apac.edu.au/>
30. KEK High Energy Accelerator Research Organization, <http://www.kek.jp/intra-e/>
31. BaBar, <http://www-public.slac.stanford.edu/babar/>
32. Taiwan National Digital Archives Program,
http://www.sinica.edu.tw/~metadata/project/project-homepage/m1-3-AssociateCatalog_eng.html
33. Deutsch, A., Liying Sui, Victor Vianu, Dayou Zhou, "A System for Specification and Verification of Interactive, Data-driven Web Applications", SIGMOD 2006 Demo.
34. Deutsch, A., Liying Sui, Victor Vianu, Dayou Zhou, "Verification of Communicating Data-driven Web Services", PODS 2006.
35. Moore, R., M. Smith, "Assessment of RLG Trusted Digital Repository Requirements," JCDL workshop on "Digital Curation & Trusted Repositories: Seeking Success", June 2006, Chapel Hill, North Carolina.
36. Rajasekar, A., M. Wan, R. Moore, W. Schroeder, "A Prototype Rule-based Distributed Data Management System", HPDC workshop on "Next Generation Distributed Data Management", May 2006, Paris, France.