CRIS - Computational Research Infrastructure for Science

Invited Paper

Eduard C. Dragut[1#], Peter Baker[2#], Jia Xu[3#], Muhammad I Sarfraz[4*], Elisa Bertino[5#*], Amgad Madhkour[6*], Raghu Agarwal[7*], Ahmed Mahmood[8*], Sangchun Han[9*]
[#]*Cyber Center, Purdue University*
[*]*Computer Science Department, Purdue University*
[1]*edragut@purdue.edu,* [2]*pnbaker@purdue.edu,* [3]*xu222@purdue.edu,* [4]*msarfraz@purdue.edu,*
[5]*bertino@purdue.edu,* [6]*amgad@purdue.edu,* [7]*raghuagarwal@purdue.edu,*
[8]*amahmoo@purdue.edu,* [9]*han84@purdue.edu*

Abstract

The challenges facing the scientific community are common and real: conduct relevant and verifiable research in a rapidly changing collaborative landscape with an ever increasing scale of data. It has come to a point where research activities cannot scale at the rate required without improved cyberinfrastructure (CI). In this paper we describe CRIS (The Computational Research Infrastructure for Science), with its primary tenets to provide an easy to use, scalable, and collaborative scientific data management and workflow cyberinfrastructure for scientists lacking extensive computational expertise. Some of the key features of CRIS are: 1) semantic definition of scientific data using domain vocabularies; 2) embedded provenance for all levels of research activity (data, workflows, tools etc.); 3) easy integration of existing heterogeneous data and computational tools on local or remote computers; 4) automatic data quality monitoring for syntactic and domain standards; and 5) shareable yet secure access to research data, computational tools and equipment. CRIS currently has a community of users in Agronomy, Biochemistry, Bioinformatics and Healthcare Engineering at Purdue University (cris.cyber.purdue.edu)

Keywords: Scientific activity management; cyberinfrastructure (CI); provenance; workflow; data dictionaries; data sharing.

1. Introduction

Despite many advances in data management and workflow technologies, scientific cyberinfrastructures (CIs) currently in use are typically composed of a diverse set of rather ad-hoc components. As a result, these CIs suffer from several major shortcomings: (1) ad hoc management of research data – data is placed on shared drives and personal laptops, resulting in highly distributed and unmanaged collections of data; (2) limited data re-use as data is not adequately described; (3) lack of documentation of computational stages (e.g., data conversion among different file formats), resulting in untraceable modifications and thus limiting the exploration of unexpected research results; and (4) limited community interaction – data and computational tools are operated on isolated servers, inaccessible to the broader community. These issues are more prevalent in the so called "long tail of small science", because large numbers of small projects produces ever larger volumes of data. Yet they lack data repositories, community standards for data structures and metadata, and data management expertise [12, 16]. The unfortunate consequence of inadequate CI solutions is compromised research efficiency and integrity.

Significant work has already been done to assist in solving CI related problems. For example, workflow management systems such as Pegasus [19], Taverna [17] and Kepler [2] as well as scripting languages help improve the repeatability and control of experiments. National data repositories such as DataONE [2] and dbGap [15] provide storage locations for long term collaborative access. Data descriptors such as OWL [1] facilitate universal access to data and metadata. Specialized database engines such as MonetDB [14] and SciDB [7, 6] provide the ability to accelerate research. Online information management communities such as HUBZero (http://hubzero.org/) and PBWorks.com provide a framework for exchanging curated information. While data curation methodologies such as DataONE and

[1] http://www.w3.org/TR/owl-features/

Data Curation Profiles [21] provide expertise for long term storage. All of these provide pieces for solving the data and computational management puzzle.

The reality is that a typical scientific researcher does not have the computational expertise, funding, or time necessary to find, assess, and combine these independent building blocks into a cohesive CI to aid his/her research, data curation, and collaborative efforts. Without a suitable CI, their research can be inefficient, unable to easily build upon prior work, and provides limited extensibility by the broader community.

In this paper we describe CRIS (The Computational Research Infrastructure for Science), with its primary tenets to provide an easy to use, scalable, and collaborative scientific data management and workflow CI for scientists lacking extensive computational expertise. We have built and currently support CRIS for an initial user community at Purdue University in Agronomy, Biochemistry, Bioinformatics and Healthcare Engineering; however the infrastructure is designed to support applications from a much broader set of scientific domains. To support such varied communities, CRIS provides an extensible suite of tools to:

- describe data using domain specific vocabularies;
- support configurable and interactive workflows for seamless operations from the raw data through analysis and visualization;
- automatically validate the quality of scientific data;
- automatically capture, transform, and analyze data and metadata with associated provenance;
- facilitate cross-domain scientific collaboration;
- provide long term storage and access to organized and managed data, leading to efficient and verifiable research.

Our philosophy is that any CI must: (1) have low barriers to entry (i.e. easy to use and configure); (2) be inexpensive; (3) provide added value (i.e. incorporate the relevant scientific tools); and (4) be interoperable with other systems (i.e., seamless integration with other systems). **In CRIS we are not re-inventing the wheel, but developing an infrastructure in which existing and new techniques are integrated and customized, driven by the specific requirements of scientific research processes.** To the best of our knowledge, there is no other CI that has the same capabilities of CRIS. We give the details of CRIS in the rest of the paper.

The remainder of this paper is organized as follows. Section 2 illustrates the need for suitable CI with a concrete scenario of a scientific research process. Section 3 provides an overview of the CRIS architecture. Section 4 describes the tools used to implement CRIS. Section 5 depicts the motivating scenario in Section 2 when CRIS is used. Section 6 concludes the paper.

2. A Motivating Scenario: A Typical Small Science Research Process

Small science CI is traditionally comprised of shared network folders for data storage; limited access lab servers or laptops; isolated computational tools; under-documented file and data formats; FTP/email/sneakernet transmission of information; limited or non-existent system backups and revision control; and manual provenance collection. As a result, valuable information is often discarded or underutilized, and research efficiencies and integrity compromised. To explain the motivation behind CRIS, we will detail a typical small science research environment for a scientist Mark. This is a typical experimental setting for Mark, a biochemist, who is currently using CRIS. Mark is conducting an experiment to analyze the mineral uptake within the plant Arabidopsis, and correlating phenotypic variations with genetic information. Although his research is unique, his CI requirements and current processes are widely applicable to a broad range of scientific environments.

Mark first needs to capture metadata about the experiment, including information such as the seed lines, photoperiod, temperature, soil conditions, and growth chambers. Second, he runs physical samples of the plant through a mass spectrometer (an instrument which measures the masses and relative concentrations of atoms and molecules) in order to identify the chemical components in the sample. The equipment produces a data file in a proprietary format, which needs to be converted to a more universally acceptable format for his analysis software (e.g., from .raw format to .mzdata format). This requires a new program that Mark does not currently have, and therefore searches the internet to find a shareware version of software that accomplishes the conversion. However after finding, he still has to install on a Linux PC (he has limited experience with Linux), and get familiar with these formats to be able to carry out the implementation. As fourth step, the converted file needs to be processed by a protein search engine, to find specific proteins within the sample. Mark needs to implement an intricate piece of software that performs the following tasks: specifies the parameters, sends the parameters along with the file to a protein search engine, and remotely launches the protein search engine. Fifth, after the search is complete, Mark copies the results from the remote computer to a flash drive, transferring the results back and storing with the rest of the experimental data on an external hard drive in his lab. Finally, Mark has to input the file to a proprietary tool and analyze the results visually. Then this process is repeated hundreds of times,

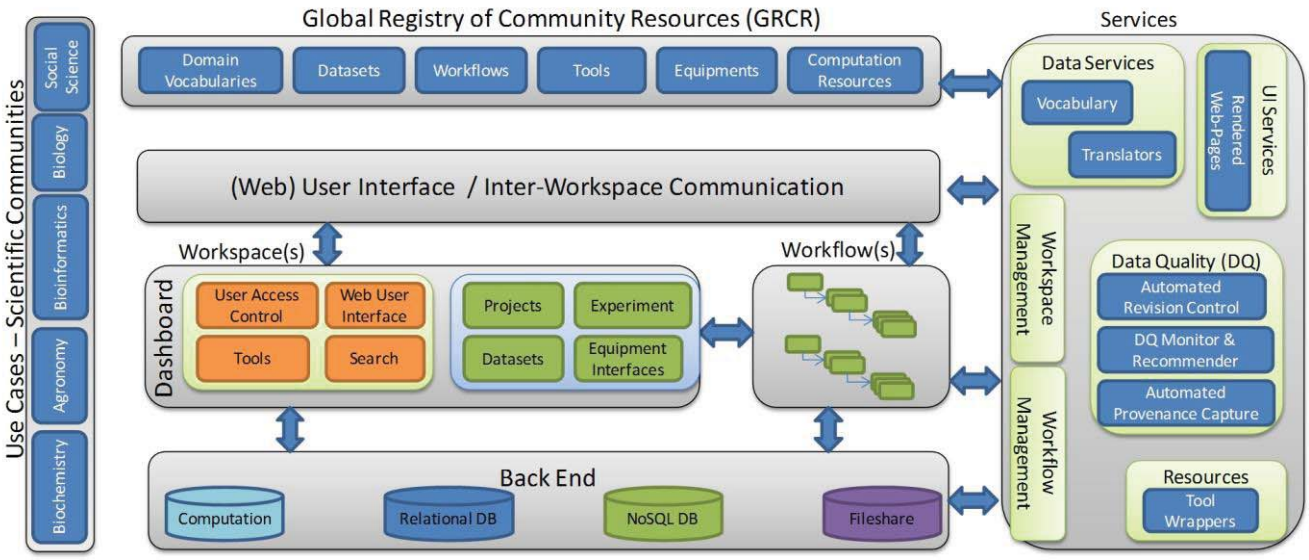

Figure 1: A high level overview of CRIS architecture.

with minor variations to the experiment setup. The data and experimental results must be kept private on his local computers until his findings can be analyzed, compiled and published. After which he needs some manner to make the original data available to the research community.

All these computer and engineering challenges sidetrack Mark from his main research activity: biochemistry. Even if Mark is successful in implementing all these steps he will still not have a complete solution, or a solution that his collaborators can leverage. Mark still needs to worry about automatic provenance collection (e.g., for experiment repeatability) [8], quality control (e.g., to avoid scientific fraud) [3], data loss (when his external hard drive fails). Furthermore, Mark lacks the infrastructure to publish his dataset or even the workflow to the broader community, resulting in others possibly repeating his work.

This scenario illustrates why it is critical to provide an easy to use and cost-effective CI which enables the seamless integration of data with existing computational tools, automates manual processes, provides broad yet secure access, and allows scientists to focus their efforts on understanding their research dom*ain in new and innovative ways.* We will revisit this scenario (Section 5) once we introduce CRIS.

3. CRIS Architecture

CRIS is a web-based application whose architecture is shown in Figure 1. We provide the details for its key components and implementation in this section.

3.1. Workspace

A workspace is the "face" of CRIS, and acts as a container for all activities and data to be managed for a single group of scientists. Each workspace includes a dashboard offering a simple view of the relevant information (Figure 2), and provides the tools and controls necessary to conduct research such as: project and experiment containers; configurable workflows; access controls; computational tools; visualization and reporting; and connectors to automatically exchange information with external systems.

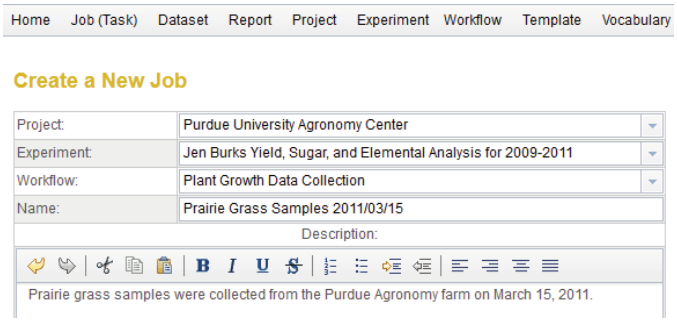

Figure 2: A fragment snapshot of CRIS's workspace

Workspaces are seamlessly connectable to diverse external data sources and computational tools. CRIS will soon support interconnected workspaces. This will provide an efficient and distributed architecture that can operate in an isolated manner, if desired, and instantly leverage other workspace resources when required. This is particularly beneficial for research projects with multiple people: e.g., projects that involve several institutions, or span diverse scientific domains. In addition to support access to remote data sources, CRIS will allow one to remotely execute computational tools within other

workspaces. Thus "owners" of computational tools can make their resources available to the broader community, without having to manage difficulties of cross platform compatibility.

3.2. Scientific Workflows

A key impediment for scientists is how to automate their manual scientific processes. To support these efforts, CRIS allows seamless definition of workflows. Workflows usually consist of several (independently developed) pieces of software, metadata capture, external resources, and datasets. However for a workflow to function properly, and to support broad reuse, the output produced in one step must be "compatible" with the input expected in the subsequent step. CRIS is built using a "common vocabulary" (Section 3.3.1) for the data interface.

Additionally, in order to make it possible to easily incorporate existing computational tools into the CI, CRIS provides "wrapping" support for tools (Services, Fig. 1). This basically helps the user to construct a definition for the inputs and outputs of a tool in terms of the common vocabulary of the project, and provides a standard mechanism to convert the data into the specific format required by the tool. In this way, the wrapper seamlessly ensures that the tool is not only usable within a particular workflow, but it can be reused in any other workflow in the scope of the project.

Notice that CRIS is not a dedicated workflow system, rather CRIS integrates workflows along with many other features (e.g., provenance, data versioning, secure access) required to ease the daily research activity of scientists and their groups. And in that light, external workflow engines (Pegasus, Kepler, etc.) can be added into CRIS as a workflow step, allowing complex data manipulation efforts to be seamlessly integrated in with other process and data management efforts.

3.3. Data Management

A basic premise of CRIS is to provide a CI with enough tangible benefits that scientists will want to use, with the net result being the efficient and organized management of their research data (for free). To accomplish this task, CRIS utilizes the following key concepts: (1) the definition of a **Domain Vocabulary** to support coordinated exchange and validation of information; (2) utilization of **Key/Value Pairs** for identification of information; (3) automatic **Rendering of Web-Pages** from template definitions; (4) integrated **Storage, Retrieval and Provenance**; and (5) **Reusable Resources**.

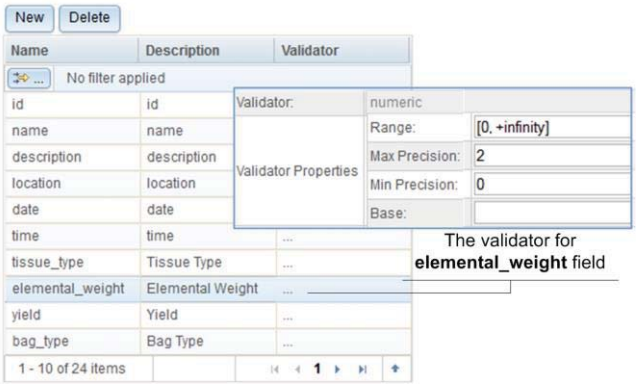

Figure 3: A Snapshot view of the vocabulary interface.

3.3.1. Domain Vocabulary. At the core of any effective exchange of information is a defined vocabulary. For example within the human perspective, the various languages throughout the world constitute defined vocabularies. They will change over time as a result of a variety of influences, but are the base reference for all interpretations. In a similar vein, CRIS provides a framework for the creation and evolution of domain vocabularies. They are best defined by a member of the research team who understands the scope of the project; Fig. 3 shows a fragment of the vocabulary from an agricultural project currently supported by CRIS.

Each domain vocabulary contains the unique definitions of relevant data elements, along with the definition of how to *validate* the information. To insure uniqueness across domains and throughout the workflows, we utilize a UUID as the primary reference identifier for each data element. For example, a definition of "Human Age" is as an integer with the range of [0-150]. However this must be kept distinct from the similar "Human Age" within a social science domain, which is instead defined as a list of ["Infant", "Child", "Adolescent", "Teenager", "Adult", "Senior"]. So understanding that these similar definitions will exist, we incorporate Translators.

Within each vocabulary data element, validators are defined allowing CRIS to automatically evaluate any data entered or ingested into the system. Validators define basic data types (integer, string, list, etc.) along with corresponding ranges. Additionally, regular expressions can be constructed and used to validator more complex data values (i.e. zipcodes, phone numbers, IP address, etc.). It is also possible to extend the internal validators to communicate with external data validation sources (e.g. verifying a term exists in a pre-defined dictionary at a web url). Finally, complex data quality measurement and detection can be implemented as a scriptable solution within a workflow step. For example, if it is necessary to detect when data values in an input stream start no longer change as expected (i.e. a rain gauge sensor freezes), then

a script can be written to check for this specific condition. When found, then the workflow step can notify the user to take corrective action.

Analogous to their human language counterparts, translators support the exchange of information between domain vocabularies. So in the above example, an integer age of [0-1] can be converted to "Infant", [13-19] can be converted to "Teenager", etc. (and also in reverse although a lossy translation). Importantly, CRIS records that the translation has occurred for future inspection. As vocabularies and translators are developed, they are made available to a broader audience through the GRCR repository (Section 3.4.2).

Future efforts will include a Guided Vocabulary Service which checks definitions created by a user against the GRCR repository of existing definitions. Whenever a definition provided by a user "matches" an existing definition, the latter is suggested to the user as an alternative definition. This is useful to avoid bloating the number of synonymous terms in the system, and reduce the number of required translators.

3.3.2. Key/Value Pairs. In the experiences with our initial user community, the set of information a researcher desires to capture is often completely different at the beginning of a project than at the end. So instead of fighting change and multiple updates to database schemas, we embraced change through simple definitions using key/value pairs. In this manner, the vocabulary element UUID is the key, and the value is the stored data. Thus, it becomes a simple task to add or remove desired information from the scientific workflow. We do however group information according to the project/ experiment/ job/ workflow step containers to support accurate retrieval of the desired information, and to provide some contextual information for individual workflow steps. The back end database is also designed to support such dynamic modifications to the captured data (see Section 3.3.6)

3.3.3. Rendered Web-Pages. To support the tenet of "an easy to use CI for scientists lacking in-depth computational expertise", CRIS automatically renders the HTML web-pages from underlying XML templates. So the researcher simply adds the desired vocabulary terms to a desired workflow step, CRIS then renders the necessary HTML, and the workflow engine validates the entered data using validation parameters specified in the vocabulary term. The net result is a minimal effort by the scientists to modify their CI requirements.

3.3.4. Storage, Retrieval, and Provenance. At each step in a workflow, in addition to the expected data files and metadata, as much provenance as possible is captured for long term storage and retrieval using the back-end storage services [20]: e.g., the specific version of a computational tool, the person entering the data, geographic location, etc. It additionally tracks revisions to the information, storing who updated specific data elements and why [9]. Then customary browse, search, and export functions are available to retrieve relevant information. This is an area for future extensions to CRIS.

3.3.5. Reusable Resources. As discussed earlier, it is important to provide resources which can be re-used by a broad set of groups. For example, Mass Spectrometry equipment is widely used by scientists. In CRIS, a basic equipment category resource is created that identifies the physical attributes (equipment owner, physical location, manufacturer, etc.) and data attributes (LAN network location, data format type and version, etc.). Once created, anyone can simply locate the resource within the CRIS GRCR (Figure 1), and pull it into his/her workflow. Hence when the Mass Spectrometry step is executed during a workflow, the data and metadata associated with the equipment are automatically retrieved and stored. This result is a fully abstracted view of the Mass Spectrometer, which allows the scientist to simply "use" the equipment while automatically gaining a rich set of experiment information. Currently, CRIS expects users to manually create proper data definitions. CRIS will incrementally be equipped with data definition assisting tools.

3.3.6. Back End. The back end services consist of a relational SQL database instance, a NoSQL database instance, a distributed file system infrastructure, and computational processing nodes. It should be noted that each described component can be deployed beyond a single server, therefore supporting isolation of data sets as required by an individual research (i.e., regulatory compliance issues), as well as scalability as the number of users increases. The SQL database is used to store data internal to CRIS (e.g., workspaces, tools, user accounts, etc.) [7]. The NoSQL database is used to store scientific datasets through the prior noted key/value pairs (Section 3.3.2). The distributed file system is used to store files of moderate size in their original form. And the computational nodes are used to run computational algorithms, translators, and any other specialized workflow steps to avoid significant loading on the web server. Future capabilities are expected to include a Hadoop cluster for storing large sized files in an efficient and scalable manner, and the appropriate hooks are already in place.

3.4. Services

We describe some of the key services of CRIS here.

3.4.1. Templates. The vocabulary is the basis for defining an element of data within a specific scientific domain, and becomes the method to validate as well as search and exchange information between domains. In the hierarchy of the system, templates are then assembled from a set of vocabulary terms, and workflows combine multiple templates. Templates are used to form a defined collection of vocabulary terms which have combined relevance within a workflow step. They inherit the definitions from each included vocabulary term, and provide a method to individually over-ride specific properties. HTML web pages can be auto-generated by CRIS from the template definition, thus providing a straightforward mechanism for a user to enter relevant information. Templates are typically designed for a specific user workflow, but can be shared with other users in a similar domain.

3.4.2. Global Registry of Community Resources (GRCR). The GRCR is CRIS's central repository for collecting and exchanging information about any first-class citizen of a CRIS workspace: domain vocabularies, datasets, tools, workflows, equipment, and computational resources. It is the component that will make it possible to easily share and search these objects across scientists and projects. GRCR will be governed by a Resource Access Control mechanism (Section 3.5) that ensures that resources are shared according to the levels and permissions prescribed by their owners [8].

3.4.3. Search. To realize collaborative opportunities beyond a single researcher's workspace, and to promote exploratory investigation, search capabilities are very necessary. Based upon our design criteria of stability, scalability, integration with NoSQL databases, and ability to extract contents directly from ingested files (Word, PDF, etc.), we have integrated the ElasticSearch[2] engine in CRIS prototype. Its schema free approach, built upon the Apache Lucene high performance search engine, is an ideal match to the data components within CRIS.

CRIS currently provides the following search and browse capabilities: (1) browse access via a hierarchical structure which closely matches the research process; and (2) keyword search based upon all stored information and vocabulary definitions. Future versions of CRIS will be equipped with more sophisticated search features, such as, search by datasets similarity, e.g., the user points out a dataset X and CRIS retrieves all the datsets similar to X.

3.5. Access and Security

The organization and sharing of large sets of heterogeneous scientific datasets pose non-trivial access control challenges. An inadequate or unreliable authorization mechanism can significantly increase the risk of unauthorized use of scientific data. This section highlights the security management issues that impact the design of an authorization model in CRIS.

3.5.1 Authorization Requirements and Model. We assume a general notion of authorization, by which an authorization is defined in terms of a subject, a permission, an object, an object owner and an object class. An inefficient way to implement an authorization mechanism is to explicitly store all authorizations for all system subjects and objects. In contrast, the concept of implicit authorizations makes it unnecessary to store all authorizations explicitly [18]. The main idea is that a permission of a certain type defined for a subject on a certain object implies other authorizations, which means that authorizations can be automatically propagated. Hence, the authorization mechanism can compute authorizations from a minimum set of explicitly stored authorizations in order to prevent unauthorized access.

The domain of subjects is organized in groups and authorizations are associated to groups, thus reducing the number of explicit individual authorizations. The idea of groups is similar to user-role assignment in Role Based Access Control (RBAC) [11]. The groups form a Group Hierarchy (GH) where a node in the hierarchy represents a group and a directed arc from group *A* to group *B* indicates that an authorization for group *A* subsume that of *B*. A permission in our model is stored as a cumulative permission represented by an integer bit mask where each bit represents a permission. Since only one entry is needed to store an authorization for a particular object, this reduces the need for implicit authorization along the domain *P* and hence implication between two authorizations does *not* occur along the domain *P*. The domain *O* of objects is organized as a rooted acyclic graph, in which each node is a Project, Experiment, Job or Workflow. An arc from node *A* to node *B* in the graph indicates that authorizations for object *A* imply authorizations for object *B*.

CRIS allows a user to develop a computational tool and then grant the execute permission on this tool to other users. A user having the authorization to execute a tool does not automatically have any authorization to directly read or modify the datasets accessed by the tool [4]. The user needs to possess the appropriate authorization on the datasets to execute the tool on them.

3.5.2 CRIS Access Control System Since CRIS is built using the Spring framework, we adopt the access control module of Spring Security[3] and customize it to meet the requirements of CRIS. The access control module of

[2] http://www.elasticsearch.org

[3] http://www.springsource.org/ spring-security

Spring Security provides comprehensive authorization services, is widely used in enterprise applications and is the de-facto standard for securing Spring-based applications.

CRIS has a set of explicit authorizations, called *authorization base* (AB). This consists of four tables provided by the default implementation of Spring Security as discussed below:

- *acl_sid* uniquely identifies any group in the system. Spring Security also provides support for group hierarchies and allows one to configure the containment relationship between groups.
- *acl_class* uniquely identifies any domain object class in the system.
- *acl_object_identity* stores information for each unique domain object along with its parent, owner and whether authorization entries inherit from any parent.
- *acl_entry* stores the individual permissions assigned to each principal or authority.

CRIS has an Authorization Module that gives users the ability to create and store authorizations in the authorization base for the various objects in the users' workspace and consequently allows access to authorized objects. If the authorization specified by the user is not already stored in AB or implied by an existing authorization in AB, the authorization is inserted into AB. In the case of tools, an additional check is done in order to ensure that grant and revoke authorizations on the dataset(s) associated with the tool are done properly.

3.6. User Interface

CRIS is intended to be routinely used by scientists lacking in-depth computer and system administration expertise. Its current user interface offers all basic functionalities and strives as much as possible to hide all the complexities of data management and computation. Our goal is to continually improve the user interface so that it will be very easy to use and consistent with the user's own mental model of their data, their activities, and the workflow as a whole.

3.7. Iterative development

CRIS is an ongoing project with a growing community of users. It is currently operational (available at: cris.cyber.purdue.edu) with the initial user community mentioned in Section 1, and several other components with partially implemented functionalities (e.g., the Resource Access Control of the GRCR, Guided Vocabulary Service, Hadoop Cluster, and Search Recommendations). We believe that an iterative development methodology delivers a robust CI, provides for immediate basic support for the scientists, and allows for comprehensive user feedback and improvements of the base system as new requirements are encountered. Additionally, it provides a unique proving ground for future research opportunities.

4. System Implementation

CRIS has been implemented using open source software and free Web APIs. The back end uses PostgreSQL database [4], MongoDB database [5], Hadoop HDFS [6] and Activiti workflow engine [7]. Activiti is used within CRIS as the workflow engine due to its ease of use in defining, changing, and sharing user workflows. CRIS is written in JAVA, with Spring and Hibernate as main frameworks.

The implementation of the front end of CRIS utilizes a number of open source APIs, mainly Spring MVC [8] and dojo javascript [9]. Most of the communication with the server is through Ajax to make the application more responsive.

5. CRIS in Action

Let's now revisit our scenario in Section 2 with Mark using CRIS for his research activity. With CRIS, Mark first defines the metadata about the experiment on a rendered Web page. Second, the mass spectrometer is "wrapped" in CRIS and Mark can run the physical sample from CRIS and CRIS will ensure that the outcome of the analysis is automatically imported in Mark's workspace. To convert the data files from .raw format to .mzdata format Mark can now use GRCR. He searches for a suitable program to see if anyone has shared such a program. Suppose such a program exists in GRCR, but it has to be executed remotely on the processing node where it is currently installed. To require such a remote execution, Mark only needs to drag the program into his workflow and CRIS takes care of transferring the file to the remote processing node, launching the program to convert the file, and transferring the converted file back to CRIS storage. Recall that in the fourth step, the converted file needed to be processed by a protein search engine. Mark can now add this step to his workflow, specifies the parameters, and CRIS does the rest: transfers the file to another remote processing node, sends the parameters,

[4] http://www.postgresql.org

[5] http://www.mongodb.org

[6] http://hadoop.apache.org/hdfs/

[7] http://activiti.org/

[8] http://www.springsource.org

[9] dojotoolkit.org/ framework

and launches the protein search engine. This part of the scenario shows how a computational node is abstracted and transparently invoked from within CRIS. Fifth, after the search is complete, CRIS automatically transfers the results back and stores them for long term archival. Mark can invoke a visualization tool in CRIS (assuming it was previously "wrapped") to explore the results. Finally, Mark can publish his data set and the workflow in GRCR to be used by the community.

If Mark is a user that belongs to the "Admin Group", then Mark has complete authorization on all objects during the course of an experiment and is not subjected to permission checking. If Mark is a normal user, then prior to the utilization of an object for an experiment, a permission check is done in order to ensure that Mark has the appropriate authorization on the requested object.

6. Conclusion

In this paper, we describe CRIS. CRIS currently provides support for 1) automatic capture, definition and processing of research data, 2) easy integration of existing data and computational tools on local or remote computers, 3) automatic data quality monitoring for syntactic and domain standards, and 4) secure access to research data, computational tools and equipment.

CRIS is an ongoing and long term project with a growing community of users at Purdue University. CRIS is continually being improved and extended with new components and functionalities such as Guided Vocabulary Service, Hadoop Cluster and Provenance at all levels.

We believe that CRIS is a step forward in the ongoing endeavor of the scientific community to build tools that allow scientific discovery through data exploration and community collaboration [13].

7. References

[1] S. Allard. "*DataONE: Facilitating eScience through Collaboration*". Journal of eScience Librarianship. 2012.

[2] [ALT04] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludäscher, S. Mock. "Kepler: An Extensible System for Design and Execution of Scientific Workflows". SSDBM, 2004.

[3] C. Batini, C. Cappiello, C. Francalanci, and Andrea Maurino. "*Methodologies for data quality assessment and improvement*". *ACM Comput. Surv.* 41, 3, 2009.

[4] E. Bertino. "Data Hiding and Security in Object-oriented Databases", ICDE, 1992.

[5] E. Bertino, G. Ghinita and A. Kamra. Access Control for Databases: Concepts and Systems. Foundations and Trends in Databases. 2011.

[6] G.P. Brown. "Overview of sciDB: large scale array storage, processing and analysis". SIGMOD. 2010.

[7] P. Cudre-Mauroux, et al. 2009. A Demonstration of SciDB: A Science-Oriented DBMS. VLDB.

[8] S. B. Davidson and J. Freire, "Provenance and scientific workflows: challenges and opportunities". SIGMOD. 2008.

[9] M. Eltabakh, W.G. Aref, et al. 2011. HandsOn DB: Managing Data Dependencies involving Human Actions, Purdue CS Tech. Report.

[10] M. Eltabakh, M. Ouzzani, and W. Aref, et al. 2008. Managing biological data using bdbms, ICDE.

[11] D. Ferraiolo, R. Sandhu, S. Gavrila, D. Kuhn & R. Chandramouli. "Proposed NIST Standard for Role Based Access Control", TISSEC, 4(3), 2001.

[12] B. Heidorn. "Shedding light on the dark data in the long tail of science". Library Trends. 2008. Vol. 57, No. 2.

[13] T. Hey, S. Tansley, and K. Tolle, editors. The Fourth Paradigm: Data-Intensive Scientic Discovery. Microsoft Research, 2009.

[14] M. Ivanova, N. Nes, R. Goncalves, M. Kersten. 2007. MonetDB/SQL meets SkyServer: the challenges of a scientific database. SSDBM Conference.

[15] [NCB07] M.D. Mailman, et al. "The NCBI dbGaP Database of Genotypes and Phenotypes", Nat Genet. 2007.

[16] T. Malik and I. T. Foster. "Addressing data access needs of the long-tail distribution of geoscientists". IGARSS. 2012.

[17] [OIN06] T. Oinn, et al. "Taverna: lessons in creating a workflow environment for the life sciences", Concurrency and Computation: Practice & Experience, 2006.

[18] F. Rabitti, E. Bertino, W. Kim & D. Woelk. "A Model of Authorization for next-generation Database Systems." TODS, 16(1), 1991.

[19] M. Sonntag, D. Karastoyanova and E. Deelman. "BPEL4Pegasus: Combining Business and Scientific Workflows", *ICSOC*, 2010.

[20] S. Sultana, M. Shehab and E, Bertino. Secure Provenance Transmission for Streaming Data. TKDE, (to appear).

[21] [WIT09] M. Witt., JR. Carlson, MR. Cragin, MR. and DS. Brandt, "Constructing Data Curation Profiles", International Journal of Digital Curation, 4(3), 2009.