Technical Report: NCSA Cyber-environments Division, ISDA01-2005, December 30, 2005

A Meta-Workflow Cyber-infrastructure System Designed for Environmental Observatories

Peter Bajcsy[1], Rob Kooper, Luigi Marini, Barbara Minsker and Jim Myers
National Center for Supercomputing Applications (NCSA)
University of Illinois at Urbana-Champaign (UIUC)

Abstract

The purpose of this white paper is to outline computer science issues related to the Task entitled "Create end-to-end meta-workflow demonstration" which is one part of the NCSA Environmental Cyber-Infrastructure Development (ECID) effort. Our goal is to research and develop meta-workflow architectures to support a set of environmental science and hydrology demonstrations in the short term and to support a spectrum of application communities in the long term. From the NCSA institutional view point, this white paper documents our design phase and provides an overview of meta-workflow definitions, previous work on workflows, a set of requirements, proposed meta-workflow architecture, and the current features of the prototype meta-workflow implementation called CyberIntegrator.

From the computer science view point, the paper presents the problem of designing a highly interactive scientific meta-workflow system that aims at building complex problem-solving environments from heterogeneous tools. Driven by systems-science use cases and complex informatics problems, we identify the dimensions along which current workflow technologies must grow to become a robust cyber-infrastructure capable of scaling to meet the national needs. Being able to join workflows developed using modules from the multiple open source and commercial workflow systems in use in various sub-disciplines is an obvious need. Less obvious but also critically important are abilities to describe and share workflow fragments, to execute portions of workflows on different appropriate hosts, or to provide security, provenance and fault-tolerance features of software execution. We introduce the term meta-workflow to refer to workflow systems designed to meet these end-to-end needs. We then discuss the architecture and implementation of a meta-workflow prototype called CyberIntegrator developed at NCSA. Our current meta-workflow architecture enables users (1) to browse registries of data, tools and computational resources, (2) to create meta-workflows by example or for batch processing, (3) to re-use and re-purpose meta-workflows, (4) to execute meta-workflows locally or remotely, and (5) to incorporate heterogeneous tools and link them transparently. The contribution of our work is (a) in defining the meta-workflow concept focused on science requirements and (b) in architecting technology and prototyping CyberIntegrator software supporting environmental observatories and other applications.

[1] Corresponding Author: Peter Bajcsy; email: pbbajcsy@ncsa.uiuc.edu

1 Introduction

Today the word “informatics” has become present in a multitude of application areas. Informatics refers to the increasing amounts of often highly complex data that have to be analyzed; information has to be interactively extracted from raw data and understood. In all application areas, scientists desire to learn from their data about a spectrum of complex phenomena surrounding us, and workflows and other software applications (including numerical and statistical models) have become their tools to support informatics, interactive data analysis and visualization, automation of large-scale data calibration and data reduction processes in Grids, etc. The challenges in any X-informatics system, where X stands, for instance, for bio, hydro, medical image or sensor [3], [4], are usually related to (a) data volume and computational requirements, (b) data, analysis and resource complexity management, and (c) the heterogeneity of information technologies supporting scientists. Our goal is to support X-informatics scientists to overcome these challenges by building a meta-workflow system.

The motivation for introducing the term scientific “meta-workflow” is driven by the need for information technology that could support scientists in their endeavor to solve highly complex problems with sophisticated tools running in heterogeneous computer environments. Problem complexity could be understood as a continuous variable that can reach infinity [12]. In a typical data-driven scientific exploration scenario, complexity of a problem refers to the complexity of (a) data sources and data organizations, (b) models developed from the data, (c) computer tools and hierarchical workflows to perform analyses, (d) computational resources involved in the analyses or (e) expertise necessary to comprehend and link all pieces of information. It is the scientific problem complexity and the broad spectrum of computer environments in the current scientific use that lead us to think about a higher layer of abstraction than a workflow, such as a meta-workflow. This could be illustrated in an environmental context with the example where Kepler software performs basic geo-spatial data retrieval and cleaning, ArcGIS integrates data, and D2KToolkit reports the results of data-mining of observations [40], [41]. Kepler has to be executed close to the data source due to size, ArcGIS has run on a server with a license, and D2KToolkit needs to run on a larger computer cluster due to processing requirements of data- mining.

The term meta-workflow has been introduced in the past in multiple contexts. First, it is viewed in the context of workflow aggregations or hierarchy of workflows [15] (a workflow is an aggregation of tasks, a meta-workflow is an aggregation of workflows or a hierarchy of workflows). Second, it is presented in the context of a process management, where large activities have to be integrated, executed and evaluated in a process of conducting electronic commerce [13],[14], [16]. It is apparent that the term meta-workflow has been used to refer to either flows of tasks or processes of activities that are more complex and include a set of autonomous flow/process entities. The formation and coordination of these complex processes led to introduction of other terms, such as workflow agents that would pose the properties of Atomicity, Consistency, Isolation, and Durability (ACIDity) [17], [18], [14]. Given these complex processes, we need to clearly extend the definition beyond a hierarchy of workflows by detailing

changes to the interface to support hierarchies versus low level modules, and to include the other problem dimensions.

We introduce a definition based on meta-workflow dimensions, and its functionality in these dimensions. The meta-workflow dimensions include (1) hierarchical structure and organization of software, (2) heterogeneity of software tools and computational resources, (3) usability of tool and workflow interfaces (e.g., workflow by example), (4) community sharing of fragments and publications, (5) user friendly security and provenance, (6) built-in fault-tolerance, etc. For example, the hierarchy dimension could be illustrated by widely applicable low-level modules (also called actors) in contrast to specialized workflow fragments. Furthermore, combinatorial explosion of module connection means that there are many more possible fragments than modules and hence there is a need for new mechanisms to organize fragments. One would also expect that fragments are produced by a larger set of people than modules so new description, publication, validation, and discovery mechanisms will be needed. Higher order workflows also have more scientific/discipline meaning. Thus, there is the potential (a) to organize them into categories related to data ingestion, cleaning, aggregation, etc. and (b) to adjust user interfaces to build scientific processing pipelines with the meaningful fragments defined by a community rather than with low level modules that lead to completely unstructured processing graphs (e.g., by using semantics to organize modules [8]). The heterogeneity dimension could be demonstrated by the number of different engines and software applications used by people for a reason, for instance, because of user friendliness, complementing functionality of modules, interactive versus batch model execution, execution scalability, ability to disconnect from running processes, etc.

Given multiple meta-workflow dimensions, meta-workflow functionality is defined by the fact that it has to provide (1) a mechanism for defining hierarchies of structures, (2) ways to integrate execution across tools and heterogeneous environments, (3) user-friendly interfaces, (4) single sign-ons across heterogeneous engines and distributed systems, (5) fault-tolerance (handle error conditions) across a meta-workflow and eventually optimization of service selection, etc. In this context, the tools are understood as software applications, workflows, web services, communication mechanism with instruments, etc, and it is expected that the tools would likely perform very sophisticated tasks. Heterogeneous environments refer to multiple application executors, workflow engines, operating system platforms, or hardware components. We derived the name meta-workflow from the term workflow since it represents the notion of structured work procedure (www.wikipedia.org) and coordinated execution of multiple activities [10] and includes a number of humans, databases, and specialized applications [18]. The prefix ‘meta’ indicates the fact that a single scientific tool or a workflow would not be sufficient and friendly enough to enable scientists to solve highly complex problems with sophisticated tools needed for the analyses and running in heterogeneous environments. As the definition of the word ‘meta’ suggests, our definition of a meta-workflow refers to a workflow of more highly organized or specialized form.

While there is an abundance of meta-workflow applications, the meta-workflow design is viewed as a computer science problem that creates a cyber-infrastructure component

supporting scientists trying to solve complex X-informatics problems. In order to demonstrate our meta-workflow prototype, we selected environmental, hydrological and medical applications. The application drivers pose a series of scientific questions about coastal management (hypoxia, elevated zinc levels in oysters, eutrophication, i.e. increased levels of nutrients such as nitrogen and phosphorus), river basin and intensively managed landscape management (extensive modification of the land for agriculture and urban use), and medical structures relevant to diagnoses and discoveries (in vitro and in vivo 3D tissue reconstruction). The investigations of the above scientific questions call for providing an interactive style software environment where analyses, including access and utilization of data, computational tools and hardware resources, should be enabled transparently so that scientists can gain phenomenon understanding while addressing their problems at the high abstract level. In our work, we derived the requirements for meta-workflow design by exploring our specific application drivers and the common requirements found in other published work.

Given the meta-workflow definition and a set of requirements, our work focuses on the architecture of a meta-workflow. The architecture is comprised of meta-workflow core and extension system components that communicate with global registries of available data, tools and computational resources. Due to the modular architecture, it is possible to substitute meta-workflow components in the future if anyone desires to do so. We present the modular meta-workflow architecture, where scientists can register their favorite data, tools or computational resources in the prototype meta-workflow registries to leverage all available resources using the developed meta-workflow system. With these community registries of heterogeneous data, tools and computational resources, we currently enable (1) meta-workflow creation by example (select and execute each processing step), (2) meta-workflow formation for batch execution (create a sequence of processing steps first and then execute them), (3) meta-workflow representation for re-using and re-purposing, (4) data structure conversions across heterogeneous tools based on syntax mapping, and (5) local or remote meta-workflow execution using heterogeneous executors and services already installed on multiple known computers. Our current prototype does not address semantic conversions (e.g., multiple names for the same physical variable) and operates under the underlying assumption of moving data to available computational resources rather than moving codes corresponding to tools to large data repositories. The reasoning behind these assumptions is explained later in the document.

This paper is organized as follows. Section 2 describes related work. Section 3 presents the meta-workflow requirements and introduces meta-workflow as one part of NCSA Cyber-environments in Section 4. We outline the meta-workflow architecture called CyberIntegrator in Section 5 and demonstrate key capabilities of our preliminary prototype implementation in Section 6.

2 Related Work

In general, meta-workflows and workflows are similar in their multi-tier structure, and in their structure (both contain three basic building blocks, such as editor, representation and

engine). The main dissimilarities could be described as follows. Workflows are composed of software components (called modules or actors) that would be subsets of meta-workflow components (tools) in terms of their semantic meaning and software complexity. Workflows provide a mechanism to develop and connect modules that comply with specific application programming interface (API) followed by developers. Meta-workflows that support multiple engines provide a user friendly registry-based mechanism to add tools that could be executed by the engines, and then connect the tools by application scientists. A workflow execution is typically performed using a homogeneous environment that consists of a single workflow engine. A meta-workflow execution is performed using heterogeneous environments that consist of multiple executors. From a user's perspective, meta-workflows operate at a coarser level of software functionality than workflows, transitions between heterogeneous environments are hidden, and incompatible data syntax is guided by user-friendly interfaces and available data converters. Currently, a user still has to address incompatible data semantics. In summary, meta-workflows could be discriminated from workflows by their semantic abstraction of tools, their flow graphs corresponding to higher conceptual levels of problem solutions, the scalability for investigating large ensembles and event-triggered runs, the simplicity of human computer interfaces and interactions, and the scale of community users and contributors.

We could relate other aspects of workflows to the concept of meta-workflows, such as the aspect of the balance between job characteristics and optimal execution. For example, jobs could be described as compute-intensive, data-intensive, analysis-intensive, coordination-intensive or visualization-intensive [8]. The optimal execution would include mechanisms for finding geographical proximity and temporal availability of computational resources, data sources and domain expertise. The meta-workflow design does not assume any specific type of job characteristics since the types will be depend on each community use. The optimal execution currently relies on each already implemented execution of a tool. Nevertheless, the meta-workflow executor takes a list of hardware resources where any particular tool can be executed and tries to execute the tool in the order provided by the un-sorted list in the case of failure. The list can be sorted based on user and community preferences in the future.

In the remainder of this section, we compare the concept of meta-workflows with the existing workflows first, then classify our meta-workflow based on four workflow paradigms and illustrate complexity aspects of meta-workflow.

2.1 Comparison with Existing Workflows

We start with definitions of a workflow since the word meta-workflow contains the word "workflow". According to www.wikipedia.org, "a workflow is the operational aspect of a work procedure: how tasks are structured, who performs them, what their relative order is, how they are synchronized, how information flows to support the tasks and how tasks are being tracked". According to the scientific literature [10], "Workflows are activities involving the coordinated execution of multiple tasks performed by different processing

entities", where the processing entities can be people or computer programs. The past workflow environment efforts have led to several workflow prototypes that should be compared with the described meta-workflow. We compare those workflows from the long list of possible candidates that we are familiar with, such as Kepler [2], [8], D2K [23], D2KSL, OGRE, Ensemble Broker [34], and ArcGIS ModelBuilder [35]. Comparisons with other workflows might be performed in a similar way, for instance, with SciFlo [28],[29], DAGMan [32], CCA [33] or Taverna [31].

The discrimination between the meta-workflow presented in this work and any single engine workflow, lies clearly in the number of execution engines used and workflow-to-workflow translations. There are numerous efforts to allow module-reuse across engines including one to link D2K [23] and Kepler [2], [8] occurring within NLADR [38]. These efforts start to address issues of matching the syntax of data structures and error reporting. Our meta-workflow effort in comparison with other efforts of module-reuse across workflow engines could be described as the development of a framework for (a) consistent inclusion of many workflow engines, (b) automatic syntactic data structure translations between heterogeneous workflows, (c) user friendly inclusion of new tools and creation of workflows, and (d) formation of workflows from tools with higher semantic meaning than the meaning of components in workflows.

The discrimination between the meta-workflow and linear workflow environments is in the capability of meta-workflows to form direct acyclic graphs (DAGs) of workflows as opposed to just linear graphs of workflows. For example, D2KSL is one effort that has tried to address the need for a simpler interface for end users constructing large-scale end-to-end processes. It incorporates some workflow-by-example (task graph is in a separate pane and not in the main user interface), but limits itself to a linear task model.

In the Grid community [5], several workflow environments have been developed to support high performance computing aspects of scientific analyses. The focus of these efforts is primarily on the execution of workflows, for example, OGRE and Ensemble Broker developed for the NSF ITR LEAD project [34]. The work is currently focused on large scale executions (thousands of runs in ensembles) on the grid of computers but otherwise the effort shares the goals of our work. Our meta-workflow currently focuses more on the simplicity of end user interactions with information technologies while utilizing all execution mechanisms transparently. In our meta-workflow approach we concentrate on simplifications of all human interaction interfaces with technology components and attempt to hide all software setup, representation and execution intricacies from the end users in order for scientists to solve complex domain scientific problems at a high level of abstraction. We expect continuing interaction between the development teams and sharing of design concepts and technologies so that the meta-workflow efforts can merge in the future (e.g., the meta-workflow engine of Ensemble broker and the meta-workflow editor and information browsers from our effort could form the next generation meta-workflow framework).

With the increased maturity of web services, workflow environments could also be built by wrapping all existing functionality into web services and orchestrating workflow

executions with web service layers to tools. The objective is to wrap fragments of code (modules) in web services and integrate them module-by-module. This approach is appealing mainly to the business community and commercial applications, e.g., ArcGIS ModelBuilder [35] as an example of geospatial data analysis provider. The past several efforts led to business process workflow architectures, such as FlowMark (later became IBM Web Sphere), WSFL (later merged with XLANG) and BPEL. In parallel, there were efforts to develop scientific workflows from scratch since the inclusion of grid computing that has not been aligned with web services requires new development [1]. The list of scientific workflow architectures includes DAGMan, Taverna, GridFlow and Grid Ant, Triana and GSFL, just to mention a few. The most adapted standard for web services has become the Business Process Execution Language (BPEL)[36]. The BPEL development focus has been so far primarily on the executors. The existing BPEL editors for forming BPEL-based workflows are built for developers rather than for scientists. Emmerich et al [1] asked a set of questions about BPEL suitability for scientific workflows with some positive answers.

Various projects and efforts in the past have chosen the direction towards web and grid services. However, these projects did not necessarily address (a) exploring the fragment structure hidden within web services (the hierarchical structure and organization of software and semantics), or (b) interacting and modifying the fragment run by the services. The former one was addressed in the Taverna project [31], where the effort has made progress on describing services semantically and allowing researchers to describe their processes at a higher level. Thus, researchers could focus on their scientific process rather than on the selection of particular modules/services. The latter one was addressed in a research prototype of 3D medical volume reconstruction using D2K web services [21], [22]. Nonetheless, based on the current web service technology state, the web service approach to meta-workflows currently poses difficulties when it comes to incorporating friendly user interactions, scalability for large data sets or fault-tolerance. Given the nature of scientific interactive explorations and discovery processes that define meta-workflow requirements, we decided to build our meta-workflow architecture by supporting inclusion of web services but not by demanding code fragments to be web services.

2.2 Comparison with Current Workflow Paradigms

In general, one could follow four types of workflow paradigms when designing a new meta-workflow system. These four workflow paradigms refer to the fact that depending on what is the “flowing” element within a workflow environment workflows could be denoted as data-flows, code-flows, control-flows and hardware-flows. The flowing element could be (1) data to hardware and software resources for data-flows (e.g., 3D volume reconstruction needing computational resources and sophisticated registration software), (2) software code to data locations and hardware resources for code-flows (e.g., data mining of large databases of water quality data), (3) control commands to cooperatively use generated data with distributed software and hardware resources for control-flow (e.g., booking a hotel and a rental car after airline ticket reservation) , and

(4) hardware to data locations and software resources for hardware-flows (e.g., spatial re-configuration of 'smart' wireless sensors to confirm indoor hazards).

Data-flows are primarily used in the scientific community (e.g., e-Science, global grid forum) while the control-flows are popular in the business community (e.g., BPEL2WS, IBM's WSFL, Microsoft's XLANG). Code-flows pose challenges on prior installations of software environments where any code could be executed and hence have not been used too frequently. The presentation of the hardware-flow concept is novel in this paper since the 'smart' wireless sensor technology has not reached the maturity to address the hardware-flow execution. Given the workflow paradigm classifications, our meta-workflow can be viewed as a combination of data-flow and control-flow as it will be explained in the next section.

2.3 Complexity Comparisons

The concept of meta-workflows can also be illustrated in terms of software complexity (see Table 1). The columns of Table 1 represent granularity of software complexity. The rows correspond to characteristics of software relevant to workflow environments. Each cell in the table gives a very short description of how characteristics are supported at a given complexity level.

From the software complexity view point, a meta-workflow application programming interface (API) is not more complex than a program function API. However, meta-workflow representation and execution deal not only with a module API that addresses data going in/out and error/security mechanisms similar to every workflow but also with engine type and registry information in the API, coupled with mechanisms to translate/integrate the syntax of data flow between engine types and to move data.

Table 1: Illustration of increasing software complexity. The abbreviations WF and MWF denote workflow and meta-workflow respectively. API stands for application programming interface.

Characteristics\ Complexity	Program Function	Software Application	Workflow	Meta-Workflow
API	Function name and arguments	Class and function list	WF composition and module list	MWF composition and tool list
Execution	Single programming language executor	Multiple programming language executors	Single WF composition executor	MWF composition executor
Error recovery and check pointing	N/A	Between function calls	Between modules	Between tools
Security (communication	Execution	Execution and communication	Execution and communicatio	Execution and communicatio

and execution)		between function calls	n between modules	n between tools
Documentation	e.g., Java doc	Get Application info	Get WF info	Get MWF info
Discovery of functionality and API	N/A	Reflection	Module Hierarchy	Registries of tools, data and resources

We also present the concept of meta-workflows in terms of the level of problem abstraction (see Table 2). From this view point, Table 2 illustrates three application driven examples. Let us suppose that one would like to solve a set of complex Maxwell's equations given some boundary conditions. The software and problem solution have to contain the implementations of a program function "addition". The addition function is one part of a calculator performing any algebraic operation. The calculator becomes a part of a workflow that could partially solve one Maxwell's equation. In order to solve all Maxwell's equations simultaneously, the workflow becomes one part of the composition of multiple workflows that is denoted as a meta-workflow. Similarly, one could describe the other examples in Table 2 about understanding causes and consequences of hydrologic variables, and predicting bacterial loadings to coastal waterbodies.

In these examples, a meta-workflow solution has to deal with discovery and organization of modules that would be not only under the control of a small group similar to workflow solutions but also under the research and development of communities and third parties. Meta-workflow also has to incorporate changes represented and managed using registries of third party fragments, and aggregate across registries that might appear for particular engines, or include special security requirements.

Table 2: Illustrations of the application driven meta-workflows

Example\Complexity	Program Function	Software Application	Workflow	Meta-Workflow
Solve Maxwell's equations to perform radar electro-magnetics prediction of a vehicle	Operation plus	Calculator for performing (A operand B)	Complicated formula requiring multiple calculators	Multiple complicated formulas that have to be solved simultaneously
Understanding causes and consequences of hydrologic variables	Geographic coordinate projection	Spatial integration of raster data	Spatial, temporal and spectral integration of raster and vector data	Prediction modeling and spatial analysis of integrated data sets
Predicting bacterial loadings to coastal waterbodies	Delineate watershed	Calculate point and non-point bacterial	Schematic processor model of bacterial	Optimization of processor model parameters

		loading	loading	

Finally, the concept of meta-workflows is illustrated in terms of complexity of user interactions. User interactions include user inputs related to a domain-specific problem, and interfaces to formation and execution of a sequence of processing steps. If we consider the problem of understanding causes and consequences of hydrologic variables then user interactions could be described by a flow of domain-specific user inputs (see Figure 1) and by interfaces to formation and execution of a linked graph of tools (see Figure 2). Meta-workflow has to provide low-complexity user-friendly interfaces that would be understandable by scientists from multiple communities. The meta-workflow interfaces has to address not only linking of tools (e.g., drag and drop followed by connecting inputs and outputs of tools) similar to workflow solutions but also better searching, filtering and sorting capabilities to find data, tools and computational resources, as well as improved capabilities for creations of meta-workflows by example and for batch processing with a rich set of personalized settings and enabled system feedback options..

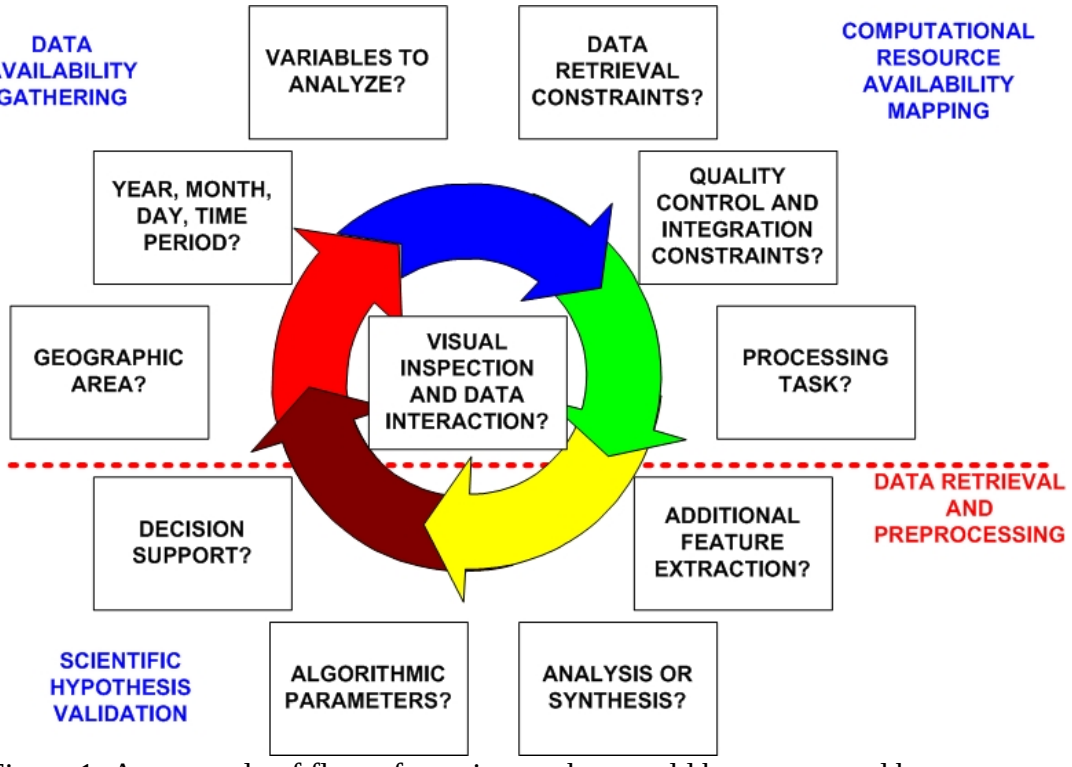

Figure 1: An example of flow of user inputs that would be represented by meta-workflows.

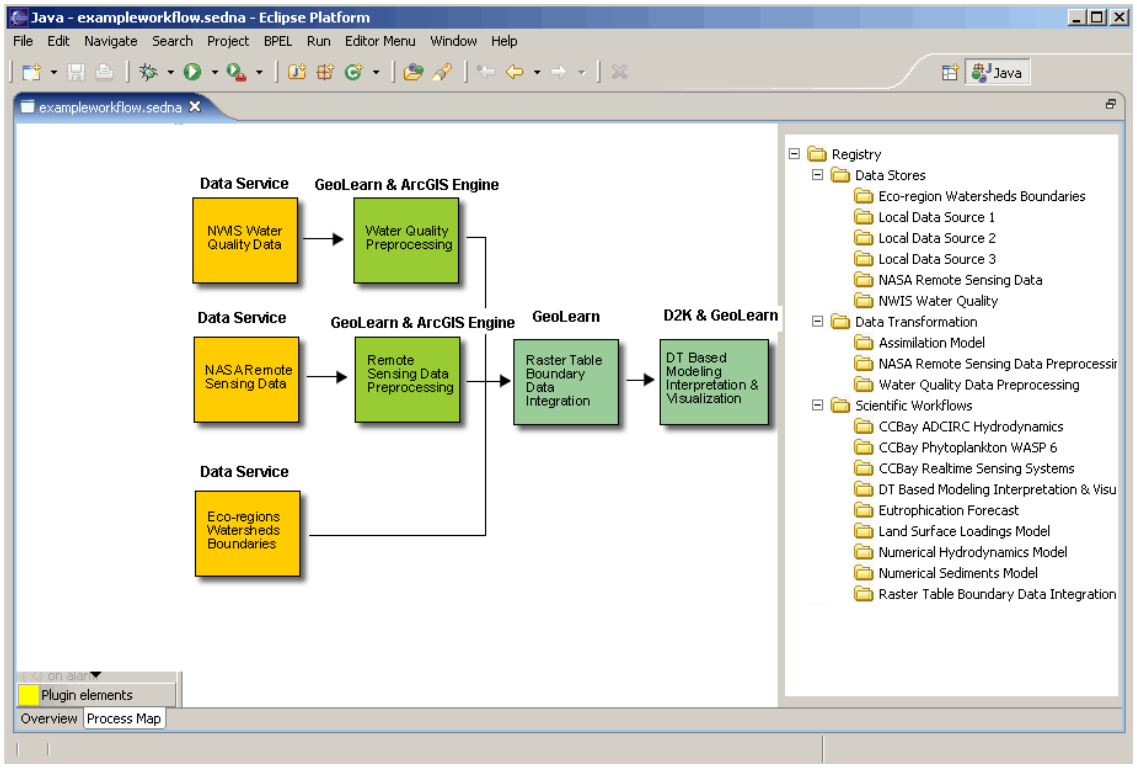

Figure 2: A mockup example of a composition of a connected graph of semantically high level tools. The example shows tool descriptions (inside of each box) and heterogeneous engines (above each box). The tools can be selected from a registry organized in this example into 'Data Stores', 'Data Transformations' and 'Scientific Workflows'.

3 Meta-Workflow Requirements

In this section, our goal is to go from a meta-workflow concept to a set of general requirements and then to a set of minimal requirements for an implementation. Our goal is to overview past workflow project requirements to identify a common set of general requirements, investigate specific requirements defined by environmental, hydrological and medical applications and then arrive to a set of minimal requirements for a meta-workflow implementation.

3.1 Published Workflow Requirements

Several publications have been published that list requirements of scientific or grid workflows. We list three sets of past workflow requirements. First, according to [2], the list of requirements for grid workflows includes: (1) authentication, (2) data movement, (3) remote service execution, (4) grid job submission, (5) job scheduling and resource management, (6) fault tolerance, (7) logging and provenance, and (8) user interaction and reporting. Second, according to [8], the requirement list includes: (1) seamless access to resources and services, (2) service composition & reuse and workflow, (3) scalability, (4)

detached execution, (5) reliability and fault-tolerance, (6) user interaction, (7) “smart” re-runs, (8) “smart” (semantic) links, (9) data provenance. Third, according to [1], the requirement list for scientific workflow management includes: (1) defining scientific workflows, such as (a) basic activities (orchestration of grid services, such as control and data flow between basic activities, invocation of grid services, synchronization, assembly of the message content for grid services, extraction of results from grid services responses and raising faults), (b) control flow and data flow, (c) hierarchical composition, (d) failure handling, (2) deployment of scientific workflows, such as (a) testing grid services, (b) deployment, (3) enactment of scientific workflows, such as (a) concurrency, (b) scalability, (c) monitoring.

3.2 Meta-Workflow Requirements

The specific requirements from our specific meta-workflow application drivers can be summarized as follows: (1) highly interactive scientific workflow system, (2) problem-solving environment, (3) integrative system of multiple workflows, (4) environment for re-use and re-purposing of existing software, and (5) system including data and workflow provenance (provenance indicates the series of interconnected experimental and processing steps that have produced the data). These requirements can be related to the common workflow requirements by an approximate consolidation of terminology.

Our current set of minimal requirements for a meta-workflow implementation is based on combining the past workflow requirements and prioritizing them according to our specific meta-workflow application drivers. Thus, the consolidated requirements could be described as follows: (1) create an editor for user friendly meta-workflow composition, (2) integrate multiple workflow engines, (3) support re-using and re-purposing meta-workflows, (4) enable run-time user interaction and a batch mode execution, (5) provide interoperable access to data and tools (web services), (6) provide registry of tools, (7) support provenance, (8) use standards and standard technologies (XML, web services), (9) incorporate error handling and check pointing, (10) incorporate security, and (11) incorporate grid computing.

While other requirements, like smart reruns, might set directions where workflow development should go, we have concentrated on defining minimal requirements for a state-of-the-art meta-workflow implementation. The ranked ordered list of requirements above represents our directions for including functionalities in our NCSA meta-workflow implementation.

We foresee additional requirements coming from other types of high-level science processes, where meta-workflows might be more appropriate than workflows. For instance, Monte Carlo simulation or optimization tasks could serve as examples, where one workflow/tool needs to be wrapped around another or multiple tools form a closed loop (output of one tools is fed to input of another tool). While we have not included support of wrappers and loops in our list of specific requirements, we plan on learning the needs of communities in the future to support hierarchical structures of meta-workflows

(wrappers of meta-workflow can form workflows) and looping structures of meta-workflows (closed loops of interconnected tools), as well as classification rules of communities to define a semantically meaningful tool.

4 NCSA Meta-Workflow as Part of Cyber-Environments

The core of NCSA's efforts over the next several years will be to provide scientific and engineering communities with an integrated set of tools and services *Cyber-environments* that allow these communities to realize the full potential of the national cyber-infrastructure in their research, development, and teaching activities (see Appendix A). Cyber-environments will provide an interface to local and shared instruments and sensor networks, data stores, computational capabilities, and analysis and visualization services within a secure framework enabling management of complex projects, automation of processes, and collaboration with colleagues both near and far.

While cyber-environments are often currently thought of as gateways to large-scale computational capabilities and community data stores, or as collaboration spaces, this definition will need to be extended in a number of directions to adequately match the needs of scientists and engineers over the next 5 years. As researchers attempt to understand more complex phenomena, e.g. ones involving very high-dimensional phenomena and networks of interacting processes, and attempt to apply their research to the solution of societal concerns, cyber-environments must support them in managing larger-scale and more complex scientific projects and processes. Cyber-environments must support users in managing the more diverse and larger-scale experimental, computational, and data resources required to characterize and model complex phenomena. Cyber-environments must bridge local, institutional, and national CI to create a seamless environment that assures the most powerful and effective techniques are always brought to bear and that enables researchers to quickly scale their techniques from local proof-of-concept to full-scale analyses. Cyber-environments must assist in the bi-directional connection between raw/group research artifacts (data, notes, plans, etc.) and published artifacts (vetted data, annotations, best practices, reviews, and papers) to enhance the flow of information between basic research and application. Similar mechanisms will be required to close the gap between textbook science and current research and engineering understanding and practice, thereby enhancing our ability to train the next generation of researchers and inform the general public.

To support these capabilities, the NCSA meta-workflow effort aims at providing users with higher-level abstractions, above the currently available cyber-infrastructure, and additional capabilities for automating or semi-automating processes. Additional tools for meta-workflow will be needed for, for example, integrating many types of data, planning optimal experiments, and monitoring, sharing, and validating, research protocols. The concept of Visual Knowledge Discovery, using data analytics to categorize, cluster, and extract features from large data sources coupled with interactive visualization to allow users to quickly digest data and build understanding, is a prime example of this. Similarly, capabilities to manage semantic information about data and resources will be a

general enabler of higher-level capabilities such as provenance tracking, annotation, and collaborative data curation capabilities, as well as recommender systems using social network analysis and network analysis of data quality. These capabilities have been incorporated into the NCSA meta-workflow design.

Additionally, cyber-environment design and development methodologies need to change to support large scale deployment and to significantly reduce the costs of coordination involved in creating, adapting, and evolving cyber-environments. Cyber-environments must transition from being thought of as the product of one-time development projects to being considered as living infrastructure that will evolve with technology and with the scientific and engineering discoveries and understanding over decades. Towards this end, NCSA must build cyber-environments on the principles of sustainability and adaptability using current and emerging techniques such as web and grid services, translating/integrating middleware (e.g. MyProxy), global unique identifiers and metadata, workflow, meta-workflow and provenance, and semantic descriptions of resources and data. These types of technologies lower the architectural coupling of cyber-infrastructure and cyber-environment components while maintaining end-to-end capabilities.

5 Architecture of NCSA Meta-workflow Prototype Called CyberIntegrator

The objective of the NCSA meta-workflow prototype called CyberIntegrator is to create a quick practical mechanism for (a) integrating software tools across a particular set of engines, (b) investigating ideas to simplify the interface and (c) providing initial integration into community cyber-environments. The goal of the NCSA meta-workflow effort is not to design a next generation engine but rather to enable an easy use of many existing heterogeneous tools in several scientific communities. Furthermore, our effort is not a multi-year fundamental basic research about optimal workflow mechanisms but rather an applied computer science research that is guided by deep theoretical understanding, and driven by providing experimental features on time to our initial communities.

In this section, we describe the design logic behind our meta-workflow architecture that is rooted in our theoretical analysis of the problem. We start with a system view of meta-workflow since it helps us design the architecture. With this view, an ideal meta-workflow could be viewed as a system for (a) browsing and searching available data, tools and computational resources, (b) accessing available data sets, tools and computational resources, (c) bringing them together using one of the workflow paradigms; (d) executing any tool or a sequence of tools, (e) monitoring and controlling executions and (f) efficiently utilizing available data, tools and computational resources. This system view allows us to describe issues related to meta-workflow architecture driven by multiple requirements (like in the Pegasus work [39]), and the key capabilities implemented in CyberIntegrator. When selecting the key capabilities for CyberIntegrator,

we looked for a reasonable functionality set that would fit into the overall web portal and grid cyber-environments effort at NCSA as described in the previous section.

First, we explain the design of our meta-workflow system architecture called CyberIntegrator based on its browsing, and data-flow and control-flow characteristics since the consideration of these characteristics and the application-driven meta-workflow requirements has led us to the meta-workflow building blocks. The browsing characteristic refers to the fact that a user has to browse registries of data, tools and computational resources to form meta-workflows. The data-flow and control-flow characteristics define what parts of the system are dynamic (data and control commands) and static (code and computational resources). Thus, the building blocks of meta-workflow architecture are described based on browsing, and data-flow and control-flow characteristics in Sections 5.1 and 5.2. Second, we address the roles and functionalities of meta-workflow building blocks in Section 5.3

5.1 Meta-Workflow Architecture Including Browsers of Distributed Registries

To build a meta-workflow architecture supporting characteristics of an ideal system, one has to establish certain access and retrieval mechanisms, management practices, software models, and communication protocols that are followed in order to perform integration of data, tools and computational resources (see commercial solutions to process management [16]). One could draw some parallels between the requirements imposed on meta-workflow development and perhaps the past requirements on the Internet Mosaic browser development developed at NCSA. It became clear that a browser would provide a mechanism for accessing and retrieving documents from URLs but the URLs had to conform to a certain representation.

The design of meta-workflow architecture is explained by referring to the parallels mentioned above. We believe that the registries of data, tools and computational resources would provide the URL-like representations of all available meta-workflow components that could be easily created and updated by scientists as is true in the case of web pages. Within the context of this comparison, the meta-workflow becomes a browser that (a) displays available components (data, tools and computational resources), (b) 'hyper-links' components (including access, retrieval, execution, management, communication and user interface), (c) stores component settings and references to user's favorite meta-workflow components locally (like browser cookies with passwords and bookmarks), (d) enables searches through the registries of data, tools and computational resources (Google-like search, filter and sort capabilities), (e) provides history of meta-workflow steps and (f) incorporates security and fault-tolerance features. Similarly to the mosaic browser development, meta-workflow component representations (the URL-like representations) have to be widely accepted to the advantage of all scientific communities. The word 'hyper-links' also indicates that triplets of data, tool and computational resource have to be linked in terms of control communication, security, data transfer and data conversion (if necessary).

This view of the meta-workflow design leads to establishing browsers of distributed registries of data, tools and computational resources as the building blocks. It

also indicates the needs for searching (e.g., find matching tools), filtering (e.g., narrow down the number of tools based on hierarchical community grouping of tools) and sorting capabilities (e.g., alphabetical order or based on availability). Furthermore, it is important to consider the needs for secure access (data, tools and computational resources), and information transfers (data, control commands), as well as the needs for system state reporting about data and process provenance, steps and graphs of processing operations, on-line help, and logging statements.

5.2 Meta-Workflow Architecture Supporting Data- and Control-Flow

Once a triplet of data, tool and computational resource has been selected, the meta-workflow has to accommodate data- and control-flows. Based on our assumption that a tool and its associated computer resource are always together, we include two meta-workflow building blocks, such as tool & computational resource (TCR) block, and data block. In addition, to control the meta-workflow, a block for coordinating the data flow is needed as illustrated in **Figure 3**. **Figure 3** provides a system architecture where the building blocks (a) access information (browsers access distributed registries), (b) pass a user-selected triplet of input data, tool and computational resource (browsers provide inputs to data and TCR blocks), (c) communicate about data availability (data flow control block coordinates input and output data handout), (d) pass data to execute a processing step (delineated around each data, TCR and data block sequence) and (e) register output data in one of the data registries after completing the execution.

The data-flow aspect of meta-workflow is represented by the fact that data are passed from data location to tool & computational resource (TCR) location. The control-flow aspect is established by communication of two consecutive TCR blocks, for example, about the availability of input/output data. According to **Figure 3**, when the first TCR block starts its execution, it registers output data in one of the data registries (properties and location) and other TCR blocks in the sequence can request their input data from the data flow control (DFC) block. The DFC block is notified by the first TCR block when the output data is ready and the DFC block can then notify all waiting blocks that their input data is ready.

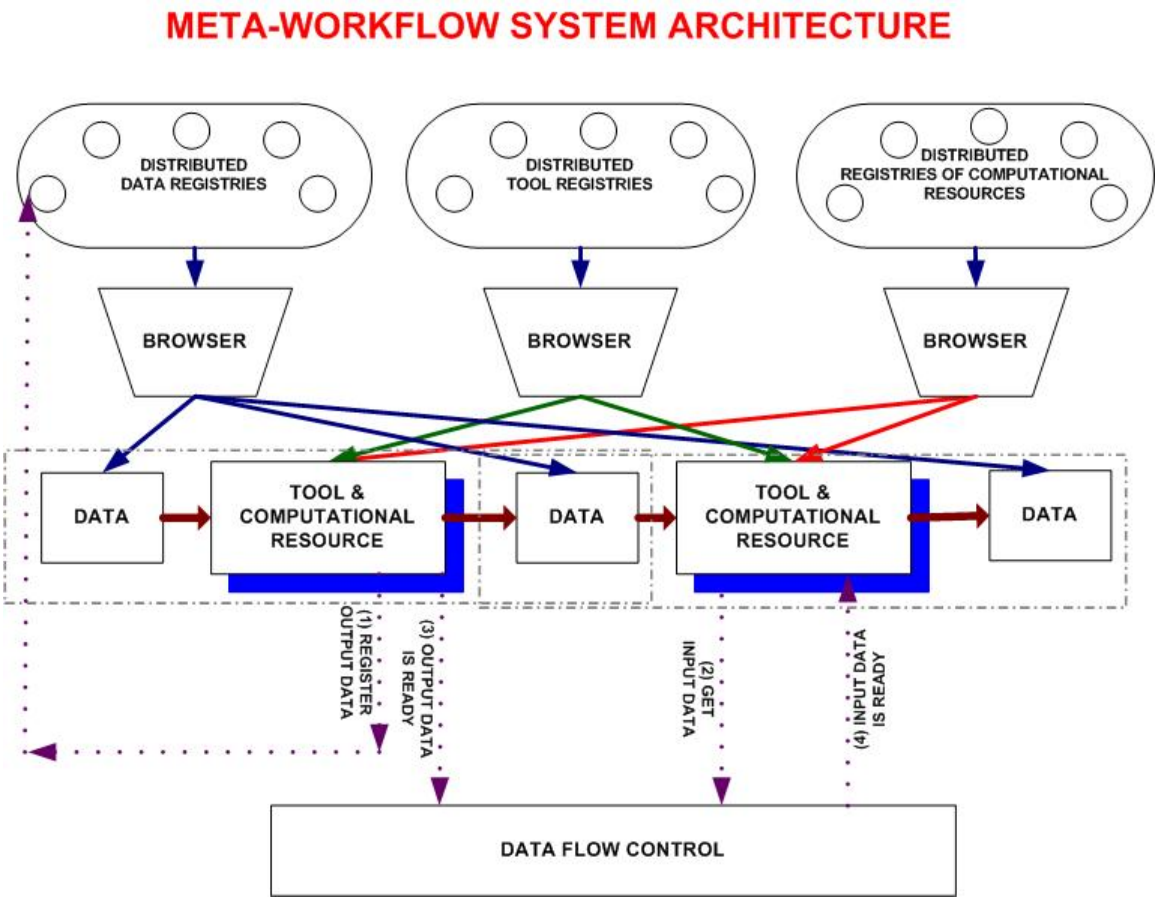

Figure 3: Meta-workflow system architecture.

Within this system architecture, a meta-workflow facilitates data transfers and communication control but not code transfers between multiple TCR blocks. The computational resources, necessary software installations, error handling, security and other internal TCR block issues would not be of concern to a meta-workflow. The issues will be handled only at the meta-workflow level. For example, the issues related to security of data information would be handled by data registries divided into local and remote parts with different security access.

5.3 Descriptions of Meta-Workflow Building Blocks

The meta-workflow system architecture can also be thought of in terms of user interactions and expertise. From this perspective, there are components where (a) a single scientist works alone and interacts with meta-workflow (e.g., browsers of registries and editors for flow formation, re-use and re-purposing), (b) multiple scientists work independently and add to or modify registries of data, tools and computational resources (e.g., editors for registries), (c) a scientist works collaboratively with other scientists by using the meta-workflow information about system states, provenance (e.g., consoles for meta-workflow feedback), (d) software developers add support for new tools that are incompatible with all meta-workflow supported tools (e.g., developer's programming

environment), and (e) software developers that modify or substitute execution and coordination of heterogeneous TCR blocks (meta-workflow engine and data flow control), and internal meta-workflow data representation. This description of meta-workflow building blocks is important for meeting the design requirements of a highly interactive and user-friendly system (see our meta-workflow definition).

Another presentation of meta-workflow building blocks is based on a coarse classification of the components based on their role in the system, such as (1) information registries, (2) system core and (3) system extensions. Information registries include information about data, tools and computational resources. A system core consists of a user friendly meta-workflow editor (including for instance, model viewer showing steps of meta-workflow, and on-line help), data structure representation, model of tools and computational resources, and execution engine. Finally, system extensions contain components providing additional features, such as data and process provenance, security, and fault-tolerance. It is assumed that a bare bone system will operate without system extensions. However, the bare bone system would have very limited capabilities since most remote tools will require security, as well as provenance for scientific collaboration purposes. The extensions should definitely be one part of the meta-workflow architecture design. The rest of this section describes each component based on this classification.

5.3.1 Meta-workflow Information Registries

We are planning to build a simple capability to access a local and/or grid community repository for data sets, with location based on user preference. This capability can eventually be extended to support hierarchical registries, sophisticated filtering, movement between registries based on community requirements for data lifecycle and so on. Our goal is to understand community data and tool cycles in the future to be able to extend our current prototype in future research and development efforts. For this purpose, we plan on using the knowledge and provenance information gathered from the users of our CyberIntegrator prototype.

Data registry: This registry describes information about data sets that are registered. Every output data set of a tool is automatically registered in this registry. The registry could be on a local machine or in a central location and a user can decide where to register his data sets. To narrow down the list of data sets of interest, filters should be available for selecting data sets based on data structures, names, size, access security or geographic locations. Community based filters could select data based on their use (validation or testing), quality (uncertainty and error) or instrument types (image, point, boundaries). Many other search, filter and sort capabilities could be inserted based on individual and community preferences of data.
Tool registry: This registry contains information about tools that are available for forming meta-workflows. A list of tools is retrieved from the tool registry and presented to a user. The list can either be located on the user's machine, or can be kept in several central locations (e.g., community central locations). The central location could be used by workflow engines to publish their workflow. The tool registry could also contain a list of computational resources. When a user requests the list of tools the information could also be combined with the information about computational resources where the tool can

be executed. In our first prototype, we do not expect to have a large list of computational resources. However, in the future, we would explore methods for filtering this list. For example, we envision developing community based filters, where a community would list tools that are commonly used and useful for a class of analyses. Filters could also be based on individual preferences of tools and computational resources, as well as based on availability of tools and computational resources to a user. The tool registry will be extended by a security component to enable registry information retrieval based on access privileges and hence making other tools in the registry invisible to a user.

Computational resource registry: This registry contains information about computational resources that are available for running each tool. Similar to tool registry filters, computational resource registries have to have filters in place as well. For example, a filter could be based on community allocated resources, security, computational load, geographical proximity, processor speed, shared memory size, number of processors, and so on. This registry is extended by a security component.

5.3.2 Meta-workflow Core

Meta-workflow editor: The role of an editor is to enable user-friendly interaction with meta-workflow formation and execution. The editor allows a user to browse the registry of data, tools and computational resources, and select a data set of interest, a tool of choice and a computational resource associated with the tool to complete one step of processing. Once the user has created a sequence of steps, a meta-workflow representation (data structure) can be saved and re-loaded later. The editor depends on all other components of meta-workflow architecture since it has to create the meta-workflow data structure.

Meta-workflow data structures: The role of a meta-workflow data structure is to contain the information about processing steps (an execution of a set of tools). The data structure stores the sequence of tools, properties of each tool, and information about input and output data of a tool. Examples of tool properties are filenames of input-data, algorithmic parameters, and any other variables used in a tool. All other meta-workflow core components depend on the information stored in this data structure. For instance, one of the uses of the information in this data structure is for selecting sub-lists of tools that can operate on a particular data set of choice.

Meta-workflow model of tools and computational resources: The role of this component is to enable an integration of new tools into the meta-workflow framework. Our model of a tool (and the underlying computational resources) can be thought of as a black box that has either inputs or outputs or both, as well as properties (see **Figure 4**). The properties also include the control parameters for each specific tool (e.g., output data is ready). A tool can be a workflow that is executed by a workflow engine, a web-service that is called, an application (such as a numerical model) that is launched, etc.

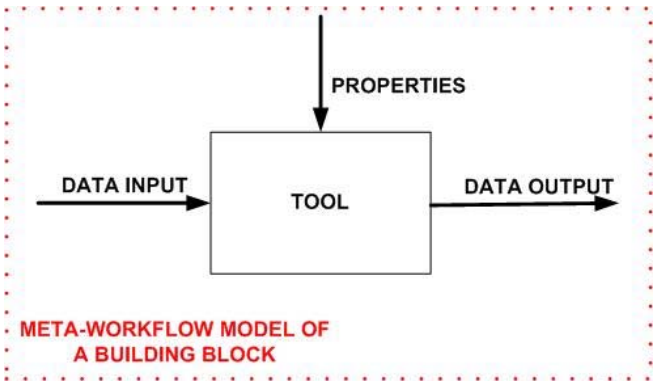

Figure 4: An underlying model of a tool in the meta-workflow architecture.

If a tool does not have any inputs and outputs it is considered to be a stand-alone application. One can still use such a tool in our meta-workflow architecture but it would not allow for easy linking to other tools. In this case, a user will need to select the location on a local disk, or on a network, where the input data and output data are stored. The model of meta-workflow tools and computational resources is embedded into the meta-workflow data structures to preserve the information about input and data, the engine to execute the tool, and the security to access data and computational resources needed.

Meta-workflow engine: The role of an engine is to integrate and execute tools on heterogeneous computer platforms, consisting of but not limited to different workflow engines, software libraries, programming languages, hardware platforms and operating systems. To allow for this, the meta-workflow engine needs to be able to translate data from one representation to another and to execute different applications on different platforms. The engine takes information about the input and output data structures, the tool to be executed and its computational resources for executing from all information registries and performs the execution.

5.3.3 Meta-workflow Extensions

Security: Some tools used in the meta-workflow environment might have restrictions on execution, some data might have limited access only, and some computational resources might be available only to certain users. Security plays an important part in the meta-workflow environment, and the meta-workflow architecture incorporates security as an extension. Since multiple tools will use different authentication systems, the architecture should provide a single sign-on mechanism that would forward the right credentials to the component requesting security clearance. This component will depend on outside single sign-on mechanisms like MyProxy. Given security requirements in many applications, most other meta-workflow components will depend on the security component.

Provenance: The meta-workflow architecture is one part of a larger cyber-infrastructure. In scientific, business, archival and other communities, there is a need to obtain a better understanding of the scientific analyses, to monitor business processes and to preserve flows, that leads us to data provenance and information gathering about meta-workflow processes. For example, the provenance component records what tools a user executed, and what data are used and created. Similar to the security component, the provenance

component is used by many other meta-workflow components. All components should record provenance information about user's activities in a consistent way.
Fault-tolerance: Another important extension is fault-tolerance to prevent losing intermediate results and avoiding wasting computational resources. This problem is addressed at the meta-workflow level by reporting job status of each processing step and registering output data in the data registry. Thus, if a tool failed then the status would be displayed in the sequence of steps and the intermediate result from the previous tool would be available.

6 Key Capabilities of CyberIntegrator

We have prototyped CyberIntegrator, a highly interactive scientific workflow system that aims at building complex problem-solving environments from heterogeneous tools. The editor of CyberIntegrator is shown in **Figure 5**. The description of how to use the editor and what system information can be obtained is provided in Section 6.1. Section 6.2 outlines two use cases that were used for CyberIntegrator prototype development and testing.

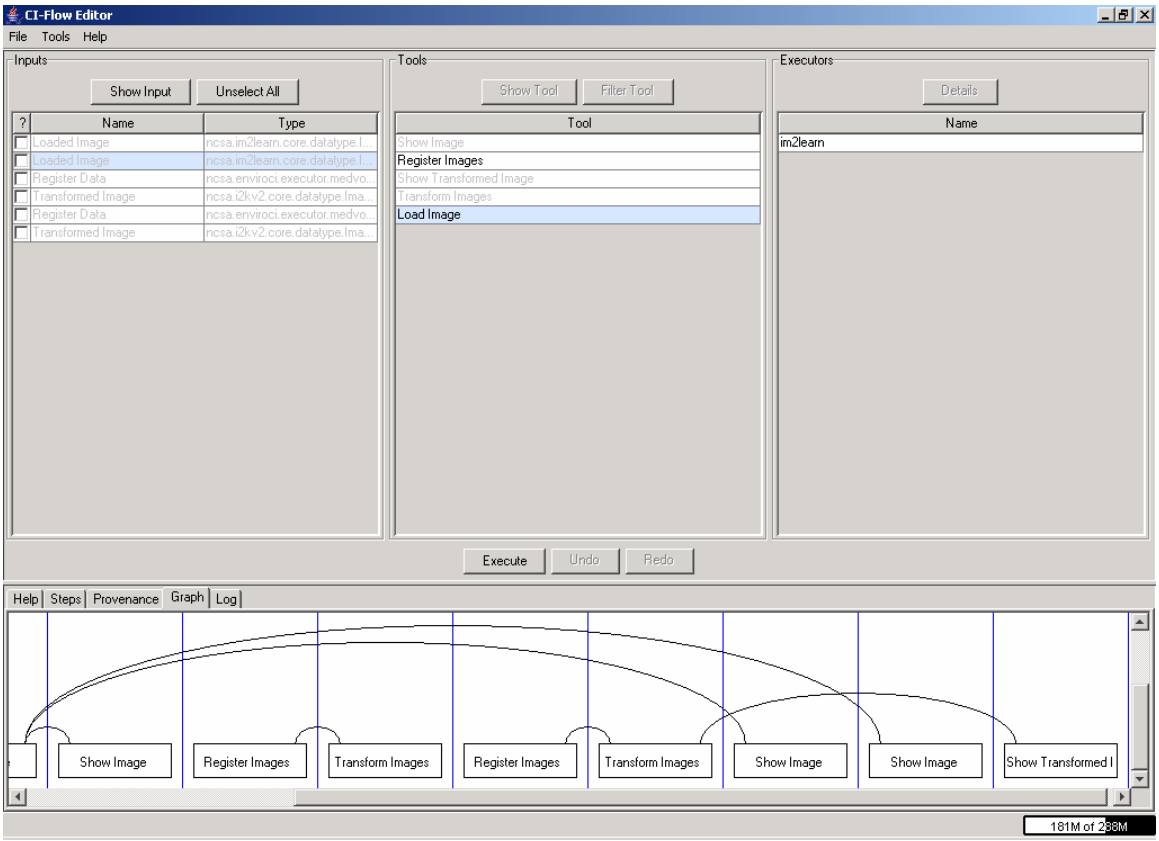

Figure 5: A screen capture of a prototype meta-workflow editor.

6.1 CyberIntegrator Prototype Functionality

This section will go into detail about the current implementation and functionalities of CyberIntegrator meta-workflow editor, engine and registries. The current meta-workflow editor (see **Figure 5**) includes three browsers of information registries (data left, tools middle and executors right), execution control (below browsers) and presentation of system information (bottom). The browser also provides filtering capabilities based on data structure types and grays out tools with incompatible input and output data structures. This is viewed as a visual guide to data and tool selection. The preliminary implementation of registries is using an XML file describing properties of available data, tools and computational resources. We plan on following the development of community registries, for example, for publishing and run-time access to registries developed for the GIS applications based on OGC (Open GIS Consortium) and WRS (Web Registry Service) recommendations (see example registry services presented in [44]). The concept of local and central (likely remote) registries is prototyped by checking multiple XML files, where some of the files are specified by their URL locations. Similarly, available tools can be provided to a user by the executor if an executor has a limited set of local tools available, or can be loaded from an external file. These sets of files are currently a place holder for the tool registry and computational resource registry, but in the future the information about data, tools and computational resources will be loaded from the registries themselves. An example of a tool entry in the tool registry is shown in Figure 6.

```
<tool name="Load Table" uuid="0">
      <output name="Loaded Table"
      type="ncsa.d2k.modules.core.datatype.table.Table" id="0"/>
      <executor>
            <d2ktoolkit itinerary="data/LoadTable.itn">
                  <output alias="Parse File To Table" port="0"
                  refid="0"/>
            </d2ktoolkit>
      </executor>
      <help>This will load a table from disk.</help>
</tool>
```

Figure 6: An example of a ‘Load Table’ tool in a tool registry.

The CyberIntegrator execution engine currently supports tools that came from our image analysis libraries (Im2Learn, GeoLearn, MedVolume and I2K) [25][26][21][24], and visual programming workflow environment developed for data mining applications (D2K) [23]. Other tools that are based on Kepler and Ptolemy II (a scientific workflow environment supporting geospatial and other applications), MS Excel (macro execution), NIH ImageJ (image manipulation), and ArcGIS (georeferencing tools) are currently being investigated for inclusion. Currently to support the above workflow engines, web services and libraries with the meta-workflow engine, an abstraction was made in such a way that each workflow engine is wrapped as an executor. The executor knows how to start any specific workflow environment and how to pass the arguments and properties to this environment. The amount of work and the level of expertise to add any new tool are being evaluated for the tools listed above. The solution to the problem of converting semantically similar data structures that have dissimilar syntax but are needed for linking tools has not been fully implemented. The on-going effort is to develop a linked graph of

data structure conversions that could automatically convert semantically compatible data structures if a path connecting two nodes (structures) exists in the graph.

The meta-workflow formation by example (step by step process with an immediate execution) is enabled by choosing a triplet of data, tool and executor (name of the machine where the tool will run) and pressing the Execute button. The formation for batch execution is achieved by setting the editor to a dummy engine mode. The dummy engine will take a tool and 'pretend' that the tool actually ran. It will create new meta-workflow data structures to enable linking tools, but will not actually execute the tools or store any actual data in the meta-workflow data structures. The meta-workflow that is created this way can be stored and reloaded at a later time. If it is reloaded in the immediate execution mode then each step in the meta-workflow would be executed.

The presentation of system information (bottom of the editor shown in **Figure 5**) currently includes on-line help for each tool (help tab), a sequence of executed steps (steps tab), provenance about user's activities (provenance tab), meta-workflow model viewer of the linked sequence of tools (graphs tab) and logging information about job execution (log tab). The on-line help is important to provide a friendly use of tools. Each tool found in the tool registry comes with its on-line documentation. The steps tab shows information about the sequence of processing steps (time, execution status, input and output) as illustrated in **Figure 7**.

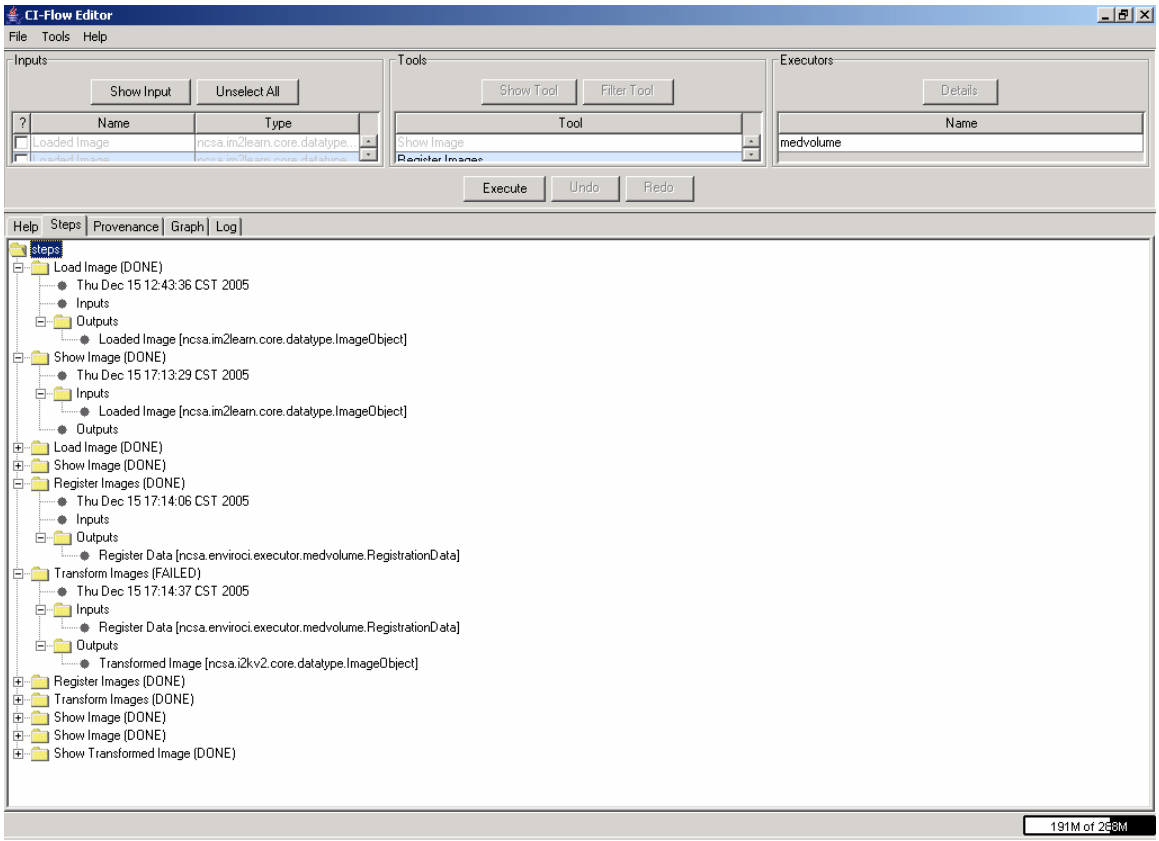

Figure 7: Meta-workflow information about a sequence of processing steps and its execution status.

The provenance tab provides meta-data about user's activities. We view the provenance information as our input to other cyber-infrastructure components that focus more on meta-data (semantic web aspects of CI) and a community-based recommendation mechanism (social network aspects of CI). The provenance data are represented currently as triples with (subject, predicate and object) [27]. More elaborate taxonomy and ontology representations [45], as well mechanisms for interfacing the provenance meta-data [42], [43], would be considered in the future. In our current case, the triplets are extended with a timestamp and the name of the user that created the triplet. The editor will receive the provenance information and make it visible to an end user in the provenance view. While it is important to have unique persistent names for each object and action, we plan on mapping the terms to more user friendly titles for display. The current provenance view uses the following output format:

Time: Function [Subject (Engine), Predicate (action), Object (tool & input)] by User

For example, an example of provenance information would be decomposed into sub-strings as follows:
Provenance information=1134672216511 : ADD [http://enviroci.ncsa.uiuc.edu/md/MultiThreaded_55ba25f4-2585-4811-bda6-e927329e8390 startExecute http://enviroci.ncsa.uiuc.edu/md/step_43dae4b3-4f71-4a9e-add8-0e1542a064e1] by http://enviroci.ncsa.uiuc.edu/md/anonymous

Provenance information sub-strings:
time=1134672216511
function= ADD
Subject(engine)= http://enviroci.ncsa.uiuc.edu/md/MultiThreaded_55ba25f4-2585-4811-bda6-e927329e8390,
Predicate (action) = startExecute
Object (tool & input) = http://enviroci.ncsa.uiuc.edu/md/step_43dae4b3-4f71-4a9e-add8-0e1542a064e1
by User = http://enviroci.ncsa.uiuc.edu/md/anonymous

The graph tab is illustrated in **Figure 5**. It shows how inputs and outputs of multiple tools were inter-connected. Finally, the log tab is also available for informing a user about his job execution. While executing meta-workflow, each tool of the meta-workflow framework might report messages. These messages could be relevant to the performed analyses (e.g., preliminary numerical results) or correspond to the state of the execution (e.g., debug statements and internal code warnings). A user can set the type of log messages in the meta-workflow editor. An example of log messages is provided below, where the syntax follows the structure (Time | Message type | Message).

16379350 DEBUG [Timer-1] ncsa.d2k.ws.client.JobStatusChangeMonitor - Status monitor is shutting down.
16377648 DEBUG [Timer-1] ncsa.d2k.ws.client.JobStatusChangeMonitor - Job status changed ncsa.d2k.ws.server.JobBean@1c47573

16377567 DEBUG [Timer-1] ncsa.d2k.ws.client.JobStatusChangeMonitor - In StatusPollTask.run()

6.2 Meta-workflow Use Cases

During the meta-workflow prototype development and testing, we used two sets of tools driven by (1) the 3D medical volume reconstruction problem [22], [21]and (2) the problem of understanding inter-variable relationships among terrestrial and hydrological variables [19], [20].

First, the 3D volume reconstruction problem consists of linking together three tools. Tool#1 allows a user to load images to be registered, and performs image preprocessing to support semi-automated control point selection for registration (image segmentation pre-processing to compute centroid locations of image structures). Using this tool, a user selects at least three pairs of control points for estimating an affine transformation. Then, Tool#2 is executed by taking the control points selected from Tool#1 and using web services and high-performance computing (HPC) resources at NCSA to compute the affine transformation parameters and transforming sets of images accordingly.
A user can execute Tool#1 and Tool#2 until the results are satisfactory. Tool#3 retrieves registered data from Tool#2 and displays them. The execution of these three tools integrates I2K image analysis library, and D2K web services. Tool#2 can be executed either by using D2K web services or D2KToolkit (workflow environment engine). The test images came from a fluorescent confocal laser scanning microscope and were acquired by scanning multiple cross sections of uveal melanoma tissue at UIC.

Second, the problem of understanding inter-variable relationships was analyzed using data sets obtained from NASA remote sensing sources (e.g., the MODIS AQUA and TERRA satellites). Tool#1 allows a user to pre-process NASA HDF EOS remote sensing data. This step is about extracting multiple raster files (images) from HDF EOS, pre-processing the files based on quality control requirements and integrating them. Tool#2 pre-processes NASA SRTM elevation maps to extract features, such as slope, aspect, flow accumulation etc. Tool#3 takes outputs of Tool#1 or Tool#2 or loads any other geo-spatial raster data and integrates them into a stack of raster variables with a consistent spatial and temporal resolution, as well as geographic location (mosaicking problem) and geographic projection. Tool#4 allows loading vector files (boundaries), and integrating raster and vector files resulting in a boundary membership mask. Tool#5 enables a user to select boundaries of interest and extract variables from the stack of images to a table (table rows represent pixel locations and columns represent variables). Tool#6 lets a user select input and output variables for data-driven modeling. Tool#7 takes tabular data and the list of input and output variables, and computes a data-mining model. Tool#8 computes input variable relevance value with respect to the predicted output variables and visualizes the model prediction, prediction error and input variable relevance in geo-spatial domain. The execution of these tools integrates GeoLearn, Im2Learn and D2K.

We have not delved so far into the issue of the semantic classification of solutions into tools. The tools in the test scenarios were formed based on separating heterogeneous

execution technologies, e.g., I2K, D2KToolkit and D2KServer. The tools described for the second test example are very likely at too fine a semantic and complexity level. For example, the tools #6 and #7 should be combined in a single tool, as well as potentially tools #3, #4 and #5. We believe that the semantic classifications would be more appropriate in the future. The classification will have to come from scientific communities and would be based on semantics and complexity levels of problems and their associated sub-solutions.

7 Summary

This paper focused primarily on the problem of designing a highly interactive scientific meta-workflow system. We explained the need for a meta-workflow view of scientific problems that addresses a class of complex informatics problems. We reviewed thoroughly workflow requirements presented in the past to formulate our set of meta-workflow requirements. Next, we presented the NCSA cyber-environments and the role of meta-workflow in it. We introduced the design logic behind the NCSA meta-workflow architecture called CyberIntegrator. Based on multiple views of the system architecture, the meta-workflow architecture building blocks were described. Finally, key capabilities of our meta-workflow prototype implementation called CyberIntegrator were demonstrated with two use cases. The two use cases include 3D medical volume reconstruction and a data-driven investigation of inter-variable relationships among terrestrial and hydrological variables. The characteristics of our current meta-workflow architecture enable users (1) to browse registries of data, tools and computational resources, (2) to create meta-workflows by example or for batch processing, (3) to re-use and re-purpose meta-workflows, (4) to execute meta-workflows locally or remotely, and (5) to incorporate heterogeneous tools and link them transparently. In the future, we plan on implementing all features described in the meta-workflow architecture to provide a practical tool for communities we have been working with.

8 Appendix A: URLs of Institutions Working on Cyber-infrastructure

This is the list provided at the NSF web site:

- National Center for Supercomputing Applications (NCSA), http://www.ncsa.uiuc.edu.
- Ohio Supercomputing Center, http://www.osc.edu .
- Pittsburgh Supercomputing Center, http://www.psc.edu .
- San Diego Supercomputer Center (SDSC) at UCSD, http://www.sdsc.edu.
- Extensible TeraGrid Facility, http://www.teragrid.org.
- Internet2, http://www.internet2.org.
- NSF Middleware Initiative (NMI), http://www.nsf-middleware.org .
- Condor, http://www.cs.wisc.edu/condor .
- Globus Alliance, http://www.globus.org .
- Globus World website, http://www.globusworld.org.
- AccessGrid, http://www.access.org

- Web Services Activity, http://www.w3.org/2002/ws
- Semantic Web Activity, http://www.w3.org/2001/sw.
- DataGrid Project, http://eu-datagrid.web.cern.ch/eu-datagrid.
- Global Grid Forum, http://www.gridforum.org
- Center for Embedded Networked Sensing, http://cens.ucla.edu
- Science of Collaboratories, http://www.scienceofcollaboratories.org/.
- InfoVis Cyberinfrastructure, http://iv.slis.indiana.edu
- Workshop on Synthesizing Management Models for Cyberinfrastructure, http://www.si.umich.edu/cyber/july292003

9 References:

[1] Emmerich W., B. Butchart, L. Chen, B. Wasserman and S. L. Price, (2005), " Grid Services Orchestration using the Business Process Execution Language (BPEL), Research Note RN 05/07, Dept. of Comp. Science, University of College London, Gower St. London, WCIE 6BT, UK

[2] Altintas I. et al. (2005), A Framework for the Design and Reuse of Grid Workflows, in SAG 2004, LNCS 3458, Springer-Verlag Berlin, Heidelberg, pp. 120-133, 2005

[3] Kumar P., J. Alameda, P. Bajcsy, M. Folk and M. Momcilo, Hydroinformatics: Data Integrative Approaches in Computation, Analysis, and Modeling, published by CRC Press LLC, October, 2005, 534p.

[4] Bajcsy P., J. Han, L. Liu and J. Young, "Survey of Bio-Data Analysis from Data Mining Perspective," Chapter 2 of Jason T. L. Wang, Mohammed J. Zaki, Hannu T. T. Toivonen, and Dennis Shasha (eds.), Data Mining in Bioinformatics, Springer Verlag, 2004, pp.9-39.

[5] Foster I. and C. Kesselman. "Computational Grids," Chapter 2 of "The Grid: Blueprint for a New Computing Infrastructure", Morgan-Kaufman, 1999.

[6] Karo M., C. Dwan, J. Freeman, J. Weissman, M. Livny, E. Retzel, "Applying Grid technologies to bioinformatics," Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing, 2001 pp. 441 –442.

[7] Fensternmacher D., "Introduction to Bioinformatics," Journal of the American Society for Information Science and Technology, 65(5): 440-446, 2005.

[8] Ludäscher B., I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, Y. Zhao (2005), "Scientific Workflow Management and the KEPLER System," Concurrence and computation: Practice and Experience, Special Issue on Scientific Workflows (to appear), available at http://users.sdsc.edu/~ludaesch/Paper/kepler-swf.pdf

[9] Leavit N. (2004), "Are Web Services Finally Ready to Deliver?" Computer, November 2004, pp14-18.

[10] Rusinkiewicz M. and A. Shetz (1995) Specifications and executions of transactional workflows, In modern Database systems: The object model, interoperability and Beyond, W. Kim Ed., ACM Press , New York, 592-620.

[11] Peltz C. (2003), "Web ser vices orchestration, a review of emerging technologies, tools, and standards," Tech report, Hewlett Packard, Co., January 2003

[12] Rosnay J., “THE MACROSCOPE: A NEW WORLD SCIENTIFIC SYSTEM,” Harper & Row, Publishers, New York 10022, 1997 (translated from Le Macroscope. Vers une vision globale. Editions du Seuil, 1975).
[13] Schuler C., S. Heiko, G. Alonso, and H-J. Schek, “
Workflows over Workflows: Practical Experiences with the Integration of SAP R/3 Business Workflows in Wise,” In: Proceedings of the Informatik'99-Workshop "Unternehmensweite und unternehmens"ubergreifende Workflows: Konzepte, Systeme, Anwendungen". Paderborn, Germany, October 1999. Nr. 99-07, Ulmer Informatik-Berichte, University of Ulm.
[14] Gill, M., Gregg, D., Schuff, D., Goul M., and Santanam, R., “Automating distributed workflow for electronic commerce: A model for building meta-workflow components,” AIS Americas Conference on Information Systems, Milwaukee, WI. August 13-15, 1999. pp. 871-873
[15] Thimm R. and T. Will,” AMIRA, A State of the Art Analysis of
Workflow Modelling and Agent-based Information-Systems,” Technical report at the University of Trier, Department of Business Information Systems II, October 29, 2004 http://www.wi2.uni-trier.de/publications/2004_TechnicalReport_thimmWill.pdf
[16] The URL of EGIP, a technological leader for web-based solutions for innovation- and process management, http://www.meta-workflow.com/
[17] M.P. Singh and M.N. Huhns, “Automating Workflows for Service Order Processing: Integrating AI and Database Technologies,” IEEE Expert, Vol. 9, No. 5, Oct. 1994, pp. 19–23. 5. J. Gray and A. Reuter,
[18] M. N. Huhns and M. P. Singh, “Workflow Agents,” IEEE Internet Computing, Vol. 2. No 4, ISSN 1089-7801, JULY - AUGUST 1998, pp94-96,
[19] Kumar P., P. Bajcsy, D. Tcheng, D. Clutter, V. Mehra, W-W Feng, P. Sinha and A. White, ”Using D2K Data Mining Platform for Understanding the Dynamic Evolution of Land-Surface Variables,” the 2005 Earth-Sun System Technology Conference, University of Maryland, MD, June 28-30, 2005.
[20] Kumar P., P. Bajcsy, D. Tcheng, D. Clutter, V. Mehra, W-W Feng, P. Sinha and A. White, ”Data Driven Discovery,” Hydrologic Information System, Status Report, CUAHSI Universities allied for water research, Editor: David Maidment, Version 1, September 15, 2005, pp.188-203.
[21]Kooper R., A. Shirk, S-C. Lee, A. Lin, R. Folberg and P. Bajcsy, “3D Volume Reconstruction Using Web Services,” The 3rd IEEE International Conference on Web Services (ICWS 2005), July 12-15, 2005, Orlando, Florida, pp. 709-716.
[22] Bajcsy P., S-C. Lee, A. Lin and R. Folberg, “3D Volume Reconstruction of Extracellular Matrix Proteins in Uveal Melanoma from Fluorescent Confocal Laser Scanning Microscope Images,” Journal of Microscopy, (accepted September 2005).
[23] Welge, M., Hsu, W.H., Auvil, L.S., Bushell, C., Martirano, J., Redman, T.M., Tcheng, D. 1999, Data to Knowledge (D2K): A Rapid Application Development Environment for Knowledge Discovery in Database, Technical Report, National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, Champaign.
[24] Bajcsy, P., Kooper, R., Clutter, D., Lee, S-C., Ferak, P. (2005), Image to Learn user manual, Tech report ALG-2005-005, URL:
http://i2k.ncsa.uiuc.edu/Im2Learn/Im2LearnManual_v1.pdf.

[25] Bajcsy P. et al. (2003), Image to Knowledge user manual, http://i2k.ncsa.uiuc.edu/i2kmanual/
[26] GeoLearn user manual (software tools for data-driven modeling from remote sensing data) http://i2k.ncsa.uiuc.edu/GeoLearn/GeoLearn_userGuide_v8.pdf
[27] Resource Description Framework (RDF) , http://www.w3.org/TR/rdf-concepts/#section-Graph-Literal
[28] Wilson, B.D. (2003), GENESIS SciFlo: Enabling Multi-Instrument Atmospheric Science Using Grid Workflows, poster SF31A-0716 0800h at American Geophysical Union (AGU) Fall Meeting Special Focus: Advances in Data Acquisition, Management, Analysis and Display, pp. 13-17.
[29] Yunck, T., Wilson, B., Braverman, A., Dobinson, E. Fetzer, E. (2004), GENESIS: The General Earth Science Investigation Suite, The fourth annual NASA's Earth Technology Conference.
[30] Goderis A., U. Sattler, P. Lord and C. Goble. Seven bottlenecks to workflow reuse and repurposing. Accepted for the 4th Int. Semantic Web Conf., Galway, Ireland, 6-10 Nov. 2005
[31] TAVERNA, workflow environment for bioinformatics developed by eScience community, http://taverna.sourceforge.net/
[32] DAGMan (Directed Acyclic Graph Manager) is a meta-scheduler for Condor, http://www.cs.wisc.edu/condor/dagman/
[33] Common Component Architecture Forum, http://www.cca-forum.org/
[34] Alameda J. S. Hampton, B. F. Jewett, A. Rossi, and R. B. Wilhelmson, "Ensemble Broker Service Oriented Architecture for LEAD," to be presented at the 22nd International Conference on Interactive Information Processing Systems for Meteorology, Oceanography, and Hydrology, Jan 28-Feb.2, 2006
[35] ArcGIS ModelBuilder supported by ESRI, http://support.esri.com/index.cfm?fa=downloads.geoprocessing.gateway
[36] Business Process Execution Language (BPEL), http://www-128.ibm.com/developerworks/library/specification/ws-bpel/
[37] Kooper R, L. Marini, J. Myers and P. Bajcsy, "A Meta-Workflow System Designed for Solving Complex Scientific Problems using Heterogeneous Tools," the IEEE Workshop on Workflow and Data Flow for Scientific Applications, Atlanta, GA, April 2006, (submitted December 2005).
[38] NLADR, National Laboratory for Advanced Data Research, URL: http://www.nladr.org/
[39] Pegasus (Planning for Execution in Grids), URL: http://www.isi.edu/ikcap/pegasus/research.html
[40] CLEANER (Collaborative Large-scale Engineering Analysis Network for Environmental Research), URL: http://cleaner.ncsa.uiuc.edu/home/
[41] CUAHSI (Consortium of Universities for the Advancement of Hydrological Science), URL: http://www.cuahsi.org/
[42] Pfister R., R. Ullman, and K. Wichmann, "ECHO Responds to NASA's Earth Science User Community", available at URL: http://www.prism.washington.edu/bloodstream/NASA-ECHO.html
[43] ECHO: Earth observing system ClearingHOuse, URL: http://www.echo.eos.nasa.gov/

[44] Zhao P., A. Chen, Y. Liu, L. Di, W. Yang, P. Li, “Grid Metadata Catalog Service-Based OGC Web Registry Service,” Proceedings of the 12th annual ACM international workshop on Geographic information systems, Washington DC, USA, November 12–13, 2004, Pages: 22 – 30. ISBN:1-58113-979-9.
[45] Semantic Web for Earth and Environmental Terminology (SWEET) http://sweet.jpl.nasa.gov/ontology/