**State of Workflow Systems for eScience**

Beth Plale, Geoffrey Fox, Stacy Kowalczyk, Kavitha Chandrasekar, Bina Bhashar

Indiana University

**June 28, 2011 -- DRAFT VERSION**

# Introduction

The concept of workflow was first developed in the Workflow Management Coalition [[6](#_ENREF_6)] which has existed for almost 20 years and generated standard reference models, documents and a substantial industry of tools and workflow management support products. Although originating in the business sector, workflows have become an important component of digital scientific research. This report is a synthesis of multiple initiatives and studies to provide an overview the research on and the state of workflow systems.

# Overview of Workflow Systems

Most workflows can be described by a graph as illustrated in figure 1 that specifies the interaction between the multiple services or activities. One important technology choice is the mechanism for transferring information between the nodes of the graph. The simplest choice is that each node reads from and writes to disk and this allows one to treat the execution of each node as an independent job invoked when all its needed input data are available on disk. The cost of reading and writing is often quite acceptable and allows simpler fault tolerant implementations. Of course, one can use messaging systems to manage data transfer in a workflow and in extreme cases, simple models where all communication is handled by a single central “control node”. Obviously this latter could lead to poor performance that does not properly scale as workflow size increases.

|  |
| --- |
|  |
| **Figure 1. A workflow graph can include subgraphs, pipelines and loops** |

There are in fact often two communication systems in workflow environments corresponding to “control” and “data” respectively. Obviously the control communication would usually have small messages and very different requirements from the data network. In this regard one should mention the “proxy model” which is often used in Grid architectures and workflow. The information flowing between proxy nodes is all essentially control information. This proxy model can also be considered as an agent framework [[35](#_ENREF_35)].

Some workflow systems are built around the *dataflow* concept with this being the original model [[10](#_ENREF_10), [11](#_ENREF_11), [54](#_ENREF_54)] with the interaction scripted in languages like JavaScript or PHP. Other workflow approaches extend the “remote method invocation” model coming from the distributed object paradigm. This model underlies the Common Component Architecture CCA [[57](#_ENREF_57), [58](#_ENREF_58)]. The Business Process Execution Language (BPEL), an OASIS [N] standard executable language for specifying actions within business processes with web services, specifies the control and not data flow of a workflow. Of course the control structure implies the dataflow structure for a given set of nodes; one simple but extremely important workflow structure is the pipeline. A more general workflow structure is that of the *directed acyclic graph* (or DAG) which is a collection of vertices and directed edges, each edge connecting one vertex to another, such that there are no cycles. That is there is no way to start at some vertex V and follow a sequence of edges that eventually loops back to that vertex V again. Dagman [[59](#_ENREF_59)] used in Condor [[60](#_ENREF_60)] is a sophisticated DAG processing engine. This leads to a class of workflow systems like Pegasus [[44](#_ENREF_44)] aimed at scheduling the nodes of DAG based workflows. Karajan [[61](#_ENREF_61)] and Ant [[62](#_ENREF_62)] can also easily represent DAG’s.

Important workflow systems based on dataflow technologies are the Kepler [[38](#_ENREF_38)] [[39](#_ENREF_39)] [[40](#_ENREF_40)] and Triana [[41-43](#_ENREF_41)] projects; Kepler is still actively being developed. Pegasus is an active system implementing the scheduling style of workflow [[44](#_ENREF_44)]. Taverna [[27](#_ENREF_27), [45](#_ENREF_45)] from the myGrid project [[46](#_ENREF_46)] is very popular in the bioinformatics community and substantial effort has been put by the UK OMII effort [[47](#_ENREF_47)] into making the system robust. An innovative extension of this project is the myExperiment scientific social networking site [[48](#_ENREF_48)], which enables sharing of workflows.

|  |
| --- |
|  |
| **Figure 2. Workflow System Components** |

Workflow systems have four major components: workflow composition, workflow orchestration, task scheduling, and one or more application task interfaces. Workflow composition is the process by which a user chooses functions, describes inputs and outputs, and determines dependencies to create the workflow. This process can be accomplished by a graphical user interface, a command line interface, a set of configuration files or any combination of these. Workflow orchestration is the process by which a workflow works– that is how the processes are initiated and executed. Task scheduling is the process by which individual steps in a workflow are managed determining the start time, marshaling the resources necessary, and coordinating multiple threads. An application task interface is the manner in which workflow systems communicate with applications, web services, plugins, and other domain specific executables.

# Sub-workflow Interoperability

Gannon and Klingenstein raised the question of workflow interoperability through organizing a 2008 NSF/Mellon workshop (sci-workshop). While the workshop raised the question, the issue required a more nuanced form before it could be studied. An advancement in characterization of workflow interoperability was undertaken by Elmroth et al. (elmroth2010three). Elmroth et al. suggest that interoperability can take several forms, expressed as three Levels. Level 1: workflow system coordinates activities that are designed for another system. Level II interoperability or Level II: sub-workflow interoperability is where sub-workflows are shared between systems. The third level of interoperability is Level III: complete workflow interoperability where it is possible to execute a workflow designed for one system by another.

In (Plale eScience 2011) we posit that of the three forms of workflow interoperability, Level I, II, and III, sub-workflow interoperability (level II) is likely to have the longest lasting value for the following reason. Most reuse of workflows in myExperiment.org is of sub-workflows, as has been noted by the authors, that is, users share portions of workflows, and these sub-workflows are being picked up for adoption at higher rates than are full workflows. Suppose further then that a workflow system guarantees reproducibility of its sub-workflows (or full workflows). If a new user were to include the sub-graph into her workflow, it is reasonable then that she would be inclined to run the sub-workflow where it is guaranteed to run. Suppose now that the researcher has two such sub-workflows whose guarantees are provided by two different workflow systems. It is reasonable then for her to create a single workflow that uses the two component workflow systems where the guarantee is strongest. Another use case for the sub-workflow approach is the case where there are certain scientific workflow cases that are unique and require unique capabilities. Parametric sweeps for millions of parameters required for the earthquake simulation requires managing millions of small jobs. For such cases specialized systems are needed for efficient execution (deelman2005pegasus), (zhao2007swift), (frey2002condor).



**Figure 3. Sub-workflow interoperability shares workflows between systems in this diagram adapted from Elmroth 2010. Activities B and F are called from System 1 by instructing Systems 2 and 3 respectively to execute the activities. B, while seen as a single activity in System 1, is actually a subworkflow. F is an activity that runs on grid middleware.**

While we posit that sub-workflows will have longest lasting value, to our knowledge there is no good comparative data on the costs, both quantitative and qualitative of adopting the strategy. We undertook to fill in the gap in knowledge through a performance evaluation and qualitative evaluation of sub-workflow interoperability that fixes the high level system at a user desktop workflow engine, and explores the performance and programming impact of various forms of remote activity. Adhering to the categorization and terminology of Elmroth et al. 2010 and shown in Figure 3, System 1 is hosted on a user's desktop machine. In this model, workflow activities run where the engine runs. This is the case for activities A and E. Two forms of sub-workflow interoperability are shown in the figure. Activity B is called from System 1 by instructing System 2 to execute the activity. B, while seen as a single activity in System 1, is actually a sub-workflow consisting of activities C and D. Activity F is called from System 1 by instructing System 3 to execute the activity. F, which is seen as an activity in System 1 calls out to grid middleware to carry out the execution of the activity. Other forms of sub-workflow interactivity can exist, but a system that can utilize local machine resources for simple execution and use remote resources for more complex tasks is simpler in the simple case. Remote access workflow systems are complex distributed systems, and that programming complexity should not hurt the simple case, enforcing the adage what a user doesn't know should not hurt them. In our evaluation we hold System 1 constant as a user desktop workflow system, specifically, the Trident Scientific Workflow Workbench (Trident). Trident was chosen because it is easy to use and as a Windows desktop solution could benefit from sub-workflow interoperability as there is a preponderance of scientific functionality running on Linux based systems. Trident uses the .NET Workflow Foundation for workflow execution.

We undertook a qualitative and quantitative evaluation of sub-workflow interoperability. Using the model in Figure 3 for the System 2 case we evaluate the performance of the Kepler workflow system (Altintas, 2004), and the Apache ODE (apache-ode) workflow tool. For the System 3 case where remote services are invoked directly we evaluate GFac (Kandaswamy Gfac) and Opal (Krishnan 2006). We also evaluate each system for compatibility with the Model of Computation (MOC) (Elmroth, 2010) of the host workflow system. The model of computation, according to Elmroth, considers the definition of interaction between activities. Intuitively, the MOC gives interpretation to the edges that connect two vertices of a workflow graph (i.e., the activities). We are interested in determining how compatible a remote system MOC is to the local system MOC with respect to typing, control or data flow, and scheduling of activities. <forthcoming>

Through experimental evaluation, we found that the remote engine model compares favorably to the remote grid middleware in terms of performance for both Kepler and ODE stacks. The overhead of executing a workflow that has a remote component (either activity or sub-workflow) was fairly low, about 5%.

The nuances enter into the qualitative aspects of sub-workflow interoperability. The model of computation (MOC) of a workflow system captures such aspects of a workflow system as the execution models supported by a system, the node scheduling algorithm, and typing restrictions on edges. As such, the MOC defines the expressiveness of a workflow system by addressing the kinds of execution models are supported. Does a system support parallel execution for instance? Control flow or data flow execution of DAGs, finite state machines? The MOC is defined in Elmroth et al. (Elmroth et al. 2010), and the MOC of several systems differentiated in Goderis et al. (Goderis et al. 2009). In this study we fix the top level workflow system as the Trident desktop workflow engine, and study various forms of sub-workflow interoperability. Further, Elmroth et al. state that "sub-workflows are seen as black boxes and their internal MoC is not important to workflows higher up in the hierarchy", meaning that we need not consider the internal edges of the subgraph (sub-workflow). The black-box nature of the sub-workflow model has advantages and disadvantages. The advantage factors into the picture when one examines the MOC of Trident. Trident is a control-flow workflow system. All scheduling decisions are based on static information and this information is used to generate a Actor/activity firing schedule in the form of a Windows Workflow Foundation run-time script before it starts execution.

Goderis et al. define a process network model where each actor executes in a Java thread, and all actors execute concurrently. Trident does not natively support the process network model. There is only one thread executing when a Trident workflow is executed. As a result, in case of a parallel workflow in Trident, there is no concurrent execution, but only interleaving execution of activities using the same thread. However, Trident can support a process network model by each parallel activity to spawn a child thread, then define a reduce or join type of process that waits for completion of these concurrent asynchronous threads. This is how we executed the parallel workflows of this study. The fact that the sub-workflow is a black box means that the Trident model of computation does not extend, nor limit the MOC used within the sub-workflow.

# Workflow System Features

Most sophisticated workflow systems support hierarchical specifications – namely the nodes of a workflow can be either services or collections of services – namely sub-workflows. This is consistent with the Grid of Grids concept described at the start of section 3.5. Other key aspects of workflow are security [[63](#_ENREF_63)] and fault-tolerance [[5](#_ENREF_5)]. Within each of these major functions, we further demarcate the different workflow system functionalities:

* *Integral domain independent workflow tasks, nodes, or activities.* What functions are built in to facilitate workflow creation? How can these functions be compounded to build complex activities?
* *Data movement/access between workflow nodes.* Can the workflow tasks access files easily? What in memory functions are available? (For example in-memory streaming through distributed brokers, centralized allocation server, or other technologies.)
* *Provenance and metadata collection.* What data is automatically collected to provide information on the execution, the purpose, and the results of the workflow?
* *Fault tolerance.* How well do these systems recover from error? With a workflow? Within a task or activity? From system errors? From application errors?
* *Parallel execution of workflows.* To what extent can workflows be run in parallel?
* *Sharing workflows*. How can researchers share components of workflows, complete workflows, and the output data from workflows?

## Current Workflow Systems

We determined the most widely used workflow systems most used in research and business environments based on a literature review. These workflow systems focus on different segments of the market which in turn drives the functionalities implemented and the technologies used. Below is a brief overview of the 10 major workflow systems.

**Kepler** is adata-flow oriented workflow system with an actor/director model that is used in ecology and geology domains.

**Taverna** is primarily focused on supporting the Life Sciences community (biology, chemistry and medical imaging) and uses a data-flow oriented model.

**Swift** is data-flow oriented workflow that uses a scripting language called Swiftscript, to run the processes in the workflow.

**Ode** is not a full system, but a workflow engine that needs to be supported by other tools and components to design and execute workflows. It can be used with front end tools such as XBaya described below.

**VisTrails** is a scientific workflow and provenance management system that provides support for data exploration and visualization.

**Trident** is a workflow workbench developed by Microsoft Research and relatively newer among the other workflow systems. It is based on the Windows Workflow Foundation (WF).

**IBM smash** makes use of enhanced BPEL Web 2.0 workflow environment for building and running dynamic Web 2.0-based applications using SOA principle.

**Lims** has elements of a workflow system, but is primarily designed as a laboratory information system to analyze experimental data using “G”, a data-flow oriented language.

**Inforsence** is a BI solution that enables organizations to use a drag and drop visual environment to build predictive and statistical models and other analytical applications on dynamically combined data from a variety of sources such as data warehouses, spreadsheets and documents as well as web services.

**Pipeline pilot** has an integrated set of applications which model and simulate informatics and scientific businesses intelligence needs of research and development organizations.

**Triana** is a graphical workflow and data analysis tools for domains including signal, text, and image processing. It includes a library of tools and users can integrate their own tools, Web, and Grid Services. Triana is a Java application and will run on almost any computer. It hides the complexity of programming languages, compilers, debuggers, and error codes.

**XBaya**  is a graphic workflow front end for backend engines such as ODE and ActiveBPEL. It can be used as a standalone application or as a Java Web Start application.

## Workflow Standards

The period 2000-2005 produced a number of workflow standards that were viewed as essential to enable the Web Service dream of interoperability by complete specification of service features. Recently there has been a realization that this goal produced heavyweight architectures where the tooling could not keep up with support of the many standards. Today we see greater emphasis on light weight systems where interoperability is achieved by ad hoc transformations where necessary. A significant problem of the standardization work was that it largely preceded the deployment of systems; the premature standardization often missed key points. This background explains the many unfinished standards activities in Table 1. The successful activities have a Business process flavor; for scientific workflows, the most relevant standard is BPEL [[20-23](#_ENREF_20)], which was based on earlier proposals WSFL and XLANG.

XML is not well suited to specifying programming constructs. Although XML can express data structures well, it is possible but not natural to express loops and conditionals that are essential to any language and the control of a workflow. It may turn out that expressing workflow in a modern scripting language is preferable to XML based standards. However, exporting data or workflows as part of ad hoc transformations for interoperability might be an appropriate use of XML in workflow systems.

Table 1. Workflow Related Standards

|  |  |  |
| --- | --- | --- |
| **Standard** | **Link** | **Status** |
| **BPEL** Business Process Execution Language for Web Services (OASIS) V2.0 | <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html>; <http://en.wikipedia.org/wiki/BPEL> | April 2007 |
| **WS-CDL** Web Service Choreography Description Language (W3C)  | <http://www.w3.org/TR/ws-cdl-10/>  | November 2005 Not final |
| **WSCI** Web Service Choreography Interface V1.0 (W3C) | <http://www.w3.org/TR/wsci/> | August 2002. Note only |
| **WSCL** Web Services Conversation Language (W3C) | <http://www.w3.org/TR/wscl10/> | March 2002 Note only |
| **WSFL** Web Services Flow Language | <http://www.ibm.com/developerworks/webservices/library/ws-wsfl2/> | Replaced by BPEL |
| **XLANG** Web Services for Business Process Design (Microsoft) | <http://xml.coverpages.org/XLANG-C-200106.html> | June 2001 Replaced by BPEL |
| **WS-CAF** Web Services Composite Application Framework including **WS-CTX, WS-CF** and **WS-TXM**   | <http://en.wikipedia.org/wiki/WS-CAF>  | Unfinished |
| **WS-CTX** Web Services Context (OASIS Web Services Composite Application Framework TC)  | <http://docs.oasis-open.org/ws-caf/ws-context/v1.0/OS/wsctx.html> | April 2007 |
| **WS-Coordination** Web Services Coordination (BEA, IBM, Microsoft at OASIS)  | <http://docs.oasis-open.org/ws-tx/wscoor/2006/06>  | February 2009 |
| **WS-AtomicTransaction** Web Services Atomic Transaction (BEA, IBM, Microsoft at OASIS)  | <http://docs.oasis-open.org/ws-tx/wsat/2006/06>  | February 2009 |
| **WS-BusinessActivity** Description: cWeb Services Business Activity Framework (BEA, IBM, Microsoft at OASIS) | <http://docs.oasis-open.org/ws-tx/wsba/2006/06>  | February 2009 |
| **BPMN** Business Process Modeling Notation (Object Management Group OMG) | <http://en.wikipedia.org/wiki/BPMN>; <http://www.bpmn.org/> | Active |
| **BPSS** Business Process Specification Schema (OASIS) | <http://www.ebxml.org/>; <http://www.ebxml.org/specs/ebBPSS.pdf> | May 2001  |
| **BTP** Business Transaction Protocol (OASIS)  | <http://www.oasis-open.org/committees/download.php/12449/business_transaction-btp-1.1-spec-cd-01.doc>  | Unfinished |

# Studies

To evaluate the existing field of workflow systems, three studies have been developed. In the fall of 2010, we completed a heuristic evaluation of 6 workflow systems; and in spring 2011, we completed a hand-on usability study of 4 workflow systems. A quantitative survey is planned for summer 2011in which we will survey users of a variety of workflow systems.

## Heuristic Evaluation

In the heuristic evaluation, 6 workflow systems were reviewed to determine the functional and technical capabilities. The systems reviewed were Trident, Swift, VisTrails, Kepler, Taverna, and Ode. This study evaluated each system based on the functions described above: specialized activities, the underlying back end engine, data movement functions, the ability to interface to application code, the ability to share workflows, fault tolerance, and provenance and metadata collection. A summary of the evaluation follows. The full report is attached in Appendix A.

## Evaluation Summary

* **Specialized Activities.** Trident and Taverna provided many standards based interfaces. Trident allows for a standard scientific semantic and syntactic data format (NetCDF). Kepler is specifically tuned tows scientific workflows with internal functions for grid access, mathematical and statistical interfaces, and support for multiple programming languages.
* **Underlying Back-end Engine.** Each of the workflow systems in this study has their own engines. Trident uses the Windows Workflow Foundation. VisTrails has a cache manager model. Kepler uses the Ptolemy engine. Ode has a JDBC data store with a data access layer and the BPLE engine.
* **Data Movement.** Trident has limited set of libraries for data movement out of the box. However it is flexible with the .Net framework to use memory, files, or databases for data movements. Swift has a set of libraries for data movement including functions that map data and provide a number of grid functions including gridftp. Kepler provides access to data within scientific domain specific repositories and has component libraries to read, write, and manipulate domain specific standard data formats.
* **Application Code Interface.** Trident uses web services to interface to application code and can execute any function using the .Net framework. Swift has a propriety scripting language as well as a Java API to interact with grid services. VisTrails supports web services and python scripts. Kepler and ODE have APIs. Taverna has remote execution services that allow it to be invoked from other applications.
* **Workflow Sharing.** Ode and Swift did not provide functions that allow for easy workflow sharing, while Trident, Taverna, VisTrails, and Kepler did. Kepler provided a robust tagging and retrieval function for its sharing environment. VisTrails allows for versioning of workflows. In addition, Trident, Kepler, and Taverna can share workflows via the myExperiement website.
* **Fault Tolerance.** Error recovery in the current Trident (v 1.2.1) is to restart the entire workflow. Swift supports internal exception handling and resubmission of jobs. Kepler supports dynamic substitution at the task level, rescue at the workflow level and restart of the failed component. Taverna supports task level restart with dynamic substitution. Ode has a “failure policy” that can support activity retry, fault or cancel.

* **Provenance and Metadata Collection**. Trident uses semantic tagging for provenance data. Swift, VisTrails, and Kepler have separate components for tracking provenance data.Taverna supports a sophisticated multi-level provenance data collection, storage, and tracking mechanism. Ode stores provenance data within its internal data storage scheme of Data Access Objects.

## Discussion

All of these workflow systems have robust infrastructures and architectures that provide libraries and functions for data transfer management and job submission and execution management. Several of the systems repurpose engines such as the Windows Workflow Foundation or Ptolemy. While each may be tuned to a specific function set or market segment, all could be implemented and used by a wide range of users.

The major differentiators in the workflow systems studied are provenance collection and fault tolerance. Although provenance is supported by most of the systems, the level of data collected, the data format and manner of storage, and the retrieval and display of the data varies widely. It is the post-process use of provenance that is both intriguing and underutilized. How can this provenance data be used to recreate experiments or workflows? How could this data be used to determine which data should be contributed to national reference databases (such as Protein Data Bank or the DataConservancy). Fault tolerance is a second differentiator. Providing a robust set of tools, functions, and options for recovering from failure, at the task or workflow level, is a significant user function that needs to be incorporated into all workflow systems. This set of services needs to be visible to the user and simple to configure and execute as well as be able to provide clear, consistent and usable feedback for correction of user contributed code, configuration errors, workflow errors, data errors, and operating system issues.

## Usability study

The second study complete is a hands-on usability study. Three Computer Science Masters students in the Data to Insight Center of Indiana University installed, configured, and used 4 workflow systems and evaluated their experiences. Trident, IBM Smash, Taverna, and Triana were used in this study. The primary evaluation criteria for this study were the ease of installation, the ease of creating and running a simple workflow, the ability to integrate a well-known external application into a workflow, and overall usability including support options. A summary of the results of the evaluation follows. For the full report, please see Appendix B.

## Evaluation Summary

1. Ease of setup is defined as the total time to download all of the required software, install all of the components, run the setup and configuration process.

**Trident**. The Trident application itself was easy to download. But additional Microsoft packages were required. The other packages were in numerous locations and took significant time to find. Installing the Trident application was simple and took less than 2 minutes; but the other packages required more effort to install and configure. We discovered that the Trident documentation was out of date and the version of SQL Server that was downloaded was incorrect. We had to download the new version of SQL Server, reinstall SQL Server, reconfigure SQL Server, and reinstall Trident. The total process took over 4 hours.

**IBM** **Smash**. The download and installation took less than 1 minute to download and less than 2 minutes to install. However, since it only operates in a 32-bit environment, we had to install a Virtual-PC with a 32-bit OS.

**Taverna**. Taverna was simple to download and install. The entire operation took less than 4 minutes.

**Triana**. The base software was simple to download; however, many files were missing. The installation environment was difficult to use and was not well documented.

1. Get a simple workflow working. For this study, we designed a simple conditional workflow to add two integers. After implementing this workflow in each system, we evaluated the amount of effort to develop the code, the ability to collect provenance and metadata, and built-in fault tolerance

**Trident**. The sample workflow process required 40 lines of code in c#.NET and took approximately 30 minutes to write. To create and execute the workflow activity took less than 30 seconds. The internal built-in functions were geared towards oceanographic work. Trident has an extensive and structured provenance data for the workflows and the data and manages versions to allow for tracking data changes. Trident has significant internal fault tolerance supporting failure tracking, new resource allocation, and workflow restart.

**IBM** **Smash**. The sample workflow took approximately 6 lines of code to implement in Smash. The workflow required additional 10 lines of code. The documentation describing the input processing was incomplete and made this task more difficult. Smash had a number of built-in functions but most of them are orientated towards business applications rather than scientific functions. Smash does not support provenance data although it does have an extensive error logging process. It does have support workflow restarts and has low fault tolerance.

**Taverna**. To create the workflow required 20 lines of code and took approximately 15 minutes. Taverna has a wide selection of built-in functions as well as user-authored functions. Provenance management provides information on all data transforms as well as on the entire workflow process. Taverna has extensive fault tolerance and workflow restart.

**Triana**. To build a sample workflow required using the internal functions that are combined in a drag and drop environment. Triana has a wide range of built-in functions and provides users with the ability to input new toolboxes of functions. Provenance is collected at the organizational level and has no capability to collect provenance at the data level. Triana has no support for workflow restart and has not internal fault tolerance.

1. Integrate the workflow systems with BLAST – The Basic Local Alignment Search Tool – a biological sequence application supported by the National Institutes of Health and the National Center for Biotechnology Information (NCBI).[[1]](#footnote-1)

**Trident**. To plug in another executable into Trident an argument written in c#.NET must be developed. This requires programming expertise.

**IBM** **Smash**. To integrate an external application in Smash requires a PHP or Groovy script or it can be executed from the command line.

**Taverna**. In Taverna, a beanshell script must be created to invoke an external application.

**Triana**. Triana is designed to support plugin applications.

1. User experience includes documentation, user support, and interface usability.

**Trident**. Trident has excellent documentation with many examples and code samples. The user support for Trident is a less active user forum. Trident has a very easy to use GUI, which is intuitive. But the .NET pre-requisite is a barrier.

**IBM** **Smash**. Smash has poor documentation and no viable web presence. Smash has both phone and email support as well as a moderated forum. Smash has an easy to use GUI as well as a command line interface.

**Taverna**. Taverna has excellent documentation with good examples and is integrated with the myExperiments portal. The user support via phone and email is prompt and accurate. The GUI for Tavern is complex and requires some effort to learn.

**Triana**. Triana has minimal documentation that hampers its usefulness. There is no discernable user support for Triana. Triana has a very easy to use interface that allows users to drag and drop objects from the toolbox to create workflows.

## Discussion

### Scientific workflow systems are like tools for domain scientists, tools that allow them to “plug together” components for data acquisition, transformation, analysis and visualization to build complex data-analysis frameworks from existing building blocks, including algorithms available as locally installed software packages or globally accessible web services. Installing and configuring these systems is not a trivial activity and requires an understanding of software components, database administration, scripting, and in some cases, programming with sophisticated languages. This is a significant barrier to use by many researchers, particularly those not in computationally based sciences. Accessing the robust functionality of the systems often required scripting or programming again posing barriers to researchers. As many domains embrace in silico research, the technical skills of researchers will increase and perhaps the barriers will not be as high. But in this transition phase, these tools may cost too much in terms of time and staff to implement.

## Planned Quantitative Study

We are planning to conduct an additional study to determine use patterns of current users of the 10 major workflow systems. We are interested in understanding which workflow system(s) are being used, the frequency of use, the purpose of use, ease of use, and the computing platforms used. The survey will also help us determine the feature sets that are important to researchers and to determine if the importance of features is dependent on scientific domain, by researcher position, or by type of project.

All of the data will be collected via a web-based survey instrument (see Appendix C). The participants will be asked to identify their research position (Principle investigator, Researcher, Post Doc, Ph.D. candidate, Student, Other), their primary research institution, and scientific domain. The survey will be available on the web for 6 weeks after the first solicitation emails are sent. Since we want to understand the use patterns and the expectations of users of workflow systems, we must target people who have used these types of systems. We will send participation requests to the listservs of each of the workflow systems listed below.

Table 2. System Users to be Surveyed

|  |  |
| --- | --- |
| System | URL |
| Ode | <http://ode.apache.org/> |
| IBM smash |  <http://www-01.ibm.com/software/webservers/smash/>  |
| Inforsence | <http://www.inforsense.com/technology/agile_analytical_workflow/index.html> |
| Kepler |  <https://kepler-project.org/>  |
| Lims |  <http://www.cambridgesoft.com/solutions/details/?fid=189>  |
| Pipeline pilot | <http://accelrys.com/products/pipeline-pilot/>  |
| Swift |  <http://www.ci.uchicago.edu/swift/index.php>  |
| Taverna  |  <http://www.taverna.org.uk/>  |
| Trident |  <http://tridentworkflow.codeplex.com/> |
| VisTrails |  <http://www.vistrails.org/index.php/Main_Page>  |

**Status of the new study**

 The survey instrument have been developed and tested. The study has been submitted to the submitted to the Indiana University Human Subjects Review Board. Due to new processes, we are waiting for the results of a prerequisite test before we can proceed. We expect that results can be available from this study 3 months after the Human Subjects approval.

 Week 1 – 4 Administer the survey

 Week 5 – 8 Statistical analysis of the results

Weeks 9 – 12 Write up results

# Workflow Systems in the Research Environment

Fully incorporating workflow systems into the research environment requires several different approaches that include the classroom, research agendas, and research communities.

## Workflows in the Classroom

Integrating research into the classroom is an important component in disseminating new knowledge and engaging students. This spring 2011, Professor Plale offered a graduate level class in the School of Informatics titled ***CSCI B669, Scientific Data Management and Preservation.*** Readings were taken from “Scientific Data Management: Challenges, Technology, and Deployment” by A. Shoshani and D. Rotem Eds. CRC Press. 2010 and The Fourth Paradigm <http://research.microsoft.com/en-us/collaboration/fourthparadigm/>. Trident was one of the platforms upon which students could base their final project.

Astrophysics PhD student, Cameron Pace, chose to develop a Trident workflow to simplify the process of calculating the magnitudes of telescopic observations, specifically nightly extinction coefficients. The workflow applyies the transformation to the raw data as well as determines if the nightly data set is good. This process can be used in discovery process of new black holes and refining the understanding of the energy jets they radiate. Cameron commented on the ease of use of Trident, despite his self-proclaimed weak computer science background. Cameron had to learn basics of C#, and had examples to draw from but nonetheless had something running in a short period of time (a couple weeks at the end of a semester). Trident will only produce one plot per workflow whereas Cameron would like to make several plots per workflow, each pot corresponding to a given star field. Our software engineer gave him code that allowed concurrent execution of threads from Trident, but he replied

*“ I learned a lot as I worked on it, and I feel that the astronomy community can benefit from using workflows.  […] However, I haven't decided if I want to release my project to the general astronomy community.  Most astronomers use a Linux system or a Mac since* [*IRAF*](http://en.wikipedia.org/wiki/IRAF)*, which is the bread & butter astronomy program, won't run on Windows and many astronomers are unaware of the likes of Cygwin.  I get the feeling that my workflow would therefore be underutilized.”*

## Workflow Research Agenda

We are developing two research agenda for workflow systems. One stream involves the workflow interoperability. Ongoing research into the viability of sub-workflow interoperability where sub-workflows are shared between systems. We are looking to develop comparative data on the costs, both quantitative and qualitative of adopting the strategy. We undertook to fill in the gap in knowledge through a performance evaluation and qualitative evaluation of sub-workflow interoperability that fixes the high level system at a user desktop workflow engine, and explores the performance and programming impact of various forms of remote activity.

The second research stream involves using workflow services in a new domain – digital curation. It is becoming a commonly held position that preserving digital preservation needs to be integrated into the data creation process. Developing and releasing workflow subcomponents that would create and monitor the necessary data and metadata for preservation as well as workflow activities to deposit data into domain repositories would provide an excellent test bed for an active curation model.

We expect to develop projects and publish our findings on these two research streams.

## Workflows Communities

In the spring of 2011, we carefully examined with public face of Trident – the CodePlex site. We reviewed the interaction with the research community and determined that while the Trident Workflow System is an excellent product, developing a robust community of researchers will take effort. The full report can be found in Appendix D. In summary, there are three major barriers to overcome: communication, code contributions, and the creating new custom code.

To facilitate communication, Trident should follow the lead of all of the other scientific workflow systems with which we are familiar and develop a community listserv. Email via listserv is the normal communication medium for academic scientific communities. While the Trident/CodePlex systems allows for the discussion to be read via email, it is not possible to contribute to the conversation without going to the Trident CodePlex site, signing in, going to the right tab, and then contributing. For most researchers, the number of steps and the time required will inhibit their contributions. Trident CodePlex needs a simple listserv that will allow 2-way communication within email along with a simple to access archive of old threads and conversations.

Currently, it is difficult for knowledgeable people like our own developers to navigate the complex Microsoft/CodePlex organization, to get authorized ids, communicate the nature of the update (base code, not a sample). Compared to other open source sites, CodePlex is completely opaque. We acknowledge that controls need to be in place to monitor code contributions, but the current restrictions are too much. The barriers for contributing code are too great overcome, even for a dedicated, power user. Unlike contributing to the base source code, contributing samples should be simple and without overt approval. The community can police samples by commenting, wiki text updates, and discussion. Currently, contributing samples has the same issues as contributing source code.

 As described in the previous section, Trident requires that all executable code be in the .NET framework. While a powerful and highly useful technology, it can prove to be a barrier to use by non-programmers. It would be very useful to have a simpler way for researchers to develop code.

**Outreach to New Communities of Researchers**

We propose to develop three new workshops designed to introduce workflows in general and Trident specifically to research scientists. Two workshops would be held in conjunction with established conferences and one workshop would be held as an independent event. The first workshop is entitled “Introduction to Trident Scientific Workflows” which would be conducted in conjunction with an existing conference. We would like to find a venue that is a new community for workflow systems such as the digital library community, which could greatly benefit from a standard way to automate processing flows. There are several opportunities up coming.

The second workshop would be entitled “Trident Scientific Workflows for Biology” which would be conducted as an independent event in Indianapolis at the IUPUI Conference Center and would be a joint project of the Data to Insight Center and the Indiana Clinical and Translational Sciences Institute (CTSI). This workshop would incorporate the Microsoft Biology Foundation toolkit with Trident.

The third workshop would be entitled “Trident Scientific Workflows for Climate Studies” which would be conducted in conjunction with an existing conference. We are continuing to look for an appropriate venue for this workshop. We plan on incorporating the Weather Research & Forecasting Model (WRF), a major weather and climate modeling engine, into this workshop.

 The full proposal for the Trident workshops can be found in Appendix E.

# Recommendations for Trident

There are very many approaches to workflow which are largely successful in prototype one-off situations. However experience has found that most are not really robust enough for production use outside the development team. This observation motivated Microsoft to put their Trident workflow environment [[24](#_ENREF_24), [25](#_ENREF_25)] into open source for science. Trident is built on the commercial quality Windows Workflow Foundation.

Through the two studies already completed and our analysis of the wider scientific community, we have developed a list of recommendation for Trident. While an excellent workflow system, we believe that with minimal effort, Trident can be improved to be useful to more researchers.

Better installation package that includes all required software components and the compatible versions of everything. The installation process should install and configure all software components (SQL Server as an example) (see section 2.1.2 Summary item 1).

Several of the workflow systems have integrated scientific functions, most notably Kepler (see section 2.2.1 Specialized Activities). Trident could benefit by having more built-in functions for scientific data. Integrating the MBF would be an excellent first step.

The ability to use scripting languages as well as .NET would make Trident more accessible to non-programming researchers. As described in section 2.1.2 Summary item 2, Trident required significantly more code to implement a simple function within a workflow than did other systems that supported scripting languages.

Improve the CodePlex site to better facilitate communication with the research community and to allow for easier code sharing as discussed in section 3.3.

Trident is an easy to use workflow system that has potential to significantly improve both the productivity of researchers and the quality of research for research in social sciences, environmental sciences, social-ecological research, operations hurricane prediction centers and other areas where Windows are part of the compute platform upon which research/operations is conducted.

# References

Web Services Business Process Execution Language Version 2.0. <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-specification-draft.html>

W. van Der Aalst, A. Ter Hofstede, B. Kiepuszewski, and A. Barros. Workflow patterns. Distributed and parallel databases, 14(1):5{51, 2003.

Yu and R. Buyya. A taxonomy of scientific workflow systems for grid computing. ACM SIGMOD Record, 34(3):44-49, 2005.

I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. Mock. Kepler: An extensible system for design and execution of scientific workflows. In Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on, pages 423-424. IEEE, 2004.

Y. Zhao, M. Hategan, B. Clifford, I. Foster, G. Von Laszewski, V. Nefedova, I. Raicu, T. Stef-Praun, and M. Wilde. Swift: Fast, reliable, loosely coupled parallel computation. In Services, 2007 IEEE Congress on, pages 199{206. IEEE, 2007.

M. Zur Muehlen. A Framework for XML-based Workflow Interoperability: The AFRICA Project. In Americas Conference on Information Systems. Citeseer, 2000.

R. Barga, J. Jackson, N. Araujo, D. Guo, N. Gautam, and Y. Simmhan. The trident scientific workflow workbench. IEEE International Conference on eScience, 0:317{318, 2008.

G. Kandaswamy and D. Gannon. A Mechanism for Creating Scientific Application Services on Demand from Workflows. In International Conference on Parallel Processing Workshops, pages 25-32, 2006.

E. Elmroth, F. Hernandez, and J. Tordsson. Three fundamental dimensions of scientific workflow interoperability: Model of computation, language, and execution environment. Future Generation Computer Systems, 26(2):245-256, 2010.

1. <http://blast.ncbi.nlm.nih.gov/Blast.cgi> [↑](#footnote-ref-1)