Workflow in Grid Systems

Geoffrey Fox[1,2,3], *Dennis Gannon*[2]
[1]Community Grids Laboratory, Indiana University
501 N Morton Suite 224, Bloomington IN 47404
[2]Computer Science Department and School of Informatics, Indiana University
[3]Physics Department, Indiana University

***Abstract*—This paper provides an overview of the Concurrency&Computation: Practice&Experience special issue on workflow in Grid systems. It is based on discussions at the Global Grid Forum GGF10 Workflow workshop in Berlin, March 2004 and subsequent analysis of the final papers. We describe the background from both a Grid and Global Grid Forum perspective and the distinctive features of the application requirements. We categorize different types of workflow emphasizing the important input from the distributed computing community. We discuss separately both the different workflow systems (often called (run-time) enginres) and the expression languages like BPEL. We highlight some important outstanding research issues in the conclusion.**

***Index Terms*: Workflow, Grid Application, Grid Services, Web Services.**

I. Introduction

On Mach 9, 2004 the Global Grid Forum hosted a workshop on the topic of workflow in Grid systems. Our goal in organizing this workshop was twofold. First, we wanted to survey and contextualize the very large spectrum of work already going on within the Grid research community on workflow programming and enactment [Yu]. Second, and perhaps more important, we wished to understand and to articulate the major problems that remain to be solved in this area. If the GGF community can bring some clarity to these outstanding research themes, we may be able to help focus the community on solutions.

The fact that the Grid research community has such a strong desire to define the role of workflow in Grid systems may come as some surprise to people in the business world. The Workflow Management Coalition (WfMC) has existed for over 10 years and they have standard reference models, documents and a substantial industry of tools and workflow management support products. Why has the Grid community

This work was supported in part by the U.S. Department of Energy under Grant DE-FC0201ER25451 and the NSF under Grants ANI-0330613 and ATM-0331480.

e-mail: gcf@indiana.edu, gannon@cs.indiana.edu.

not adopted these existing standards? While it is not uncommon for the scientific community to re-invent technology rather than purchase existing solutions, there are issues involved in the technical applications of Grid systems that are unique to science and go beyond the models of workflow of the past. For example, in 1996 the WfMC defined workflow as:

The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules.[allen].

Clearly, this definition does not accurately capture the current business world, which is in the process of being transformed by the needs of e-commerce and a move to web services. In the case of workflow in science and engineering, the primary topic of the papers here, workflow concepts have evolved from distributed programming using systems like Linda [Linda], AVS [avs] and Khoros [khoros] and complex shell scripts to a suite of sophisticated programming systems described below. Similarly, the Grid world has evolved from simple toolkits to authenticate users on remote supercomputers, to a service-based architecture intended for activities ranging from supporting large, distributed virtual organizations for e-science to autonomic and on-demand computing for commercial enterprises. The applications of Grids are equally diverse. In the current Grid context one may postulate the definition of workflow to be:

The automation of the processes, which involves the orchestration of a set of Grid services, agents and actors that must be combined together to solve a problem or to define a new service.

What this workshop has accomplished is to explore the various meanings of "combine", "service" and "actor" in this vague definition, and to shed some new light on nature of the problems that we seek to solve.

Thirty five papers were submitted to the workshop and 10 papers were selected for full presentation and an addition 11 authors were represented in focused panel sessions. This special issue contains an additional 7 papers that were submitted in response to the call for this publication. The GGF meeting followed another workshop at the Edinburgh e-Science Center which was held in December of 2003 and a summary of that meeting was presented by Dave Berry to start the GGF workshop. The workshop had one invited presentation by Frank Leymann from IBM. In the paragraphs that follow, we attempt to present an overview of the findings of the workshop and discuss the contributions made by the papers presented in this special issue.

II. CHARACTERIZING THE APPLICATIONS

It is important to note that the workshop focused primarily on the problems of workflow in scientific applications. This, in part, reflects the origins of the GGF but it also reflects the fact that most commercial and enterprise-level applications seem to be well served by emerging standards like the Business Process and Execution Language for Web Services WS-BPEL [bpel]. However, this may be an illusion. In fact, the penetration of Grid technology into science is much greater than it is in the commercial sector. And what may seem like a solved problem in the commercial-grid sector may, in fact, be a problem with dimension not yet fully understood in the Grid science community.

The applications described in the workshop included:

1. The search for gravitational waves coming from compact binary star systems.
2. Bioinformatics and systems biology.
3. The genetic analysis of autoimmune diseases.
4. Large scale data analysis for particle physics.
5. Obtaining bayesian networks from data
6. Ecological studies of invasive species.
7. Geological investigations of igneous rocks.
8. Computational fluid dynamics
9. Prediction of short range weather phenomena.

The nature of these applications of workflow varied dramatically. The time it takes to execute workflows from this collection ranged from seconds to months and the number of entities participating in the workflows went from two or three to tens of thousands. In some cases people are part of the process. For example, some workflow may require human approval or intervention at some point during the workflow enactment. In some cases the workflow is monitored and controlled by a user in real time, but it is often submitted as a "batch" process to be enacted where and when the resources are available. In other cases the work flow may need to be defined by classes of requirements for outcomes, which, in turn must be compiled into specific enactment steps.

A. A Workflow Hierarchy

Workflow, as practiced in scientific computing, derives from several significant precedent programming models that are worth noting because these have greatly influenced the way we think about workflow in scientific Grid applications. We can call these the "dataflow" model in which data is streamed from one actor to another. As previously mentioned AVS and Khoros were early important examples of this model. The other influence comes from something we can call "distributed parallel programming" in which actors

invoke each other via remote procedure calls. The Common Component Architecture as implemented in XCAT [xcat], HeNCE [hence] and the ICENI [iceni] project are examples of such systems. A major structural difference between these to models is the way they are synchronized and controlled. In a purely dataflow based system, the synchronization is fully distributed: the activities are determined by the arrival of data and there is no "central" control that is explicit. In the distributed parallel programming model there is often an explicit component that is the centralized control that sequences the interactions between the component agents. While the pure dataflow concept is extremely elegant, it is very hard to make work in practice because distributing control in a distributed system can create applications that are not very fault tolerant. Consequently, many of the systems that are described here that use a dataflow model for expressing the computation may have an implicit centralized control program that sequences and schedules each action. As we shall see in the systems described here, the "heritage" of the programming model will often be obvious and can help us understand programming philosophy of the system. It is interesting to contrast the dataflow and graphical programming approach between the parallel programming and distributed systems environments. The "scalable" (large number) of decomposed parts and strict synchronization of most large parallel scientific codes has made the graphical GUI and dataflow paradigm unpopular in this field. However they clearly are very effective in a broad class of (functional rather than data parallel) distributed or Grid applications.

Of the many possible ways to distinguish workflow computations on Grids, one is to consider a simple complexity scale. At the most basic, and arguably the most common, level one can consider simple linear workflows in which a sequence of tasks must be performed in a specified linear order. The first task transforms an initial data object into new data object which is the "input" to the next data-transformation task, etc. The execution time for the entire chain of tasks may be a few minutes, or it may be days. In the cases where the execution time is short, the most common workflow programming tool is a simple script written in Python or Perl or even Matlab. The case of longer running workflows often requires more sophisticated tools that are described below.

At the next level of complexity, one can consider workflows that can be described by an *acyclic graph*, where nodes of the graph represent a task to be performed and edges represent dependencies between tasks. This is harder to represent with a scripting language without a substantial additional framework behind it, but it is not at all difficult to represent with a tool like Ant [karajan, ogre] and it is the foundation of the DagMan [deelman] workflow system used by the Condor project. Applications that follow this pattern can be characterized by workflows in which some tasks depend upon the completion of several other tasks which may be executed concurrently. For example, parameter studies in which a number of identical tasks must be initiated with slightly different input conditions and, when they are all complete, a summary task is needed to finish the work.

The next level of workflow complexity can be characterized *cyclic graphs*, where the cycles represent some form of implicit or explicit loop or iteration control mechanisms. In this case the workflow "graph" often describes a network where the nodes are either services or some form of software component instances or represent more abstract control objects. The graph edges represent messages or data streams or pipes that channel work or information between services and components. Many of Grid workflow tools, including many described at the workshop and in this volume, correspond to this model. We will describe these in greater detail below. What distinguishes these applications from the acyclic graph of tasks model is that the nodes represent services that are "connected" to other services. This communication often takes the form of a dialog or sequences of transactions. It is often said that the paradigm in [xcat, iceni] can be likened to "composition is space" to distinguish it from the "composition in time" that characterizes an acyclic graph of tasks where edges represent execution order dependences. Data decomposition in parallel computing corresponds to "composition in space". Another distinguishing feature of some systems – especially the acyclic graphs – is the critical importance of "job-scheduling" of the type seen in Condor [Condor].

This cyclic graph model is also often considered to be a type of Grid-level distributed static dataflow. In some cases this is exactly what it is: each node is processing a stream of messages and pushing results streams to its downstream connected neighbors. For example, a distributed tool to process events in "real time" from instrument sources.

It can be argued that this cyclic graph of communicating services/components can be "unwound" into an acyclic graph of tasks. This is correct only if there is an a priori way to tell how many cycles are executed in the graph prior to termination of the workflow.

The final level of workflow is one in which a compact graph model is not appropriate. This may be the case when the graph is simply too large and complex to effectively "program" it as a graph. (However, some tools allow one to turn a Graph into a new first-class component or service, which can then be included as a node in another graph (a workflow of workflows). This technique allows graphs of arbitrary complexity to be constructed.) A more significant case in which a static graph model fails is when the very structure of the workflow is not static. It may be the case that the workflow is driven by events that determine its structure. It may also be the case that the workflow structure is defined by processes that are negotiated at runtime. For example, suppose one component service must be replaced by another and that new service requires other new services to be connected into the picture. These new services may also require a different organization of the upstream components of the workflow. Or, for example, a workflow may be able to dynamically optimize its structure when it sees that a component service is not needed.

While applications of workflows of this complexity may appear to be science fiction at the present time, the application of Grids to autonomic systems may require this level of intelligence and adaptability.

Finally, the static graph model may fail as a means to characterize the workflow in the case that the graph is implicit as in the case when the workflow is expressed as a set of desired outcomes that can be satisfied by a number of different workflow enactments. For example, a data query may be satisfied moving a large data set across country or it may be cheaper to recreate the data in a local environment. This is a simple example of the Griphyn virtual data concept [griphyn].

III. The Workflow Systems.

The workshop had presentations on a number of significant workflow systems. Many of these address the "cyclic graph model" with a compositional tools based on graphical layout system that allows users to move "components", which represent tasks or services, from a palette to an assembly panel. Using typed input and output ports, the programmer connects together the graph and then executes it. This is certainly not a new idea. The AVS graphics system [avs] was an early component based framework for building graphical scientific application. In the cases of distributed systems a number of previous systems operate in this mode [manish, xcat, webflow]. However, this new generation of tools is the first to directly address the issues of Grid programming.

The **Triana** system, described in the paper "Programming Scientific and Distributed Workflow with Triana Services" [triana], is an excellent example of this concept. At the user's level Triana provides an elegant and well tested composition tool and a large toolbox of ready-to-use components. For Grid application, Triana uses a software layer called the GAP to distribute subsystems of the workflow graph to remote Grid resources for execution. Triana has a mechanism to move a subgraph of the workflow to remote hosts for execution. Triana also has a mechanism that allows an arbitrary remote web and grid services to be imported into the computation by transforming the WSDL document for the service into a local proxy component.

The paper "Scientific Workflow Management and the Kepler System" [kepler] describes **Kepler**, which approaches many of the same problems as Triana. Kepler is used in several large Grid projects where the management of biological data analysis workflows is critical. The approach Kepler takes is based on an actor-oriented model which allows hierarchical modeling and dataflow semantics. The Kepler tools support a well-designed graphical composition interface that is very intuitive and easy to use. To support the interaction with web services Kepler uses a form of actor proxy for each web services that is invoked.

In addition they have created a set of Grid actors for doing GridFTP file management and Globus GRAM execution. The paper also provides a very elegant study of the connection between the Kepler's foundations and higher-order functional programming.

Another workflow system, **Taverna**, that has been extensively used in life science applications is described in the paper "Taverna: Lessons in creating a workflow environment for the life sciences" [taverna]. Taverna is part of the myGrid project, which is building middleware to support data-intensive experiments in molecular biology. The Taverna team's excellent paper provides many insights into the problems of doing scientific workflows in a web service oriented environment and several are worth repeating here. Building workflows based on service composition is a powerful approach and they have over 1000 services that can be used as components in workflows. However, solving the problems of service discovery and selection become non-trivial parts of the process when the potential catalog of workflow components is large. Another issue they point out is that many stand-alone services have powerful data analysis and visualization user interfaces that are provided to the user. By use the service as a single component in a workflow, one bypasses these capabilities. This passes the problem of doing final data analysis and visualization to the end of the workflow process where it is often hard to replace what has been given up along the way. A major problem that Taverna addresses is that of capturing the full metadata context including the provenance of all aspects of the scientific experiment that the workflow represents. This includes the data derivations and the workflow's audit trail of invoked services. A critical feature of e-science is the ability to enable the repeatability of experiments.

Several papers address the role of Petri nets in Grid workflows. In "User Tools and Languages for Graph-based Grid Workflows", [hoheisel] Hoheisel addresses the problem of graph complexity by dynamic workflow refinement as part of the process of transforming the abstract Petri net graph into a concrete one used for execution. He also shows how the Petri net model can incorporate both implicit and explicit exception management.

In the paper "Grid-Flow: A Grid-Enabled Scientific Workflow System with a Petri Net-Based Interface" [gridflow] Guan and co-authors describe another Petri net-based workflow system called **Grid-Flow**. Their approach to Petri net-based tools is to employ a generic modeling system called GMI to define the instance of the Petri net user interface tools. The Petri net specification is then compiled into a lower level Grid-Flow Description language which is executed by the workflow engine. The paper "Workflow applications in GridLab and PROGRESS projects" [progress] describes another Petri net inspired workflow system **GRMS**. They describe this system and compare it to Dagman and Triana and discuss its implementation in the PROGRESS portal.

In the paper "ScyFlow: An Environment for the Visual Specification and Execution of Scientific Workflows" [skyflow] the authors describe another directed graph based workflow tool that is designed to manage NASA's large scale simulation and data analysis work. **SkyFlow** can handle both control flow and parameterized data flow within any given workflow. In many ways, this is a very powerful, lightweight alternative to many more complex systems. In the paper, "Automatic Grid Workflow Based on Imperative Programming Languages" [cepba] the authors take the approach of parallelizing sequential programs to generate Grid workflows and they compare their approach to many other systems described here.

In "What makes workflows work in an opportunistic environment?" [deelman] the authors concentrate particularly on issues of data management and we draw from the experiences with mapping and execution systems: **Pegasus, DAGMan** and **Stork**. They address the particularly important problem of resource allocation and planning for workflow execution involving many large data-intensive tasks.

IV. Expressing Workflow

As noted above there are a host of very powerful and useful workflow composition tools with excellent graphical user interfaces. These are intended for the class of workflows that are modest size, i.e. less than a few thousand nodes, cyclic or acyclic graphs.

For larger systems, or for workflows of most complex dynamically structured type, something that resembles a complete programming language is required. Analogously in conventional programming, one distinguishes user environment (Interactive Development Environment IDE), programming language (say Java or C#) and runtime. The latter is called "enactment engine for Grid workflow. BPEL is the leading candidate for a "complete programming language" for workflows. In fact, BPEL provides two usage patterns for expressing workflow behaviors. The first usage pattern is based on the concept of an abstract process which represents a role, such as "buyer" or "seller" in a business process and the "graph" based language to describe their interaction. These interactions are defined by partner links and BPEL provides the language to describe the public aspects of the protocol used in the interaction between these partners. The other usage pattern of BPEL involves the "algebraic" or algorithmic language needed to define the logic and state of an executable process in terms of web service resources and XML data, and tools to deal with exceptions and failures.

From the perspective of Grid workflows BPEL brings several important features together that are not well represented in most of the other systems described here. First, a BPEL workflow, once enacted, can be a very long-lived entity. In high quality implementations, it exists as a web-service that is executed by a workflow engine that is capable of suspending the state of the instance in a database. This means a workflow instance can wait weeks or months for an event to happen that can cause it to be re-activated and

respond. Furthermore, BPEL has control structures that allow for very dynamic responses and fault recovery.

Three papers in the workshop discuss the role of BPEL. Frank Leymann from the University of Stuttgart and IBM described BPEL in both philosophy and design. Leyman's article [leyman] addresses how BPEL can be used in concert with the implied resource pattern of WSRF. He shows how resource lifetime and transactions can be addressed using BPEL. He also considers the problem of monitoring, but concludes that additional support is needed to solve this problem. Francisco Curbera, Rania Khalaf, William Nagy, Sanjiva Weerawarana provided a paper [papco] that describe the issues involved in the implementation of BPEL4WS and Aleksander Slominski [alek] also describes the issues involved with integrating BPEL with the OGSI and WSRF Grid standards. He also considers some extensions to BPEL to support data management through a type of pseudo partners. Finally Dieter Cybok's paper [cybok] describes a very BPEL-like language called GWEL which was designed to test workflow concepts in the context of OGSI and the Globus version 3 toolkit.

In the course of the workshop several significant issues were raised about workflow expression. First among these is the issue of the level of abstraction needed to describe workflows. The graph based construction toolkits like Triana, Taverna, ICENI and others allow the user to describe workflow at the level of a graph of components. It is then left to other tools to map this abstraction to specific resources. BPEL is at a different level of abstraction in which the programmer specifies processes and their logic in terms of services that are bound at a later time.

At a higher level we have systems like Chimera [chim] in which the workflow is defined in terms of virtual data requirements. Chimera abstract dependencies are mapped to lower level involved with service discovery, planning and execution. Chimera uses Pegasus with Dagman [deelman], which is a combination of two systems that incorporates adaptive resource planning and scheduling into the execution of the workflow. Using this approach they have been able to dynamically schedule very large (many thousands of task nodes) workflow on production grids.

An important question that was raised several times at the meeting was the role of BPEL as a standard language for orchestrating services-based Grid applications. There are two important approaches to using BPEL in other Grid systems. The first of these is using BPEL's own extensibility to create "towers" specific to particular Grid styles or application domains. The second approach is to consider BPEL a target language for higher level languages such as those discussed here. This type of use for language is common in conventional programming where on distinguishes the languages used in the different phases of a programming task – those used for compilers, script engines, virtual machines, machine code are rather

different. One could imagine using a dataflow GUI as the programming model and BPEL as the Grid machine code. Further especially at the user level, one has many different languages expressing different application requirements and programming models; one should expect multiple languages to be needed for Grid workflow.

V. Issues in Workflow Enactment

There are several significant issues that have been raised for workflow enactment in the context of Grid systems. Of course for those interested in scientific applications, performance is always an issue. In the case of workflow enactment, there are two aspects to this: efficiency and robustness. In terms of efficiency, the critical issue is the ability to quickly bind workflow tasks to the appropriate Grid resources. It also depends very heavily on the mechanisms used to move data between tasks and services that need them at various stages of the enactment. One cannot assume that web service protocols like SOAP should be used in anything other than "control" and simple message delivery. Real data movement between components of the workflow must be either via an interaction with a data movement service, or through specialized binary-level data channel running directly between the tasks involved.

Robustness is another issue. We must assume that various parts of a workflow will fail. BPEL and a few other workflow systems have extensive exception handling capabilities. It is essential that exception handling include mechanisms to recover from failure as well as detecting it. Also failure is a something that can happen to a workflow enactment engine. If we assume that the workflow lasts for a long period (days or months) we must assume that its execution will not be a continuously running process. It may more closely resemble a document that is updated over time as its author has occasion to work on it.

A related issue is the monitoring of the workflow. In addition to being able to restart the workflow from a failure checkpoint, the user may wish to track progress of the enactment. In some cases the workflow is event driven and a log of the events that trigger the workflow processing can be used to "replay" an animation of how the workflow progressed. This is also an important aspect of debugging a workflow. A user may wish to "single step" the workflow execution to understand potential errors in the flow logic.

Alternatively, the user may actually wish to "steer" the workflow as it approach various points where human intervention about resource or algorithm decisions.

VI. General Research Issues

There are a number of significant general issues for workflow authoring and enactment in Grid systems. Many of these point to general research problems for the future.

One area that has received little attention in the current Grid workflow projects is the issue of security. Howard Chivers observed "workflow systems are often interposed between users, data and services without considering the trust responsibilities that this design imposes on planning and enactment systems." He goes on to suggest, "we need a Systems Design approach that separates enactment and protection by refactoring the protection requirements away from planning and enactment and into the distributed system." It is dangerous to trust a complex workflow enactment to assure resources are protected. To what extent can we delegate a user's identity/authority to a complex workflow enactment that may require a large set of capabilities to complete a set of distributed tasks? Chivers' paper [chivers] is an updated reflection on many of these themes.

Another interesting area of research involves the way in which we can use a workflow document as part of the scientific provenance of a computational experiment? Under what conditions can we publish a workflow script as a document that can be used by others to verify a scientific claim? If the workflow was triggered by sequences of external events, can the monitoring of the workflow capture these events well enough so that the enactment can be repeated?

There are a variety of places where Semantic Grid research will have an impact on workflow in Grid systems. The provenance questions stated above are certainly among these. Other include finding better ways for our workflow to automatically generate the metadata about the way the work was done so that a later search for a method to solve a particular problem can turn-up the appropriate patterns of solution. More generally, how can a system discover/synthesize a workflow if the only thing the user provides is the desired outcome of the enactment? This may involve an integration of case-based reasoning and a very rich service-based Grid. Can we build systems that allow workflows to automatically do incremental self-optimization? Can we automatically discover new properties and services of a Grid that enable such self-optimization?

Another important problem involves the manner in which we bind data sources to workflow patterns and templates. In many scientific applications, the individual workflow components are complex applications that require dozens of parameter settings. In some cases, the parameter settings must be propagated to multiple services and in other cases, the ability to set a particular parameter may depend upon the authorization level of the user. The difficult part is that in many cases, the workflow instance is created on

demand from a user at a Grid portal. The user may not know that the workflow exists: to the user it is a web application. How do we collect all the required parameters needed as inputs to all the services in the workflow and present this to the user as a coherent interface? Clearly it is possible for the designer of the workflow template to also build a web-form page to supply the needed parameters, but there is a need to be able to automate this interface generation process.

There are clearly many more research problems to be addressed. Many topics, both short and long term are addressed in the workshop papers.

ACKNOWLEDGEMENTS

We would like to thank the other workshop organizers for their hard work in putting this event together. They are: Abbas Farazdel, IBM, Carole Goble, University of Manchester, Ewa Deelman, USC ISI, Dave Berry, NeSC, David Angulo, Depaul University, Virinder Batra, IBM, Simon Cox, Southampton Regional e-Science Centre, Francisco Curbera, IBM Research, Tomasz Haupt, MsState University, Piyush Mehrotra, NASA Ames Research Center and Ravi Subramaniam, Intel. We also thank the Global Grid Forum for sponsoring this event.

REFERENCES

1. [alek] Aleksander Slominski, On Using BPEL Extensibility to Implement OGSI and WSRF Grid Workflows, Concurrency and Computation: Practice and Experience [this issue].
2. [allen] R. Allen, “Workflow: an Introduction”, Workflow Handbook, 2001, Workflow Management Coalition.
3. [avs] Craig Upson, Thomas Faulhaber, Jr., David H. Laidlaw, David Schlegel, Jefrey Vroom, Robert Gurwitz, Andries van Dam, The Application Visualization System: A Computational Environment for Scientific Visualization, IEEE Comput. Graph. Appl., vol. 9, no. 4, 1989, pp 30-42. IEEE Computer Society Press.
4. [bpel] Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana Trickovic, and Sanjiva Weerawarana. Business Process Execution Language for Web Services, May 2003. http://www.ibm.com/developerworks/webservices/library/ws-bpel/
5. [cepba] Raul Sirvent, Josep M. Perez, Rosa M. Badia, and Jesus Labarta, Automatic Grid Workflow Based on Imperative Programming Languages, Concurrency and Computation: Practice and Experience [this issue].
6. [chim], Ian T. Foster, Jens-S. Vockler, Michael Wilde, Yong Zhao, Chimera: AVirtual Data System for Representing, Querying, and Automating Data Derivation, SSDBM '02: Proceedings of the 14th

International Conference on Scientific and Statistical Database Management, 2002, pp. 37--46, IEEE Computer Society.

7. [chivers] Howard Chivers, Refactoring Service-Based Systems: How to Avoid Trusting a Workflow Service, Concurrency and Computation: Practice and Experience [this issue].
8. [Condor] Condor Home Page http://www.cs.wisc.edu/condor/condorg/
9. [cybok], Deiter Cybok, A Grid Workflow Infrastructure, Concurrency and Computation: Practice and Experience [this issue].
10. [deelman] Ewa Deelman, Tevfik Kosar, Carl Kesselman, Miron Livny, What makes workflows work in an opportunistic environment?, Concurrency and Computation: Practice and Experience [this issue].
11. [gridflow] Zhijie Guan, Francisco Hernandez, Purushotham Bangalore, Jeff Gray, Anthony Skjellum, Vijay Velusamy, Yin Liu, Grid-Flow: A Grid-Enabled Scientific Workflow System with a Petri Net-Based Interface, Concurrency and Computation: Practice and Experience [this issue].
12. [griphyn] Ewa Deelman, James Blythe, Yolanda Gil, Carl Kesselman, Workflow management in GriPhyN, in Grid resource management: state of the art and future trends, 2004, pp. 99--116, Kluwer Academic Publishers.
13. [hence] Adam Beguelin, Jack Dongarra, G. A. Geist, HeNCE: A User's Guide, Version 2.0, http://www.netlib.org/hence/hence-2.0-doc-html/hence-2.0-doc.html
14. [hoheisel] Andreas Hoheisel, User Tools and Languages for Graph-based Grid Workflows, Concurrency and Computation: Practice and Experience [this issue]
15. [iceni] A. Mayer, S. McGough, N. Furmento, J. Cohen, M. Gulamali, L. Young, A. Afzal, S. Newhouse, and J. Darlington, ICENI: An Integrated Grid Middleware to Support e-Science, in Component Models and Systems for Grid Applications. V. Getov and T. Kielmann, editors. volume 1 of CoreGRID series, p. 109—124, Springer, June 2004
16. [karajan] Gregor von Laszewski and Mike Hategan, Java CoG Kit Karajan/Gridant Workflow Guide, http://www.cogkit.org
17. [Kepler] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Yang Zhao, Scientific Workflow Management and the Kepler System.
18. [khoros] J. Rasure and S. Kubica. The Khoros Application Development Environment. Khoral Research Inc., Albuquerque, New Mexico, 1992.
19. [leyman] Frank Leyman, Choreography for the Grid: Towards Fitting BPEL to the Resource Framework, Concurrency and Computation: Practice and Experience [this issue].
20. [Linda] Nicholas Carriero and David Gelernter *Linda in Context*, Communications of the ACM 32 , 4 (April 1989) pages 444 - 458
21. [manish] Bhat, V. and Parashar, M.. Discover Middleware Substrate for Integrating Services on the Grid. *Proceedings of the 10th International Conference on High Performance Computing (HiPC 2003), Lecture Notes in Computer Science,* Editors: T.M. Pinkston, V.K. Prasanna, Springer-Verlag,

Hyderabad, India, Vol. 2913, pp 373 – 382, December 2000

22. [ogre] Albert L. Rossi, Open Grid Runtime Environment, OGRE, http://corvo.ncsa.uiuc.edu/ncsa-tools
23. [paco], Francisco Curbera, Rania Khalaf, William A. Nagy, and Sanjiva Weerawarana, Implementing BPEL4WS: The Architecture of a BPEL4WS Implementation, Concurrency and Computation: Practice and Experience [this issue].
24. [progress] Michal Kosiedowski, Krzysztof Kurowski, Cezary Mazurek, Jarek Nabrzyski, Juliusz Pukacki, Workflow applications in GridLab and PROGRESS projects, Concurrency and Computation: Practice and Experience [this issue].
25. [skyflow] Karen M. McCann, Maurice Yarrow, Adrian DeVivo, Piyush Mehrotra, ScyFlow: An Environment for the Visual Specification and Execution of Scientific Workflows, Concurrency and Computation: Practice and Experience [this issue].
26. [taverna] Tom Oinn, Mark Greenwood, Matthew Addis, Justin Ferris, Kevin Glover, Carole Goble, Duncan Hull, Darren Marvin, Peter Li, Phillip Lord, Matthew R. Pocock, Martin Senger, Anil Wipat and Chris Wroe, Taverna: Lessons in creating a workflow environment for the life sciences, Concurrency and Computation: Practice and Experience [this issue].
27. [triana] David Churches, Gabor Gombas, Andrew Harrison, Jason Maassen, Craig Robinson, Matthew Shields, Ian Taylor, Ian Wang, Programming Scientific and Distributed Workflow with Triana Services, Concurrency and Computation: Practice and Experience [this issue].
28. [webflow] Dimple Bhatia, Vanco Burzevski, Maja Camuseva, Geoffrey Fox, Wojtek Furmanski, Girish Premchandra WebFlow: A Visual Programming Paradigm for Web/Java Based Coarse Grain Distributed Computing (1997) .Concurrency - Practice and Experience 9:6, 555-577, June 1997.
29. [xcat] Sriram Krishnan, Randall Bramley, Dennis Gannon, Rachana Ananthakrishnan, Madhusudhan Govindaraju, Aleksander Slominski, Yogesh Simmhan, Jay Alameda, Richard Alkire, Timothy Drews, and Eric Webb. The XCAT Science Portal. In Scientific Programming, volume 10(4), pages 303-317, 2002
30. [Yu] Jia Yu and Rajkumar Buyya, A Taxonomy of Workflow Management Systems for Grid Computing, Technical Report, GRIDS-TR-2005-1, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, March 10, 2005. http://www.gridbus.org/reports/GridWorkflowTaxonomy.pdf